TVORBA A PREZENTÁCIA DÁT

DANIELA BEZÁKOVÁ, DANA HORVÁTHOVÁ, ANDREA HRUŠECKÁ, ROMAN HRUŠECKÝ, ĽUDMILA JAŠKOVÁ, MONIKA TOMCSÁNYIOVÁ, PATRIK VOŠTINÁR

EURÓPSKA ÚNIA
Európsky sociálny fond
Európsky fond regionálneho rozvoja

OPERAČNÝ PROGRAM ĽUDSKÉ ZDROJE

MINISTERSTVO ŠKOLSTVA, VEDY, VÝSKUMU A ŠPORTU SLOVENSKEJ REPUBLIKY

it akadémia

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje

www.minedu.sk www.employment.gov.sk/sk/esf/ www.itakademia.sk

Tvorba a prezentácia dát

Spracované v rámci národného projektu IT Akadémia – vzdelávanie pre 21. storočie

Bratislava 2021

Tvorba a prezentácia dát

Spracované s finančnou podporou národného projektu IT Akadémia – vzdelávanie pre 21. storočie

Autori: PaedDr. Daniela Bezáková, PhD., Ing. Dana Horváthová, PhD., PaedDr. Andrea Hrušecká, PhD., PaedDr. Roman Hrušecký, PhD., doc. RNDr. Ľudmila Jašková, PhD., doc. PaedDr. Monika Tomcsányiová, PhD., PaedDr. Patrik Voštinár, PhD.

Recenzenti: Ing. Martin Drlík, PhD., Ing. Martin Džbor, PhD., Ing. Majka Majherová, PhD.,

Neprešlo jazykovou úpravou.

Vydavateľ: Centrum vedecko-technických informácií SR, Bratislava

Rok vydania: 2020

Vydanie : 1. vydanie

ISBN 978-80-89965-67-0 EAN 9788089965670

Bratislava 2020

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu v rámci Operačného programu Ľudské zdroje.

OBSAH

ZOPÁR DÔLEŽITÝCH INFORMÁCIÍ PRE UČITEĽA

Poradie a nadväznosť kapitol

V učebnici sme sa rozhodli paralelne zaoberať obsahom stránky aj jej vzhľadom (zdá sa nám, že to je pre žiakov motivujúcejšie), preto sme pomerne skoro zaradili kapitolu o kaskádových štýloch – už ako tretiu v poradí. Od tejto kapitoly sa teda venujeme nielen HTML jazyku (oboznamujeme žiakov s novými elementami), ale priebežne aj CSS štýlom (oboznamujeme žiakov s novými vlastnosťami elementov).

- Kapitoly 1-9 odporúčame preberať v takom poradí, ako sú uvedené. Je možné vymeniť poradie kapitol 3. a 4. (v kapitole 4 sa štýly nevyužívajú).
- **Kapitoly 10 a 11** sú pomerne náročné a **odporúčame** ich **zaradiť len pre pokročilých žiakov**.
- Kapitolu 12 môžeme zaradiť už po 9. kapitole, nevyužíva nič z toho, čo sa nachádza v 10. a 11. kapitole.
- Kapitolu 13 o validite stránok je možné zaradiť aj skôr: časť o HTML validite je možné zaradiť už po 4. kapitole, časť o CSS validite po 7. kapitole. Možno by bolo ideálne „rozpustiť“ túto kapitolu do všetkých ostatných v zmysle, že validitu stránok by sme mali kontrolovať pravidelne.
- Kapitolu 14 o publikovaní na webe môžeme tiež zaradiť už skôr (napríklad po kapitole 9). Je na zvážení učiteľa, či a kedy bude chcieť, aby žiaci svoje stránky publikovali.
- V kapitole 15 sa používa čítač obrazovky NVDA a program Color Contrast Analyser na tvorbu simulácií porúch farebného videnia. Práca s týmito programami je jednoduchá, ale odporúčame učiteľom, aby sa s nimi dobre oboznámili vopred. Postačí, ak vyriešia úlohy určené pre žiakov. Užitočné informácie nájdu v metodických pokynoch na konci kapitoly.
- V kapitolách 15, 16 a 18 sa pri riešení úloh používajú reálne webové stránky. Na niektorých je potrebné nájsť ťažko prístupné objekty pre niektoré skupiny používateľov. Je možné, že autori časom stránky zmenia a v čase, keď budú žiaci riešiť úlohy, už nebudú takéto objekty obsahovať. V takom prípade odporúčame nájsť iné stránky s ťažko prístupnými objektami. Vodítkom môžu byť metodické pokyny na konci kapitol.
- V kapitole 17 sa používa online automatický nástroj na testovanie prístupnosti WebAim. Práca s ním je jednoduchá, ale aj napriek tomu odporúčame učiteľom, aby sa s týmto nástrojom oboznámili vopred a otestovali ním stránky, ktoré použijú na hodine.
- Kapitolu 19 považujeme za teoretický vstup do nasledujúcich kapitol 20 – 23 a odporúčame ju preskočiť len v prípade, že žiaci majú osvojený pojmový aparát z multimédií.
- V kapitolách 20 a 21 sa využívajú voľne dostupné editory Inkscape a VSDC. Úlohy v týchto kapitolách sa venujú len vybraným, menej štandardným témam. Kapitoly na seba nijako nenadväzujú. Spája ich len spoločná téma pizzérie, ktorá sa vnára do značnej časti učebných materiálov.

- Kapitolu 22 je najlepšie preberať následne po zoznámení sa s multimédiami a s tvorbou videa, avšak je možné ju zaradiť už skôr. Je na zvážení každého učiteľa, kedy chce pracovať s audio a video elementami na stránke, so streamovaním videa a podobne.
- Kapitolu 23 odporúčame zaradiť len pre pokročilých študentov, ktorí sa chcú venovať programovaniu multimédií v jazyku Python. V takom prípade je vo webovej aplikácií k dispozícií vyše 100 lekcií s návodom, ako programovať a spracovávať mediálne elementy pomocou jednotlivých modulov (Pillow, pi3d, Pydub a MoviePy).

Jednotlivé témy majú rôznu (niekedy až veľmi odlišnú) **časovú náročnosť.** V žiadnom prípade neplatí pravidlo, čo téma, to dvojhodinovka.

Čo sa týka **vstupných vedomostí a zručností žiakov** postačujúcich k úspešnému zvládnutiu učiva:

- v kapitolách 1 až 18 ide o základné zručnosti práce s internetom, webovým prehliadačom a textovým editorom,
- v kapitolách 19 až 22 predpokladáme, že študenti spĺňajú štandardy stanovené pre tematickú oblasť Reprezentácie a nástroje v Rámcovom vzdelávacom programe pre gymnáziá,
- pre lepšie porozumenie učiva prezentovaného v kapitole 23 je vhodné, aby mali študenti zvládnuté aspoň základy dátových typov, riadiacich štruktúr (cyklov), zoznamov, funkcií a metód v jazyku Python.

K prehliadačom

- V učebnici sme to nijako nešpecifikovali, ale stránku si môžu žiaci pozerať v ľubovoľnom prehliadači. Ak to softvérové vybavenie školy dovoľuje, je vhodné upozorniť žiakov na to, aby si stránku vyskúšali vo viacerých prehliadačoch (netreba zakaždým, ale napríklad predtým, než ju budú publikovať).
- Väčšina obrázkov je zosnímaná z prehliadača Chrome, zobrazenie v iných prehliadačoch sa môže (len veľmi) mierne odlišovať.

Práca v JSFiddle

V učebnici pomerne často využívame prácu v editore JSFiddle (https://jsfiddle.net/). Používame ho najmä na experimentovanie s elementami, skúmanie vlastností elementov a podobne. **Kódy z JSFiddle** (či už HTML alebo CSS) **si neukladáme do súboru.** V učebnici sa často striedajú príklady, v ktorých využívame JSFiddle editor a príklady, v ktorých využívame klasický editor a editujeme nejaký konkrétny súbor. **Žiakov by sme mali upozorniť, aby editor JSFiddle počas hodiny nezatvárali**, aj keď idú riešiť príklad, v ktorom sa nevyužíva - ten nech si pozrú v inom okne prehliadača. Často totiž neskôr nasleduje príklad, ktorý zasa využíva editor JSFiddle a nadväzuje na zdrojový kód, ktorý sme v ňom napísali predtým. Medzi súbormi pre učiteľa sú aj príklady a riešené úlohy, v ktorých využívame JSFiddle. Tie môže dať učiteľ žiakom v prípade potreby – napr. ak si niekto omylom zatvorí JSFiddle alebo ak na hodine potrebuje nadviazať na to, čo robili žiaci v JSFiddle na predchádzajúcej hodine.

V prípade, že z nejakého dôvodu nechcete používať editor JSFiddle, môžete namiesto neho využívať klasický editor. V takom prípade by však bolo vhodné mať pripravenú šablónu so základnou štruktúrou webovej stránky (z konca kapitoly 1), do ktorej budú žiaci dopisovať.

Súbory pre žiakov

Texty jednotlivých kapitol bez metodických odporúčaní pre učiteľa sú uložené v samostatných súboroch pomenovaných ako *kapitola-cislo_kapitoly.pdf*. Sú určené pre žiakov, ak sa učitelia rozhodnú poskytnúť im ich ako študijný materiál.

Ku každej kapitole sú vytvorené zdrojové súbory pre žiakov v súbore *cislo_kapitoly*.zip. Na tieto sa odvolávame v učebnici.

Počas celej učebnice vyvíjame stránku virtuálnej pizzerie IT Pizza. Pri tvorbe stránky IT Pizza poskytujeme ku každej kapitole zdrojový súbor index.html, ktorý obsahuje všetko to, čo sa so stránkou robilo v predchádzajúcej kapitole, môžu sa v nich však vyskytnúť aj drobnejšie úpravy a odlišnosti vhodné pre potreby aktuálnej kapitoly. Je preto na zvážení učiteľa, či pri prechode na novú kapitolu budú žiaci pokračovať vo svojom vlastnom súbore (čo je možno pre niektorých motivujúcejšie, ale môžu sa vyskytnúť mierne odlišnosti vo vzhľade aj obsahu) alebo v tom, ktorý dodávame s učebnicou (všetci žiaci budú na začiatku kapitoly na rovnakej úrovni). Každopádne odporúčame žiakom sprístupňovať zdrojové súbory postupne po kapitolách, nie zo všetkých kapitol naraz, aby sme sa vyhli tomu, že zdrojové súbory z učebnice začnú využívať ako vlastné riešenia.

Súbory pre učiteľov

Ide o zdrojové kódy príkladov a úloh z učebnice (úlohy sú vyriešené). Na tieto súbory sa v učebnici neodvolávame, sú ale pomenované a očíslované tak, aby bolo zrejmé, ku ktorému príkladu či úlohe v učebnici patria. Sú určené:

- ako pomôcka pre učiteľov na riešenie úloh, treba si však uvedomiť, že ukazujú len jedno možné riešenie, neznamená to, že žiadne iné neexistuje,
- v prípade potreby učitelia môžu konkrétny súbor použiť ako východzí súbor pre žiakov (ak si žiaci zabudli uložiť a pod.)

Projekty

Na formatívne hodnotenie žiakov neodporúčame použiť testy, ale preferujeme tvorbu miniprojektov, prípadne rozsiahlejších projektov. Na miniprojekty možno použiť niektoré úlohy, na ktoré sa odvolávame v metodických pokynoch pre učiteľa na konci každej kapitoly. Zadania rozsiahlejších projektov v tomto texte neuvádzame, ale sú v súbore *projekty*-pre_ziakov.zip a zadania spolu s riešeniami sú v súbore *projekty*-pre_ucitelov.zip.

TVORBA WEBU

DANIELA BEZÁKOVÁ, ANDREA HRUŠECKÁ, ROMAN HRUŠECKÝ, ĽUDMILA JAŠKOVÁ, MONIKA TOMCSÁNYIOVÁ

1 ÚVOD DO TVORBY WEBOVÝCH STRÁNOK

S webovými stránkami (alebo HTML stránkami) sa dnes stretávame denne: hľadáme na nich informácie, komunikujeme s kamarátmi, nakupujeme v online obchodoch, počúvame hudbu, ... Webové stránky majú školy, mestá, osoby, banky, obchody ...

Webová stránka je kombinácia textu, obrázkov, animácií, hypertextových odkazov a ďalších objektov, ktoré sú webové prehliadače schopné zobraziť. Vytvoriť webovú stránku znamená vytvoriť jej obsah („vložiť " texty, obrázky, tabuľky, odkazy a ďalšie spomínané objekty) a rozhodnúť, akým spôsobom sa bude tento obsah zobrazovať, t.j. ako budú jednotlivé objekty na stránke rozložené, aké budú mať vlastnosti, napr. aké farby, aký druh, štýl či veľkosť písma, riadkovanie, medzery medzi jednotlivými prvkami na stránke, atď.

Zdrojový kód stránky, jazyk HTML

Webová stránka je v skutočnosti obyčajný textový súbor, v ktorom je pomocou špeciálnych **značiek** popísané, čo sa má na webovej stránke zobraziť a ako sa to má zobraziť – tzv. **zdrojový kód** stránky.

Webovú stránku môžeme vytvárať dvoma spôsobmi:

1) priamo (pomocou vhodných nástrojov) vkladáme do stránky texty, obrázky, či iné objekty a nastavujeme ich vzhľad. Môžeme to prirovnať k tvorbe textového dokumentu v prostrediach ako MS Word, Libre Office …. V priebehu vytvárania stránky ju už vidíme tak, ako bude vyzerať vo webovom prehliadači. Takýmto spôsobom si môžeme vytvoriť stránku napr. pomocou sites.google.com, estranky.sk či webnode.sk.
2) vytvárame zdrojový kód stránky, teda len **popisujeme**, aké objekty (texty, obrázky, odkazy, ...) budú na stránke a akým spôsobom sa budú zobrazovať.

Predpokladáme, že s tvorbou stránok prvým spôsobom, ste sa už stretli. V tejto učebnici sa budeme venovať druhému spôsobu, teda naučíme sa vytvárať zdrojový kód stránky.

ÚLOHA 1.1

V prehliadači otvorte nasledujúce stránky www.aupark.sk, www.fll.sk a zobrazte pre obe zdrojový kód (v prehliadačoch Chrome, MozillaFirefox aj MS Edge na to môžeme použiť klávesovú skratku **Ctrl+U** alebo **F12**).

- Preskúmajte ho. Viete v ňom nájsť texty, ktoré ste videli priamo na stránke v prehliadači?
- Všímajte si texty obalené v zátvorkách <>, napr. `<head>`. Nájdite aspoň 10 rôznych takýchto značiek. Ktoré z nich sa nachádzajú v oboch zdrojových kódoch?
- Všimnite si, že, mnohé značky sa vyskytujú v kóde v dvoch tvaroch: `<head>` a `</head>`, alebo `<title>` a `</title>`. Nájdite v kódoch oboch stránok značky `</body>`, `</html>`. Kde v rámci súboru sa nachádzajú?

V zdrojových kódoch stránok z *úlohy 1.1* sme mohli vidieť nasledujúce značky: `<!DOCTYPE html>`, `<html>` a `</html>`, `<head>` a `</head>`, `<title>` a `</title>`, `<meta`

`charset="utf-8">`, `<script>` a `</script>`, `<body>` a `</body>`, `<div>` a `</div>` a veľa ďalších. Všetky tieto značky sú prvkami tzv. **značkovacieho jazyka HTML**.

HTML (HyperText Markup Language) je značkovací jazyk určený na definovanie štruktúry a vzhľadu webových dokumentov. Tvoria ho značky, pomocou ktorých definujeme prvky (elementy) stránky a ich rozmiestnenie na stránke. Jazyk HTML nie je programovací jazyk, ani nie je určený na tvorbu či spracovanie textu. V tejto učebnici sa budeme zaoberať jazykom HTML vo verzii **HTML5**.

ÚLOHA 1.2

Pozrite si zdrojový kód stránky svojej školy a nájdite v ňom predchádzajúce značky.

ÚLOHA 1.3

Pozrite si zdrojový kód stránky, na ktorú chodíte najčastejšie. Dá sa v tomto kóde orientovať? Ak áno, pokúste sa v ňom nájsť nejaké značky, ktoré ste našli v zdrojových kódoch stránok z *Úlohy 1.1.*

Teraz skúsime vytvoriť vlastnú webovú stránku pomocou značiek jazyka HTML. Čo budeme potrebovať?

- Zdrojový kód stránky môžeme písať v ľubovoľnom **textovom editore**. Najlepšie je však používať taký editor, ktorý pozná HTML značky, pomáha nám pri ich písaní, farebne ich zvýrazňuje (napr. Atom, Notepad++, PSPad, BlueFish, WebStorm).
- Ak si chceme pozrieť výsledok našej práce, musíme si stránku otvoriť vo **webovom prehliadači** (Mozilla Firefox, Chrome, Opera, Microsoft Edge a iné).

ÚLOHA 1.4

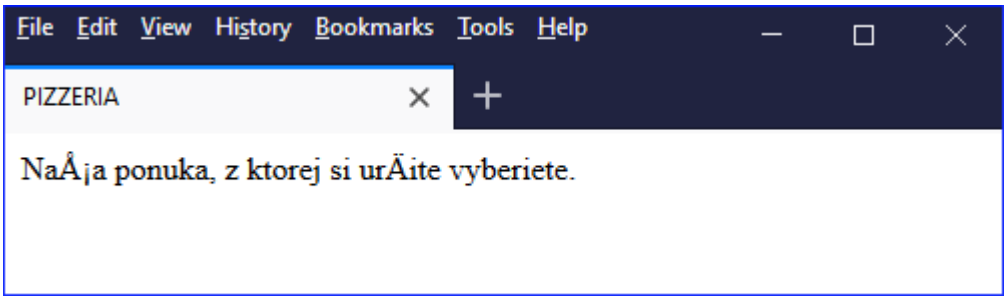
- (v škole) Zistite, aký textový editor vhodný na tvorbu webových stránok, máte na školských počítačoch.
- (doma) Zvoľte si niektorý z editorov uvedených v predchádzajúcom odseku, nájdite ho na internete, stiahnite a nainštalujte na svoj počítač.

PRÍKLAD 1.5

Vytvorme textový súbor (webovú stránku) s nasledujúcim kódom.

```
<html>
<head>
  <title>PIZZERIA</title>
</head>
<body>
  Naša ponuka, z ktorej si určite vyberiete.
</body>
</html>
```

Súbor uložíme pod názvom `index.html` a následne otvoríme vo webovom prehliadači. Mali by sme dostať „výsledok“ podobný tomu na *obrázku 1.1*. To, že sa nezobrazuje správne diakritika (slovenské znaky s dĺžňami a mäkčeňmi), si zatiaľ nebudeme všímať.



Obrázok 1.1 Webová stránka zobrazená vo webovom prehliadači.

ÚLOHA 1.6

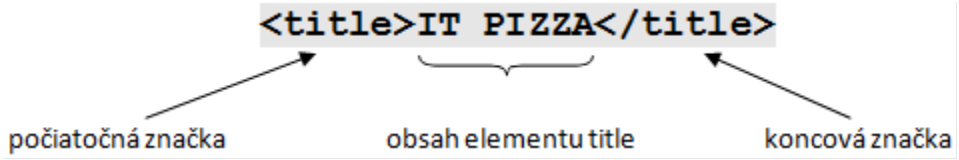
- Doplňte text medzi značkami `<body>` a `</body>` (napr. dopíšte štyri názvy pízz, ktoré poznáte), uložte súbor a znovu ho zobrazte vo webovom prehliadači (ak ste prehliadač nezavreli, stačí obnoviť stránku napr. stlačením **F5**).
- Zmeňte text PIZZERIA medzi značkami `<title>` `</title>` na IT PIZZA, uložte súbor a zobrazte ho vo webovom prehliadači. Zmenilo sa niečo? Kde sa zobrazujte text, ktorý sa nachádza medzi značkami `<title>` `</title>`?

POZNÁMKA

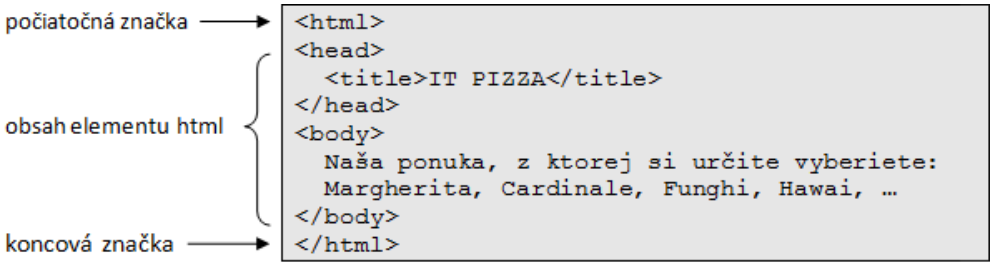
V ďalších úlohách budeme automaticky predpokladať, že ak čitateľ zmení súbor so zdrojovým kódom stránky, tak tieto zmeny uloží a následne si výsledok pozrie vo webovom prehliadači.

HTML elementy

Vráťme sa ešte k jazyku HTML. Základnou jednotkou jazyka HTML je **element**. HTML element sa zvyčajne skladá z počiatočnej značky, koncovej značky a obsahu, ktorý je medzi týmito dvoma značkami. Neskôr uvidíme aj elementy, ktoré majú len počiatočnú značku a žiaden obsah.



Obrázok 1.2 Element `title`



Obrázok 1.3 Element `html`

ÚLOHA 1.7

- Aké elementy sme použili v našej webovej stránke (v zdrojovom kóde z *Príkladu 1.5*)?
- Pre každý z nich určte, čo je počiatočná značka príslušného elementu, čo koncová a čo obsah elementu.

Ďalej v texte sa budeme odvolávať na konkrétne elementy iba pomocou ich počiatočnej značky (napr. element `<html>`), resp. len ich názvu (napr. element `html`). Vždy však budeme mať na mysli celý stavebný blok.

POZNÁMKA

Názvy elementov môžeme písať malými aj veľkými písmenami, teda zápisy `<html>`, `<HTML>` aj `<Html>` sú korektné.

Vidíme, že obsahom jedného elementu môžu byť iné elementy: v našom zdrojovom kóde (Obrázok 1.3) element `<html>` obsahuje elementy `<head>` a `<body>`, element `<head>` obsahuje element `<title>`. Hovoríme, že elementy do seba **vnárame**. Neskôr uvidíme, že toto vnáranie elementov nie je ľubovoľné, má svoje pravidlá a obmedzenia.

Základná schéma webového dokumentu

Časti stránky: hlavička, telo, titulok

V predchádzajúcej kapitole sme vytvorili jednoduchú webovú stránku pomocou elementov `<html>`, `<head>`, `<title>`, `<body>`. Tieto elementy tvoria základ webového dokumentu. Ich význam popisujeme v nasledujúcej tabuľke.

Element	Popis
`<html></html>`	začiatok a koniec celého dokumentu
`<head></head>`	hlavička dokumentu (skrytá v prehliadači)
`<title></title>`	titulok dokumentu, zobrazuje sa v titulnom riadku prehliadača
`<body></body>`	telo dokumentu, zobrazuje sa v okne prehliadača

Správne usporiadanie webového dokumentu je nasledovné:

```
<html>
<head>
<title>Toto je titulok</title>
</head>

<body>
  Toto je telo dokumentu. Do tejto časti sa píše väčšina elementov.
  Obsah tejto časti sa zobrazí v prehliadači.
</body>
</html>
```

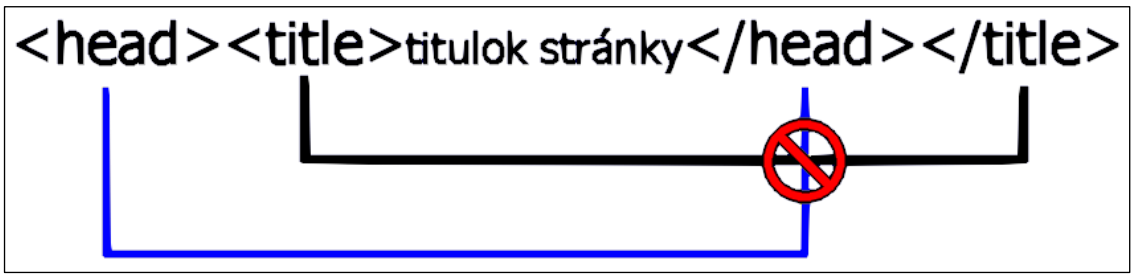
POZNÁMKA

Pri definovaní elementov si musíme dávať pozor na poradie počiatočných a koncových značiek. Platí, že najskôr musíme ukončiť (koncovou značkou) ten element, ktorý sme začali (počiatočnou značkou) ako posledný.

Správne poradie: `<head><title>titulok stránky</title></head>`

Nesprávne poradie: `<head><title>titulok stránky</head></title>`

Ak by sme spojili čiarami začiatočné a koncové značky elementov, tak pri správnom poradí elementov by sa čiary nepreťali.



Obrázok 1.4 Nesprávne poradie elementov.

ÚLOHA 1.8

Vyskúšajte nejaké zlé poradie značiek (napr. v kóde nižšie) a zistite, ako sa stránka zobrazí.

```
<html>
<head>
  <title>IT PIZZA</head>
</title>
<body>
  Naša ponuka, z ktorej si určite vyberiete: Margherita,
Cardinale, Funghi, Hawai, …
</body>
</html>
```

Štandard dokumentu – element <!DOCTYPE>

V úvode sme spomenuli, že stránky budeme vytvárať podľa štandardu **HTML5**. Štandard stránky určujeme pomocou elementu **`<!DOCTYPE>`**, ktorý musí byť uvedený **ešte pred elementom** `<html></html>`. Keďže štandardov na tvorbu webových stránok je niekoľko, musíme v elemente `<!DOCTYPE>` určiť, ktorý konkrétny štandard budeme používať. Použitie štandardu HTML5 definujeme elementom: **`<!DOCTYPE html>`**. Element `<!DOCTYPE>` sa zvyčajne píše s veľkými písmenami.

PRÍKLAD 1.9

Doplňme element `<!DOCTYPE>` so štandardom HTML5 do súboru `index.html`.

```
<!DOCTYPE html>
<html>
<head>
  <title>IT PIZZA</title>
</head>
<body>
  Naša ponuka, z ktorej si určite vyberiete: Margherita,
Cardinale, Funghi, Hawai, ...
</body>
</html>
```

Všimnite si, že element `<!DOCTYPE>` má len počiatočnú značku a žiaden obsah. Elementy, ktoré nemajú žiaden obsah, nazývame **prázdne elementy**.

Kódovanie stránky

Stránka IT Pizza, ktorú sme vytvorili v úvodnej časti, sa v niektorých prehliadačoch zobrazovala nesprávne (pozri Obrázok 1.1) - niektoré slovenské znaky s diakritikou boli nahradené inými znakmi. Aby sme zabezpečili správne zobrazovanie slovenského textu vo webovom prehliadači, musíme v zdrojovom kóde nastaviť tzv. **kódovanie**. Tým vlastne oznámime prehliadaču, akú znakovú sadu má použiť pri zobrazovaní textu. Existuje niekoľko slovenských kódovaní: windows-1250, iso-8859-2, utf-8. My budeme používať univerzálne kódovanie **utf-8**.

Kódovanie nastavujeme pomocou elementu **`<meta>`**, ktorý musí byť umiestnený v hlavičke webového dokumentu, t.j. vnútri elementu `<head>`. Je vhodné uvádzať ho skôr ako ďalšie elementy v hlavičke. Ak chceme používať kódovanie utf-8, použijeme takýto tvar elementu: `<meta charset="utf-8">`.

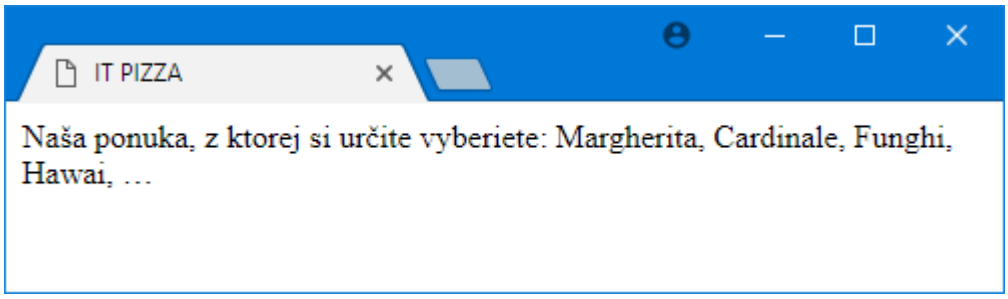
PRÍKLAD 1.10

Pridajme do svojej webovej stránky nastavenie kódovania utf-8.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>IT PIZZA</title>
</head>
```

```
<body>
  Naša ponuka, z ktorej si určite vyberiete: Margherita,
Cardinale, Funghi, Hawai, …
</body>
</html>
```

Element `<meta>` nemá koncovú značku. Na Obrázku 1.5 vidíme správne zobrazenie stránky v prehliadači.



Obrázok 1.5 Stránka s diakritikou.

ÚLOHA 1.11

Vyskúšajte kódovania windows-1250 a iso-8859-2.

ÚLOHA 1.12

Viete si predstaviť vytváranie stránky napr. v gréčtine? Vytvorte stránku so základnou schémou a ako obsah skopírujte text z nejakej gréckej stránky. Otestujte, či kódovanie utf-8 bude postačovať na zobrazenie tohoto textu.

Po doplnení elementov na nastavenie štandardu a kódovania dostávame kompletnú základnú schému webového dokumentu, ktorú môžeme využívať pri tvorbe ďalších stránok.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>titulok</title>
</head>
<body>
   Vlastný obsah stránky
</body>
</html>
```

Pri voľbe elementu pre nadpis by sme sa mali riadiť výhradne logickou hierarchiou nadpisu, nie podľa veľkosti písma. Ak ide o hlavný nadpis, mali by sme použiť element `<h1>`, pre podnadpis element `<h2>`, atď. Veľkosť nadpisu si môžeme prispôsobiť – naučíme sa to neskôr.

Metodické pokyny pre učiteľa

CIEĽ

Cieľom tejto kapitoly je priblížiť tvorbu statických webových stránok editovaním zdrojového kódu stránky. Študenti sa dozvedia o nástrojoch vhodných na editovanie HTML kódu, oboznámia sa so základnou schémou HTML dokumentu, vytvoria prvú webovú stránku a naučia sa jej nastaviť kódovanie.

VÝKLAD

V úvode sú úlohy zamerané na skúmanie zdrojového kódu reálnych webových stránok. Neskôr študenti vytvárajú jednoduchú webovú stránku, ktorej nastavia titulok, štandard HTML5 a správnu znakovú sadu.

Editory na tvorbu webových stránok

- Atom - https://atom.io , nevýhoda – veľmi dlho sa spúšťa
- Notepad++ - https://notepad-plus-plus.org/
- PSPad - http://www.pspad.com/sk/
- BlueFish - http://bluefish.openoffice.nl/index.html
- WebStorm - https://www.jetbrains.com/webstorm/
  Webstorm je komerčný produkt, avšak jeho edukačná verzia je pre študentov vlastniacich ISIC (aj učiteľov s ITIC) zadarmo. Treba ho však sťahovať zo stránky https://www.jetbrains.com/buy/classroom/?fromMenu

K úlohe 1.4

Táto úloha nie je povinná. Treba zvážiť podľa situácie na škole, či ju žiakom zadať.

K príkladu 1.5

Kým na stránke nie je nastavené kódovanie, nemusí sa správne zobrazovať diakritika. Žiakov na to môžeme zatiaľ len upozorniť, opravíme to na konci kapitoly. V učebnici sme zámerne odfotili ukážku zobrazenia stránky v prehliadači Mozilla, kde sú znaky s diakritikou zobrazené zle. Niektoré prehliadače, napr. Chrome, však akoby podľa znakov vedeli rozoznať, aké kódovanie pre stránku treba použiť, a zobrazia znaky s diakritikou správne, aj keď ho nemáme nastavené pomocou elementu `meta`.

K úlohe 1.12

Úloha nie je povinná. Môžeme ju využiť pre rýchlejších alebo zvedavých žiakov.

2 ŠTRUKTÚROVANIE TEXTU NA WEBOVEJ STRÁNKE

Keď vytvárame nejaký textový dokument, zvykneme text v ňom rozčleňovať do odsekov.

PRÍKLAD 2.1

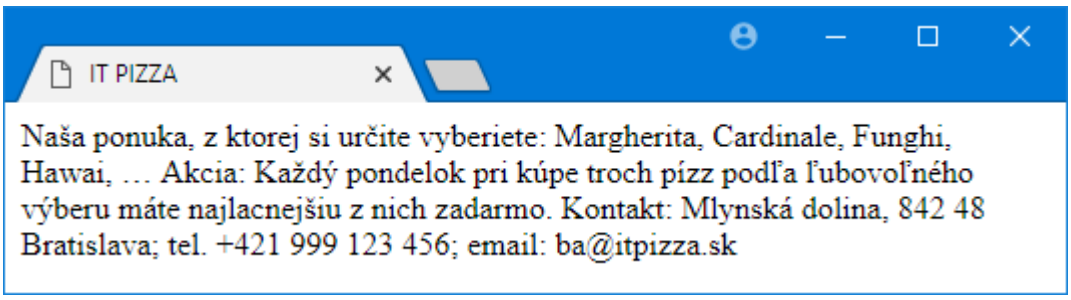
Pokračujme v tvorbe stránky IT Pizza (súbor `02/index.html`). Do elementu `<body>` za informácie o ponuke pizzerie pridáme:

- informácie o aktuálne prebiehajúcich akciách,
- kontaktné informácie.

Ponuku, akcie a kontaktné informácie dáme v zdrojovom kóde na nový riadok (prípadne oddelíme prázdnym riadkom).

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>IT PIZZA</title>
</head>
<body>
  Naša ponuka, z ktorej si určite vyberiete: Margherita,
Cardinale, Funghi, Hawai, …
  Akcia: Každý pondelok pri kúpe troch pízz podľa ľubovoľného
výberu máte najlacnejšiu z nich zadarmo.
  Kontakt: Mlynská dolina, 842 48 Bratislava; tel. +421 999 123
456; email: ba@itpizza.sk
</body>
</html>
```

Po pozretí stránky v prehliadači zistíme, že celý text sa nám zobrazil ako jediný odsek.



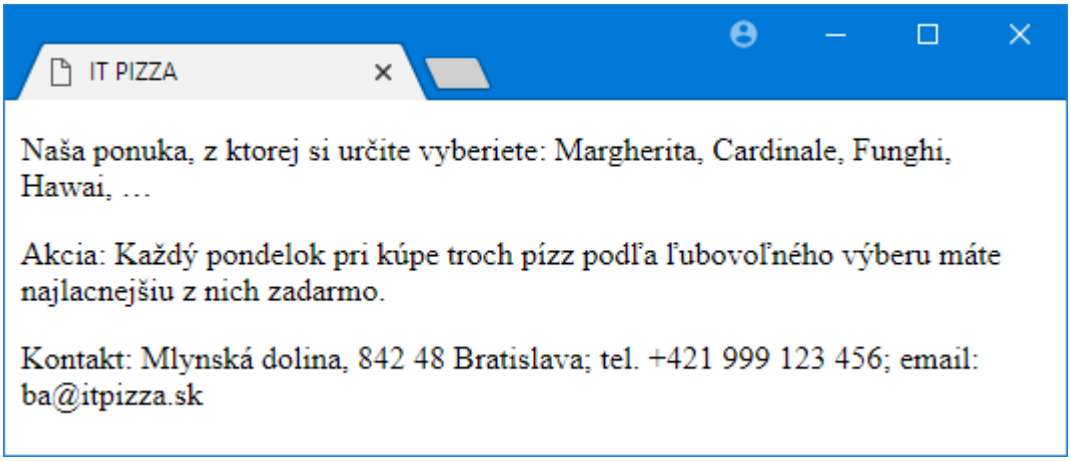
Obrázok 2.1 Stránka v prehliadači.

Na vytvorenie nového odseku nestačí v zdrojovom kóde stránky prejsť na nový riadok, ani vytvoriť medzi odsekmi prázdne riadky. Odsek v jazyku HTML definujeme pomocou elementu `<p></p>`, pričom obsahom tohto elementu je text odseku. Neskôr uvidíme, že obsahom odseku môže byť nielen text, ale napr. aj obrázok.

PRÍKLAD 2.2

Opravíme zdrojový kód stránky IT Pizza použitím elementu `<p>`.

```
...
<body>
  <p>Naša ponuka, z ktorej si určite vyberiete: Margherita,
Cardinale, Funghi, Hawai, …</p>
  <p>Akcia: Každý pondelok pri kúpe troch pízz podľa ľubovoľného
výberu máte najlacnejšiu z nich zadarmo.</p>
  <p>Kontakt: Mlynská dolina, 842 48 Bratislava; tel. +421 999
123 456; email: ba@itpizza.sk</p>
</body>
</html>
```



Obrázok 2.2 Stránka s odsekmi.

ÚLOHA 2.3

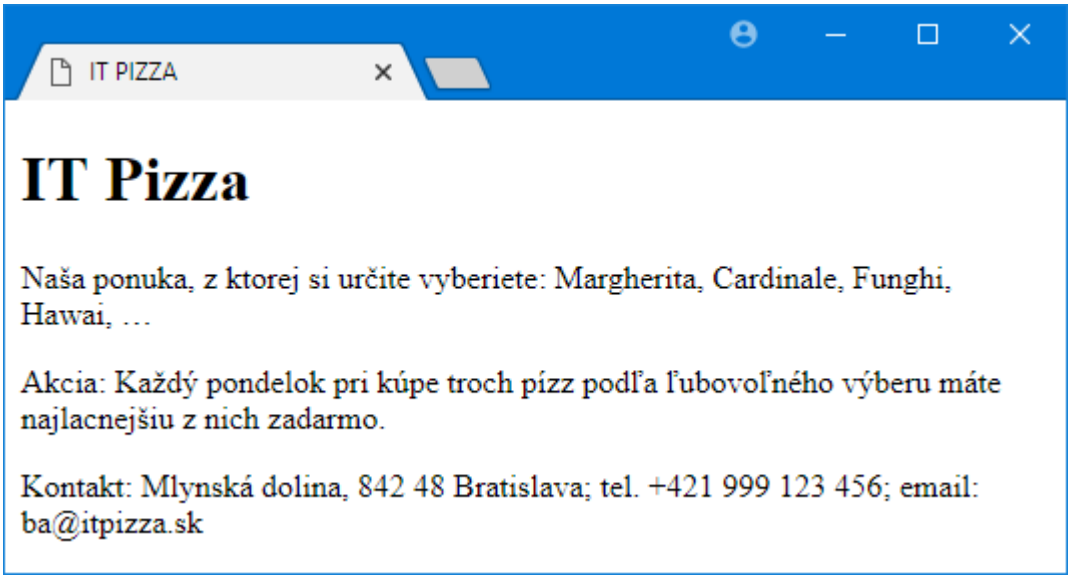
Pred prvý odsek pridajte nový odsek s textom IT Pizza.

PRÍKLAD 2.4

Chceli by sme, aby text IT Pizza nebol len obyčajným odsekom, ale aby to bol nadpis. Na vytvorenie nadpisu použijeme element `<h1></h1>`.

Nahraďme značky `<p>`, `</p>` definujúce prvý odsek stránky značkami `<h1>` a `</h1>`.

```
...
<body>
  <h1>IT Pizza</h1>
  <p>Naša ponuka, z ktorej si určite vyberiete: Margherita,
Cardinale, Funghi, Hawai, …</p>
  <p>Akcia: Každý pondelok pri kúpe troch pízz podľa ľubovoľného
výberu máte najlacnejšiu z nich zadarmo.</p>
  <p>Kontakt: Mlynská dolina, 842 48 Bratislava; tel. +421 999
123 456; email: ba@itpizza.sk</p>
</body>
</html>
```



Obrázok 2.3 Stránka s nadpisom.

Elementy `<p>` a `<h1>` patria medzi tzv. **blokové elementy**. Blokové elementy vždy začínajú na novom riadku a štandardne zaberajú celú šírku stránky, teda nemôžu byť vedľa seba.

Popis najpoužívanejších blokových elementov uvádzame v nasledujúcej tabuľke.

Element	Popis
`<h1></h1>`	Nadpis úrovne 1 – najvyššia
`<h2></h2>`	Nadpis úrovne 2
`...`	
`<h6></h6>`	Nadpis úrovne 6 – najnižšia
`<p></p>`	Odsek (akási základná jednotka textu)
`<address> </address>`	Adresa
`<blockquote> </blockquote>`	Citácia z iného zdroja (zvyčajne blok textu), väčšinou je odsadená od okolitého textu
`<pre></pre>`	Zobrazí text presne tak, ako je v zdrojovom kóde (vrátane medzier a nových riadkov)

ÚLOHA 2.5

Postupne meňte element `<h1>` v zdrojovom kóde stránky IT Pizza na elementy `<h2>`, `<h3>` až `<h6>`. Ako sa jednotlivé nadpisy líšia?

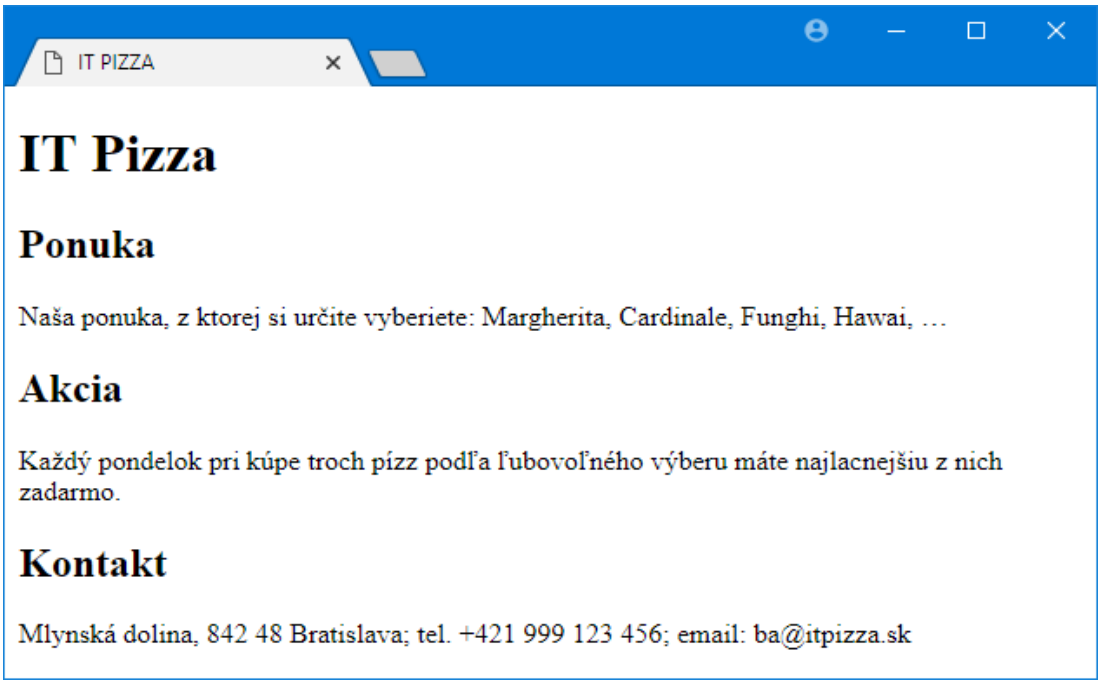
Pri voľbe elementu pre nadpis by sme sa mali riadiť výhradne logickou hierarchiou nadpisu, nie podľa veľkosti písma. Ak ide o hlavný nadpis, mali by sme použiť element `<h1>`, pre podnadpis element `<h2>`, atď. Veľkosť nadpisu si môžeme prispôsobiť - naučíme sa to neskôr.

ÚLOHA 2.6

Text IT Pizza naformátujte ako nadpis úrovne 1. Doplňte podnadpisy Ponuka, Akcie a Kontakt pred zodpovedajúce odseky. Ktorý z nadpisových elementov pre ne zvolíte?

ÚLOHA 2.7

Na odsek s kontaktnými informáciami použite element `address`.



Obrázok 2.4 Stránka s nadpismi.

ÚLOHA 2.8

Na experimentovanie s html kódom môžeme použiť online html editor JSFiddle na stránke https://jsfiddle.net/. Editor je rozdelený na niekoľko častí, nás v tejto chvíli zaujímajú hlavne časti HTML a Result. Do časti **HTML** skopírujte základnú schému stránky (na konci 1. kapitoly). Výsledok uvidíte v časti **Result** po stlačení tlačidla **Run**. Ako obsah elementu `<body>` postupne vložte nasledujúce 3 kódy s rôznymi vnoreniami elementov. Zakaždým najskôr preskúmajte zdrojový kód, pokúste sa odhadnúť jeho výsledok, až potom zobrazte výsledok v okne **Result**. Zhodujú sa vaše predstavy s tým, čo zobrazil prehliadač?

```
<p>v tomto odseku je vnorený <h3>nadpis úrovne 3</h3> text za nadpisom</p>
```

```
<h1>nadpis nadpis <p>odsek odsek odsek odsek</p> nadpis nadpis</h1>
```

```
<h1>jednotka <h2>dvojka</h2> jednotka</h1>
```

POZNÁMKA

Elementy nemôžeme do seba vnárať úplne ľubovoľne. Nemali by sme napríklad vnútri elementu `<p>` používať nadpisy, alebo v rámci nadpisu nejakej úrovne definovať nadpis inej úrovne. Môžeme však časti textu v rámci odseku či nadpisu zvýrazniť nejakým iným spôsobom, použitím tzv. *riadkových elementov*.

Riadkové elementy

Riadkové elementy nezačínajú na novom riadku a zaberajú len nevyhnutnú šírku. Používame ich zväčša na nejaký druh zvýraznenia menšej časti textu (slovo, veta) v rámci blokového elementu.

ÚLOHA 2.9

V online editore JSFiddle vyskúšajte nasledujúce kódy:

```
<p><strong>Adrenalín</strong> je <em>hormón</em>, ktorý sa tvorí v dreni <em>nadobličiek</em>.</p>

<p>Stojí, stojí mohyla,<br>na mohyle zlá chvíľa.<br>Na mohyle tŕnie chrastie ...</p>

<p>a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup></p>

<p>for cyklus v Pyhtone: <code>for i in range(10):</code></p>
```

Vedeli by ste na základe vašich skúseností povedať, na čo slúžia elementy `<em>`, `<strong>`, `<sup>`, `<br>` a `<code>`?

V nasledujúcej tabuľke uvádzame zoznam a popis niektorých riadkových elementov. Kompletný zoznam elementov na formátovanie môžeme nájsť v technickej špecifikácii štandardu HTML5 alebo na stránke w3schools.com (http://www.w3schools.com/tags/default.asp).

Element	Popis
`<em></em>`	Zvýraznené písmo (zvyčajne šikmé)
`<strong></strong>`	Veľmi zvýraznené písmo (zvyčajne tučné)
` `	Nový riadok v rámci nejakého sekcie/odseku/časti textu
`<sub></sub>`	Dolný index
`<sup></sup>`	Horný index
`<cite></cite>`	Citácia (napr. názov práce, nie osoby)
`<code></code>`	Zdrojový kód
`<samp></samp>`	Ukážka, príklad

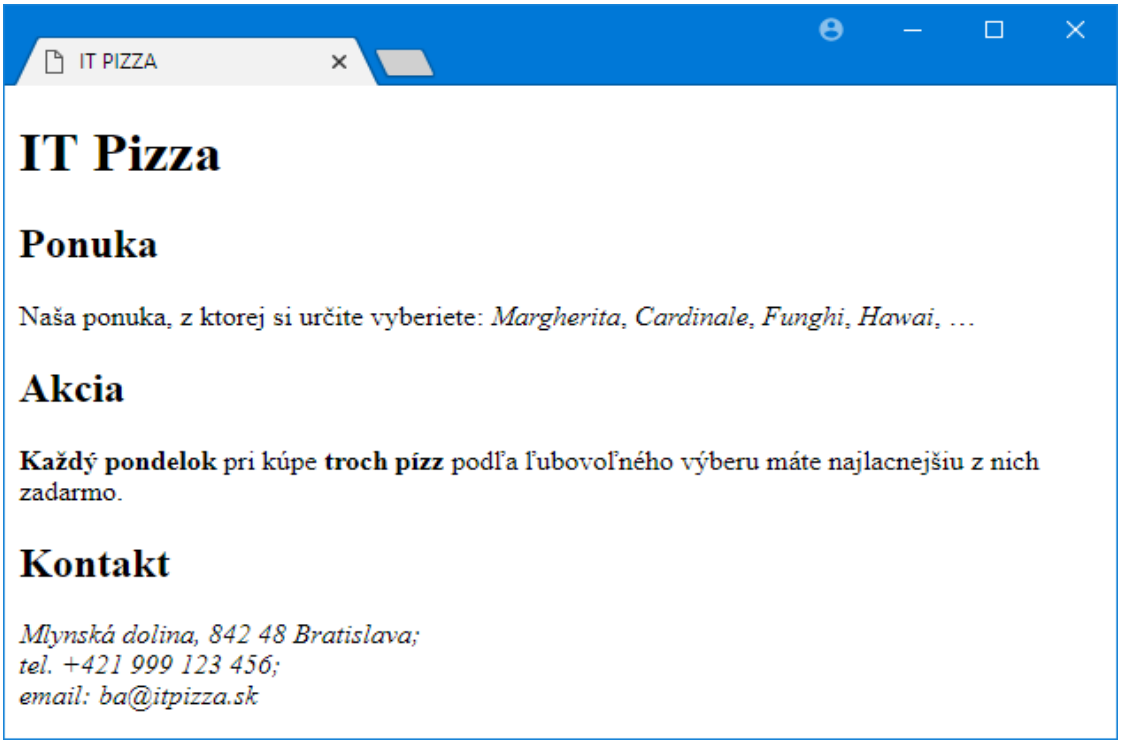
Element `<br>` nemá vplyv na štruktúru stránky, len vloží nový riadok v aktuálnom odseku/sekcii/bloku. Jeho použitie by sme mali veľmi zvážiť a nenahradzovať ním napr. odsek (element `<p></p>`).

ÚLOHA 2.10

Upravte stránku IT Pizza (jej zdrojový kód) tak, aby:

- všetky názvy pízz boli zvýraznené šikmým písmom,
- údaje v kontaktoch: adresa, telefón a email sa zobrazovali na novom riadku, ale nie ako nové odseky,
- v odseku za nadpisom *Akcia* boli tučným písmom zvýraznené spojenia „každý pondelok“ a „troch pízz“.

Pomocou ktorého riadkového elementu by ste zvýraznili telefónne číslo?



Obrázok 2.5 Stránka s vyžitím riadkových elementov.

ÚLOHA 2.11

Pohľadajte na internete ďalšie riadkové elementy a niektoré vyskúšajte.

Metodické pokyny pre učiteľa

CIEĽ

Cieľom je, aby sa študenti zoznámili so základnými HTML elementami umožňujúcimi štruktúrovať text na webovej stránke (blokové elementy) a elementami, ktoré umožňujú nejakým spôsobom zvýrazňovať/odlíšiť časti textu (riadkové elementy).

VÝKLAD

Dôležité je, aby študenti experimentovali s blokovými a riadkovými elementami a pochopili, aký je ich význam a tiež aký je medzi nimi rozdiel. Vhodnou pomôckou je online HTML editor JSFiddle.

Študenti možno poznajú riadkové elementy `<b>` a `</b>`, prípadne `<i>` a `</i>`, či `<u>` a `</u>`. Treba ich upozorniť, aby tieto značky nepoužívali, pretože nie sú v súlade so štandardom HTML5.

Nadväznosť na predchádzajúce kapitoly

Kapitolu **je možné prebrať až po predchádzajúcej kapitole**.

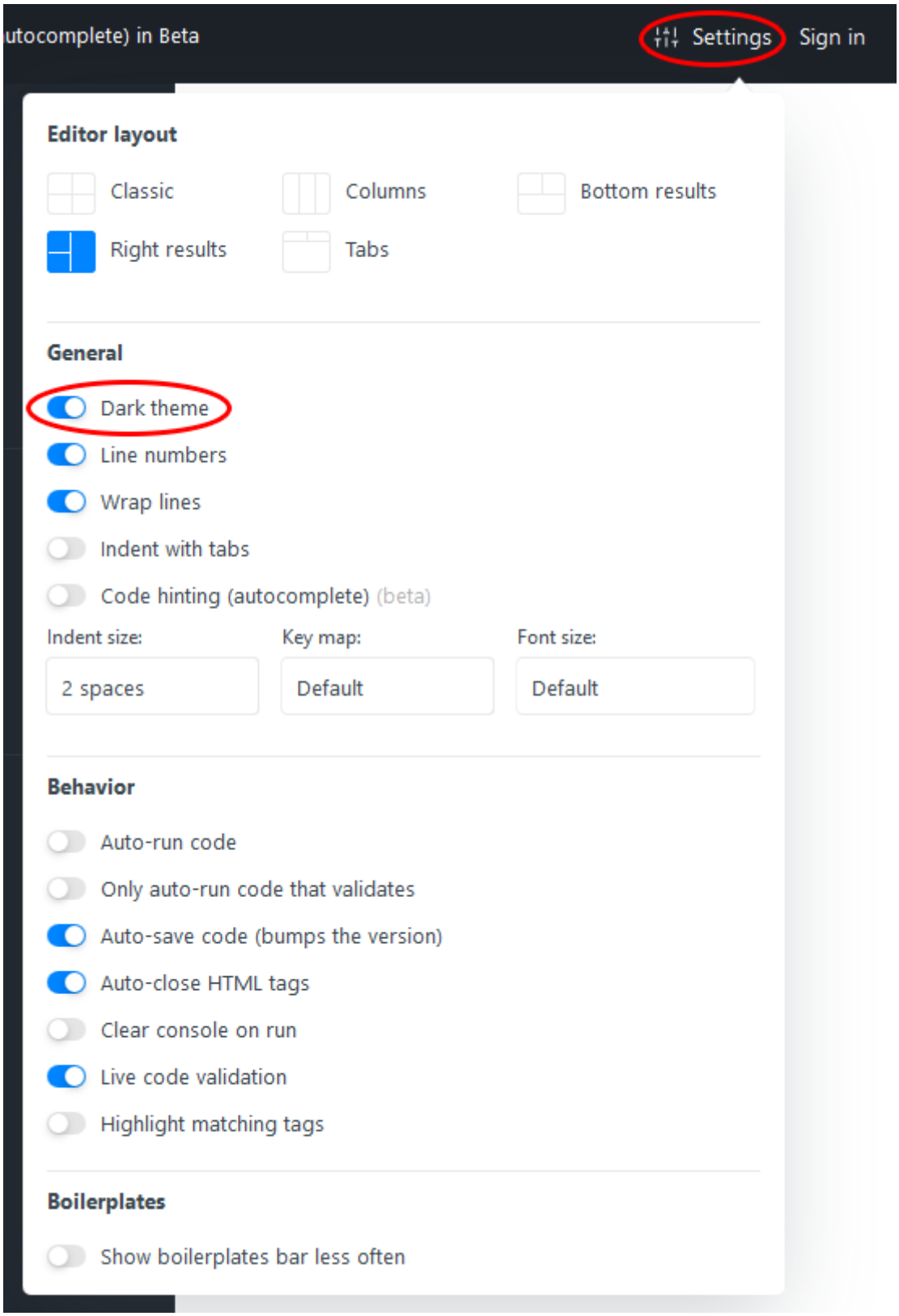
K príkladu 2.1

Žiaci môžu buď pokračovať vo vlastnom súbore z predchádzajúcej kapitoly, alebo použiť súbor `02/index.html`, v ktorom je stránka IT Pizza v takom stave, v akom bola vytvorená v kapitole 1.

K editoru JSFiddle (príklade 2.8)

Editor **JSFiddle je vhodný na experimentovanie** s HTML kódom aj CSS. Výhodou je, že vidíme všetky časti kódu (HTML aj CSS) aj výslednú stránku v jednom okne. Tiež, ak si chceme len rýchlo niečo vyskúšať, nemusíme napísať celý kód stránky (elementy `html`, `head`, `body`...), stačí len obsah elementu `body`. Editor zvýrazňuje syntax, čiastočne pomáha pri písaní kódu a tiež pomôže odhaliť niektoré jednoduché chyby, napr. ak použijeme inú koncovú značku elementu ako počiatočnú.

Editor sa pri prvom načítaní zobrazí s čiernym pozadím, to je však možné zmeniť v časti Settings.


utocomplete) in Beta
Settings
Sign in
Editor layout
Classic
Columns
Bottom results
Right results
Tabs
General
Dark theme
Line numbers
Wrap lines
Indent with tabs
Code hinting (autocomplete) (beta)
Indent size:
2 spaces
Key map:
Default
Font size:
Default
Behavior
Auto-run code
Only auto-run code that validates
Auto-save code (bumps the version)
Auto-close HTML tags
Clear console on run
Live code validation
Highlight matching tags
Boilerplates
Show boilerplates bar less often

3 ÚVOD DO KASKÁDOVÝCH ŠTÝLOV

V predchádzajúcich častiach sme vytvorili základnú štruktúru stránky IT Pizza, ale bez zmeny vzhľadu. Aktuálnym trendom tvorby webových stránok je vytváranie vzhľadu len pomocou tzv. kaskádových štýlov. **Použitie kaskádových štýlov nám umožňuje oddeliť štruktúru stránky od jej vzhľadu, meniť vzhľad stránky bez zásahu do jej obsahu a štruktúry.** Kaskádové štýly, skrátene len štýly (CSS – Cascading Style Sheets) popisujú spôsob zobrazenia HTML elementov na obrazovke počítača či v inom médiu (napr. tablet, smartfón, tlačiareň).

V tejto kapitole si ukážeme, ako definovať kaskádové štýly a ako ich vložiť do webovej stránky. Pomocou kaskádových štýlov nastavíme farby pozadia a textu a veľkosť písma. Budeme vychádzať zo súboru `index.html` v priečinku `03`.

ÚLOHA 3.1

Otvorte si online html editor na stránke https://jsfiddle.net/. Do okna HTML skopírujte obsah elementu `<body>` zo súboru `03/index.html`. Do okna CSS vložte nasledujúci kód a pozorujte, čo sa udeje:

```
h1 {
  color: orange;
  background-color: black;
}
```

Nezabudnite kliknúť na tlačidlo ▷ Run, ktorým „spustíte“ kód stránky.

ÚLOHA 3.2

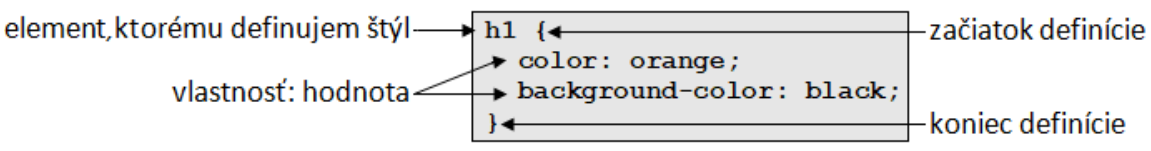
V kóde z predchádzajúcej úlohy skúšajte meniť názvy farieb (zadávajte anglické názvy farieb, napr. `red`, `yellow`...) a zistite, či ich prehliadač/editor použije.

ÚLOHA 3.3

V kóde z úlohy 3.1 nahraďte `h1` niektorým z `body`, `h2`, `p`, `em`, `strong` a pozorujte, čo sa udeje po „spustení“ kódu. Ktoré časti stránky sa zmenia?

Vytvorenie kaskádových štýlov

V predchádzajúcich úlohách sme vytvorili definície kaskádových štýlov. Zjednodušene by sme ich mohli vysvetliť nasledovne:



Obrázok 3.1 Definovanie kaskádových štýlov

Uvedený kód môžeme čítať ako „pre element `h1` nastav farbu textu na `orange` a farbu pozadia na `black`".

Odborne hovoríme, že vlastnostiam `color` a `background-color` sme nastavili hodnotu (farbu). Medzi vlastnosťou a hodnotou musí byť dvojbodka `(:)` a táto dvojica musí byť ukončená bodkočiarkou `(;)`. Všetky nastavenia vlastností musíme uzatvoriť do zátvoriek `{ }` (aj keď budeme definovať len jednu vlastnosť štýlu).

PRÍKLAD 3.4

Urobme kópiu štýlov z úlohy 3.1 a vložme ju znovu pod definíciu `h1`. Zmeňme druhé `h1` na `h2` a vymeňme farby textu a pozadia, aby boli nadpisy rôzne.

```
h1 {
  color: orange;
  background-color: black;
}
h2 {
  color: black;
  background-color: orange;
}
```

Čo sa udialo so stránkou? Ktoré nadpisy sa zmenili?

ÚLOHA 3.5

- Pre nadpisy `h2` zmeňte farbu pozadia na inú.
- Zrušte vlastnosť `color` pre `h2` (vymažte riadok). Ako sa zmení stránka?

Definície kaskádových štýlov, ktoré sme vytvorili, sa použijú pre každý element `h1` a každý element `h2`, ktoré máme na stránke.

Vloženie kaskádových štýlov do stránky

Poďme teraz upraviť stránku IT Pizza (nie v systéme JSFiddle) a pridať do nej kaskádové štýly.

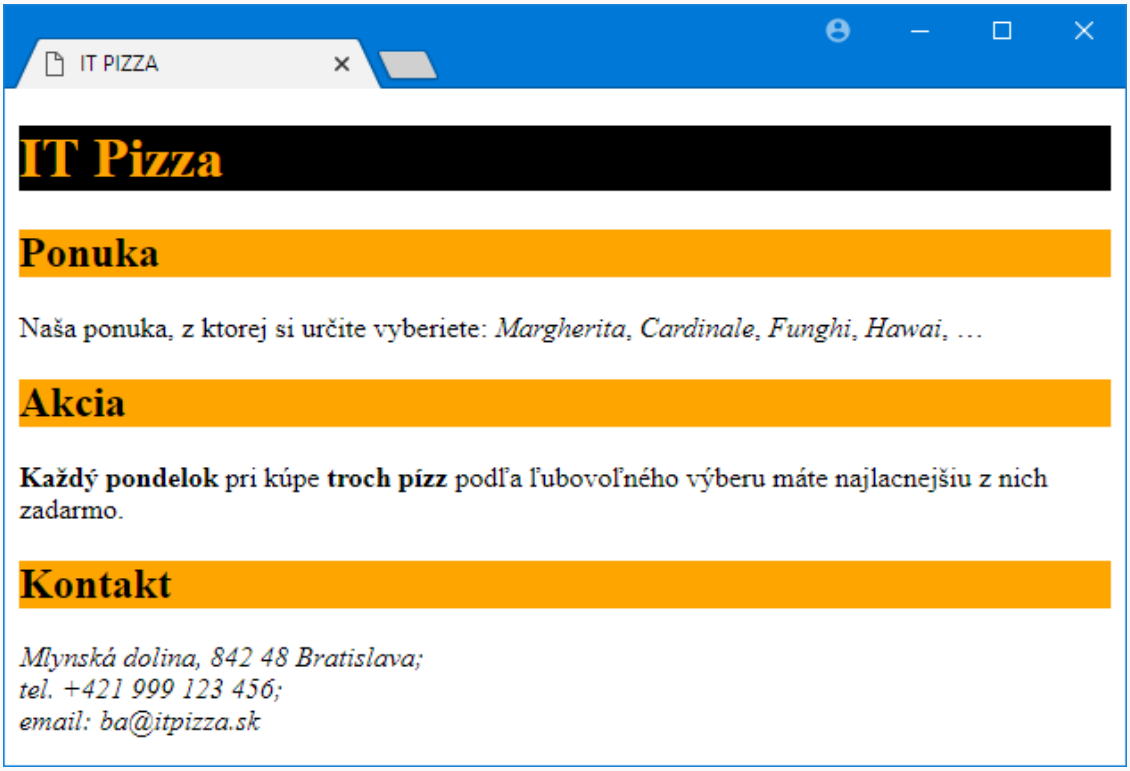
Ak chceme do existujúcej stránky vložiť definície kaskádových štýlov, môžeme využiť element `<style></style>`, ktorý sa vkladá do hlavičky webovej stránky (element `<head>`).

```
<head>
  <meta charset="utf-8">
  <title>IT PIZZA</title>
  <style>
  </style>
</head>
```

PRÍKLAD 3.6

Otvorme súbor `03/index.html`. V ňom do elementu `<head>` vložíme prázdny element `<style>`. Do neho skopírujeme kaskádové štýly z *príkladu 3.4*. Súbor uložíme a pozrieme si webovú stránku v prehliadači.

```
<head>
  <style>
    h1 {
      color: orange;
      background-color: black;
    }
    h2 {
      color: black;
      background-color: orange;
    }
  </style>
</head>
```



ÚLOHA 3.7

Vyskúšajte rôzne zmeny štýlov v súbore `index.html`, ktoré ste skúšali v editore JSFiddle, a overte, či sa správajú rovnako.

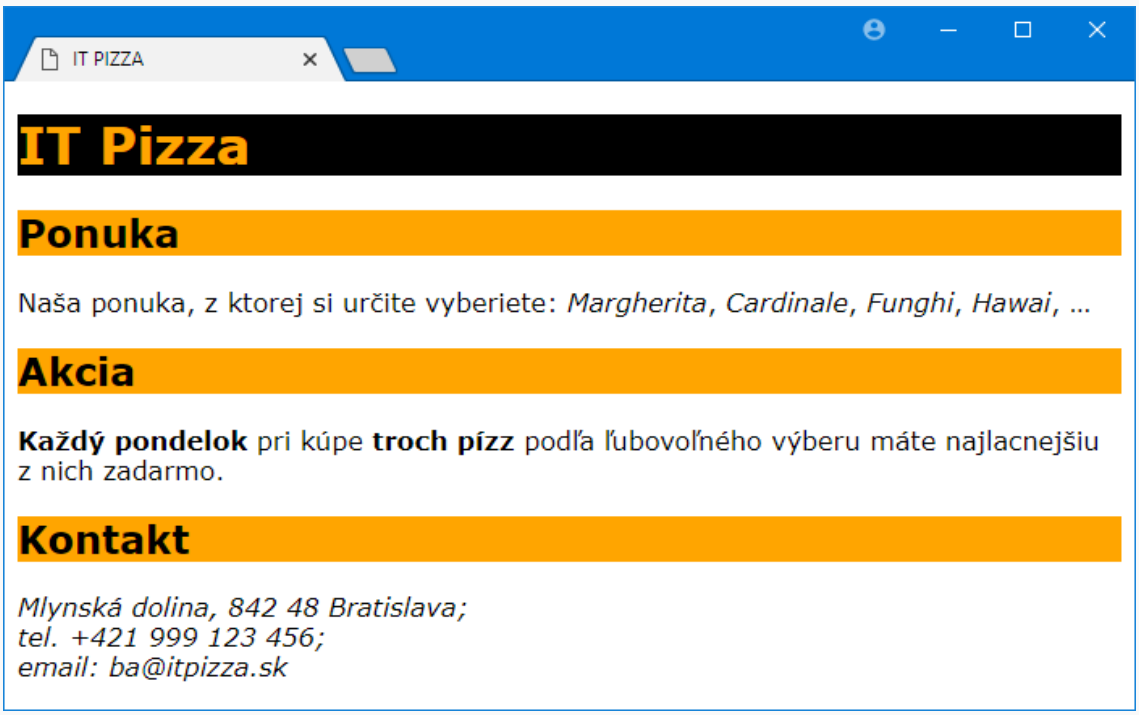
Zmena písma na stránke

Okrem farieb môžeme zmeniť aj typ písma rôznym elementom. Doteraz sme nikde písmo nenastavovali, takže prehliadač použil vlastné základné nastavenie.

PRÍKLAD 3.8

Nastavme typ písma na celej stránke na *Verdana*. Ku kaskádovým štýlom na stránke IT Pizza doplňme nasledujúci štýl pre element `body`.

```
<style>
  ...
  h2 {
    color: black;
    background-color: orange;
  }
  body {
    font-family: Verdana;
  }
</style>
```



ÚLOHA 3.9

Zmeňte typ písma:

- nadpisov `h2` na `Arial`,
- odsekov `p` alebo zvýraznených textov `(em)` na `"Times New Roman"`,
- elementov `address` na `"Courier New"`.

Viacslovné názvy písma musia byť v úvodzovkách alebo v apostrofoch.

POZNÁMKA

Ak definujeme typ písma, je vhodné uvádzať nie jeden typ písma, ale viacero typov pre prípad, ak by prvé písmo v poradí nebolo dostupné na danom počítači, kde sa stránka zobrazuje. Ďalšie písma oddeľujeme čiarkami, napr. `font-family: Arial, Verdana, sans-serif`.

ÚLOHA 3.10 (OPAKOVANIE)

Nájdite na internete text básne Ranená breza alebo Môj macík. Vytvorte stránku s touto básňou tak, aby každý verš bol v samostatnom riadku a každá sloha bola samostatným odsekom. Na názov básne a meno autorky využite vhodné nadpisové elementy. Pomocu kaskádových štýlov definujte vzhľad stránky – snažte sa využiť všetko, čo ste sa doteraz naučili – zmenu typu písma, farby pozadia aj písma.

Metodické pokyny pre učiteľa

CIEĽ

Cieľom je získať prvé skúsenosti s definovaním kaskádových štýlov.

MOTIVÁCIA

Myslíme si, že žiaci už od začiatku tvorby stránky budú mať záujem tvoriť nielen obsah, ale aj dizajn stránky. Keďže dnešným trendom je definovať dizajn pomocou CSS a nie elementov v rámci HTML, zaradili sme prvé stretnutie s CSS už ako 3. kapitolu.

Nadväznosť na predchádzajúce kapitoly

Kapitolu **je možné preberať až po kapitolách 1 a 2**.

K definovaniu štýlov

Sú ďalšie dva spôsoby vloženia štýlov k stránke:

- ukladanie štýlov do externých súborov a ich pripojenie pomocou elementu link
- definovanie štýlu priamo v definícii elementu pomocou atribútu style.

V učebnici uvádzame len jeden spôsob vkladania štýlov. Keďže je to len úvod do CSS, nemali by sme žiakov „zahltiť“ encyklopedickým vymenovaním všetkých spôsobov, ako vkladať štýly.

Možnosť definovania štýlov pomocou elementu <style></style> v rámci elementu head sme zvolili preto, lebo:

a) kým majú žiaci relatívne malú stránku, je výhodnejšie mať všetko spolu v jednom dokumente (preto nie externé)
b) považujeme za vhodnejšie mať všetky vlastnosti definované pokope na jednom mieste, a nie rozhádzané po jednotlivých elementoch, kde by sa často opakovali rovnaké definície vlastností (preto nie atribút style v elementoch).

K definovaniu vlasností písma

Ak sa študenti v minulosti stretli s nastavovaním vlastností písma pomocou atribútov HTML elementov, treba ich upozorniť, aby to nerobili, pretože to nie je v súlade so štandardom HTML5.

Očakávané problémy

zabudnutie pravej zátvorky } , zabudnutie dvojbodky, preklepy v názvoch vlastností

4 OBRÁZKY

Bežnou súčasťou webových stránok sú obrázky. V tejto kapitole sa naučíme vkladať obrázky do webového dokumentu, oboznámime sa s ich dôležitými nastaveniami, ukážeme si, ako môžeme do webovej stránky umiestňovať rôzne ikonky a symboly.

Vkladanie obrázka do webového dokumentu

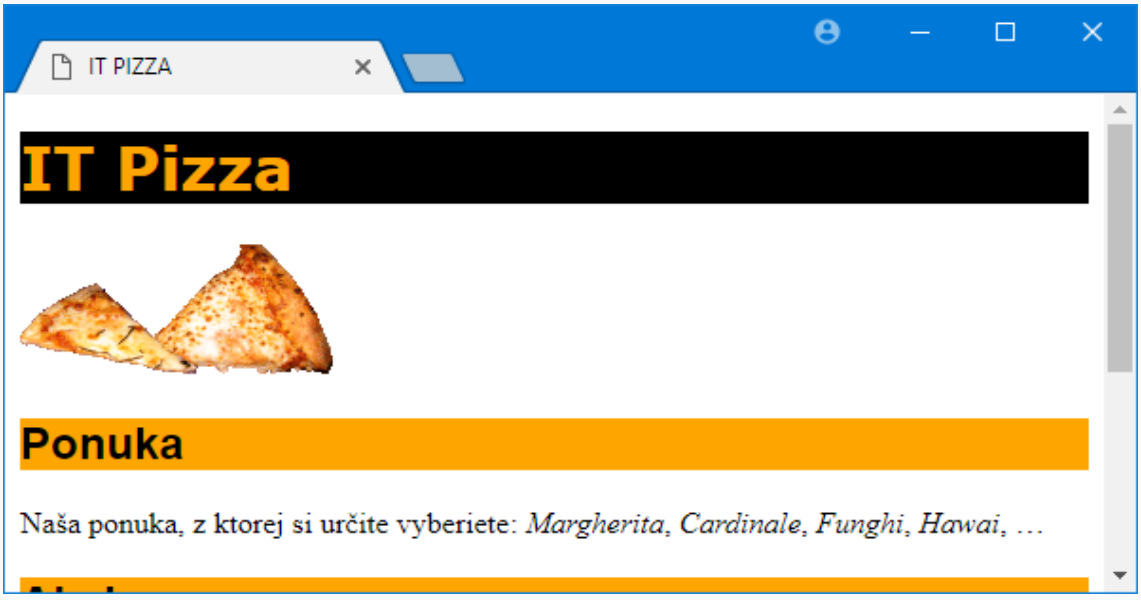
Na webovú stránku našej virtuálnej pizzerie pridáme jej logo a obrázky jednotlivých pízz. Musíme si uvedomiť, že obrázok v počítači je vlastne súbor, v ktorom je obrázok nejakým spôsobom zakódovaný. Do HTML kódu nevkladáme priamo obsah tohto súboru, ale taký element, ktorým oznámime prehliadaču, aby nám na dané miesto vložil obrázok z daného súboru.

PRÍKLAD 4.1

04/logo.gif

Na stránku IT Pizza (súbor `04/index.html`) vložíme logo pizzerie – obrázok zo súboru `logo.gif` v priečinku `04`. Do zdrojového kódu stránky IT Pizza, pod element `<h1>`, pridajme nasledujúci kód: `<img src="logo.gif" alt="logo">`

```
<body>
  <h1>IT Pizza</h1>
  <img src="logo.gif" alt="logo">
  <h2>Ponuka</h2>
  <p>Naša ponuka, z ktorej si určite vyberiete:
    <em>Margherita</em>, <em>Cardinale</em>,
    <em>Funghi</em>, <em>Hawai</em></p>
...
</body>
```



Obrázok 4.1 Stránka IT Pizza s obrázkom.

ÚLOHA 4.2

- Zmeňte v elemente `<img>` názov obrázkového súboru `logo.gif` na `logo2.png`. Čo sa stalo?
- Zmeňte v elemente `<img>` názov súboru `logo2.png` na `logo3.png`. Čo sa stalo?
- Zrušte v elemente `<img>` nastavenie `alt="logo"`. Aký efekt to malo na stránke?
- Opravte element `<img>` na pôvodnú verziu z príkladu 4.1.
- Povedzte vlastnými slovami, aký význam majú nastavenia `src` a `alt`?

V HTML jazyku definujeme obrázky pomocou elementu `<img>`. Element `<img>` je prázdny, ale ako ste si mohli všimnúť v príklade, súčasťou značky `<img>` je definovanie určitých vlastností elementu:

- `src="logo.gif"` – týmto prehliadaču povieme, že má na stránke zobraziť obrázok zo súboru `logo.gif`,
- `alt="logo"` – týmto nastavením definujeme alternatívnu textovú informáciu k obrázku.

ÚLOHA 4.3

- V priečinku `04` nájdete obrázky `margerita.jpg` a `cardinale.jpg`. Upravte zdrojový kód stránky IT Pizza tak, aby sa tieto obrázky zobrazili na stránke v odseku, kde je vymenovaná ponuka pízz (t.j. v rámci už existujúceho elementu `<p>`). Nezabudnite nastaviť alternatívne texty pre oba obrázky.
- Zobrazujú sa vaše dva obrázky vedľa seba alebo pod sebou? Viete toto zobrazenie ovplyvniť?

Vlastnostiam elementov, ako napr. `src` či `alt`, hovoríme **atribúty**. S atribútmi sme sa stretli už v *1. kapitole*: v elemente `<meta>` sme pomocou atribútu `charset` nastavovali kódovanie stránky.

Väčšina atribútov musí mať nastavenú nejakú **hodnotu**[1]. Napr. v zdrojovom kóde stránky IT Pizza sme:

- v prvom elemente `<img>` nastavili hodnotu atribútu `src` na `logo.gif` a hodnotu atribútu `alt` na `logo`,
- v elemente `<meta>` sme nastavili hodnotu atribútu `charset` na `utf-8`.

Vo všeobecnosti atribúty, ktoré majú hodnotu, definujeme v tvare:
`nazov_atribútu="hodnota_atribútu"`
Atribúty spolu s ich hodnotami vždy definujeme **v otváracej značke elementu**.

[1] Sú aj také atribúty, ktoré hodnotu mať nemusia, nazývame ich booleovské. Stretneme sa s nimi v kapitole 12.

ÚLOHA 4.4

V zdrojovom kóde stránky IT Pizza zmeňte v prvom elemente `<img>` poradie atribútov na: `<img alt="logo" src="logo.gif">`. Prejavilo sa to nejako na výslednej stránke?

Atribúty elementu môžeme uvádzať v ľubovoľnom poradí, medzi nimi však musíme dávať medzeru. **Hodnoty atribútov musíme uvádzať v úvodzovkách alebo apostrofoch**.

Atribúty elementu img

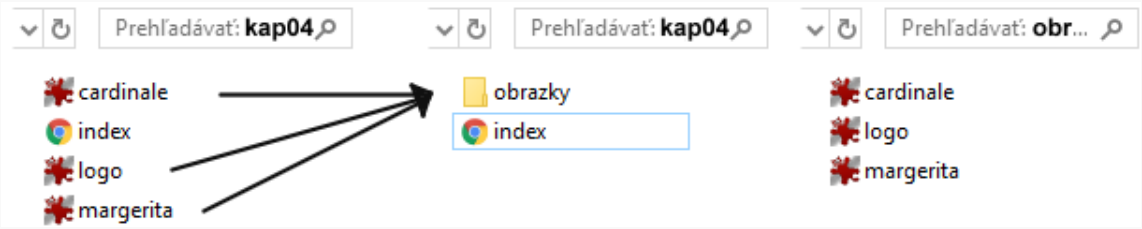
Atribút src

Povinným atribútom elementu `<img>` je atribút `src`. Hodnotou atribútu je tzv. cesta k obrázku, napr. `"logo.gif"`, alebo `"obrazky/logo.gif"`, ale napr. aj `"http://oi.sk/oi/oi-logo.gif"`. Cesta k obrázku definuje jeho umiestnenie na počítači (niečo ako adresa, t.j. v ktorom priečinku, podpriečinku, atď. sa obrázok nachádza). Cesta môže byť daná absolútne alebo relatívne. Podrobnejšie informácie o relatívnom a absolútnom adresovaní sú uvedené v 6. kapitole v časti *Relatívne a absolútne adresovanie*.

PRÍKLAD 4.5

Obrázky, ktoré patria k stránke, sa zvyknú ukladať do samostatného priečinka, resp. priečinkov.

Vytvorme v priečinku so súborom `index.html` nový priečinok `obrazky` a presťahujme doň všetky doteraz použité obrázkové súbory (logo, obrázky jednotlivých pízz).



Obrázok 4.2 Vľavo pôvodný obsah priečinka kap04, vpravo jeho obsah po presunutí obrázkových súborov do podpriečinka obrazky.

Ak teraz zobrazíme stránku v prehliadači, neuvidíme žiaden obrázok, pretože v zdrojovom kóde máme nastavené zlé cesty k obrázkovým súborom.

Vo všetkých `<img>` elementoch opravíme hodnotu atribútu `src` tak, aby obsahovala správnu relatívnu cestu k príslušnému obrázku, vrátane názvu priečinka, v ktorom sa obrázky nachádzajú.

```
<body>
  <h1>IT Pizza</h1>
  <img src="obrazky/logo.gif" alt="logo">
  <h2>Ponuka</h2>
  <p>Naša ponuka, z ktorej si určite vyberiete:
    <em>Margherita</em>, <em>Cardinale</em>,
    <em>Funghi</em>, <em>Hawai</em>, …<br>
    <img src="obrazky/margerita.jpg" alt="pizza margherita">
    <img src="obrazky/cardinale.jpg" alt="pizza cardinale">
  </p> ...
```

Atribút alt

Ďalším **povinným atribútom** elementu `<img>` je `alt`. Tento atribút obsahuje alternatívnu textovú informáciu, ktorá sa zobrazí vtedy, ak sa nezobrazí obrázok. Je dôležitá napr. pre hendikepovaných, alebo pre zariadenia, ktoré obrázky nezobrazujú (vyhľadávače, čítače obrazoviek pre nevidiacich, atď.). Neslúži na zobrazovanie informácie v bublinovej nápovede. K tomu slúži iný atribút (pozri ďalej).

ÚLOHA 4.6

Skontrolujte, či všetky obrázky na stránke IT Pizza majú nastavený atribút `alt`. Ak nie, doplňte ho.

Atribúty width a height

ÚLOHA 4.7

V editore JSFiddle (https://jsfiddle.net/) vložte do HTML časti nasledujúci zdrojový kód a pozrite si výsledok.

```
<body>
  <img src="http://oi.sk/oi/oi-logo.gif" alt="inf. olympiáda"
     width="300" height="206">
</body>
```

- V elemente `<img>` zmeňte hodnotu atribútu `width` z 300 na 150 a hodnotu atribútu `height` z 206 na 103. Ako sa zmenil výstup?
- Experimentujte s hodnotami atribútov `width` a `height`.

Atribútmi `width` a `height` určujeme, v akom veľkom priestore sa má obrázok zobraziť. Hodnoty oboch atribútov sú v obrazových bodoch (pixeloch). Uvedením rozmerov obrázka uľahčíme jeho zobrazenie vo webovom prehliadači. Ten si môže vyhradiť priestor na stránke ešte pred načítaním obrázka zo servera.

ÚLOHA 4.8

Zistite rozmery obrázkov pízz, ktoré ste použili na stránke IT Pizza a doplňte atribúty `width` a `height` do všetkých `<img>` elementov.

ÚLOHA 4.9

Nájdite na internete obrázky pízz Hawai a Funghi, uložte ich do priečinka `obrazky` a pridajte na stránku IT Pizza. Obom obrázkom (elementom `<img>`) nastavte nielen povinné atribúty `src` a `alt`, ale aj atribúty `width` a `height`.

POZNÁMKA

Napriek tomu, že atribúty `width` a `height` nám umožňujú nastaviť, v akej veľkosti sa obrázok zobrazí na stránke, nemali by sme tieto parametre zneužívať na to, aby sme obrázky na stránke zmenšovali (ako to robia niektorí začiatočníci napr. pri tvorbe fotogalérií). Ak totiž vložíme na stránku veľký obrázok (napr. fotku z digitálneho fotoaparátu) a len atribútmi `width` a `height` ho „zmenšíme", v skutočnosti sa musí stiahnuť celý súbor (veľká fotografia) a až na stránke sa akože zmenší.

V prípade, že máme na stránke veľa takýchto "umelo zmenšených" obrázkov, môže to významne zväčšiť čas načítania stránky. Odporúčame najskôr upraviť obrázky do takej veľkosti, v akej ich chceme na stránke použiť a v atribútoch `width` a `height` uvádzať ich skutočnú veľkosť. V prípade tvorby fotogalérií si vytvoríme malé obrázky (náhľady), z ktorých spravíme odkazy na veľké fotografie.

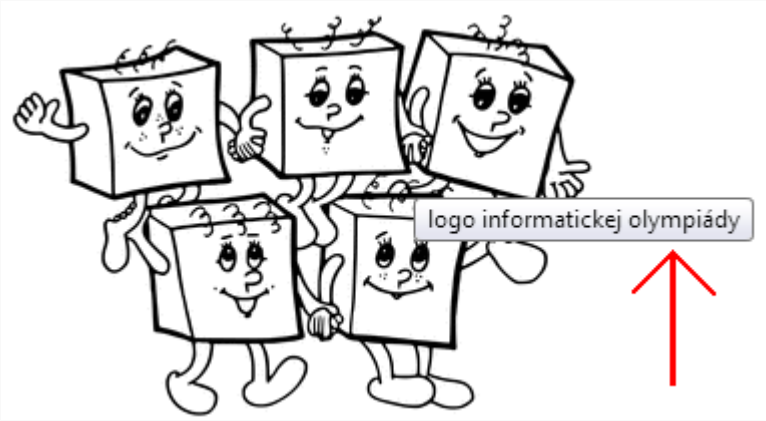
Atribút title

PRÍKLAD 4.10

V editore JSFiddle doplníme element `<img>` o nový atribút `title` nasledovne:

```
<img src="http://oi.sk/oi/oi-logo.gif" alt="inf. olympiáda" width="300" height="206" title="logo informatickej olympiády">
```

Zobrazte výsledok a nadíďte myšou nad obrázok. Čo sa zobrazilo?



Atribút `title` používame, ak chceme k obrázku pridať ďalšiu doplňujúcu informáciu, ktorá by sa používateľovi zobrazila v bublinovej nápovede.

ÚLOHA 4.11

Zmeňte hodnotu atribútu `title` v JSFiddle a znova vyskúšajte.

ÚLOHA 4.12

Všetkým obrázkom na stránke IT Pizza pridajte atribút `title` s vhodnou hodnotou. Pozrite si stránku v rôznych prehliadačoch a zistite, či jednotlivé prehliadače interpretujú atribút `title` rovnako.

Formáty obrázkov, autorské práva

Na stránkach nemôžeme použiť obrázky ľubovoľných formátov, ale len také, s ktorými vedia pracovať webové prehliadače. Oficiálne povolené formáty sú GIF, JPG, PNG. Iné formáty by sme nemali používať, aj keď ich môžu niektoré prehliadače zobrazovať (napr. prehliadač Microsoft Edge/Internet Explorer dokáže zobraziť formát BMP, čo je však absolútne nevhodný formát pre webové stránky – súbory tohto typu sú príliš veľké).

Pri vkladaní obrázkov nesmieme zabudnúť na dodržiavanie autorských práv. Používame buď vlastné obrázky alebo obrázky s voľnou licenciou. Na internete nájdeme množstvo stránok ponúkajúcich obrázky na voľné stiahnutie, napr. pixabay.com, freeimages.com, clipart-library.com a ďalšie. Dodržiavanie autorských práv sa samozrejme týka nielen obrázkov, ale akéhokoľvek obsahu, ktorý na stránku dávame (texty, videá, a pod.).

Ikony a symboly v HTML

Pri komunikácii na sociálnych sieťach dnes často používame emotikony/smajlíky, či iné obrázkové symboly, napr. ikonky zvieratiek, kvetov, športov, jedál, atď. Niektoré z nich majú v HTML jazyku svoje špeciálne kódy. Ak ich chceme vložiť do stránky, nemusíme teda použiť element `<img>`, ale zapíšeme ich ako obyčajný text.

PRÍKLAD 4.13

V online editore JSFiddle vytvoríme takýto odsek.

```
<body>
  <p>Dnes o &#128354; som bol vyvenčiť svojho &#128054;.<br>
  Vonku svietilo &#127774;. Kúpil som si &#127789; a
&#127861;.</p>
</body>
```

Výsledok vidíme na obrázku:

ÚLOHA 4.14

Zistite, čo sa stane s obrázkami, ak namiesto elementu `<p>` použijete niektorý z nadpisových elementov.

ÚLOHA 4.15

Vyskúšajte zobraziť predchádzajúci kód v rôznych prehliadačoch. Zoznam ikoniek a ich kódov si môžete pozrieť na http://html-css-js.com/html/character-codes/icons/.

ÚLOHA 4.16

Vytvorte stránku s krátkym príbehom, či básničkou. Vaša stránka musí byť členená na nadpis, odseky, musíte na nej použiť štýly na nastavenie farby textu či pozadia a zmeniť písmo (`font-family`) aspoň v jednom odseku. Čo najviac podstatných mien v texte sa snažte nahradiť ikonkami zo zoznamu http://html-css-js.com/html/character-codes/icons/ alebo obrázkami.

HTML entity

Niektoré znaky sú v HTML rezervované na zápis HTML značiek. Ak by sme napríklad potrebovali v texte použiť znak *menší ako* (<) alebo *väčší ako* (>), webový prehliadač by tieto znaky pochopil ako súčasť HTML značiek a nezobrazil by ich správne. Na zápis takýchto rezervovaných znakov sa používajú HTML entity alebo znakové entity.

V nasledujúcej tabuľke uvádzame niektoré rezervované znaky a ich zápis pomocou HTML entít.

zápis v HTML	znak	popis
`<`	<	menší ako
`>`	>	väčší ako
`&`	&	and

Okrem rezervovaných znakov sa pomocou HTML entít zapisujú aj znaky, ktoré nemôžeme napísať pomocou klávesnice, napríklad Euro, copyright, libra a podobne. V nasledujúcej tabuľke uvádzame niektoré takéto znaky a ich zápis pomocou HTML entít.

zápis v HTML	znak	popis
`€`	€	Euro
`©`	©	copyright
`£`	£	libra

ZAPAMÄTAJTE SI

HTML entita sa zapisuje nasledovne
`&nazov_entity;`

Nesmieme zabudnúť na začiatku napísať znak **&** a na konci **bodkočiarku**. V názve entity treba dodržiavať malé a veľké písmená.

Z predchádzajúcich kapitol vieme, že v HTML kóde je ľubovoľne dlhá sekvencia medzier a tabulátorov chápaná ako jediná medzera. Ak chceme, aby prehliadač zobrazil viac medzier za sebou, môžeme v potrebnom počte napísať za sebou HTML entitu ` ` (Non Breaking SPace – nedeliteľná medzera). Napríklad kód `     ` spôsobí, že sa napíšu tri medzery za sebou.

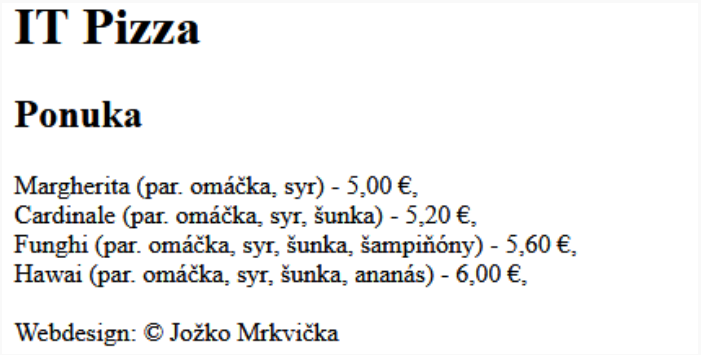
Ako už samotný názov entity napovedá, ak použijeme tento zápis medzi dvomi slovami, webový prehliadač to pochopí tak, že medzi týmito slovami nesmie zalomiť riadok. Napríklad kód `s pozdravom` zabezpečí, že reťazec „s pozdravom“ sa napíše do riadku celý tak, aby predložka s nebola na konci riadku.

PRÍKLAD 4.17

V online editore JSFiddle vytvoríme takýto odsek.

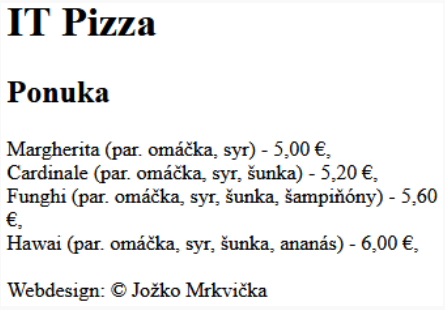
```
<body>
  <h1>IT Pizza</h1>
  <h2>Ponuka</h2>
  Margherita (par. omáčka, syr) - 5,00 &euro;, <br>
  Cardinale (par. omáčka, syr, šunka) - 5,20 &euro;, <br>
  Funghi (par. omáčka, syr, šunka, šampiňóny) - 5,60 &euro;, <br>
  Hawai (par. omáčka, syr, šunka, ananás) - 6,00 &euro;, <br>
  <br>
  Webdesign: &copy; Jožko Mrkvička
</body>
```

Výsledok vidíme na obrázku:



ÚLOHA 4.18

V editore JSFiddle s kódom z predchádzajúcej úlohy meňte veľkosť časti s výsledkom. Všimnite si, že ak sa znak € na šírku nezmestí, prejde do nového riadku.



Namiesto obyčajných medzier vložte medzi sumu a znak € (v kóde z príkladu 4.17) nedeliteľné medzery (` `). Znova meňte veľkosť časti s výsledkom a zistite, ako sa stránka správa teraz.

ÚLOHA 4.19

Vyskúšajte ďalšie HTML entity zo stránky www.w3schools.com/html/html_entities.asp.

Metodické pokyny pre učiteľa

CIEĽ

Cieľom je naučiť sa vkladať obrázky do webovej stránky, oboznámiť sa s ich atribútmi. Uvedomiť si význam atribútu alt a title.

VÝKLAD

Študentov treba upozorniť, že:

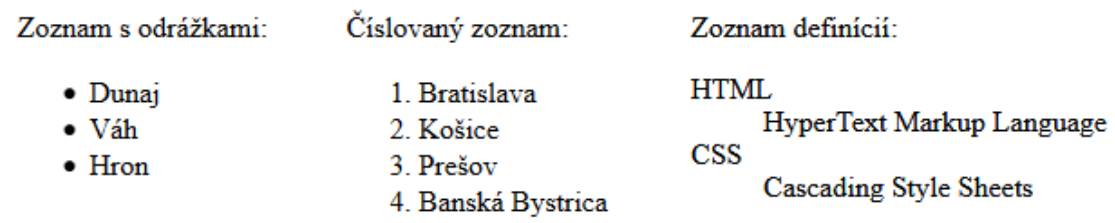
- v názvoch súborov nesmú byť medzery a znaky s diakritikou,
- cesta k súboru s obrázkom nesmie byť viazaná na umiestnenie na konkrétnom počítači (napr. file:///C:/ Web/06/obrazky/cardinale.jpg),
- obrázky si majú pripraviť v takej veľkosti, v akej ich chcú použiť na stránke, nemeniť rozmery obrázkov pomocou atribútov width a height.

Nadväznosť na predchádzajúce kapitoly: Kapitolu **je možné preberať až po kapitolách 1 a 2**. V kapitole sa nevyužívajú kaskádové štýly. Teoreticky je možné kapitolu 4 prebrať skôr ako kapitolu 3. V takom prípade nevyužite predpripravený súbor pre žiakov `04/index`, ale vlastný (resp. `03/index.html`).

Úloha 4.16 sa dá použiť ako opakovacie zadanie, alebo ako miniprojekt – možno aj pre dvojice. Ak žiaci riešili úlohu 3.10, môžu ju využiť v tejto úlohe.

5 ZOZNAMY S ODRÁŽKAMI A ČÍSLOVANÉ ZOZNAMY

V html rozlišujeme tri druhy zoznamov: zoznam s odrážkami, číslovaný zoznamy a zoznam definícií. Ich príklady môžeme vidieť na obrázku 5.1. V učebnici sa ďalej budeme venovať len prvým dvom typom, tvorbu zoznamu definícií si môže čitateľ naštudovať napríklad na http://wiki.lacko.me/html:zoznamy#zoznam_definicii alebo na https://www.w3schools.com/tags/tag_dl.asp.



Obrázok 5.1 Zobrazenie zoznamu s odrážkami, číslovaného zoznamu a zoznamu definícií v prehliadači.

Zoznamy s odrážkami

ÚLOHA 5.1

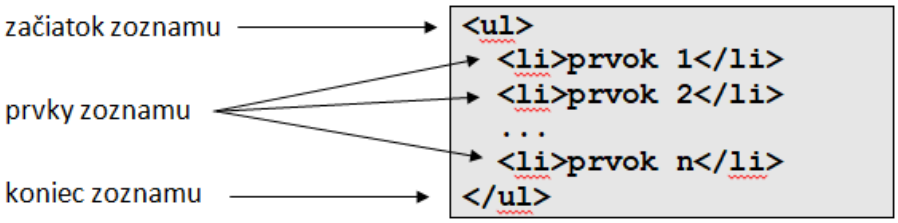
V editore JSFiddle vložte do časti HTML nasledujúci zdrojový kód.

```
<body>
  Prísady na pizzu:
  <ul>
    <li>šunka</li>
    <li>syr</li>
    <li>kukurica</li>
    <li>olivy</li>
  </ul>
</body>
```

a) Zmeňte text olivy na artičoky. Ako sa zmenila stránka?
b) Za riadok `<li>syr</li>` pridajte nový riadok `<li>saláma</li>`.
c) Zmažte riadok `<li>kukurica</li>`.
d) V riadku `<li>syr</li>` zmažte slovo *syr*.
e) Vedeli by ste sformulovať, aký význam má element `<li></li>`?
f) Zmažte značky `<ul>` a `</ul>`. Aký význam má element `<ul></ul>`?

Zoznam s odrážkami definujeme pomocou elementu `<ul></ul>`. Vnútri tohto elementu musia byť všetky prvky zoznamu, ktoré definujeme pomocou elementu `<li></li>` (Obrázok 5.2).

Zoznam s odrážkami používame vtedy, keď nám na poradí jednotlivých prvkov v zozname nezáleží (v angličtine sa mu hovorí ***u**nordered **l**ist*).



Obrázok 5.2 Schéma elementu `<ul></ul>`

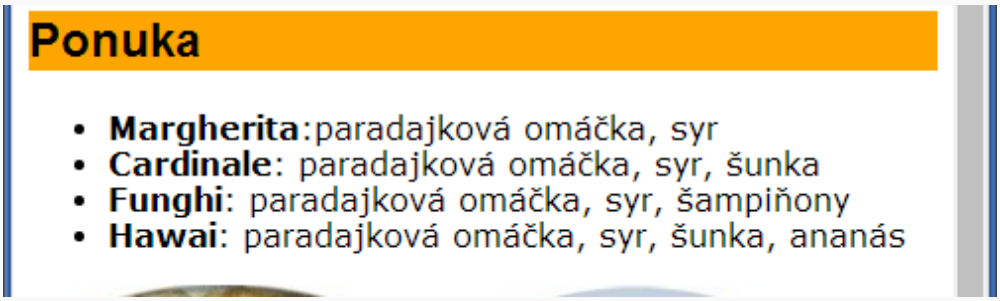
Vnútri elementu `<ul></ul>` môžu byť len elementy `<li></li>`, žiadne iné. V rámci elementov `<li></li>` už môžeme používať aj iné elementy.

PRÍKLAD 5.2

Otvorme súbor `05/index.html`. V zdrojovom kóde zmeňme odsek s ponukou pízz na zoznam s odrážkami.

```
…
<h2>Ponuka</h2>
<ul>
  <li><strong>Margherita</strong>:paradajková omáčka, syr</li>
  <li><strong>Cardinale</strong>: paradajková omáčka, syr,
šunka</li>
  <li><strong>Funghi</strong>: paradajková omáčka, syr,
šampiňony</li>
  <li><strong>Hawai</strong>: paradajková omáčka, syr, šunka,
ananás</li>
</ul>
```

Výsledok by mal vyzerať ako na obrázku:



Prvkami zoznamu môžu byť nielen texty, ale aj obrázky, či kombinácia obrázkov a textu. Vždy však musia byť definované **vnútri elementu** `<li></li>`.

PRÍKLAD 5.3

Na stránke IT Pizza pridajme do odrážky Margherita obrázok pizze Margerita. Element `<img>` s príslušným obrázkom sa už v zdrojovom kóde nachádza, stačí, ak ho presunieme do správneho elementu `<li>`.

```
<li><strong>Margherita</strong>:paradajková omáčka, syr <img
src="obrazky/margerita.jpg" alt="pizza margherita" width="165"
height="124" title="pizza margherita"></li>
```

ÚLOHA 5.4

Na stránke IT Pizza doplňte obrázok príslušnej pizze do každej odrážky (kódy elementov `<img>` sú v súbore `index.html` už vytvorené, stačí ich presunúť na správne miesta). Upravte zdrojový kód tak, aby sa obrázok pizze zobrazoval pod textom, ako na vidíme obrázku (pomôcka: využite element `<br>`).

Číslované zoznamy

ÚLOHA 5.5

V online editore JSFiddle zmeňte značky `<ul></ul>` na značky `<ol>`, `</ol>` (Ak ste medzitým editor zavreli, otvorte ho a skopírujte doň kód z úlohy 5.1). Čo sa zmenilo?

Číslovaný zoznam definujeme pomocou elementu `<ol></ol>`. Vnútri tohto elementu musia byť všetky prvky zoznamu, ktoré definujeme pomocou elementu `<li></li>`.

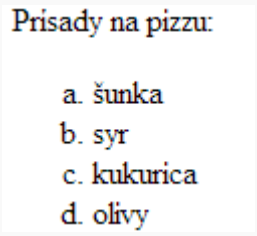
Zoznam s číslami používame vtedy, keď nám **záleží na poradí** prvkov v zozname (v angličtine sa mu preto hovorí ***o**rdered **l**ist*). Okrem čísel môžeme na označenie poradia použiť aj písmená (napr. a, b, c, …, resp. A, B, C, …), či rímske číslice.

PRÍKLAD 5.6

V zozname z *úlohy 5.5* chceme na určenie poradia namiesto čísel použiť písmená. Pridáme elementu `<ol>` atribút `type` s hodnotou `a`.

```
<body>
  Prísady na pizzu:
  <ol type="a">
    <li>šunka</li>
    <li>syr</li>
    <li>kukurica</li>
    <li>olivy</li>
  </ol>
</body>
```

Výsledok vidíme na obrázku:



ÚLOHA 5.7

V kóde z príkladu 5.6 meňte hodnotu atribútu `type` postupne na `"a"`, `"i"`, `"I"` a pozorujte zmeny na stránke.

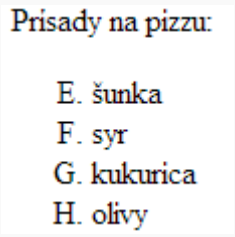
V nasledujúcej tabuľke uvádzame zoznam a popis atribútov, pomocou ktorých vieme nastaviť základné vlastnosti číslovaného zoznamu.

Atribút	Hodnoty	Popis
`type`	1, A, a, I, i	typ zoznamu (číslovaný, veľké písmená, malé písmená, veľké rímske číslice, malé rímske číslice)
`start`	*číslo*	hodnota, ktorou sa začína číslovať zoznam

ÚLOHA 5.8

Vyskúšajte si použitie atribútu `start` pre element `<ol>`. Skúšajte v kombinácii s rôznymi hodnotami atribútu `type`.

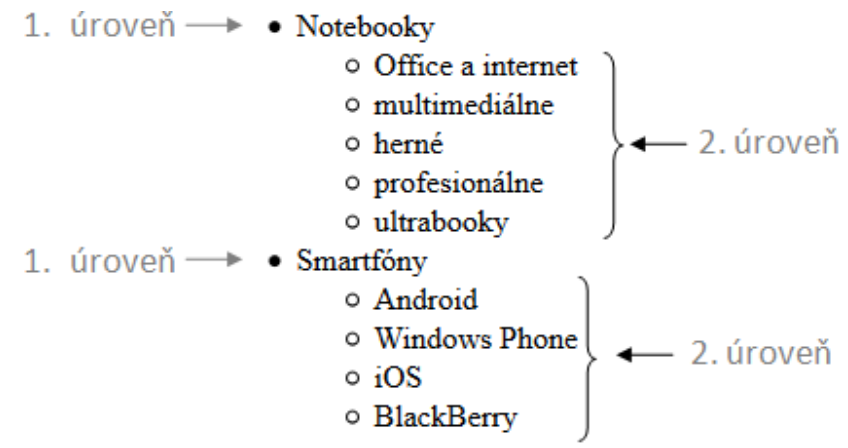
a) Pridajte elementu `<ol>` atribút `start` s hodnotou 3.
b) Upravte element `<ol>` tak, aby sa zoznam zobrazoval ako na obrázku:



Viacúrovňové zoznamy

Zoznamy aj ich prvky sú blokové elementy. Jednotlivé prvky zoznamu (t.j. obsah elementu `<li></li>`) môžeme formátovať (napr. pomocou riadkových elementov `strong`, `em`, `br`), aj do nich vkladať ďalšie bloky, napr. odseky, nadpisy, ďalšie zoznamy.

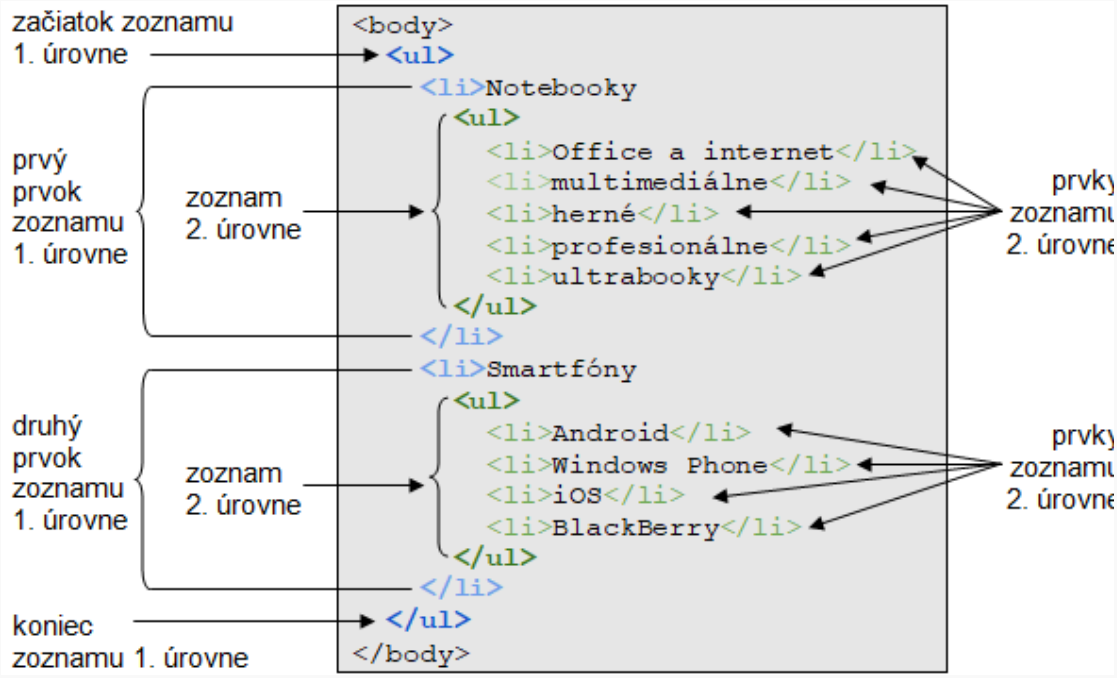
Ak prvkom zoznamu je ďalší zoznam, vzniká tzv. dvojúrovňový zoznam, ako na obrázku 5.3:



Obrázok 5.3 Ukážka dvojúrovňového zoznamu.

PRÍKLAD 5.9

Vytvorme zoznam ako na Obrázku 5.3.



Vložte tento kód do časti HTML v editore JSFiddle (kód môžete skopírovať zo súboru `05/priklad5-9.txt`).

ÚLOHA 5.10

Do zoznamu z predchádzajúceho príkladu:

- pridajte prvok *mini notebooky* do zoznamu notebookov,
- pridajte prvok *ostatné* do zoznamu smartfónov,
- pridajte na 1. úroveň nový prvok *Príslušenstvo*, ktorý bude zoznamom s tromi prvkami: *batérie*, *klávesnice* a *obaly*.

ÚLOHA 5.11

V kóde z predchádzajúcej úlohy experimentujte s rôznymi kombináciami elementov `<ul>` a `<ol>`. Napríklad:

- Nahraďte element `<ul>`, ktorý definuje zoznam 1. úrovne, elementom `<ol>`.
- Nahraďte element `<ul>` v niektorom zozname 2. úrovne (napr. pri definícii zoznamu smartfónov) elementom `<ol>`.
- Upravte viacúrovňový zoznam tak, aby zoznam 1. úrovne bol zoznam s odrážkami, ale všetky zoznamy 2. úrovne boli číslované pomocou písmen a., b., ...
- Pri riešení týchto úloh si dajte pozor na koncové značky – ak sa zoznam začína značkou `<ul>`, musí sa končiť značkou `</ul>` a ak sa začína značkou `<ol>`, musí sa končiť značkou `</ol>`.

ÚLOHA 5.12*

Pozrite si v niektorej svojej učebnici stranu s obsahom. Vedeli by ste rovnaký obsah vytvoriť pomocou jazyka HTML?

Upravte kód (doplňte vhodné elementy) zo súboru `05/obsah.html` tak, aby ste vytvorili viacúrovňový číslovaný zoznam.

Časté chyby pri tvorbe zoznamov

ÚLOHA 5.13

Preskúmajte nasledujúce zdrojové kódy. Sú podľa vás správne? Každý kód zobrazte vo webovom prehliadači, resp. „spustite“ v JSFiddle. Zobrazí prehliadač to, čo by ste očakávali?

Kódy nájdete v priečinku `05/chyby` v súboroch `01.html až 05.html`.

1.

```
<body>
 <ul>
   <li>ryby</li>
   <li>plazy
   <li>vtáky</li>
   <li>cicavce</li>
 </ul>
</body>
```

2.

```
<body>
 <ul>
   <li>ryby</li>
   <li>plazy</li>
   <li>vtáky</li>
   <li>cicavce</li>
</body>
```

3.

```
<body>
 <ul>
   <li>ryby</li>
   <li>plazy<li>
   <li>vtáky</li>
   <li>cicavce</li>
 </ul>
</body>
```

4.

```
<body>
  <li>ryby</li>
  <li>plazy</li>
  <li>vtáky</li>
  <li>cicavce</li>
</body>
```

5.

```
<body>
  <ul>
    <li>Notebooky</li>
    <ul>
      <li>herné</li>
      <li>multimediálne</li>
    </ul>
    <li>Smartfóny</li>
  </ul>
</body>
```

Najčastejšie pri tvorbe zoznamov dochádza k takýmto chybám:

- zabudneme koncovú značku `</li>` (kód 1) alebo `</ul>` (kód 2),
- namiesto koncovej značky `</li>` napíšeme počiatočnú `<li>` (kód 3) –tým definujeme úplne iný zoznam,
- úplne vynecháme element `<ul></ul>` a definujeme priamo prvky zoznamu pomocou elementu `<li>` (kód 4),
- zoznam 2. úrovne vložíme priamo do elementu `<ul>` pre 1. úroveň namiesto toho, aby sme ho vložili ako obsah elementu `<li>` 1. úrovne (kód 5).

Prehliadač zobrazí stránku aj s chybným zdrojovým kódom, zväčša sa naše chyby „snaží nejako opraviť". Nemali by sme sa však na to spoliehať, pretože nie každý prehliadač to dokáže a navyše takto „opravená" stránka nemusí zodpovedať našim predstavám.

ÚLOHA 5.14

Opravte chyby v kódoch z *úlohy 5.13*.

Nastavenie vlastností zoznamov pomocou CSS

Jazyk HTML ponúka len veľmi málo možností prispôsobenia zoznamov. Tvar odrážky v HTML5 nemôžeme nastaviť vôbec a pri číslovanom zozname máme len obmedzené možnosti. Pomocou kaskádových štýlov môžeme nastaviť viacero typov odrážok a číslovania a navyše pomocou vlastnosti `list-style-image` môžeme zoznamom nastaviť ľubovoľný obrázok ako odrážku.

PRÍKLAD 5.15

V editore JSFiddle použime ešte raz zdrojový kód z *úlohy 5.1* a zobrazme ho.

Do časti CSS pridajme nasledujúcu definíciu štýlu:

```
ul {
 list-style-type: square;
}
```

a znovu zobrazme výsledok. Odrážky sa zmenili z kruhov na štvorce.

Prísady na pizzu:

- šunka
- syr
- kukurica
- olivy

ÚLOHA 5.16

Meňte hodnotu `list-style-type` postupne na: `circle`, `disc`, `decimal`, `lower-greek`. Zmeňte tiež `<ul>` na `<ol>` a vyskúšajte znova.

Ďalšie hodnoty vlastnosti `list-style-type` si môžete pozrieť na https://www.w3schools.com/cssref/pr_list-style-type.asp

PRÍKLAD 5.17

Do zdrojového kódu stránky IT Pizza pridajme nasledujúcu definíciu štýlu (spomeňme si, že definície štýlov vkladáme do elementu `<style></style>`, ktorý je v hlavičke dokumentu):

```
...
ul {
 list-style-type: none;
 list-style-image: url('obrazky/pizza.png')
}
  </style>
```

Pri nastavovaní obrázku ako odrážky je vhodné vypnúť typ odrážky pomocou vlastnosti `list-style-type: none`.

Obrázok 5.4 Obrázok ako odrážka.

ÚLOHA 5.18

Zmeňte tvar odrážok na stránke IT Pizza na obrázok zo súboru `05/obrazky/odrazka.png`.

Metodické pokyny pre učiteľa

CIEĽ

Cieľom je naučiť sa vytvárať rôzne druhy zoznamov, pochopiť princíp vnárania zoznamov.

VÝKLAD

Žiakov necháme experimentovať v príklade 5.1, aby sami prišli na to, aký význam majú elementy `ul` a `li`, neskôr `ol`. Tiež veľa experimentujú s atribútom `type`.

Pri viacúrovňových zoznamoch treba zdôrazniť, že zoznam nižšej úrovne sa vkladá do elementu `li` zoznamu vyššej úrovne, nie priamo do elementu ul/`ol` zoznamu vyššej úrovne.

Nadväznosť na predchádzajúce kapitoly: vyžaduje všetky predchádzajúce kapitoly; ak ste vynechali kapitolu 3 so štýlmi, je treba z tejto kapitoly vypustiť časť 5.5 Nastavenie vlastností zoznamov pomocou CSS.

Úlohu 5.12 je možné využiť na zhrnutie.

V úlohe 5.18, kde žiaci mali pridať do stránky štýl

```
...
ul {
 list-style-type: none;
 list-style-image: url('obrazky/pizza.png')
}
```

môžeme so žiakmi ešte preskúmať, čo by sa stalo, keby:

- sme nezadali v štýle príkaz: `list-style-image: url('obrazky/pizza.png')` a nechali iba `list-style-type: none;`
- naopak, nechali: `list-style-image: url('obrazky/pizza.png')` a zrušili `list-style-type: none;`

Úloha 5.13 Zvýraznené chyby a ich oprava

Kód s chybami

1.

```
<body>
  <ul>
    <li>ryby</li>
    <li>plazy
    <li>vtáky</li>
    <li>cicavce</li>
  </ul>
</body>
```

chýba koncová značka

Opravený kód

```
<body>
  <ul>
    <li>ryby</li>
    <li>plazy</li>
    <li>vtáky</li>
    <li>cicavce</li>
  </ul>
</body>
```

2.

```
<body>
  <ul>
    <li>ryby</li>
    <li>plazy</li>
    <li>vtáky</li>
    <li>cicavce</li>
</body>
```

chýba koncová značka

```
<body>
  <ul>
    <li>ryby</li>
    <li>plazy</li>
    <li>vtáky</li>
    <li>cicavce</li>
  </ul>
</body>
```

3.

```
<body>
  <ul>
    <li>ryby</li>
    <li>plazy<li>
    <li>vtáky</li>
    <li>cicavce</li>
  </ul>
</body>
```

Namiesto koncovej značky počiatočná.

```
<body>
  <ul>
    <li>ryby</li>
    <li>plazy</li>
    <li>vtáky</li>
    <li>cicavce</li>
  </ul>
</body>
```

4.

```
<body>
  <li>ryby</li>
  <li>plazy</li>
  <li>vtáky</li>
  <li>cicavce</li>
</body>
```

vynechané <ul>

```
<body>
  <ul>
  <li>ryby</li>
  <li>plazy</li>
  <li>vtáky</li>
  <li>cicavce</li>
  </ul>
</body>
```

5.

```
<body>
  <ul>
    <li>Notebooky</li>
    <ul>
      <li>herné</li>
      <li>multimediálne</li>
    </ul>
    <li>Smartfóny</li>
  </ul>
</body>
```

chýba element

```
<body>
  <ul>
    <li>Notebooky</li>
    <li>
      <ul>
        <li>herné</li>
        <li>multimediálne</li>
      </ul>
    </li>
    <li>Smartfóny</li>
  </ul>
</body>
```

6 ODKAZY A NAVIGÁCIA

Odkazy nájdeme takmer na všetkých stránkach. Používame ich na prepojenie jednotlivých webových dokumentov. Prepájať môžeme:

- stránky z rôznych webových sídel: napr. keď kliknutím na odkaz zo stránky Google odskočíme na stránku školy a z nej sa zase kliknutím na odkaz dostaneme napr. na Facebook (Google, Facebook a web server, na ktorom je umiestnená stránka školy, sú určite rôzne webové sídla),
- rôzne stránky v rámci toho istého webového sídla: z hlavnej stránky školy sa dostaneme napr. na stránku s rozvrhom, stránku so zoznamom učiteľov, stránku s fotogalériou školy, atď.,
- jednotlivé časti v rámci tej istej stránky.

Zamyslite sa: Ako by ste zadefinovali odkaz?

Prepájanie webových sídiel

ÚLOHA 6.1

Pozrite si aktuálnu verziu stránky IT Pizza v súbore `06/index.html`. Rozšírili sme ponuku, doplnili fotogalériu priestorov pizzerie a v časti *Kontakt* sme pridali adresu druhej prevádzky. V úvode stránky pribudli informácie o možnosti objednania pizze a stravovacích zariadeniach, v ktorých ponúkajú našu pizzu.

Porozmýšľajte, s akými stránkami by sme mohli prepojiť našu stránku a ktoré objekty (texty, obrázky) by sme mohli využiť ako odkazy.

Text ako odkaz

PRÍKLAD 6.2

Prepojme stránku IT Pizza so stránkou stravovacieho zariadenia Free Food. Odkaz spravíme z textu Free Food v úvode stránky. Ešte potrebujeme poznať adresu stránky, na ktorú má náš odkaz viesť – tá je http://www.freefood.sk/.

Upravme časť zdrojového kódu nasledovne:

```
<h2>Ponuka</h2>
Našu pizzu si môžete vychutnať aj v stravovacích zariadeniach
<a href="http://www.freefood.sk">Free Food</a>.<br>
Objednať si môžete ...
```

Otestujme odkaz v prehliadači – zobrazme stránku a kliknime na text „Free Food“.

Prepojenia, teda hypertextové odkazy, vytvárame pomocou elementu `<a></a>`, ktorý vo všeobecnosti používame v tvare:

```
<a href="adresa">text odkazu</a>
```

kde:

- *text odkazu* je text, ktorý sa zobrazí na stránke a bude sa naň dať kliknúť,
- `href` je atribút elementu `<a>`, ktorého hodnotou je adresa stránky, na ktorú má viesť odkaz.

Všimnite si, že v zdrojovom kóde z *príkladu 6.2* sme ako adresu stránky uviedli nielen `www.freefood.sk`, ale `http://www.freefood.sk`.

Pri odkazoch na iné webové sídlo musíme uviesť aj text `http://`, práve tým totiž prehliadaču oznamujeme, že odkaz smeruje na iné webové sídlo a nie na súbor na našom počítači.

ODPOVEDZTE

Spomeniete si, čo znamená `http`? A aký je rozdiel medzi `http` a `https`?

Odkaz môžeme doplniť o nepovinný atribút `title`. Text, ktorý uvedieme v tomto atribúte, sa zobrazí v bublinovej nápovede pri prechode myšou ponad odkaz (podobne ako je to pri obrázku). Používame ho na uvedenie nejakej informácie navyše. **Upozornenie**: Nepleťme si tento atribút s **elementom** `<title>`.

PRÍKLAD 6.3

Doplňme na stránku IT Pizza odkaz na stránku `Bistro.sk`. Obom odkazom pridajme vhodný atribút `title`.

```
<h2>Ponuka</h2>
Našu pizzu si môžete vychutnať aj v stravovacích zariadeniach
<a href="http://www.freefood.sk" title="FreeFood | Super papaníčko
za skvelé ceny">Free Food</a>. <br>
Objednať si môžete cez náš objednávkový formulár alebo
prostredníctvom <a href="https://www.bistro.sk" title="Bistro.sk -
donáška, rozvoz jedla">Bistro.sk</a>.
```

Otestujme odkaz v prehliadači – zobrazme stránku a kliknime na text „Bistro.sk“.

ÚLOHA 6.4

V súbore `06/sutaze.html` je pripravený zoznam vybraných informatických súťaží. Vytvorte odkazy na ich stránky. Ku každému odkazu pridajte aj vhodný atribút `title`. Odkazy otestujte.

Obrázok ako odkaz

Už sme spomínali, že odkazom môže byť nielen text, ale aj obrázok. Takýmito odkazmi bývajú často logá reklamných partnerov, piktogramy sociálnych sietí a pod. Zdrojový kód pre obrázkový odkaz sa od zdrojového kódu pre textový odkaz líši len v tom, že medzi počiatočnú a koncovú značku elementu `<a></a>` dáme namiesto textu element `<img>`.

PRÍKLAD 6.5

Na stránke IT Pizza vytvoríme z loga sociálnej siete Facebook odkaz na stránku Facebooku (keby naša pizzeria mala vlastnú stránku na Facebooku, definovali by sme odkaz priamo na ňu, ale takto vytvoríme len odkazy na hlavnú stránku `facebook.com`).

```
<p>Sledujte nás na
  <a href="https://www.facebook.com/" title="Facebook"><img
src="obrazky/ico-fb.png" width="40" height="40" alt="ico-
facebook"></a>
  <img src="obrazky/ico-twitter.png" width="40" height="40"
alt="ico-twitter">
</p>
```

ÚLOHA 6.6

Na stránke IT Pizza v časti *Sledujte nás* doplňte odkaz na Twitter.

Možno ste si na niektorých stránkach všimli, že po kliknutí na odkaz sa stránka nezobrazí v aktuálnom okne prehliadača, ale otvorí sa v novom okne. Toto vieme zabezpečiť pomocou atribútu **`target`**, ktorému nastavíme hodnotu **`_blank`**. Schéma elementu `<a></a>` potom bude:

`<a href="adresa" target="_blank">`*`text odkazu`***`</a>`**.

PRÍKLAD 6.7

Upravme odkazy na Facebook a Twitter na stránke IT Pizza tak, aby sa príslušné stránky zobrazovali v novom okne.

```
<p>Sledujte nás na
  <a href="https://www.facebook.com/" title="Facebook"
target="_blank"><img src="obrazky/ico-fb.png" width="40"
height="40" alt="ico-facebook"></a>
  <a href="https://twitter.com/" title="Twitter"
target="_blank"><img src="obrazky/ico-twitter.png" width="40"
height="40" alt="ico-twitter"></a>
...
```

ÚLOHA 6.8

Na stránke `06/sutaze.html` (z úlohy 6.4) upravte odkazy tak, aby sa stránky príslušných súťaží zobrazovali v novom okne. Odkazy otestujte.

Prepájanie častí stránky

Určite ste si všimli, že stránka IT Pizza je už pomerne rozsiahla. Pre čitateľa, ktorý vidí našu stránku prvýkrát, nemusí byť jednoduché nájsť informáciu, ktorú potrebuje: z úvodu stránky nie je zrejmé, aké informácie sa na nej nachádzajú a tiež dostať sa napr. ku fotogalérii alebo ku kontaktom, znamená prerolovať cez celú ponuku, či iné informácie, čo nemusí byť pohodlné.

Ako vyriešime tieto nedostatky? Na úvod stránky pridáme niečo ako obsah – len krátke nadpisy, aby čitateľ vedel, z akých častí sa vlastne stránka skladá, napr. Ponuka, Fotogaléria, Akcie, Kontakt. Z týchto "nadpisov" môžeme ďalej spraviť odkazy na konkrétne časti stránky, t.j. také odkazy, na ktoré keď klikneme, stránka sa automaticky presunie na konkrétnu časť.

ÚLOHA 6.9

Zobrazte v prehliadači stránku `06/ucebnica.html` a postupne klikajte na jednotlivé odkazy v úvode. Po každom odskoku sa vráťte na úvod stránky a vyskúšajte ďalší odkaz.

Rozdelenie stránky na sekcie

Aby sme mohli vytvoriť odkazy na isté časti stránky, potrebujeme obsah stránky **logicky** rozdeliť na časti (sekcie) a jednotlivým častiam dať nejaký jednoznačný identifikátor, aby sme sa na ne neskôr vedeli odvolávať.

Sekcie v HTML definujeme pomocou elementu `<section></section>`. Obsahom tohto elementu je všetko, čo sa má v príslušnej sekcii nachádzať.

```
<section>obsah sekcie</section>
```

Použitím atribútu `id`, do ktorého priradíme vhodný reťazec, môžeme sekciu pomenovať, napr. `<section id="ponuka">obsah sekcie</section>`.

Hodnota atribútu `id` musí byť na stránke jedinečná. Atribút `id` má v HTML oveľa širšie použitie a pre jeho hodnoty platia isté obmedzenia. Ďalšie informácie o atribúte `id` nájdeme napr. na https://www.w3schools.com/html/html_id.asp.

PRÍKLAD 6.10

Na stránke IT Pizza vytvoríme sekciu s identifikátorom *ponuka*. Obsahom sekcie bude podnadpis *Ponuka* (element `<h2>Ponuka</h2>`) a nasledujúci zoznam ponúkaných pízz (element `<ul>`).

Pred zdrojový kód nadpisu *Ponuka* pridáme počiatočnú značku `<section id="ponuka">` a za text *Zoznam alergénov* doplníme koncovú značku `</section>`. Pre rozsiahlosť uvádzame v nasledujúcom zdrojovom kóde len skrátený zoznam pízz.

```
<section id="ponuka">
  <h2>Ponuka</h2>
  Našu pizzu si môžete vychutnať aj v stravovacích zariadeniach
  <a href="http://www.freefood.sk" title="Free Food | Super
  papaníčko za skvelé ceny">Free Food</a>.<br>
  Objednať si môžete cez náš objednávkový formulár alebo
  prostredníctvom <a href="http://www.bistro.sk"
  title="Bistro.sk - donáška, rozvoz jedla">Bistro.sk</a>.
  <ul>
    <li><strong>Margherita</strong><br>
      paradajková omáčka, syr <sub>1, 7</sub><br>
      <img src="obrazky/margerita.jpg" alt="pizza Margherita"
      width="200" height="110" title="pizza Margherita"><br>
      malá 3,00 &euro;<br>   veľká 4,50 &euro;
  </li>
    ...
  <li><strong>Tonno</strong><br>
    paradajková omáčka, mozarella, tuniak, cibuľa
    <sub>1, 4, 7</sub><br>
      <img src="obrazky/tonno.jpg" alt="pizza Tonno"
      width="200" height="110" title="pizza Tonno"><br>
      malá 4,00 &euro;<br>   veľká 5,50 &euro;
  </li>
  </ul>
  Zoznam alergénov
</section>
```

PRÍKLAD 6.11

Na stránke IT Pizza vytvoríme sekciu s identifikátorom *akcia*. Obsahom sekcie bude podnadpis *Akcia* a nasledujúci odsek.

```
<section id="akcia">
  <h2>Akcia</h2>
  <p><strong>Každý pondelok</strong> pri kúpe <strong>troch
     pízz</strong> podľa ľubovoľného výberu máte najlacnejšiu
     z nich zadarmo.</p>
</section>
```

ÚLOHA 6.12

Vytvorte na stránke IT Pizza sekcie pre fotogalériu (`id="galeria"`) a kontakt (`id="kontakt"`). Sekcia *Kontakt* bude obsahovať všetko od adries oboch prevádzok pizzérie až po koniec stránky.

Vytvorenie odkazov na elementy s atribútom id

Ak máme vo webovom dokumente elementy, ktoré majú svoj identifikátor (atribút `id`), môžeme na ne veľmi jednoducho vytvoriť odkaz. Vytvoríme ho rovnako ako iný odkaz pomocou elementu `<a>` s atribútom `href`. Hodnotou atribútu `href` však tentokrát bude reťazec zložený z `#` a identifikátora (`id`) elementu (teda vlastne niečo ako *adresa elementu*). Napríklad HTML kód odkazu na element `<section id="#galeria">` bude vyzerať:

```
<a href="#galeria">fotogaléria</a>
```

Vo všeobecnosti definujeme **odkaz na pomenovaný element** takto:

`<a href="#id_elementu">`*`text_odkazu`***`</a>`**

kde `id_elementu` je jednoznačný identifikátor elementu, t.j. hodnota atribútu `id` príslušného elementu.

PRÍKLAD 6.13

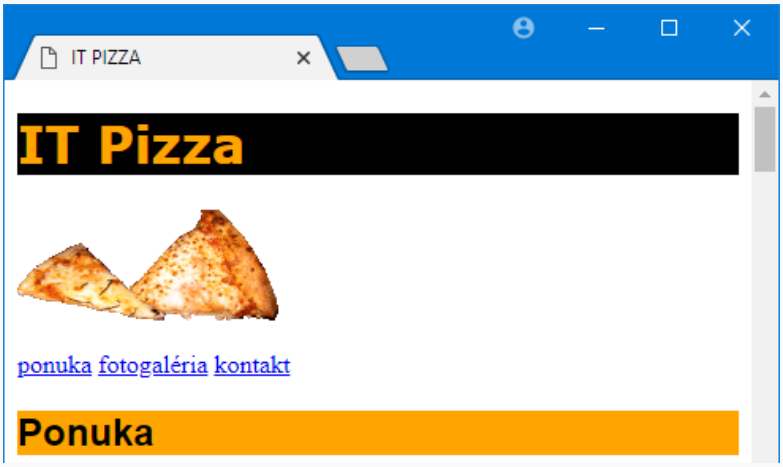
Na stránke IT Pizza za logom stránky pridáme odsek s odkazmi na časti *Ponuka* a *Fotogaléria*:

```
<h1>IT Pizza</h1>
<img src="obrazky/logo.gif" alt="logo" width="167" height="70"
title="logo">

<p>
  <a href="#ponuka">ponuka</a> <a href="#galeria">fotogaléria</a>
</p>
...
```

ÚLOHA 6.14

Otestujte funkčnosť odkazov z *príkladu 6.13*. Doplňte do stránky IT Pizza odkaz na časť *Kontakt*.



Navigácia

Skupinu odkazov, ktoré vzájomne prepájajú jednotlivé časti webového sídla (nemusia to byť nutne časti jednej a tej istej stránky, ale aj odkazy na iné stránky tvoriace istý celok) nazývame

navigácia. Navigáciu zvykneme definovať pomocou elementu `<nav></nav>`. Logicky ju tak odlíšime od iných častí stránky (napr. od section). Viac o navigácii si môžete prečítať na https://en.wikipedia.org/wiki/Web_navigation.

ÚLOHA 6.15

Pozrite si stránku svojej školy, svojho mesta, obľúbenej skupiny či inú často navštevovanú stránku a skúste identifikovať, čo je v tejto stránke navigácia. Všímajte si aj nasledujúce:

- Kde v rámci stránky sa navigácia nachádza?
- Ako sú odkazy navigácie rozložené (vedľa seba, pod sebou, inak)?
- Je navigácia v rámci toho istého webového sídla stále rovnaká (obsahuje rovnaké odkazy)?
- Smerujú odkazy z navigácie do toho istého dokumentu alebo na iné stránky?
- Vedeli by ste sformulovať nejaké zásady pre dobrú navigáciu?

ÚLOHA 6.16

V poslednej verzii stránky IT Pizza (z úlohy 6.14) nahradíme element `<p>` obsahujúci navigáciu elementom `<nav></nav>`.

```
<h1>IT Pizza</h1>
<img src="obrazky/logo.gif" alt="logo" width="167" height="70"
title="logo">
<nav>
  <a href="#ponuka">ponuka</a> <a href="#galeria">fotogaléria</a>
  <a href="#kontakt">kontakt</a>
</nav>
...
```

Stránku otestujte. Všimnite si, že sa zmenil aj vzhľad odkazov. V kapitole 7 sa naučíme prispôsobovať vzhľad jednotlivých odkazov aj navigácie ako celku pomocou kaskádových štýlov.

Prepájanie stránok v rámci sídla

Už sme sa naučili, ako prepojiť rôzne webové sídla, aj ako prepojiť jednotlivé časti v rámci tej istej stránky. V tejto časti si ukážeme, ako prepojiť stránky v rámci toho istého webového sídla.

ÚLOHA 6.17

Pozrite si obsah priečinka `06/cestovka`. Postupne zobrazte v prehliadači súbory `index.html`, `grecko.html`, `chorvatsko.html`, `objednavka.html`. Nájdite na týchto stránkach časť, ktorú by sme mohli považovať za navigáciu a preskúmajte ju – klikajte na jednotlivé odkazy.

V *úlohe 6.17* ste si určite všimli, že jednotlivé html súbory v priečinku `cestovka` sú navzájom poprepájané – z každého sa vieme kliknutím na vhodný odkaz dostať do všetkých ostatných. Podobný princíp prepájania súborov ste pravdepodobne videli aj na stránkach z *úlohy 6.15*.

Pozrime si zdrojový kód súboru `06/cestovka/index.html`. Časť s navigáciou vyzerá takto.

```
<nav>
    <a href="index.html">Hlavná stránka</a>
    <a href="grecko.html">Grécko</a>
    <a href="chorvatsko.html">Chorvátsko</a>
    <a href="objednavka.html">Objednávka</a>
</nav>
```

Ak porovnáme tento zdrojový kód so zdrojovým kódom z *príkladu 6.16*, zistíme, že sú veľmi podobné. Namiesto odkazov na konkrétne pomenované miesta v dokumente však teraz používame odkazy na súbory v rámci toho istého webového sídla.

Vo všeobecnosti definujeme odkaz na súbor v rámci toho istého webového sídla takto:

`<a href="adresa_suboru">`*`text_odkazu`***`</a>`**

kde *adresa_suboru* je jeho umiestnenie na disku vzhľadom na súbor, z ktorého vedie odkaz. Ak sú oba súbory umiestnené v tom istom priečinku, bude *adresa_suboru* zhodná s názvom súboru (vrátane prípony html). Viac o adresovaní súborov si pozri v nasledujúcej časti **Relatívne a absolútne adresovanie**.

PRÍKLAD 6.18

Na stránke IT Pizza pod ponukou pízz vytvoríme odkaz na zoznam alergénov, ktorý máme uložený v súbore `06/alergeny.html`

```
    <li><strong>Tonno</strong>...</li>
  </ul>
  <a href="alergeny.html">Zoznam alergénov</a>
</section>
...
```

Keďže súbor `alergeny.html` sa nachádza v rovnakom priečinku ako súbor, v ktorom naň vytvárame odkaz (teda súbor so stránkou IT Pizza), bude hodnotou atribútu `href` nového elementu `<a>` len názov súboru, čiže `alergeny.html`.

PRÍKLAD 6.19

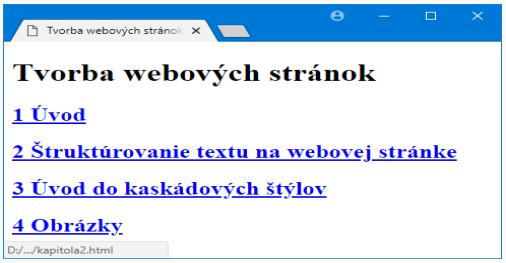
Na stránke IT Pizza takmer na konci vytvoríme odkaz na obchodné podmienky prevádzok IT Pizza. Tie máme uložené v súbore `06/obchodne_podmienky.pdf`.

```
  ...
  <a href="obchodne_podmienky.pdf">obchodné podmienky</a>
  <p>&copy; IT akadémia, 2018, Mlynská dolina, 842 48
Bratislava</p>
</body>
</html>
```

Podobným spôsobom, ako v *príkladoch 6.18* a *6.19* by sme mohli vytvoriť odkaz na ľubovoľný typ súboru, nemusí to byť len html súbor.

ÚLOHA 6.20

V priečinku `06/ucebnica` nájdete päť súborov: štyri z nich sú kapitoly učebnice (`kapitola1.html`, `kapitola2.html`, `kapitola3.html` a `kapitola4.html`), piaty je obsah tejto učebnice (`obsah.html`). Na stránke `obsah.html` vytvorte z názvov jednotlivých kapitol odkazy na príslušné kapitoly. Výsledok vidíte na obrázku 6.1.

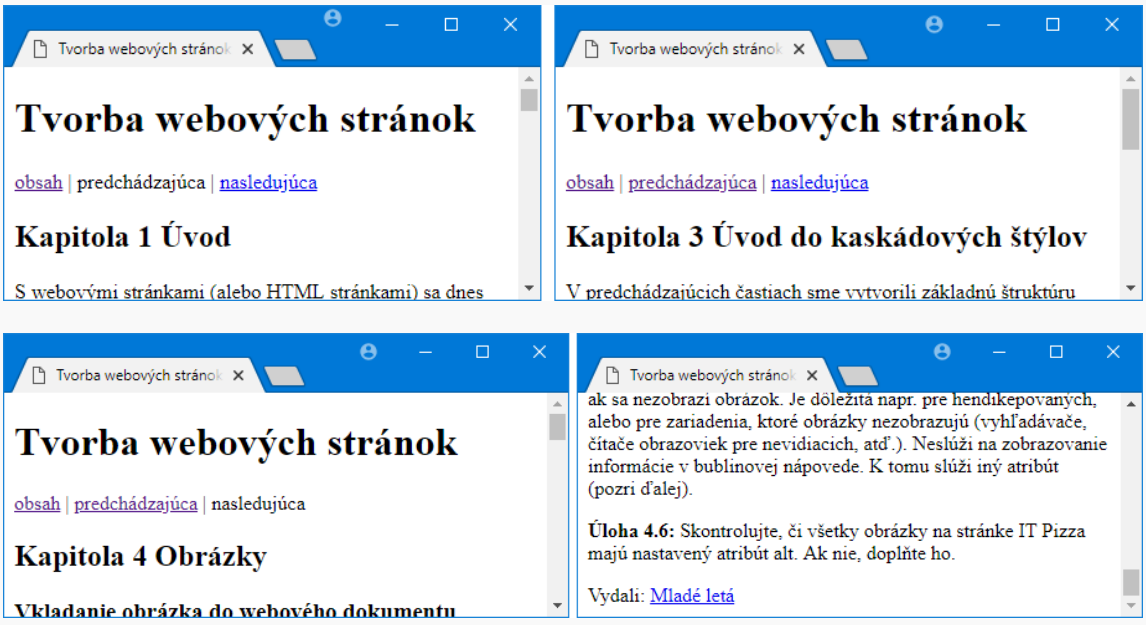


Obrázok 6.1 Ukážka odkazov na jednotlivé kapitoly.

ÚLOHA 6.21

Prepojte stránky `kapitola1.html`, `kapitola2.html`, `kapitola3.html` a `kapitola4.html` z predchádzajúcej úlohy nasledovne (pozri tiež obrázok 6.2):

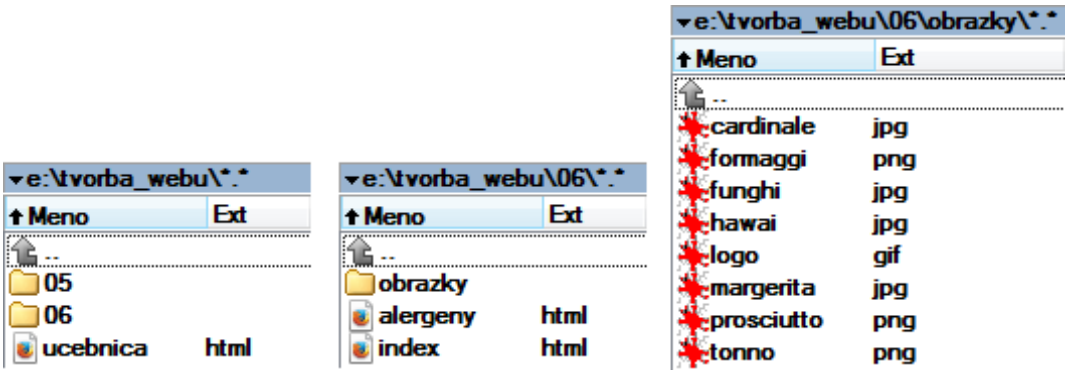
- na všetkých stránkach kapitol sa budú v navigácii nachádzať vždy tri prvky: *obsah*, *predchádzajúca* a *nasledujúca*:
- text *obsah* bude vždy odkazom na stránku `obsah.html`,
- text *predchádzajúca* bude odkazom na predchádzajúcu kapitolu, ak existuje (čiže na 1. kapitole tento text nebude odkazom)
- a text *nasledujúca* bude vždy odkazom na nasledujúcu kapitolu, ak taká existuje (čiže na poslednej kapitole tento text nebude odkazom)
- na každej stránke dolu bude odkaz na stránku vydavateľstva Mladé letá (www.mladeleta.sk). Stránka vydavateľstva sa bude otvárať v novom okne.



Obrázok 6.2 Ukážka odkazov na obsah, predchádzajúcu a nasledujúcu kapitolu.

Relatívne a absolútne adresovanie

Relatívna adresa súboru je adresa tohoto súboru vzhľadom ku konkrétnemu webovému dokumentu – tomu, v ktorom sa na daný súbor odvolávame (napr. pri tvorbe odkazu `<a>` alebo vkladaní obrázka `<img>`). Napríklad, ak máme v istom priečinku súbor `index.html` a tiež (pod)priečinok `obrazky`, v ktorom je súbor `logo.gif`, potom adresa súboru `logo.gif` vzhľadom na súbor `index.html` je `obrazky/logo.gif`. Teda, keby sme na stránku `index.html` chceli vložiť obrázok zo súboru `logo.gif`, kód by vyzeral nasledovne: `<img src="obrazky/logo.gif" alt="logo">`.



Obrázok 6.3 Štruktúra priečinkov a ich obsah.

V prípade, že sa chceme odkázať o úroveň vyššie (nadradený priečinok), použijeme znaky `../`. Napr. ak by sme v súbore `index.html` chceli vytvoriť odkaz na súbor `ucebnica.html`, ktorý je v nadradenom priečinku (pozri štruktúru priečinkov a súborov na *obrázku 6.3*), použili by sme kód `<a href="../ucebnica.html">`. Ak sa potrebujeme odkázať o viac úrovní vyššie, použijeme viackrát znaky `../`.

ÚLOHA 6.22

V štruktúre priečinkov a súborov zobrazenej na *obrázku 6.3*, aká by bola:

- relatívna adresa súboru `alergeny.html` vzhľadom na súbor `index.html`,
- relatívna adresa súboru `alergeny.html` vzhľadom na súbor `ucebnica.html`,
- relatívna adresa obrázka `hawai.jpg` vzhľadom na súbor `ucebnica.html`,
- relatívna adresa súboru `zoznamy.html`, ktorý sa nachádza v priečinku `05`, vzhľadom na súbor `ucebnica.html`,
- * relatívna adresa súboru `zoznamy.html`, ktorý sa nachádza v priečinku `05`, vzhľadom na súbor `index.html`.

Absolútna adresa súboru je úplná (webová) adresa tohoto súboru na serveri alebo lokálnom počítači. Nie je závislá od umiestnenia dokumentu, v ktorom túto adresu používame. Napr.

- https://www.nejakyserver.sk/obsah.html
- https://www.nejakyserver.sk/obrazky/logo.jpg
- https://www.nejakyserver.sk/vyucba/informatika.html

alebo pri umiestnení na lokálnom počítači:

- c://web/03/obrazky/logo.jpg
- d://ucebnice/informatika/tvorba_webu/index.html

ÚLOHA 6.23

V štruktúre priečinkov a súborov zobrazenej na obrázku 6.3, aká by bola:

- absolútna adresa súboru `ucebnica.html`,
- absolútna adresa súboru `alergeny.html`,
- absolútna adresa obrázka `hawai.jpg`,
- absolútna adresa súboru `zoznamy.html`, ktorý sa nachádza v priečinku `05`.

Ak máme všetky súbory (stránky, obrázky atď.) na našom lokálnom počítači a robíme medzi nimi prepojenia, je vhodné používať relatívne adresy.

Metodika pre učiteľa

CIEĽ

Cieľom je naučiť sa akým spôsobom v HTML definujeme odkazy. Uvedomiť si, že odkazmi môžeme spájať jednotlivé časti stránky, stránky v rámci jedného sídla, stránky z rôznych webových sidiel. Rozlišovať medzi absolútnou a relatívnou adresou.

Nadväznosť na predchádzajúce kapitoly: Nutne vyžaduje kapitoly 1, 2 a 4. V tejto kapitole sa nepracuje so štýlmi ani so zoznamami, čiže priamo nevyžaduje kapitolu 3 a 5. Štýly aj odrážky sa však objavujú v zdrojových kódoch vstupných súborov.

Možné problémy: zle uvedené adresy

Príklad 6.5: V riešení sa používa https, nie je to však v materiáli nijako zdôraznené ani tu nie je vysvetlený rozdiel medzi http a https. Predpokladáme, že žiaci už poznajú protokol http, resp. https. V prípade, že to tak nie je, odporúčame tento príklad preskočiť.

Úloha 6.9: Táto úloha slúži na to, aby pochopili/vyskúšali, ako funguje odkazovanie v rámci jedného webového dokumentu.

Odkazovanie na konkrétne časti stránky (K časti Vytvorenie odkazov na elementy s id): Na označenie miesta na stránke, na ktoré má viesť odkaz, sa kedysi používal aj element . Tento spôsob označovania však už nie je v súlade so štandardom HTML5. Správne je využívať na odkazovanie v rámci jednej stránky id atribúty elementov.

Úloha 6.21: Úloha je určená na opakovanie.

Po absolvovaní tejto kapitoly odporúčame, aby žiaci vypracovali projekt **Webová stránka mesta** alebo **Webové sídlo turistickej atrakcie**. Zadanie obidvoch projektov je v súbore *projekty*-pre_ziakov.zip. Žiaci môžu vytvárať webovú stránku alebo webové sídlo podľa vlastného výberu, prípadne môžu vytvárať stránky podľa predlohy na obrázkoch. V takom prípade môžu použiť poskytnuté dátové súbory. Učitelia môžu pri kontrole použiť výslednú stránku v súbore *projekty*-pre_ucitelov.zip.

7 CSS - VLASTNOSTI PÍSMA A TEXTU, PSEUDOTRIEDY

V tejto kapitole budeme skúmať rôzne vlastnosti textu a písma. Mnohé z nich vám budú určite známe z práce v textovom editore. Dozvieme sa viac o tom, ako sa v kaskádových štýloch dajú definovať farby. Naučíme sa tiež zmeniť správanie zvolených elementov na stránke vtedy, keď nad ne nadídeme myšou.

Vlastnosti písma a textu

V *kapitole 3* sme si ukázali, ako pomocou kaskádových štýlov nastavíme typ písma (vlastnosť `font-family`). Okrem typu môžeme rozhodnúť aj o iných vlastnostiach písma, napr. o veľkosti, či má byť písmo tučné, šikmé, podčiarknuté, prečiarknuté, atď. Niektoré z týchto vlastností vieme nastaviť pomocou HTML elementov (viete ktoré a pomocou ktorých elementov?), avšak kaskádové štýly nám ponúkajú oveľa širšiu ponuku rôznych nastavení písma a textu ako samotné HTML.

Tučné, šikmé a podčiarknuté písmo

PRÍKLAD 7.1

V editore JSFiddle skopírujeme do okna HTML zdrojový kód zo súboru `07/vitaminy.html`. Definujme kaskádové štýly tak, aby všetky nadpisy úrovne 1 boli podčiarknuté, všetky nadpisy úrovne 2 boli šikmé a všetky odkazy boli tučné. (Len pripomíname, že v editore JSFiddle píšeme definície štýlov do CSS časti.)

```
h1 {
  font-family: Verdana;
  text-decoration: underline;
}
h2 {
  font-style: italic;
}
a {
  font-weight: bold;
  color: blue;
}
```

Pomocou vlastnosti `font-weight` nastavujeme hrúbku písma. Ak chceme, aby písmo bolo tučné, použijeme hodnotu `bold`. Šikmé písmo nastavíme pomocou vlastnosti `font-style` s hodnotou `italic`. Pomocou vlastnosti `text-decoration` môžeme písmo podčiarknuť, prečiarknuť, či dokonca nadčiarknuť. Pre podčiarknuté písmo zvolíme hodnotu `underline`.

ÚLOHA 7.2

V CSS kóde z príkladu 7.1

- vyskúšajte niektoré iné hodnoty vlastnosti `font-weight` uvedené v tabuľke 7.1, napr. `normal`, `bolder`, `lighter`, `500`, …,
- experimentujte s rôznymi hodnotami vlastnosti `text-decoration`, napr. `underline`, `overline`, `line-through`.
- V tabuľke 7.1 si pozrite vlastnosti `font-variant` a `text-transform`. Vyskúšajte ich. Experimentovaním zistite, aký je rozdiel medzi `font-variant: small-caps` a `text-transform: uppercase`.

Otázka: Porozmýšľajte, ktoré elementy majú prednastavenú hrúbku písma na `bold`, a ktoré majú prednastavené šikmé písmo? Aké prednastavené vlastnosti majú odkazy?

Ako môžeme zrušiť nejakú vlastnosť elementu? Ak sme ju definovali sami v nejakom kaskádovom štýle, stačí ju z príslušného štýlu odstrániť. Ak je to nejaká prednastavená vlastnosť elementu, musíme ju „potlačiť“, t.j. použiť takú hodnotu príslušnej vlastnosti, ktorá je neutrálna – zväčša `normal` alebo `none`.

PRÍKLAD 7.3

Pokračujme v štýlovaní v editore JSFiddle. Zrušme podčiarknutie nadpisov úrovne 1, tučné písmo v nadpisoch z úrovne 2 a podčiarknutie odkazov.

```
h1 {
  font-family: Verdana;
}
h2 {
  font-style: italic;
  font-weight: normal;
}
a {
  font-weight: bold;
  color: blue;
  text-decoration: none;
}
```

Podčiarknutie nie je prednastavená vlastnosť nadpisov úrovne 1, definovali sme ju my v *príklade 7.1*, takže ju zrušíme jednoducho tým, že z definície štýlu pre `h1` zmažeme riadok `text-decoration: underline`. V prípade odkazov, ktoré majú podčiarknutie prednastavené, zrušíme podčiarknutie nastavením vlastnosti `text-decoration` na hodnotu `none`. Nadpisy (nielen úrovne 2) majú prednastavenú hrúbku písma na tučné, čo dokážeme zrušiť nastavením hodnoty `font-weight na normal`.

Veľkosť písma

Veľkosť písma nastavujeme pomocou vlastnosti `font-size`. Môžeme ju zadať v rôznych jednotkách, napr. v pixeloch (`px`), v percentách (`%`) a jednotkách `em`.

Veľkosť písma v pixeloch

PRÍKLAD 7.4

V kóde z predchádzajúceho príkladu zmeňme veľkosť písma nadpisov `h1` na `30px` a veľkosť písma odsekov na `18px`.

```
h1 {
  font-family: Verdana;
  font-size: 30px;
}
p {
  font-size: 18px;
}
```

Jednotka pixel je vám určite známa, spája sa s rastrovou grafikou a udávajú sa v nej napr. veľkosti fotografií. Prednastavená veľkosť normálneho textu (teda napr. v odseku) v prehliadačoch je `16px`. Pomocou jednotky `px` nastavujeme veľkosť písma absolútne, t.j. na konkrétnu veľkosť, bez ohľadu na to, akú veľkosť majú okolité elementy.

Veľkosť písma v percentách

PRÍKLAD 7.5

Zmeňme CSS kód z príkladu 7.4 nasledovne:

```
h1 {
  font-family: Verdana;
  font-size: 150%;
}
p {
  font-size: 90%;
}
```

Ak použijeme na nastavenie veľkosti písma percentá, nastavujeme veľkosť písma relatívne vzhľadom na nadradený element (element, v ktorom sa nachádza náš element, v našom prípade `<body>`). Ak za základ (100%) vezmeme veľkosť textu v elemente `<body>`, potom `font-size` s hodnotou 90% definuje písmo o niečo menšie ako základné a `font-size` s hodnotou 150% písmo, ktoré je o polovicu väčšie ako základné.

Veľkosť písma v em

PRÍKLAD 7.6

Príklad 7.6: Zmeňme CSS kód z *príkladu 7.5* nasledovne:

```
h1 {
  font-family: Verdana;
  font-size: 1.5em;
}
p {
  font-size: 0.9em;
}
```

Výsledná stránka v *príklade 7.6* bude rovnaká ako v *príklade 7.5*. Jednotka `em` rovnako ako `%` umožňuje relatívne nastavenie veľkosti písma vzhľadom na nadradený element. `1em` je 100%, a teda napr. `2em` je 200%, `0.5em` je 50% atď.

Upozornenie: Pri všetkých jednotkách musíme hodnotu spolu s jednotkou uvádzať ako jedno slovo (bez medzery!), napr. `2.5em`, `130%`, `800px`. Ak je hodnotou desatinné číslo, zapisujeme ho s desatinnou bodkou.

ÚLOHA 7.7

V CSS kóde z predchádzajúcich príkladov skúšajte pre jednotlivé elementy rôzne hodnoty `font-size` s rôznymi jednotkami, napr.

- pre element `h1` skúste: `2em`, `2.5em`, `3em`, `180%`, `250%`, `30px`, `40px`, ...
- pre element `p` skúste `120%`, `0.8em`, `14px`, ...
- definujte `font-size` pre element `li` alebo `a` a experimentujte.

Absolútna a relatívna veľkosť písma

ÚLOHA 7.8

V dvoch samostatných oknách (kartách) prehliadača si otvorte súbory `07/font-size-rel.html` a `07/font-size-abs.html`. Taktiež si tieto súbory otvorte v editore, v ktorom píšete HTML kód. HTML časť je v oboch stránkach rovnaká, líšia sa len v štýloch. V súbore `font-size-rel.html` sme použili relatívne nastavenia veľkostí pre elementy `p` a `strong`, v súbore `font-size-abs.html` absolútne. V oboch súboroch sme veľkosť písma v elemente `body` nastavili na `14px`.

Experimentujte s hodnotou `font-size` pre element `body`, ale tak, že nastavíte vždy rovnakú hodnotu v oboch súboroch. Jednotku `px` nemeňte. Porovnajte obe výsledné stránky.

Zarovnanie textu

Zarovnanie textu určite poznáte z textových editorov. Text v odseku môžeme zarovnať vľavo, na stred, vpravo, alebo vľavo aj vpravo (pozri *obrázok 7.1*). V kaskádových štýloch slúži na zarovnanie textu vlastnosť `text-align`. Môže nadobúdať jednu z hodnôt `left`, `center`, `right` alebo `justify`. Vlastnosť `text-align` môžeme použiť len pre blokové elementy.

Obrázok 7.1 Možnosti pre zarovnanie textu.

ÚLOHA 7.9

V editore JSFiddle v kóde z úlohy 7.7 upravte štýl pre element `<p>` – postupne vyskúšajte všetky hodnoty `text-align`. Pri skúšaní odporúčame zmenšiť šírku okna s výslednou stránkou.

V *tabuľke 7.1* sumarizujeme niektoré vlastnosti písma a ich hodnoty. Zoznam vlastností, ani ich hodnôt nie je úplný. Kompletný zoznam vlastností písma nájdete na https://www.w3schools.com/css/css_font.asp.

Tabuľka 7.1 Vlastnosti písma.

vlastnosť	hodnoty	popis, význam
`font-weight`	normal | bold | bolder | lighter | 100 | 200 | 300 | ... | 800 | 900	nastaví hrúbku písma
`font-style`	normal | italic | oblique	nastaví štýl (šikmosť) písma
`font-variant`	normal | small-caps	zobrazí text normálne, alebo ako malé KAPITÁLKY (veľkosť písmen je zachovaná, ale všetky vyzerajú ako veľké)
`font-family`	napr. Arial, Helvetica, sans-serif, "Times New Roman", Times, serif, "Courier New", Courier, mono	určuje tzv. rodinu písma viacslovné názvy musia byť v úvodzovkách alebo v apostrofoch
`font-size`	*výška_písma* v px, napr. 16px *výška_písma* v em, napr. 2em *výška_písma* v %, napr. 140%	nastaví výšku písma elementu hodnoty v jednotkách em a % sa vzťahujú k elementu, do ktorého je tento element vnorený
`color`	názov farby, napr. green RGB kód farby, napr. #FFCC00	farba textu
`text-decoration`	none | underline | overline | line-through	nastavenie podčiarknutia, nadčiarknutia, prečiarknutia a blikania textu
`text-transform`	none | capitalize | uppercase | lowercase	mení veľké a malé písmená
`text-align`	left | right | center | justify	(vodorovné) zarovnanie textu v rámci blokového elementu

Viac o farbách

Jednou z vlastností textu je jeho farba. Zatiaľ sme hodnotu farby definovali len pomocou názvu farby (napr. `black`, `green` a pod.) V HTML a CSS môžeme zapísať hodnotu farby viacerými spôsobmi. Najčastejšie používané uvádzame v *tabuľke 7.2*.

Tabuľka 7.2 Spôsoby zápisu farieb.

spôsob	zápis	príklad
meno_farby	meno/názov farby podľa štandardu	`black`, `cadetblue`

#RRGGBB	zložky r,g,b vyjadrené v hexadecimálnom tvare
	ak sú dvojice zložiek rovnaké, môžeme použiť skrátený zápis #RGB
		`#FAC305`
		`#FFCC00`
		`#FC0`

rgb(r, g, b)	zložky r,g,b vyjadrené ako čísla (0 – 255) alebo percentá (0% – 100%)	`rgb(255,104,0)` `rgb(100%, 41%, 0%)`
rgb(r, g, b, a)	zložky r, g, b ako čísla (0 – 255) alebo percentá (0% – 100%) s priesvitnosťou (0.0 – 1.0)	`rgb(255, 104, 0, 0.5)` `rgb(100%, 41%,0%, 50%)`

ÚLOHA 7.10

Na stránke IT Pizza (súbor `07/index.html`) doplňte potrebné štýly, prípadne iba vlastnosti tak, aby:

- odkazy na stránke boli sivé (`#999999`), tučné a nepodčiarknuté,
- nadpis úrovne 1 bol 3.5krát väčší ako bežný text, veľkými písmenami.

Obrázok 7.2 Nadpis a odkazy na stránke IT Pizza.

Pseudotrieda :hover

Určite ste si na mnohých stránkach všimli, že niektoré prvky v nich menia svoj vzhľad pri istých udalostiach, napr. keď na ne prídeme myšou alebo keď na ne klikneme.

ÚLOHA 7.11

Uveďte čo najviac príkladov, ako sa na stránkach zmení vzhľad odkazov, keď na ne prídeme myšou.

Asi najčastejšia zmena, ktorá nastáva v prípade odkazov, nad ktorými je myš, je zmena farby textu, zmena farby pozadia či podčiarknutie, prípadne ich kombinácie. Zmenu vzhľadu odkazov pri nadídení myši vieme zadefinovať v rámci kaskádových štýlov pomocou tzv. pseudotriedy `:hover`.

PRÍKLAD 7.12

Do HTML okna v editore JSFiddle nakopírujme obsah súboru `07/vitaminy.html` a do CSS okna obsah súboru `07/vitaminy.css`. Všetkým odkazom, ak ponad ne nadídeme myšou, zmeníme farbu na červenú. Do okna CSS pridáme nasledujúci kód a vyskúšame.

```
...
a {
  font-weight: bold;
  color: blue;
  text-decoration: none;
}
a:hover {
  color: red;
}
```

ÚLOHA 7.13

Urobte nasledujúce zmeny. Zakaždým stránku vyskúšajte.

- V definícii štýlu `a:hover` nahraďte farbu `red` inou farbou.
- V definícii `a:hover` pridajte podčiarkovanie textu.
- V definícii `a:hover` nastavte farbu textu na bielu a farbu pozadia (`background-color`) na modrú.
- V definícii `a:hover` nastavte veľkosť písma na 110%. Čo táto zmena spôsobí na stránke?

Pseudotriedy slúžia na definovanie „špeciálneho“ stavu elementu na stránke za istých okolností. Jednou z takýchto okolností je práve nadídenie myšou ponad element. Definovanie vzhľadu elementu, nad ktorým je myš, umožňuje pseudotrieda `:hover`. My sme ju v *príklade 7.12* definovali pre element `a`, čím sme nastavili vzhľad odkazov, nad ktorými je myš. Pseudotriedu `:hover` však môžeme používať s ľubovoľným elementom.

PRÍKLAD 7.14

Pri prechode myšou ponad nadpis úrovne 1 zmeníme písmo na kapitálky.

```
h1 {
  font-family: Verdana;
}
h1:hover {
  font-variant: small-caps;
}
```

Aj keď pseudotriedu `:hover` môžeme používať s ľubovoľným elementom, treba vždy zvážiť, či jej použitie bude mať praktický význam. Napr. naša posledná zmena z *príkladu 7.14* žiaden praktický význam nemá. Tiež treba dobre zvážiť, ktoré vlastnosti elementu budeme v pseudotriede `:hover` meniť. Rôzne „blikajúce efekty” či efekty „poskakujúceho textu” (ako ste videli v poslednej z úloh *7.13*) môžu na čitateľa pôsobiť skôr rušivo. Mali by sme robiť len také zmeny, ktoré nezmenia rozloženie prvkov na stránke.

ÚLOHA 7.15

Na stránke IT Pizza nastavte farbu pozadia odkazov, nad ktorými je myš, na `#ff9900`.

ÚLOHA 7.16

Na stránke `07/ucebnica.html` definujte štýly pre elementy `body`, `h1`, `h2`, `h3`, `h4`, `p`, `a`, `a:hover`. Vlastnosti a ich hodnoty si zvoľte podľa vlastného uváženia.

TVORBA WEBOVÝCH STRÁNOK

Obsah: **Úvod Štruktúrovanie textu na webovej stránke Úvod do kaskádových štýlov Obrázky Odkazy**

1. ÚVOD

S webovými stránkami (alebo HTML stránkami) sa dnes stretávame denne: hľadáme na nich informácie, komunikujeme s kamarátmi, nakupujeme v online obchodoch, počúvame hudbu, ... Webové stránky majú školy, mestá, osoby, banky, obchody ...

Webová stránka je kombinácia textu, obrázkov, animácií, hypertextových odkazov a ďalších objektov, ktoré sú webové prehliadače schopné zobraziť. Vytvoriť webovú stránku znamená vytvoriť jej obsah („vložiť " texty, obrázky, tabuľky, odkazy a ďalšie spomínané objekty) a rozhodnúť, akým spôsobom sa bude tento obsah zobrazovať, t.j. ako budú jednotlivé objekty na stránke rozložené, aké budú mať vlastnosti, napr. aké farby použijeme, aký druh, štýl či veľkosť písma, riadkovanie, medzery medzi jednotlivými prvkami na stránke, atď.

1.1 Zdrojový kód stránky, jazyk HTML

Webová stránka je v skutočnosti obyčajný textový súbor, v ktorom je pomocou špeciálnych značiek popísané, čo sa má na webovej stránke zobraziť a ako sa to má zobraziť – tzv. zdrojový kód stránky.

Obrázok 7.3 Ukážka stránky ucebnica.html s nastavenými štýlmi.

Metodické pokyny pre učiteľa

CIEĽ

Cieľom je

- spoznať niektoré vlastnosti textu a experimentovať s nimi
- oboznámiť sa, akým spôsobom sa v kaskádových štýloch zadávajú jednotky veľkosti a preskúmať, aký je medzi nimi rozdiel
- spoznať spôsoby definovania farieb v CSS

MOTIVÁCIA

Experimentovanie so vzhľadom stránky.

VÝKLAD

Nie je cieľom naučiť sa jednotlivé vlastnosti textu a písma a ich možné hodnoty naspamäť. V učebnici neuvádzame kompletný zoznam vlastností textu a písma, dokonca ani nevysvetľujeme význam všetkých tých, ktoré spomíname. Myslíme si, že najlepším spôsobom, ako spoznať tieto vlastnosti, je experimentovanie s jednotlivými vlastnosťami a ich hodnotami.

K jednotkám (napr pri veľkosti písma): Ak je hodnota nula (0), jednotku nemusíme uvádzať.

K časti Viac o farbách:

- Predpokladáme, že žiaci poznajú RGB farebný model, teda že farbu môžeme zložiť z troch zložiek: červenej, zelenej a modrej.
- Farbu môžeme zadať aj pomocou HSL farebného modelu, a to v tvare `hsl(h, s, l)` alebo `hsla(h, s, l, a)` – farba (prípadne s priesvitnosťou a) v tvare odtieň, sýtosť, svetlosť.

Rušenie vlastností: S rušením vlastností by sme mali byť opatrní. Vlastnosti totiž odlišujú jednotlivé elementy od iných elementov. Ak bol nejaký element istým spôsobom zvýraznený (napr. odkazy majú prednastavené, že sú tučné a podčiarknuté) a my to zrušíme, mali by sme pouvažovať nad iným spôsobom zvýraznenia, aby sa element nestal neidentifikovateľným (napr. aby odkazy nesplynuli s ostatným textom).

K úlohe 7.8: Pri absolútnom nastavení, nech akokoľvek zmeníme veľkosť písma v elemente `body`, veľkosť písma v ostatných elementoch zostane zachovaná. Pri relatívnom nastavení, ak zmeníme veľkosť písma v elemente `body`, zmení sa aj veľkosť písma v ostatných elementoch (lebo v našom konkrétnom príklade je element `body` nadradený elementu `p` a ten je zasa nadradený elementu `strong`).

K úlohe 7.13, k poslednému bodu: Žiaci by mali prísť na to, že to "poskakuje". Učiteľ so študentmi diskutuje, ako sa im to páči, či to nepôsobí rušivo, pre aké nastavenia by sa rozhodli oni.

K definícii pseudotried: Pseudotriedy definujeme nasledujúcim spôsobom:

```
element:názov_pseudotriedy {
  vlastnost1: hodnota1;
  vlastnost2: hodnota2;
  ...
}
```

- Názov pseudotriedy vždy uvádzame s dvojbodkou.
- Pseudotriedy nedefinujeme samostatne, ale s nejakým elementom, napr. `a:hover`. Vlastnosti pre element bez pseudotriedy sa zachovajú.

Do učebnice sme tento text nezaradili, lebo v tejto časti žiaci uvidia len jedinú pseudotriedu.

K úlohe 7.16: Je to úloha určená na opakovanie. Štýly zámerne nie sú predpísané. Stránka v ukážke bola oštýlovaná takto:

- všetok text na stránke (`body`) je písmom Arial, veľkosťou 12px,
- nadpisy `h1` majú veľkosť 2.5em,
- nadpisy `h1` a `h2` majú čierne pozadie, biely text a sú písané kapitálkami,
- nadpisy `h3` a `h4` sú šikmým písmom,
- odkazy sú tučné, nepodčiarknuté, zelenej farby a keď nad ne nadídeme myšou, budú podčiarknuté a tmavozelené,
- text v odsekoch je zarovnaný na obe strany.

Nadväznosť na predchádzajúce kapitoly: Vyžaduje všetky predchádzajúce kapitoly.

8 BLOKOVÉ PRVKY NA WEBOVEJ STRÁNKE A ICH ŠTÝLOVANIE

V tejto kapitole sa oboznámime s niekoľkými blokovými elementami na (logické) členenie textu a ukážeme si tri dôležité vlastnosti blokových elementov, a to orámovanie, vonkajšie a vnútorné okraje.

Blokové elementy

V kapitole 2 sme sa naučili, ako môžeme štruktúrovať text pomocou blokových elementov definujúcich nadpisy (`<h1>`, .., `<h6>`) a odsek (`<p>`). V kapitole 6 sme spoznali elementy `<section>` a `<nav>`. V tejto časti si predstavíme ďalšie blokové elementy, ktoré nám umožňujú rozčleniť dokument na väčšie či menšie ucelené časti, a to elementy `<header>`, `<footer>`, `<article>`, `<aside>` a `<div>`. Zoznam všetkých blokových elementov nájdete na https://www.w3schools.com/html/html_blocks.asp.

Element `<header>` používame na definovanie hlavičky dokumentu, teda akéhosi úvodného obsahu, resp. obsahu, ktorý sa opakuje v úvode každej stránky webového sídla. Zvyčajne obsahuje jeden alebo viac nadpisov, logo, prípadne nejakú navigáciu. Nemýľme si element `<header>` s elementom `<head>`. Element `<head>` definuje hlavičku po „technickej stránke", v rámci neho definujeme napr. titulok stránky, či kódovanie (pozri *Kapitolu 1*), uvádza sa **pred** elementom `<body>`. Elementom `<header>` definujeme hlavičku obsahovej časti webového dokumentu, teda **je vnorený** v elemente `<body>`.

Elementom `<footer>` definujeme pätu webového dokumentu. V päte zväčša uvádzame informácie o autoroch a vlastníckych právach, kontaktné informácie prípadne aj s mapami, mapu sídla, odkaz na úvod stránky. Podobne ako hlavička, aj päta zvykne byť spoločná pre všetky stránky webového sídla.

PRÍKLAD 8.1

Na stránke IT Pizza (súbor `08/index.html`.) definujeme elementy `<header>` a `<footer>`. Do hlavičky umiestnime hlavný nadpis (`h1`) a logo stránky (obrázok `logo.gif`). Do päty zahrnieme všetko od kontaktov na obe prevádzky až po koniec (odkazy na Facebook, Twitter a ©).

```
<body>
  <header>
    <h1>IT Pizza</h1>
    <img src="obrazky/logo.gif" alt="logo" width="167" height="70">
  </header>
  ...
  <footer id="kontakt">
    <h2>Kontakt</h2>
    <h3>Bratislava</h3>
    <address>Mlynská dolina, 842 48 Bratislava<br>
      tel. +421 999 123 456<br> email: ba@itpizza.sk </address>
    <h3>Banská Bystrica</h3>
    <address>Tajovského 40, 974 01 Banská Bystrica<br>
      tel. +421 999 123 457<br> email: bb@itpizza.sk </address>
    <p>Sledujte nás na
      <a href="https://www.facebook.com/" title="Facebook" ...
```

```
      <a href="https://twitter.com/" title="Twitter" ...
    </p>
    <p>&copy; IT akadémia, 2018, Mlynská dolina, 842 48
      Bratislava</p>
  </footer>
</body>
```

Element `<nav>` definuje množinu odkazov, ktoré tvoria tzv. (hlavnú) navigáciu stránky. Uvedomte si, že nie všetky odkazy na stránke, musia byť súčasťou navigácie.

Hlavný obsah stránky (zvyčajne medzi hlavičkou a pätou) môžeme ďalej členiť pomocou blokových elementov: `<section>`, `<article>`, `<aside>`. Medzi elementami `<section>` a `<article>` v podstate nie je žiaden rozdiel, obidvoma môžeme označiť nejakú časť dokumentu. My budeme používať `<section>` na definovanie väčších, obsahovo ucelených, častí (napr. kapitoly) a `<article>` na definovanie ich podčastí, menších celkov (napr. podkapitoly). Element `<aside>` sa používa na definovanie vedľajšieho obsahu (poznámka, vysvetlivka, odbočenie, teda niečo, čo nemusí byť priamou súčasťou hlavného obsahu, ale s týmto obsahom nejako súvisí).

ÚLOHA 8.2

Na stránke IT Pizza v časti *Ponuka* použite element `<article>` na definovanie informácií pre každú pizzu, t.j. nahraďte všetky elementy `<li>` v tejto časti elementom `<article>`. Nezabudnite zrušiť element `<ul>`, a tiež príslušný štýl pre `<ul>`.

Tiež nahraďte zvýraznenie pizze pozmocou elementu `<strong>` využitím nadpisu `<h3>` a zrušte element `<br>` za názvom pizze. Všetky nahradenia možno vidieť na *obrázku 8.1*. Odporúčame využívať automatické nahradenie textu.

```
<li><strong>Margherita</strong><br>
  paradajková omáčka, syr <sub>1, 7</sub><br>
  <img src="obrazky/margerita.jpg" alt="pizza Margherita" ...
  malá 3,00 &euro;<br>
  veľká 4,50 &euro;
</li>
```

↓

```
<article><h3>Margherita</h3>
  paradajková omáčka, syr <sub>1, 7</sub><br>
  <img src="obrazky/margerita.jpg" alt="pizza Margherita" ...
  malá 3,00 &euro;<br>
  veľká 4,50 &euro;
</article>
```

Obrázok 8.1 Nahradenie elementu <li> elementom <article> na stránke IT Pizza.

Časť stránky o aktuálne prebiehajúcich akciách definujte pomocou elementu `<aside>` (príslušné `<section>` nahraďte `<aside>`).

ODPOVEDZTE

Mali zmeny, ktoré sme spravili v *príklade 8.1* a *úlohe 8.2*, vplyv na vzhľad stránky IT Pizza?

Okrem už uvedených blokových elementov sa veľmi často pri tvorbe stránok používa element `<div>`. Umožňuje nám deliť stránku na logické celky bez toho, aby sme špecifikovali obsahový význam týchto celkov (ako to robia napr. elementy `header` alebo `footer`). Podľa w3schools `div` sa veľmi často používa ako kontajner pre iné HTML elementy v spojení s CSS na definovanie vzhľadu časti webového dokumentu alebo v spojení s JavaScriptom na definovanie nejakého správania. Elementy `<div>` môžeme do seba vnárať do ľubovoľnej hĺbky.

PRÍKLAD 8.3

Na stránke IT Pizza dáme do hlavičky stránky (`<header>`) na pozadie obrázok zo súboru `header-bg.jpg`. Do štýlov pridáme štýl pre `header`.

```
<style>
  ...
  header {
    background-image: url(obrazky/header-bg.jpg);
  }
</style>
```

ÚLOHA 8.4

Na stránke IT Pizza definujte vhodné štýly tak, aby:

- navigácia mala farbu pozadia `#FFE4C4`,
- päta stránky mala farbu pozadia `#FF9900` a farbu textu čiernu,
- element `<aside>` mal text zarovnaný na stred a pozadie farby `#F5F6F7`,
- nadpis `<h1>` nemal žiadne pozadie.

Rôzne štýly pre rovnaký typ elementu

Štýly, ktoré sme definovali doteraz, sa vždy aplikovali na všetky výskyty elementu, pre ktorý boli zadefinované. Napr. ak sme definovali štýl `h2 {font-family: Verdana}`, nastavili sme tým typ písma pre všetky nadpisy úrovne 2 na stránke. Niekedy by sme však potrebovali mať rôzne štýly pre ten istý druh elementu, napr. jeden štýl pre odkazy v navigácii a iný štýl pre ostatné odkazy.

PRÍKLAD 8.5

V kóde na *obrázku 8.2* máme tri výskyty elementu `h1`, z toho jeden je v hlavičke. Chceli by sme vizuálne odlíšiť nadpis `h1`, ktorý je v hlavičke od ostatných nadpisov `h1` (teda tých, ktoré nie sú v hlavičke).

```html
<body>
  <header><h1>Vitamíny</h1></header>

  <section id="tuky">
    <h1>Vitamíny rozpustné v tukoch</h1>
    <h2>Vitamín A</h2>
    <p>Hlavným zdrojom je plnotučné mlieko, <a href="https://sk.wikipedia.org/wiki/Vajce">
vajcia</a> a pečeň.</p>
    <h2>Vitamín D</h2>
    <p>Za normálnych okolností sa vitamín D tvorí v koži pôsobením slnečného žiarenia.</p>
  </section>

  <section id="voda">
    <h1>Vitamíny rozpustné vo vode</h1>
    <h2>Vitamín B1 - Tiamín</h2>
    <p>Hlavným zdrojom je droždie, <a href="https://sk.wikipedia.org/wiki/Obilie">obilné
klíčky</a> a syr.</p>
    <h2>Vitamín C</h2>
    <p>Jeho hlavným zdrojom sú ovocie a zelenina.</p>
  </section>

  <footer>Zdroj: <a href="https://sk.wikipedia.org/wiki/Vitam%C3%ADn">Wikipédia</a></footer>
</body>
```

Obrázok 8.2 Kód s viacerými výskytmi elementu h1.

V editore JSFiddle pomocou štýlov zmeníme zobrazenie stránky s kódom z *obrázku 8.2* (kód nájdete v súbore `08/vitaminy.html`, skopírujte ho do časti HTML). V časti CSS definujeme potrebné štýly tak, aby:

- **všetky nadpisy** úrovne 1 boli písmom `Verdana`,
- nadpisy úrovne 1, ktoré sú v hlavičke stránky (element `<header>`) mali pozadie farby `lightgreen` (*obrázok 8.3*).

```css
h1 {font-family: Verdana;}
header h1 {
  background-color: lightgreen;
}
```

Vitamíny rozpustné v tukoch

Obrázok 8.3 Rôzne oštýlované nadpisy h1.

Zápisom `header h1` definujeme štýl len pre tie elementy `h1`, ktoré sú vnorené v elemente `<header>` (elementy `h1`, ktoré nie sú v `header`, sa nezmenia). Vlastnosti, ktoré definujeme v štýle `h1`, sa aplikujú na všetky výskyty elementu `h1`, vrátane tých, ktoré sú v hlavičke.

ÚLOHA 8.6

V kóde na *obrázku 8.2* máme tri odkazy (elementy `a`): jeden z nich je v päte (`footer`) a dva v sekciách. Vizuálne rozlíšte odkazy, ktoré sú v päte od odkazov, ktoré sú mimo nej. Odkazy v päte nech sú farby `#990033`, ostatné odkazy nech sú farby `darkblue`.

Ak chceme definovať **štýl len pre jeden konkrétny výskyt elementu**, môžeme využiť identifikátor elementu.

PRÍKLAD 8.7

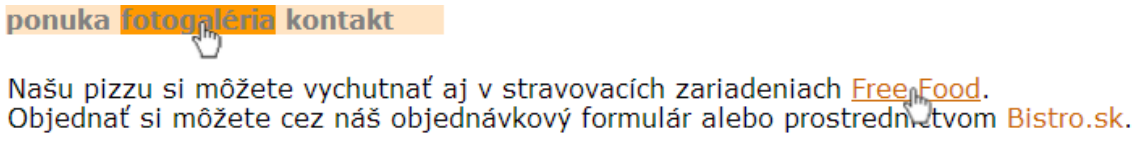
V html dokumente z predchádzajúceho príkladu máme dve sekcie, každej sme dali identifikátor. Sekcii s identifikátorom `voda` dáme svetlomodré pozadie (`#CCFFFF`), sekcii s identifikátorom tuky svetložlté pozadie (`#FFF5CC`).

```
#voda {
  background-color: #CCFFFF;
}
#tuky {
  background-color: #FFF5CC;
}
```

ÚLOHA 8.8

Na stránke IT Pizza upravte použitím štýlov:

- odkazy v navigácii: farba `#333333`, tučné, nepodčiarknuté a pri nadídení myšou sa farba pozadia zmení na `#FF9900`,
- ostatné odkazy: farba `#CC6600`, nepodčiarknuté, pri nadídení myšou sa podčiarknu.



Obrázok 8.4 Odkazy v navigácii (hore) a ostatné odkazy (dolu) na stránke IT Pizza.

Zopakujme si, akými spôsobmi už vieme definovať štýly:

- `p {...}` - tento štýl definuje vlastnosti pre všetky výskyty elementu `p` na stránke,
- `header h1 {...}` - tento štýl definuje vlastnosti pre tie výskyty elementu `h1`, ktoré sú vnorené v elemente `header`,
- `#id {...}` - tento štýl definuje vlastnosti elementu s daným `id`.

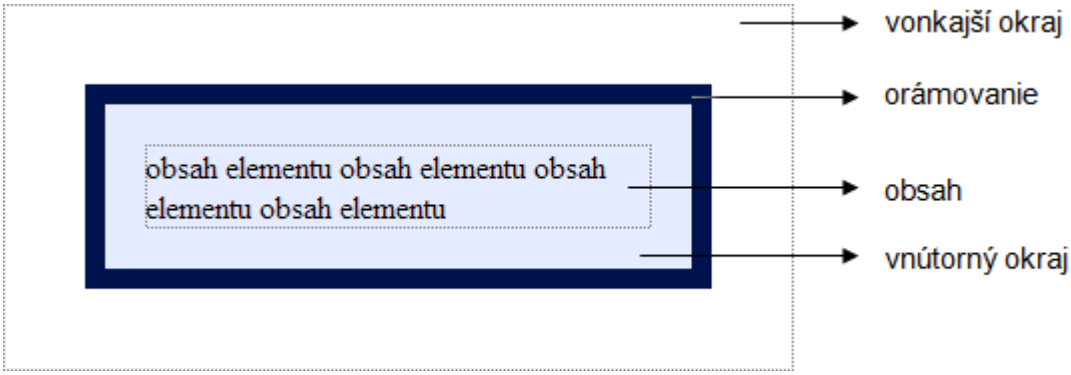
CSS – okraje elementu

Pre každý element sa automaticky generuje tzv. **box**. Box sa skladá z **obsahu**, **vnútorného okraja**, **orámovania** a **vonkajšieho okraja** (Obrázok 8.5).

Orámovanie elementu je tmavý rámik na *obrázku 8.5*. Orámovanie nemusí byť vždy viditeľné.

Priestor medzi obsahom elementu a jeho orámovaním nazývame **vnútorný okraj**.

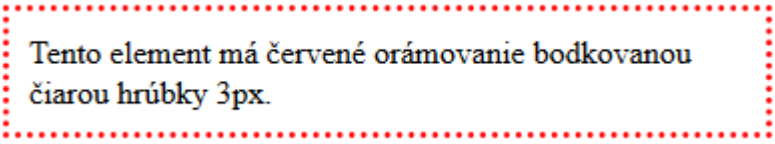
Priestor medzi orámovaním elementu a okolitými elementmi nazývame **vonkajší okraj**. Na obrázku je to vzdialenosť medzi orámovaním elementu (modrý rámik) a bodkovaným rámikom (ľubovoľný element okolo).



Obrázok 8.5 Box model (sivé bodkované čiary sú len pomocné).

Orámovanie

Pomocou vlastností `border-color`, `border-width` a `border-style` môžeme nastaviť farbu, hrúbku a štýl orámovania elementu.

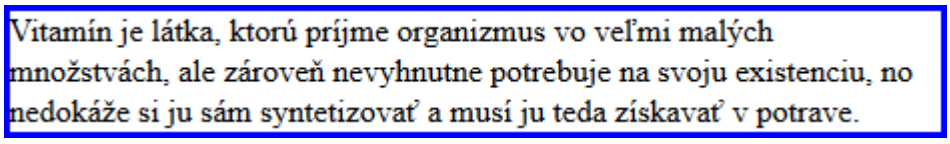


Obrázok 8.6 Orámovanie elementu.

PRÍKLAD 8.9

V editore JSFiddle definujeme v časti HTML jeden odsek (element `<p>`) s ľubovoľným textom (napr. zo súboru `08/vitamin.txt`). Pomocou kaskádových štýlov orámujeme tento odsek modrou súvislou čiarou hrúbky `3px` (*obrázok 8.7*).

```
p {
  border-width: 3px;
  border-style: solid;
  border-color: #0000FF;
 }
```



Obrázok 8.7 Orámovanie elementu.

ZAPAMÄTAJTE SI

Orámovanie elementu sa zobrazí len vtedy, ak sú nastavené farba, hrúbka aj štýl.

Okrem farby, hrúbky a štýlu vieme orámovaniu nastaviť aj mieru zaoblenia rohov. Význam všetkých vlastností orámovania a ich možné hodnoty uvádzame v *tabuľke 8.1*.

Tabuľka 8.1 Vlastnosti orámovania.

vlastnosť	hodnoty	popis, význam
`border-color`	*názov farby alebo RGB farby*	definuje farbu orámovania (pre všetky štyri okraje)
`border-style`	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset	definuje štýl orámovania (pre všetky štyri okraje)
`border-width`	*hrúbka orámovania v %, px, em* thin | medium | thick	definuje hrúbku orámovania (pre všetky štyri okraje)
`border-radius`	polomer zaoblenia v *px* alebo *%* (*alebo ľubovoľná jednotka*)	definuje polomer zaoblenia rohov

ÚLOHA 8.10

V kóde z *príkladu 8.9* experimentujte s vlastnosťou `border`:

- nastavte hodnotu `border-width` postupne na `1px`, `5px`, `10px`, `40px`, `1%`, `1em`, `1.5em`,
- vyskúšajte rôzne hodnoty pre `border-style`, napr. `dotted`, `dashed`, `double`, `groove`, `ridge`, `inset`, `outset`,
- zmeňte `border-color`.

Štýl pre element `p` z *príkladu 8.9* môžeme definovať aj pomocou združenej vlastnosti `border`.

```
p {
  border: 3px solid #0000FF;
}
```

Pri použití združenej vlastnosti musíme hodnoty zadať v poradí: *hrúbka štýl farba* (oddelené medzerou).

V prípade, že použijeme vlastnosť `border`, nastavujeme vlastnosti orámovania na štyroch stranách: *hore*, *vpravo*, *dolu* a *vľavo*. Existujú však aj vlastnosti `border-top`, `border-bottom`, `border-left` a `border-right`. Pomocou týchto vlastností môžeme samostatne definovať farbu, hrúbku a štýl orámovania pre hornú, dolnú, ľavú a pravú stranu elementu.

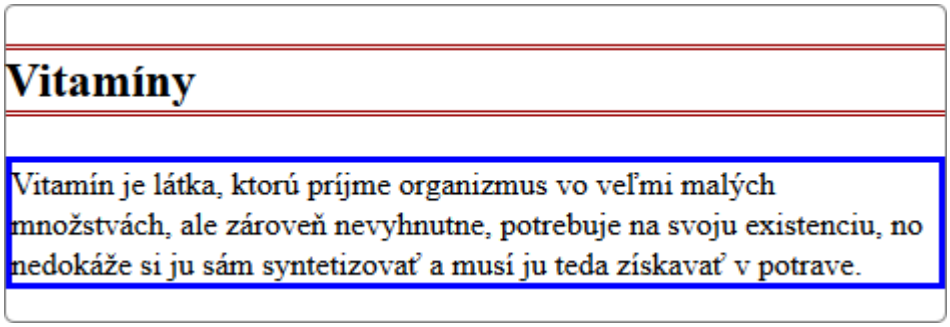
PRÍKLAD 8.11

Do HTML kódu z *príkladu 8.9* doplníme jeden nadpis úrovne 1 a nadpis aj odsek ešte vnoríme do elementu `section` (aktualizovaný HTML kód skopírujeme zo súboru `08/vitamin2.txt`). V CSS nastavíme:

- pre nadpis dolné a horné orámovanie hrúbky `3px`, dvojité, farby `#990000`,
- pre `section` tenké sivé súvislé orámovanie na všetkých štyroch stranách s mierne oblými rohmi (pozri obrázok 8.8).

```
h1 {
  border-bottom: 3px double #990000;
  border-top: 3px double #990000;
}
section {
  border: thin solid gray;
```

```
  border-radius: 5px;
}
```



Obrázok 8.8 Rôzne typy orámovania elementov.

ÚLOHA 8.12

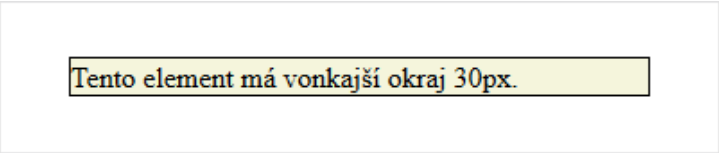
Na stránke IT Pizza (pozri obrázok 8.9) definujte pomocou štýlov:

- orámovanie okolo časti akcia (elementu `aside`) – orámovanie by malo byť zľava, sprava a zdola, rovnakej farby, ako majú nadpisy `h2` – hrúbku a štýl orámovania si zvoľte,
- orámovanie okolo informácií o každej ponúkanej pizze (t.j. okolo elementu `article`, ktorý je v `section`) – zvoľte tenké súvislé orámovanie sivej farby,
- zmenu farby pozadia bloku informácií o jednej pizze (`article` v `section`) pri nadídení myšou – farba sa zmení na svetlosivú,
- orámovanie okolo obrázkov vo fotogalérii (nie iných!) – vzhľad orámovania si zvoľte.

Obrázok 8.9 Orámovanie na stránke IT Pizza (z úlohy 8.12).

Vonkajšie okraje

Vlastnosť `margin` určuje vzdialenosť orámovania elementu od okolitých elementov na stránke vo všetkých smeroch, tzv. vonkajší okraj elementu. V prípade, že element má definovanú farbu pozadia, tak voľný priestor definovaný vlastnosťou `margin` nebude podfarbený touto farbou.

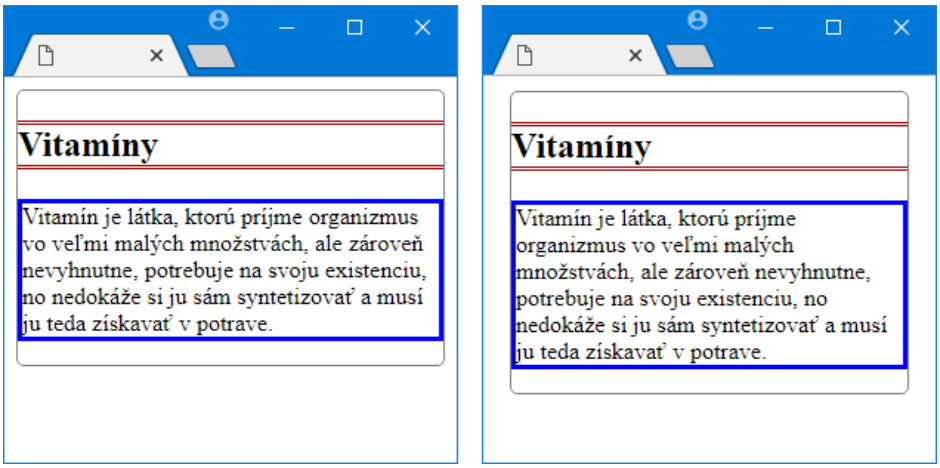


Obrázok 8.10 Vonkajší okraj elementu.

PRÍKLAD 8.13

Elementu `section` z *príkladu 8.11* nastavíme vonkajší okraj na `10px`. Rozdiel medzi `section` s nenastaveným vonkajším okrajom a s nastaveným vonkajším okrajom je na *obrázku 8.11*.

```
section {
  border: thin solid gray;
  border-radius: 5px;
  margin:10px;
}
```



Obrázok 8.11 Vľavo bez vonkajšieho okraja, vpravo s vonkajším okrajom 10px pre section.

ÚLOHA 8.14

V kóde z príkladu 8.13:

- meňte hodnoty `margin` postupne na `20px`, `30px`, `40px`, `60px`,
- meňte hodnoty `margin` postupne na `10%`, `5%`, `2%`,
- meňte hodnoty `margin` postupne na `1em`, `2em`, `2.5em`,
- vyskúšajte, čo sa stane, ak použijete záporné číslo (napr. `-10px`),
- pridajte `margin` pre element `h1`, skúšajte rôzne hodnoty,
- pridajte `margin` pre element `p`, skúšajte rôzne hodnoty.

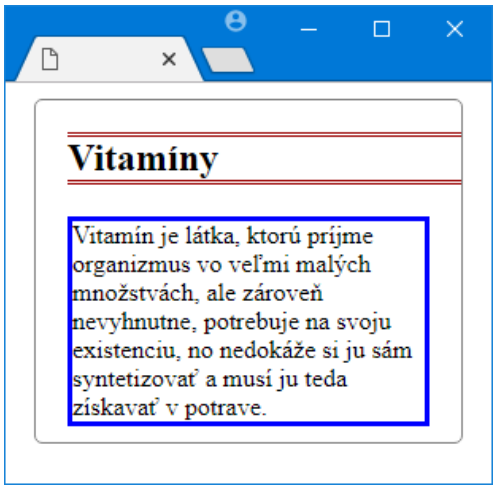
Podobne ako pri orámovaní, aj pri vlastnosti `margin` máme možnosť samostatne nastaviť horný, pravý, dolný, či ľavý vonkajší okraj elementu. Môžeme to spraviť pomocou samostatných vlastností: `margin-top`, `margin-right`, `margin-bottom`, `margin-left`.

Na nastavenie rôznych hodnôt vonkajších okrajov na jednotlivých stranách môžeme tiež použiť združenú vlastnosť `margin` tak, že namiesto jedného čísla zadáme štvoricu čísel, ktoré určujú horný, pravý, dolný a ľavý okraj (v tomto poradí). Hodnoty oddeľujeme medzerou.

PRÍKLAD 8.15

V JSFiddle doplníme kód z *príkladu 8.13*. Odseku nastavíme ľavý a pravý vonkajší okraj na `20px`, spodný vonkajší okraj na `10px`. Nadpisu nastavíme ľavý vonkajší okraj na `20px`.

```
p {
  border: 3px solid #0000FF;
  margin: 0px 20px 10px 20px;
}
h1 {
  border-bottom: 3px double #990000;
  border-top: 3px double #990000;
  margin-left: 20px;
}
```



Obrázok 8.12 Nastavenie okrajov.

ÚLOHA 8.16

Experimentujte s vlastnosťou `margin`.

- Nastavte elementu `h1` horný vonkajší okraj na 0.
- Ktoré vonkajšie okraje a na akú hodnotu treba v našom príklade nastaviť elementom `h1` a `p`, aby boli tieto dva elementy nalepené na seba?
- Pozrite si vaše pokusy aj bez nastavenia orámovania (stačí, ak vlastnosť `border` v definícii štýlu zakomentujete pomocou `/* …*/`, napr. `/*border: 3px solid #0000FF;*/`).
- Experimentovaním zistite, ktoré vonkajšie okraje majú prednastavené elementy `h1`, `p` a `section` (t.j. s hodnotou inou ako 0).

- Porozmýšľajte a experimentovaním overte, ktoré iné elementy majú prednastavené nejaké nenulové vonkajšie okraje.
- Experimentujte s vlastnosťou `margin` pre element `body`.

ÚLOHA 8.17

Na stránke IT Pizza využite vlastnosť `margin` na to, aby ste:

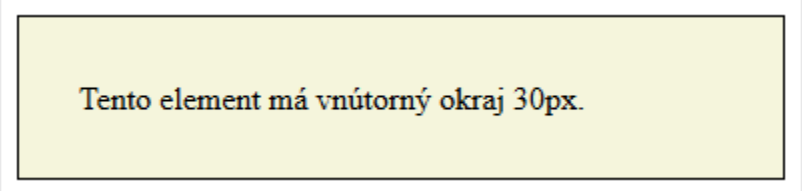
- prilepili celý obsah stránky k okrajom okna prehliadača,
- aby sa jednotlivé sekcie k sebe prilepili.

Pomôcka: Pouvažujte, ktorým elementom treba prestaviť vonkajšie okraje, či všetky alebo len na niektorej strane a na akú hodnotu. Zakaždým zvážte, či tak treba urobiť pre všetky výskyty daného elementu, alebo iba pre niektoré výskyty, ktoré sú napr. vnorené v inom elemente.

Obrázok 8.13 Využitie vlastnosti margin na stránke IT Pizza.

Vnútorné okraje

Vlastnosť `padding` určuje vzdialenosť obsahu elementu od jeho orámovania, tzv. vnútorný okraj elementu. V prípade, že element má definovanú farbu/obrázok pozadia, tak voľný priestor definovaný vlastnosťou `padding` bude tiež podfarbený.

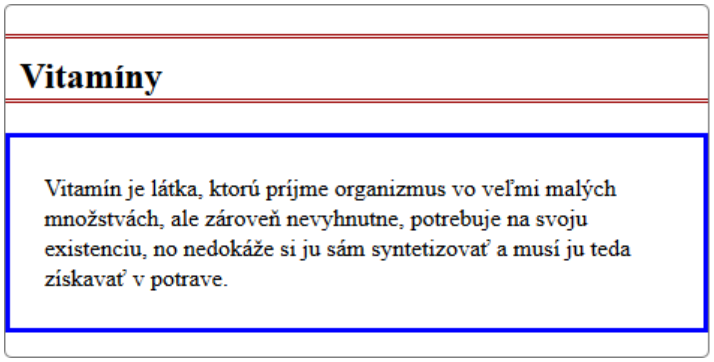


Obrázok 8.14 Vnútorný okraj.

PRÍKLAD 8.18

V editore JSFiddle pracujeme s HTML kódom z *príkladov 8.13, 8.15.* Z CSS kódu zrušme všetky nastavenia vlastnosti `margin`. Definujme elementu `<p>` vnútorný okraj na všetkých stranách na hodnotu `5%` a elementu `<h1>` ľavý a pravý vnútorný okraj na hodnotu `10px`.

```
p {
  border: 3px solid #0000FF;
  padding: 5%;
}
h1 {
  border-bottom: 3px double #990000;
  border-top: 3px double #990000;
  padding-left: 10px;
  padding-top: 10px;
}
```



Obrázok 8.15 Nastavenie vnútorných okrajov.

Pomocou vlastností `padding-top`, `padding-right`, `padding-bottom`, `padding-left` môžeme nastaviť vzdialenosť od horného, pravého, dolného či ľavého okraja elementu.

Vzdialenosti od všetkých štyroch strán môžeme tiež nastaviť pomocou združenej vlastnosti `padding`, ktorej hodnotou bude štvorica čísel oddelených medzerou v poradí: top right bottom left. Napr. `padding: 5px 10px 20px 10px` definuje vzdialenosť od horného okraja `5px`, vzdialenosť od ľavého a pravého okraja `10px` a vzdialenosť od dolného okraja `20px`. Ak zadáme iba jedno číslo, okraje budú rovnaké pre všetky štyri strany.

ÚLOHA 8.19

V CSS kóde z príkladu 8.18 experimentujte s vlastnosťou `padding`.

- Meňte hodnotu vnútorného okraja pre odsek. Skúste zmeniť jednotku `%` na `px`, či `em` a znova skúšajte rôzne hodnoty. Dá sa použiť aj záporná hodnota?
- Pre nadpis nastavte namiesto ľavého a horného vnútorného okraja pravý a dolný vnútorný okraj. Vyskúšajte rôzne hodnoty pre tieto dva okraje.
- Vyriešte predchádzajúcu úlohu pomocou združenej vlastnosti `padding` so štvoricou čísel.
- Pre element `<section>` nastavte nejaké jednoduché orámovanie a vonkajší okraj na `2%`. Pozrite si výsledok. Potom vonkajší okraj zmeňte na vnútorný a porovnajte.

ÚLOHA 8.20

Na stránke IT Pizza využite vlastnosť `padding` na odsadenie textu od okraja vo všetkých hlavných častiach stránky (hlavička, päta, jednotlivé sekcie), a taktiež v blokoch pre každú pizzu. Veľkosť vnútorného okraja si zvoľte. Pre navigáciu a odkazy v navigácii nastavte vnútorný okraj na všetkých stranách na `1em`. Zrušte oranžové pozadie nadpisov úrovne 2.

Obrázok 8.16 Využitie vlastnosti padding na stránke IT Pizza.

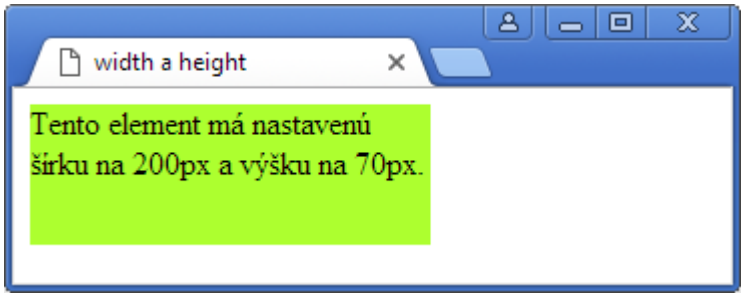
Šírka a výška elementu

Kaskádové štýly nám umožňujú nastaviť rozmery elementu. Slúžia na to vlastnosti `width` a `height`.

PRÍKLAD 8.21

V súbore `08/rozmery.html` máme definovaný jeden element `div`. Definujme mu farbu pozadia, šírku `200px` a výšku `70px`.

```
  ...
  <style>
    div {
      background-color: greenyellow;
      width: 200px;
      height: 70px;
    }
  </style>
</head>
```



Obrázok 8.17 Element s nastavenou šírkou a výškou v px.

Hodnoty vlastností `width` a `height` môžeme zadávať v dĺžkových jednotkách (napr. `px`, `em`) alebo v percentách (vzhľadom na nadradený element).

ÚLOHA 8.22

V kóde z *príkladu 8.21* experimentujte s hodnotami vlastností `width` a `height`:

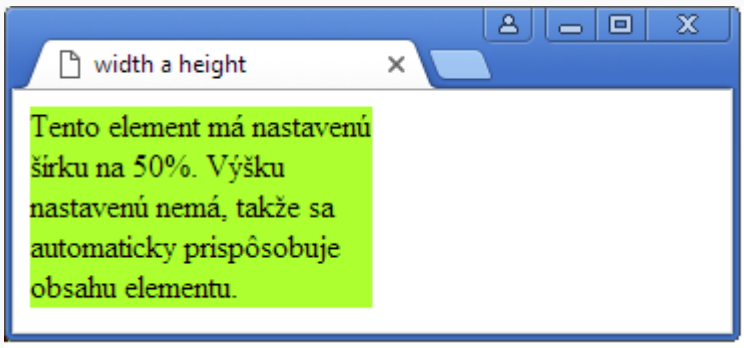
- Zmeňte šírku na `300px`, `400px`, `100px` a pozorujte. Čo sa stane, ak zvolíte príliš malú hodnotu?
- Nastavte šírku na `200px` a meňte len výšku – na `100px`, `30px`, `20px`.
- Úplne zrušte nastavenie výšky, ponechajte len nastavenie šírky. Akú výšku (tým nemyslíme číselné vyjadrenie) bude mať element `div` teraz?

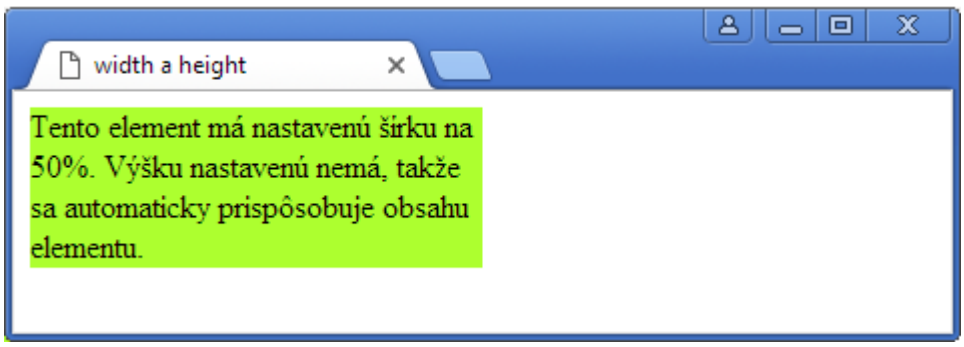
Štandardne majú vlastnosti `width` a `height` nastavenú hodnotu `auto`, čo znamená, že šírku a výšku elementu si vypočíta prehliadač, keď zobrazuje stránku.

PRÍKLAD 8.23

Pokračujeme v *príklade 8.21* (odporúčame pridať do elementu `<div>` viac textu). V CSS časti zmeníme elementu `div` šírku na `50%`.

```
div {
  background-color: greenyellow;
  width: 50%;
}
```





Obrázok 8.18 Element s nastavenou šírkou na 50% pri rôznych veľkostiach okna prehliadača.

ÚLOHA 8.24

V kóde z príkladu 8.23 experimentujte s nastavením šírky, resp. výšky elementu v `%`.

- Meňte šírku okna prehliadača a pozorujte, ako sa mení šírka elementu.
- Postupne meňte šírku elementu na `100%`, `80%`, `20%`, `5%` a pozorujte. Čo sa stane, keď zvolíte príliš malú hodnotu?
- Zopakujte predchádzajúce dva body pre výšku. Je správanie stránky rovnaké ako pri nastavení šírky?

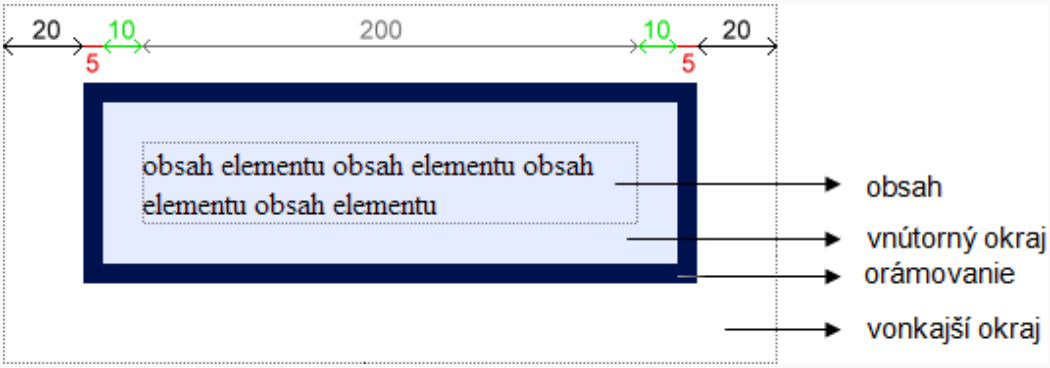
Vlastnosťami `width` a `height` nastavujeme šírku a výšku obsahu elementu, t. j. ich hodnoty nezahŕňajú veľkosť vonkajšieho okraja, orámovania a vnútorného okraja.

PRÍKLAD 8.25

Vypočítajme šírku, ktorú v prehliadači zaberá element `div` s nasledujúcimi nastaveniami:.

```
div {
  width: 200px;
  padding: 10px;
  border: 5px solid darkblue;
  margin: 20px;
}
```

Jednotlivé rozmery si znázorníme na obrázku (Obrázok 8.19). Celková šírka boxu, ktorý náš element `div` zaberá v prehliadači, bude: 20px (ľavý vonkajší okraj) + 5px (ľavé orámovanie) + 10px (ľavý vnútorný okraj) + 200px (šírka obsahu) + 10px (pravý vnútorný okraj) + 5px (pravé orámovanie) + 20px (pravý vonkajší okraj) = 270px.



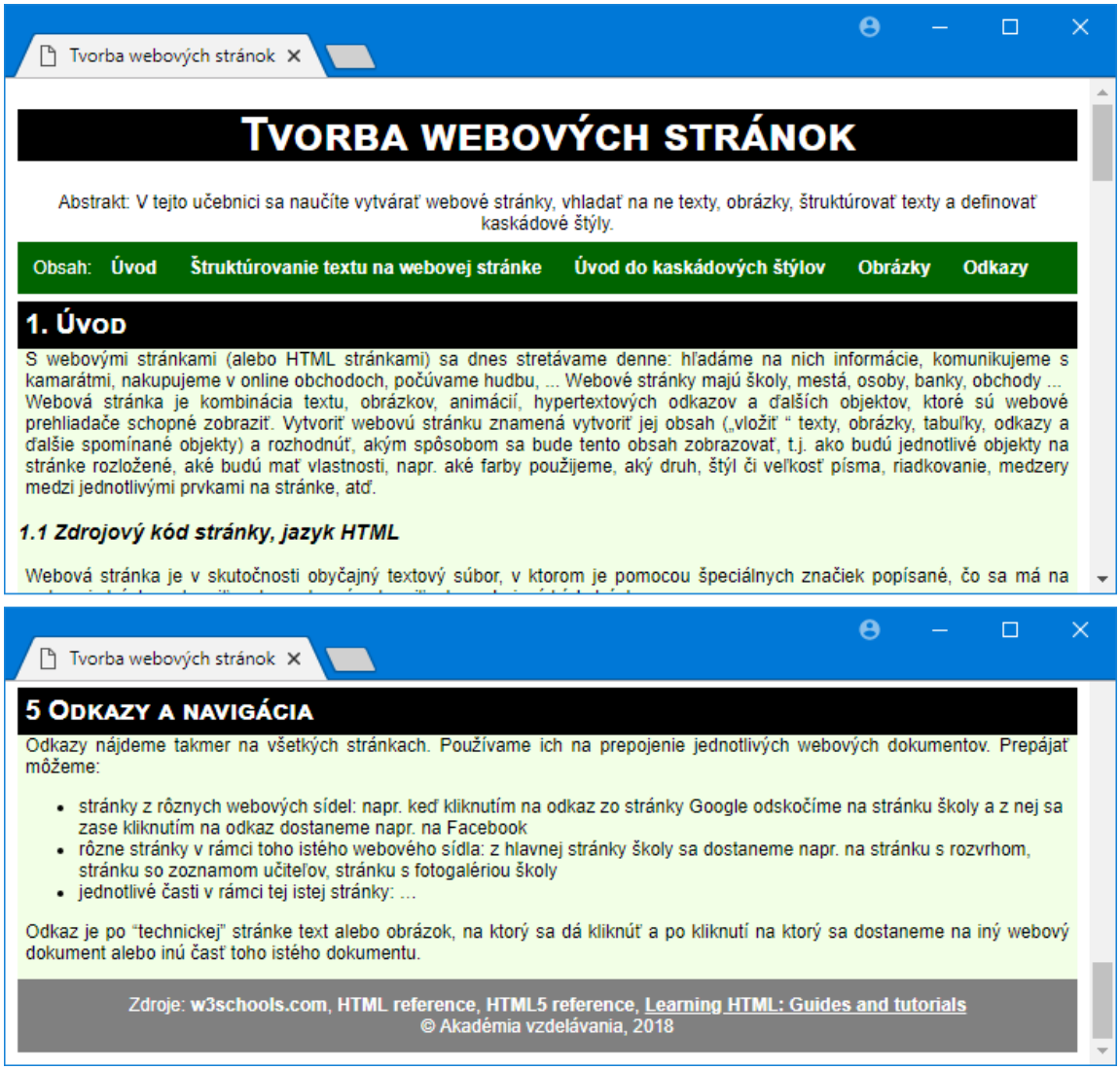
Obrázok 8.19 Výpočet šírky boxu, ktorý zaberá element v prehliadači.

Celková šírka boxu, ktorý element zaberá v prehliadači, sa vypočíta ako ľavý margin + ľavý border + ľavý padding + šírka obsahu + pravý padding + pravý border + pravý margin. Analogický výpočet platí pre výšku.

ÚLOHA 8.26

Upravte štruktúru webovej stránky `08/ucebnica.html` a oštýlujte ju:

- definujte na stránke časti `header` (nadpis a abstrakt), `nav` (odkazy na jednotlivé kapitoly), `footer` (odkazy na zdroje, autori),
- `header`, `nav` a `footer` nejako pekne oštýlujte (inšpirácia na *obrázku 8.20*), samostatne oštýlujte odkazy v navigácii, odkazy v päte a ostatné odkazy,
- zabezpečte, aby sekcie pre 1., 3. a 5. kapitolu mali pozadie inej farby (napr. `#F2FFE6`) ako sekcie pre 2. a 4. kapitolu,
- skúste vo vašich štýloch využiť aj niektoré z vlastností `border`, `padding`, `margin`, `width`.



Obrázok 8.20 Ukážka stránky ucebnica.html (hore hlavička a 1. kapitola, dolu 5. kapitola a päta).

Metodické pokyny pre učiteľa

CIEĽ

Cieľom je:

- oboznámiť sa s ďalšími blokovými elementami na logické členenie textu,
- experimentovať s vlasnosťami blokových elementov: border, padding a margin.

MOTIVÁCIA

Dizajnovanie stránky.

VÝKLAD

V úlohách so stránkou IT Pizza nie je dôležité, aby žiacke riešenia vyzerali rovnako, ako tie na obrázku. Dôležité je, aby to žiaci skúšali.

Elementy header a footer

V texte sme kvôli zjednodušeniu `header` a `footer` označili ako hlavičku a pätu celej stránky (dokumentu). Predpokladáme, že tieto pojmy žiaci poznajú z textového editora, resp. z vytvárania nejakých úradných dokumentov, ktoré majú hlavičku a pätu. Podľa definície zo štandardu, však `header` a `footer` definujú hlavičku a pätičku nielen celej stránky, ale väčšieho bloku. Môžeme ich teda na stránke použiť aj viackrát. `Header` a `footer` môže mať napr. každý `article`.

```
<body>
<header></header>
<section>
<article><header></header></article>
<article><header></header></article>
</section>
```

Príklad 8.10: Môžeme nechať žiakov experimentovať s dvojitým orámovaním rôznej hrúbky, aby sami prišli na to, pri akej minimálnej hrúbke sa prejaví to, že orámovanie bude dvojité.

Úloha 8.12: Posledný bod – treba definovať štýl pre `#galeria img`.

Vlastnosť margin:

Vlastnosť `margin` a všetky ostatné príbuzné vlastnosti môžeme aplikovať na ľubovoľný viditeľný element.

V prípade, že chceme nastaviť rovnaké medzery vo vodorovnom a zvislom smere, môžeme vlastnosti `margin` zadať dve hodnoty, napr. `margin: 2em 20px`. Táto vlastnosť nastaví

veľkosť vonkajšieho okraja vo zvislom smere na 2 znaky (2em) a vo vodorovnom smere to bude 20 bodov (20px).

Úloha 8.17: Na stránke IT Pizza využite vlastnosť margin na to, aby ste:

- prilepili celý obsah stránky k okrajom okna prehliadača:
  Riešenie:
  `body {margin: 0},`
  `header h1 { margin: 0},`
  `footer p {margin-bottom: 0}`
- aby sa jednotlivé sekcie k sebe prilepili
  Riešenie: Treba si uvedomiť, že `section`, `footer` aj `aside` majú nulový `margin`, ale v každej z nich je `h2` a ten margin má, čiže treba nastaviť `margin-top` pre `h2` na 0.

Vlasnosť padding:

Vlastnosť `padding` a všetky ostatné príbuzné vlastnosti môžeme aplikovať na ľubovoľný viditeľný element.

V prípade, že chceme nastaviť rovnaké medzery vo vodorovnom a zvislom smere, môžeme vlastnosti padding zadať dve hodnoty, napr. `padding: 2em 20px`. Táto vlastnosť nastaví veľkosť vnútorného okraja vo zvislom smere na 2 znaky (2em) a vo vodorovnom smere to bude 20 bodov (20px).

Poznámka:

V príkladoch 8.21 a 8.23 ste si mohli všimnúť, že ak element nezaberal celú šírku stránky, bol zarovnaný na ľavý okraj okna prehliadača. To môžeme zmeniť, ak elementu nastavíme vlastnosti `margin-left` či `margin-right` na hodnotu `auto`.

```
div {
  background-color: lime;
  width: 50%;
  margin-left: auto;
}
```

element bude zarovnaný vpravo

```
div {
  background-color: lime;
  width: 50%;
  margin-left: auto;
  margin-right: auto;
}
```

element bude centrovaný

Úlohu 8.26 môžeme použiť na opakovanie.

Kapitolu odporúčame realizovať na 4 (prípadne viacerých) vyučovacích hodinách.

Nadväznosť na predchádzajúce kapitoly: Kapitola vyžaduje všetky predchádzajúce kapitoly.

9 TABUĽKY

Tabuľky väčšinou využívame na prehľadné zobrazenie údajov, ktoré nejako spolu súvisia. Je to najbežnejší spôsob zobrazovania dát v mnohých oblastiach ako fyzika, chémia, štatistika, ale aj psychológia a iné. Istý čas sa tabuľky (pri tvorbe webových stránok) využívali aj na rozmiestňovanie objektov na stránke, napr. na vytvorenie dvojstĺpcového alebo trojstĺpcového vzhľadu stránky. Dnes sa už tento spôsob považuje za zastaralý a na rozmiestňovanie objektov sa využívajú predovšetkým kaskádové štýly (to si ukážeme v *kapitole 10*).

Vytváranie tabuliek

Každá tabuľka sa skladá z riadkov, v ktorých je jedna alebo viacero buniek. Tabuľky v HTML definujeme po riadkoch, nie po stĺpcoch.

Pomocou elementu `<table></table>` definujeme začiatok a koniec tabuľky. V rámci elementu `<table>` definujeme jednotlivé riadky tabuľky pomocou elementu `<tr>`. A napokon v rámci elementu `<tr>` definujeme jednotlivé bunky pomocou elementu `<td>`, resp. elementu `<th>`, ak sa jedná o bunku, ktorá je hlavičkou stĺpca alebo riadka. Od počtu buniek v riadku závisí počet stĺpcov celej tabuľky. V prípade, že nie sú žiadne bunky spojené, môžeme povedať, že počet buniek v riadku (v každom riadku rovnaký) sa rovná počtu stĺpcov tabuľky. V prípade, že sú niektoré bunky v riadku spojené, musíme si dať pozor na počet buniek v príslušnom riadku, viac pozri časť *Zlučovanie buniek*.

PRÍKLAD 9.1

V editore JSFiddle vytvoríme tabuľku o vitamínoch. Začneme prvými dvoma riadkami, ktoré budú obsahovať tieto údaje:

názov	**zdroje**	**rozpustný v**
vitamín A	plnotučné mlieko, vajcia, pečeň	tukoch

Do HTML časti napíšme nasledujúci kód:

```
<body>
  <table>
  <tr>
    <th>názov</th>
    <th>zdroje</th>
    <th>rozpustný v</th>
  </tr>
  <tr>
    <td>vitamín A</td>
    <td>plnotučné mlieko, vajcia, pečeň</td>
    <td>tukoch</td>
  </tr>
  </table>
</body>
```

Tabuľka, ktorú sme týmto kódom vytvorili, vyzerá takto:

názov	zdroje	rozpustný v
vitamín A	plnotučné mlieko, vajcia, pečeň	tukoch

Štandardne sa orámovanie pre tabuľky a bunky nezobrazuje, resp. žiadne orámovanie pre ne nie je nastavené. Ak chceme, aby tabuľka bola orámovaná, definujeme pomocou kaskádových štýlov (teda v časti CSS) vlastnosť `border` pre elementy `table`, `td` a `th`.

```
table, td, th {
  border: 1px solid black;
}
```

Poznámka: Zápisom `table`, `td`, `th` definujeme naraz štýly pre elementy `table`, `td` aj `th`. Takýto zápis s čiarkou používame vtedy, ak chceme viacerým druhom elementov definovať rovnaké vlastnosti.

ÚLOHA 9.2

Doplňte do tabuľky z *príkladu 9.1* ďalšie riadky podľa obrázka 9.1.

názov	zdroje	rozpustný v
vitamín A	plnotučné mlieko, vajcia, pečeň	tukoch
vitamín D	tvorí sa v koži pôsobením slnečného žiarenia	tukoch
vitamín B1 - tiamín	droždie, obilné klíčky, syr	vode
vitamín C	ovocie, zelenina	vode

Obrázok 9.1 Tabuľka Vitamíny.

ÚLOHA 9.3

V JSFiddle experimentujte s orámovaním tabuľky z *úlohy 9.2*:

- vyskúšajte rôzne hrúbky orámovania (napr. `2px`, `3px`, `10px`),
- vyskúšajte rôzne farby orámovania,
- nastavte orámovanie:
  - len pre table (t.j. pre `td` a `th` nie),
  - len pre `td`,
  - len pre `th`,
  - len pre `tr`,
  - pre td a `th`,
  - ako uznáte za vhodné.

Otázka: Aké štýly a pre aké elementy by sme museli definovať, aby sme okolo celej tabuľky mali hrubší rámik (napr. 5px), ale okolo jednotlivých buniek tenšie rámiky (napr. 1px)?

PRÍKLAD 9.4

Do tabuľky z predchádzajúcich úloh pridáme popis (legendu). Využijeme element `<caption>`.

```
<body>
  <table>
  <caption>Prehľad vitamínov</caption>
  <tr>
    <th>názov</th>
...
```

Popis tabuľky sa štandardne zobrazuje nad tabuľkou zarovnaný na stred.

ÚLOHA 9.5

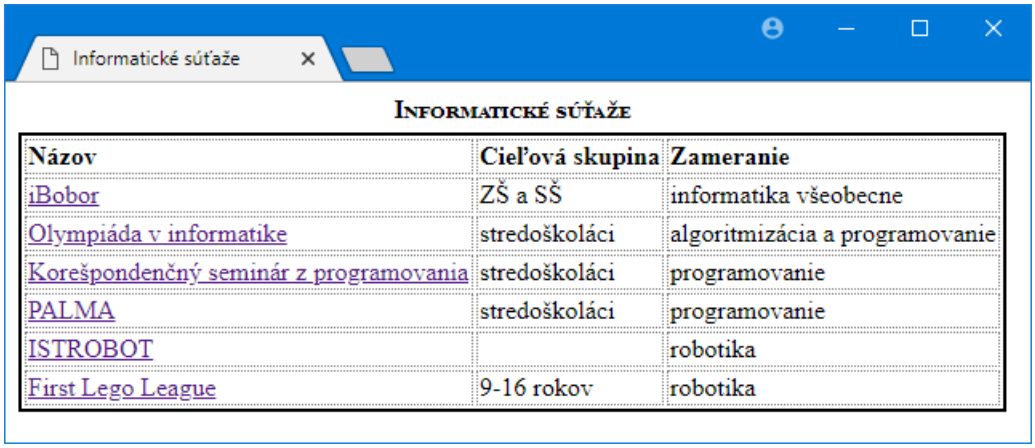
V kóde z predchádzajúceho príkladu definujte štýl pre element `caption` tak, aby popis tabuľky bol tučný a zobrazoval sa zarovnaný vľavo nad tabuľkou.

Tabuľka 9.1 Základné elementy pre tabuľku.

element	popis
`<table></table>`	začiatok a koniec celej tabuľky (vnútri musí byť všetko, čo sa týka tabuľky)
`<tr></tr>`	definuje celý riadok (vnútri sú bunky)
`<th></th>`	definuje bunku v rámci riadka, ktorá je nadpisom (zvyčajne zvýraznená) – používame len pre nadpisy riadkov/stĺpcov, nie kvôli zvýrazneniu
<td></td>	definuje bunku v rámci riadka
<caption></caption>	legenda celej tabuľky, zobrazuje sa nad okrajom tabuľky, musí byť definovaná v rámci elementu `<table></table>`

ÚLOHA 9.6

Vytvorte tabuľku informatických súťaží podľa *obrázku 9.2*. Údaje do tabuľky nájdete v súbore `09/sutaze.html`.



INFORMATICKÉ SÚŤAŽE

Názov	Cieľová skupina	Zameranie
iBobor	ZŠ a SŠ	informatika všeobecne
Olympiáda v informatike	stredoškoláci	algoritmizácia a programovanie
Korešpondenčný seminár z programovania	stredoškoláci	programovanie
PALMA	stredoškoláci	programovanie
ISTROBOT		robotika
First Lego League	9-16 rokov	robotika

Obrázok 9.2 Tabuľka Informatické súťaže.

Obsah buniek (teda obsah elementu `<td>`) môžeme ľubovoľne štruktúrovať a formátovať. Takisto môžeme do buniek vložiť obrázky, či ďalšie tabuľky (tzv. vnáranie tabuliek).

ÚLOHA 9.7

Prerobte ponuku pízz v súbore `09/ponuka.html` pomocou tabuľky (pozri *obrázok 9.3*).

- Tabuľka bude mať stĺpce názov, zloženie, obrázok, alergény a cena.
- Názvy pízz budú formátované ako nadpis úrovne 2.
- Zloženie bude zoznam jednotlivých položiek.
- Názov stĺpca Alergény bude odkazom na súbor `alergeny.html`.
- Bunky budú mať zobrazené len spodné orámovanie, tabuľka nebude mať orámovanie.
- Hlavičky stĺpcov budú zarovnané vľavo a väčším písmom (napr. 120%).

IT Pizza - ponuka

Názov	Zloženie	Obrázok	Alergény	Cena
Cardinale	• paradajková omáčka • syr • šunka		1, 7	malá 4,00 € veľká 5,50 €
Funghi	• paradajková omáčka • syr • šampiňony		1, 7	malá 5,00 € veľká 5,50 €

Obrázok 9.3 Ponuka pízz ako tabuľka.

Vlastnosti tabuliek

V predchádzajúcich dvoch kapitolách sme si ukázali, ako pomocou kaskádových štýlov nastaviť rôzne vlastnosti písma a textu, orámovanie a okraje blokov. Väčšinu týchto vlastností môžeme využiť aj pre elementy tvoriace tabuľky, nie všetky však majú zmysel pre každý z tabuľkových elementov. V tejto časti budeme veľa experimentovať s rôznymi vlastnosťami pre elementy `table`, `tr`, `td` a `th` a ukážeme si, resp. sami prídete na to, ktoré nastavenia pre ktoré elementy sú vhodné. Tiež si ukážeme nové vlastnosti špecifické len pre tabuľky.

Vlastnosti elementu table

V *časti 9.1* sme tabuľke nastavovali vlastnosť `border`. S orámovaním súvisia dve vlastnosti špecifické len pre element `table`: `border-collapse` a `border-spacing`. Ich význam si ukážeme na príkladoch.

PRÍKLAD 9.8

Vráťme sa k tabuľke vitamínov v editore JSFiddle z *úlohy 9.4* (kód je v súbore `09/vitaminy.html`). Chceme, aby tabuľka nemala separátne orámovanie pre bunky a celú tabuľku, ale aby tieto dve orámovania splynuli do jedného (porovnajte tabuľky na *obrázku 9.1* a *9.4*). Do CSS kódu doplníme nastavenie `border-collapse` pre element `table`.

```
table, td, th {
  border: 1px solid black;
}
table {
  border-collapse: collapse;
}
```

názov	zdroje	rozpustný v
vitamín A	plnotučné mlieko, vajcia, pečeň	tukoch
vitamín D	tvorí sa v koži pôsobením slnečného žiarenia	tukoch
vitamín B1 - tiamín	droždie, obilné kličky, syr	vode
vitamín C	ovocie, zelenina	vode

Obrázok 9.4 Splynutie orámovania buniek a tabuľky.

ÚLOHA 9.9

Meňte hrúbku, štýl a farbu orámovania. Prejaví sa zmena?

ODPOVEDZTE

Je možné pri splynutých orámovaniach buniek a celej tabuľky nastaviť inú farbu „vonkajšieho" rámika a inú farbu pre „vnútorné čiary"?

Hodnota vlastnosti `border-collapse` rozhoduje o tom, či budú orámovania buniek a tabuľky oddelené (hodnota `separate`, prednastavená) alebo splynú (hodnota `collapse`).

PRÍKLAD 9.10

V editore JSFiddle nahraďme vlastnosť `border-collapse` vlastnosťou `border-spacing` s hodnotu `10px`. Tým definujeme, aká má byť vzdialenosť (priestor) medzi jednotlivými bunkami (pozri *obrázok 9.5*).

```
table, td, th {
  border: 1px solid black;
}
table {
  border-spacing: 10px;
}
```

názov	zdroje	rozpustný v
vitamín A	plnotučné mlieko, vajcia, pečeň	tukoch
vitamín D	tvorí sa v koži pôsobením slnečného žiarenia	tukoch
vitamín B1 - tiamín	droždie, obilné kličky, syr	vode
vitamín C	ovocie, zelenina	vode

Obrázok 9.5 Tabuľka s nastavenou vzdialenosťou susedných buniek.

ÚLOHA 9.11

V tabuľke z *príkladu 9.10*:

- experimentujte s hodnotou `border-spacing`,
- zmeňte hodnotu `border-spacing` na dvojicu `5px 10px`,
- nastavte hodnotu `border-spacing` na dvojicu `20px 5px` a zrušte vlastnosť `border` pre `th` a `td`, pre `table` ju zachovajte!,
- vráťte vlastnosť `border` pre `td` a `th`, pridajte `border-collapse` s hodnotou `collapse` a skúšajte rôzne hodnoty pre `border-spacing`. Prejaví sa nejako zmena hodnôt `border-spacing`? Prečo asi?

ÚLOHA 9.12

V tabuľke z predchádzajúcej úlohy experimentujte s nastaveniami LEN pre element `table`:

- zrušte `border-spacing`, ponechajte `border-collapse: collapse`,
- definujte šírku (`width`), vyskúšajte napr. hodnoty `70%`, `100%`, `50%`, `300px`, `600px`, ..,
- definujte výšku (`height`), vyskúšajte napr. hodnoty `200px`, `100px`, `130px`, ...,
- nastavte vonkajší okraj (`margin`) na `20px`,
- nastavte farbu pozadia (`background-color`) na `#ECD9C6`,
- nastavte `font-family` na `Arial`,
- nastavte vnútorný okraj (`padding`) na `20px`.

Tabuľka 9.2 Vlastnosti tabuľky.

element		popis
`border-collapse`	collapse | separate	Definuje správanie orámovania buniek tabuľky. **collapse** – Orámovania tabuľky a jednotlivých buniek budú splývať. Čiary orámovania budú vycentrované v priestore medzi bunkami. **separate** – Celá tabuľka a každá bunka tabuľky má svoje vlastné orámovanie a vzdialenosť medzi bunkami je riadená vlastnosťou `border-spacing`.
`border-spacing`	*dĺžka1 dĺžka2 | dĺžka* napr. 10px 2px alebo len 5px	Určuje vzdialenosť medzi orámovaním susedných buniek. Priestor medzi orámovaním je vyplnený pozadím tabuľky. Ak zadáme dve hodnoty, prvá určuje vzdialenosť v horizontálnom a druhá vo vertikálnom smere. Ak zadáme len jednu hodnotu, určuje vzdialenosť pre oba smery.
`caption-side`	top | bottom	Určuje polohu legendy/hlavičky vzhľadom k tabuľke.

Okrem uvedených špecifických vlastností môžeme elementu `table` definovať takmer všetky nám už známe vlastnosti, napr. rôzne vlastnosti textu (`font-family`, `color`, `font-weight`, ...), farbu pozadia, vonkajšie či vnútorné okraje, šírku, výšku.

Vlastnosti elementu tr

PRÍKLAD 9.13

V tabuľke z *úlohy 9.12* nastavme prvému riadku (riadku s hlavičkami) tmavé pozadie (napr. farby `#130D06`), svetlú farbu textu (napr. `#ECD9C6`) a zarovnanie vľavo (*obrázok 9.6*).

Riešenie

```
tr {
  background-color: #130D06;
  color: #ECD9C6;
  text-align: left;
}
```

zrejme nie je úplne správne, pretože takto sme zmenili všetky riadky tabuľky. Už vieme, že ak chceme definovať štýl práve jednému výskytu elementu, musíme tento element nejako identifikovať. Definujme prvému riadku tabuľky v HTML kóde identifikátor `id="prvy"`.

```
<table>
  <caption>Prehľad vitamínov</caption>
  <tr id="prvy">
    <th>názov</th>
...
```

a v CSS kóde definujme štýl pre `#prvy` (namiesto pre element `tr`).

```
#prvy {
  background-color: #130D06;
  color: #ECD9C6;
  text-align: left;
}
```

ODPOVEDZTE

Vedeli by ste (pre našu konkrétnu tabuľku) vyriešiť *príklad 9.13* aj bez použitia atribútu `id` (resp. bez zásahu do HTML časti, len použitím CSS)? Bolo by vami navrhnuté riešenie použiteľné aj pre iné tabuľky? Pripomíname že chceme zvýrazniť len prvý riadok.

Prehľad vitamínov

názov	zdroje	rozpustný v
vitamín A	plnotučné mlieko, vajcia, pečeň	tukoch
vitamín D	tvorí sa v koži pôsobením slnečného žiarenia	tukoch
vitamín B1 - tiamín	droždie, obilné klíčky, syr	vode
vitamín C	ovocie, zelenina	vode

Obrázok 9.6 Zvýraznenie prvého riadka tabuľky.

ÚLOHA 9.14

V CSS kóde z *príkladu 9.13* pridajte do štýlu pre `#prvy`:

- nastavenie šikmého písma,
- nastavenie výšky, napr. na `30px`,
- nastavenie šírky – skúšajte rôzne hodnoty,
- nastavenia `margin`, `padding`, `border`.

Riadkom tabuľky (elementu `tr`) môžeme nastavovať rôzne vlastnosti textu, farbu pozadia a výšku. Nemá však zmysel nastavovať im akékoľvek okraje či orámovanie, ani šírku.

Otázka: Zamyslite sa (resp. vyskúšajte), čo sa stane, ak definujeme štýl pre pseudotriedu `tr:hover` (napr. `tr:hover {background-color: white}`).

Vlastnosti elementov td, th

PRÍKLAD 9.15

V editore JSFiddle v CSS časti ponechajme v štýle pre element `table` len nasledujúce vlastnosti:

```
…
table {
  border-collapse: collapse;
  width: 350px;
  background-color: #ECD9C6;
  font-family: Arial;
}
#prvy {
...
```

Tabuľku sme zámerne zúžili na 350px, aby nám vznikli bunky s viacerými riadkami textu. Všimnime si teraz napr. bunku s textom *vitamín D*. Text v nej je vodorovne zarovnaný vľavo a zvislo sa v rámci bunky nachádza v strede. Aj všetky ostatné texty sú zvislo umiestnené v strede buniek.

Zmeňme zvislé zarovnanie textu vo všetkých bunkách tabuľky tak, aby text bol umiestnený v rámci bunky hore – definujme pre `td` a `th` vlastnosť `vertical-align`.

```
...
td, th {
  vertical-align: top;
}
```

názov	zdroje	rozpustný v
vitamín A	plnotučné mlieko, vajcia, pečeň	tukoch
vitamín D	tvorí sa v koži pôsobením slnečného žiarenia	tukoch
vitamín B1 - tiamín	droždie, obilné klíčky, syr	vode
vitamín C	ovocie, zelenina	vode

názov	zdroje	rozpustný v
vitamín A	plnotučné mlieko, vajcia, pečeň	tukoch
vitamín D	tvorí sa v koži pôsobením slnečného žiarenia	tukoch
vitamín B1 - tiamín	droždie, obilné klíčky, syr	vode
vitamín C	ovocie, zelenina	vode

Obrázok 9.7 Zvislé zarovnanie textu: na stred a hore.

Pomocou vlastnosti `vertical-align` definujeme zvislé zarovnanie textu v rámci bunky. Môže nadobúdať hodnoty `middle` (teda akési zvislé centrovanie), `top` a `bottom`. Vlastnosť môžeme použiť aj pre element `tr`, nie však pre `table`.

ÚLOHA 9.16

V CSS kóde z *príkladu 9.15* experimentujte s vlastnosťami pre elementy `td` a `th`:

- zmeňte nastavenie `vertical-align` na `middle`, resp. `bottom`,
- nastavte hodnotu `padding` na `3px`, `10px`, resp. štvoricu `5px 10px 5px 10px`,

- zrušte vlastnosť `width` pre element `table` a definujte `width` pre `td` a `th` s hodnotou `150px`; experimentujte s inými hodnotami, skúste príliš malé aj príliš veľké hodnoty,
- pridajte nastavenie `height` s hodnotou napr. `30px`, `50px`, `80px`.

ODPOVEDZTE

Pre aký typ údajov by bolo vhodné použiť vodorovné zarovnanie s hodnotou `right`?

Ako by ste nastavili šírku len prvého stĺpca tabuľky na `100px`?

Bunkám tabuľky môžeme nastavovať vlastnosti textu (vrátane horizontálneho aj vertikálneho zarovnania), farbu pozadia, šírku, výšku, vnútorné okraje a orámovanie. Nemá zmysel nastavovať im vlastnosť `margin`, lebo okolie buniek určujú vlastnosti elementu `table`.

CSS – triedy

PRÍKLAD 9.17

V JSFiddle upravíme štýly pre tabuľku tak, aby sa striedali farby riadkov ako na *obrázku 9.8*. Najskôr si do CSS časti nakopírujeme štýly zo súboru `09/styl09pr17.txt`.

Prehľad vitamínov

názov	zdroje	rozpustný v
vitamín A	plnotučné mlieko, vajcia, pečeň	tukoch
vitamín D	tvorí sa v koži pôsobením slnečného žiarenia	tukoch
vitamín B1 - tiamín	droždie, obilné klíčky, syr	vode
vitamín C	ovocie, zelenina	vode

Obrázok 9.8 Tabuľka so striedajúcimi sa farbami riadkov.

S doterajšími znalosťami by sme to vedeli vyriešiť napr. takto: v HTML kóde by sme tretiemu a piatemu riadku definovali atribút `id` (pre každý z týchto riadkov rôzne, lebo `id` musí byť unikátne) a v CSS kóde by sme pre tieto dva identifikátory vytvorili štýl, v ktorom by sme nastavili farbu pozadia. Toto riešenie by bolo funkčné, ale nie veľmi efektívne. Ak by tabuľka mala viac riadkov, museli by sme pre polovicu riadkov definovať rôzne identifikátory a pridať všetky tieto identifikátory do CSS časti, čím vlastne vytvárame toľko štýlov, koľko je identifikátorov, hoci všetky tieto štýly sú rovnaké. Potrebujeme jeden spoločný štýl pre viacero výskytov (ale nie všetky) istého elementu. Na to môžeme použiť tzv. triedy.

V HTML kóde tabuľky si „označíme" všetky tie elementy `tr`, ktoré majú mať svetlejšie pozadie – v našom prípade 2. a 4. riadok. Označíme ich pridaním atribútu `class` s hodotou napr. `"svetly"`.

```
<table>
<caption>Prehľad vitamínov</caption>
<tr id="prvy">
...
</tr>
```

```
<tr class="svetly">
  <td>vitamín A</td>
  ...
</tr>
<tr>
  <td>vitamín D</td>
  ...
</tr>
<tr class="svetly">
  <td>vitamín B1 - tiamín</td>
  ...
</tr>
...
```

V CSS kóde definujeme triedu (štýl) s názvom `.svetly` a jedinou vlastnosťou – farbou pozadia. V definícii kaskádového štýlu musíme meno triedy uvádzať s bodkou na začiatku (bez medzery), napr. `.svetly`.

```
.svetly {
  background-color: #F9F2EC;
}
```

Takto definovanú triedu môžeme použiť nielen pre element `tr`, ale pre ľubovoľný element. Môžeme napríklad doplniť pred tabuľku nadpis `h1` a naň tiež aplikovať štýl definovaný triedou `.svetly`.

```
<body>
  <h1 class="svetly">Vitamíny</h1>
  <table>
...
```

ZAPAMÄTAJTE SI

V kaskádových štýloch zadefinujeme triedu spôsobom

`.nazov_triedy { vlastnosť: hodnota; ... }`

a v HTML kóde označíme elementy, pre ktoré chceme túto triedu použiť pomocou atribútu `class="nazov_triedy"`.

ÚLOHA 9.18

V súbore `09/body.html` máme definovanú tabuľku s bodmi žiakov na seminári z informatiky. Využite triedu (`class`) na to, aby stĺpec tabuľky, v ktorom je súčet bodov, mal šírku `100px` a text v ňom bol zarovnaný vpravo a tučný. Zarovnajte vpravo tiež obsah všetkých ostatných buniek, v ktorých sú čísla (nie je potrebná trieda).

Zlučovanie buniek

Pomocou HTML vieme definovať aj tabuľky so zlúčenými bunkami. Môžeme zlučovať niekoľko buniek vedľa seba v rámci riadka alebo niekoľko buniek pod sebou v rámci jedného stĺpca.

PRÍKLAD 9.19

V súbore `09/vitaminy2.html` máme podobnú tabuľku s vitamínmi, s akou sme pracovali doteraz, len s väčším množstvom údajov a takmer bez štýlov. Zlúčime v 3. stĺpci všetky po sebe idúce bunky, v ktorých je rovnaká hodnota (pozri *obrázok 9.9*). Tabuľku môžeme upravovať priamo v súbore alebo ju nakopírujeme do JSFiddle.

názov	zdroje	rozpustný v
vitamín A	plnotučné mlieko, vajcia, pečeň	tukoch
vitamín D	tvorí sa v koži pôsobením slnečného žiarenia	
vitamín B1 - tiamín	droždie, obilné klíčky, syr	vode
vitamín B2 - riboflavín	kvasnice, mlieko, mäso, ryby	
vitamín C	ovocie, zelenina	

Obrázok 9.9 Zlúčenie buniek v rámci stĺpca.

Ak chceme zlúčiť bunky pod sebou v rámci jedného stĺpca, použijeme atribút `rowspan`. Jeho hodnotou je číslo, vyjadrujúce koľko buniek pod sebou chceme zlúčiť do jednej. Atribút definujeme pre tú bunku tabuľky, ktorá je prvá (najvyššie) zo zlučovaných. Vo všetkých nasledujúcich riadkoch, ktoré sa majú zlúčiť, musíme túto jednu bunku vynechať!

```
<table>
  <tr>
    <th>názov</th>
    <th>zdroje</th>
    <th>rozpustný v</th>
  </tr>
  <tr>
    <td>vitamín A</td>
    <td>plnotučné mlieko, vajcia, pečeň</td>
    <td rowspan="2">tukoch</td>
  </tr>
  <tr>
    <td>vitamín D</td>
    <td>tvorí sa v koži pôsobením slnečného žiarenia</td>
      <td>tukoch</td>
  </tr>
  <tr>
      <td>vitamín B1 - tiamín</td>
      <td>droždie, obilné klíčky, syr</td>
      <td rowspan="3">vode</td>
  </tr>
  <tr>
    <td>vitamín B2 - riboflavín</td>
    <td>kvasnice, mlieko, mäso, ryby</td>
    <td>vo vode</td>
  </tr>

  <tr>
    <td>vitamín C</td>
    <td>ovocie, zelenina</td>
    <td>vo vode</td>
  </tr>
</table>
```

Podobne funguje zlučovanie buniek vedľa seba v rámci jedného riadka. Do najľavejšej bunky zo zlučovaných pridáme atribút `colspan`, ktorého hodnotou bude počet zlučovaných buniek.

Definície zvyšných zlučovaných buniek v danom riadku potom vynecháme (t.j. vynecháme príslušné `td` elementy).

PRÍKLAD 9.20

V súbore `09/adresy.html` zlúčime bunky, kde fakturačná adresa a doručovacia adresa sú rovnaké (pozri obrázok 9.10).

meno	doručovacia adresa	fakturačná adresa
Jožko Mrkvička	Hronská 5, 974 01 Banská Bystrica	Nevädzová 54, 821 01 Bratislava
Aneta Múdra	Alejová 7, 040 01 Košice	
Boris Malina	Chalupkova 17, 010 09 Žilina	Hečkova 4, 010 01 Žilina
Milena Veselá	L. Novomeského 22, 911 08 Trenčín	

Obrázok 9.10 Zlúčenie buniek v rámci stĺpca.

```
<table>
    <tr>
      <th>meno</th>
      <th>doručovacia adresa</th>
      <th>fakturačná adresa</th>
    </tr>
    <tr>
      <td>Jožko Mrkvička</td>
      <td>Hronská 5, 974 01 Banská Bystrica</td>
      <td>Nevädzová 54, 821 01 Bratislava</td>
    </tr>
    <tr>
      <td>Aneta Múdra</td>
      <td colspan="2">Alejová 7, 040 01 Košice</td>
      <td>Alejová 7, 040 01 Košice</td>
    </tr>
    <tr>
      <td>Boris Malina</td>
      <td>Chalupkova 17, 010 09 Žilina</td>
      <td>Hečkova 4, 010 01 Žilina</td>
    </tr>
    <tr>
      <td>Milena Veselá</td>
      <td colspan="2">L. Novomeského 22, 911 08 Trenčín</td>
      <td>L. Novomeského 22, 911 08 Trenčín</td>
    </tr>
</table>
```

ÚLOHA 9.21

V tabuľke so súťažami z *úlohy 9.6* zlúčte tie po sebe idúce bunky v stĺpci Zameranie, v ktorých je rovnaká hodnota (pozri *obrázok 9.11*).

Informatické súťaže

Názov	Cieľová skupina	Zameranie
iBobor	ZŠ a SŠ	informatika všeobecne
Olympiáda v informatike	stredoškoláci	programovanie
Korešpondenčný seminár z programovania	stredoškoláci	
PALMA	stredoškoláci	
ISTROBOT		robotika
First Lego League	9-16 rokov	

Obrázok 9.11 Tabuľka súťaží so zlúčenými bunkami.

ÚLOHA 9.22

Pomocou tabuľky vytvorte svoj rozvrh hodín. Môže byť vymyslený, s menším počtom hodín ako ten reálny, ale nech sú v ňom aspoň štyri rôzne predmety a aspoň dve dvojhodinovky. Tabuľka by mala spĺňať tieto vlastnosti:

- bude sa zobrazovať orámovanie tabuľky aj orámovanie buniek, pričom tieto čiary splynú,
- tabuľka bude zaberať 80% šírky stránky,
- obsah buniek bude zarovnaný na stred horizontálne aj vertikálne,
- obsah buniek bude od orámovania mierne odsadený,
- každý predmet bude mať inú farbu, rovnaké predmety tú istú farbu (ak máte veľa predmetov, rozdeľte ich do troch skupín, prírodovedné, humanitné a ostatné a každá skupina nech má svoju farbu),
- dvojhodinovky (prípadne viachodinovky) budú zlúčené bunky,
- hlavičky stĺpcov a riadkov budú tučným písmom.

Ďalšie vlastnosti doplňte podľa vlastnej fantázie.

Metodické pokyny pre učiteľa

CIEĽ

Cieľom je naučiť sa vytvárať tabuľku pomocou HTML jazyka, experimentovať s vlasnosťami tabuliek, preskúmať, ktoré vlasnosti majú zmysel pre celú tabuľku, ktoré pre riadky a ktoré len pre bunky.

VÝKLAD

K poznámke za príkladom 9.1: Ak chceme rovnaké vlastnosti zadefinovať viacerým druhom elementov, stačí ich oddeliť čiarkou, napr.

- `h1, h2 {...}` – tento štýl bude patriť všetkým výskytom elementov `h1` a `h2` na stránke,
- `#id1, #id2: {...}` – tento štýl sa aplikuje na element s identifikátorom `id1` a element s identifikátorom `id2`,
- `img, #galeria: {...}` – tento štýl ovplyvní všetky výskyty elementu `img` a element s identifikátorom `galeria`,
- `h1, footer p: {...}` – týmto štýlom nastavíme vzhľad elementov `h1` a tých elementov `p`, ktoré sú vnorené v elemente `footer`.

Úloha 9.6: Nie je podstatné, aby tabuľka vyzerala presne ako na obrázku, ale mali by sa aspoň trochu pohrať aj s orámovaním tabuľky a buniek. Na obrázku boli použité nasledujúce štýly:

- celá tabuľka je orámovaná hrubším súvislým čiernym rámikom,
- bunky sú orámované 1px bodkovaným sivým rámikom,
- `th` sú zarovnané doľava,
- `caption` je tučným písmom, kapitálkami a má pridaný spodný vonkajší okraj .

K tabuľke 9.2: Vlastnosť `caption-side` nefunguje v starších verziách IE, Opera.

Príklad 9.13: Treba vyskúšať, aj toto nesprávne riešenie – aspoň pekne farebne vynikne `padding`, lebo celá tabuľka má svetlohnedú farbu pozadia a riadky budú mať tmavú, tie prekryjú tú svetlohnedú, ale nie vo vnútornom okraji tabuľky.

K otázke za príkladom 9.13: Príslušný štýl môžeme nastaviť elementu `th`, ale len pre tabuľky, ktoré majú `th` len v prvom riadku (hlavičku majú len stĺpce). Pre ľubovoľnú tabuľku nie, lebo `th` môže byť súčasne použité aj ako hlavičky riadkov, a teda by sme zvýraznili okrem 1. riadku aj 1. stĺpec.

K vertical align

Vlastnosť `vertical-align` môžeme používať nielen pre tabuľkové elementy. Vlastnosť môže nadobúdať hodnoty `baseline | sub | super | top | text-top | middle | bottom | text-bottom | length | %`. Pre elementy `td`, `th` a `tr` však majú zmysel len hodnoty `middle`, `top` a `bottom`.

Dá sa to skúšať na:
https://www.w3schools.com/cssref/playit.asp?filename=playcss_vertical-align&preval=baseline

K triedam (class)

V učebnici sme definovali tzv. všeobecné triedy, také, ktoré môžeme aplikovať na ľubovoľný element. Triedu môžeme priradiť len istému elementu, napr. `p.kapitalky {font-variant: small-cass}`. V takom prípade sa trieda aplikuje len na tie elementy `p`, ktoré majú definovaný atribút `class="kapitalky"`. Ak by sme atribút `class="kapitalky"` definovali napr. pre element `h1`, nadpis by to nijako nezmenilo, lebo trieda `.kapitalky` je určená výhrane pre elementy `p`.

Jednému elementu môžeme priradiť aj viacero tried, napr. `<p class="kapitalky" class="svetly">`.

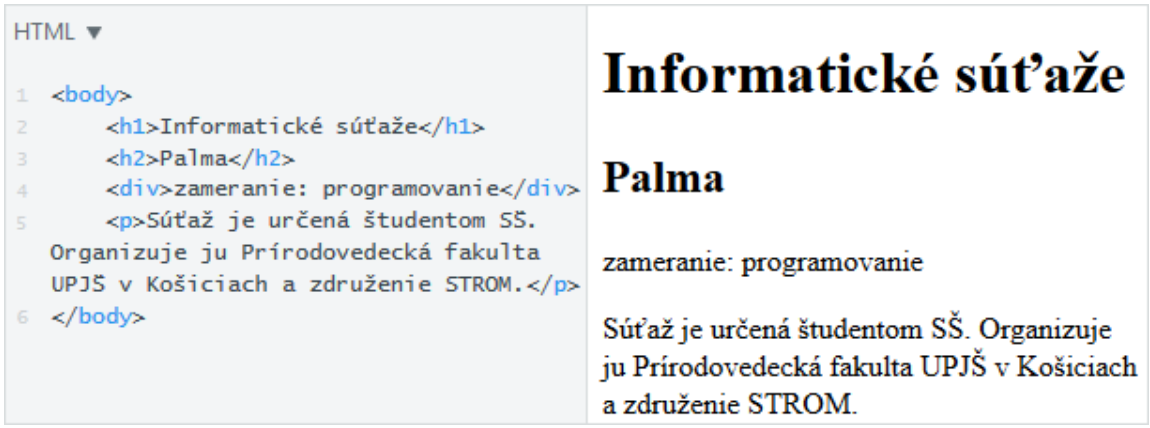
Úloha 9.22 je opakovacia. Učiteľ môže na ukážku použiť súbor `09/09-11-u-rozvrh.html`.

Nadväznosť na predchádzajúce kapitoly: Kapitola vyžaduje všetky predchádzajúce kapitoly.

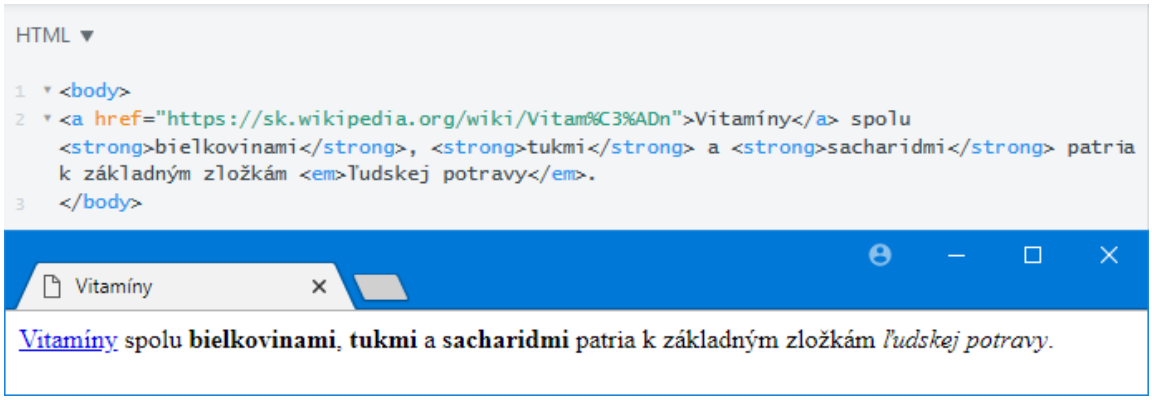
10 ROZMIESTŇOVANIE OBJEKTOV

Štandardne sa elementy, z ktorých sa skladá stránka, zobrazujú v prehliadači „jeden za druhým" v takom poradí, v akom sú definované v HTML kóde. Od typu elementu, t.j. či je element *blokový* alebo *riadkový*, závisí, či sa elementy zoraďujú pod seba, alebo vedľa seba.

- Blokové elementy sa radia za sebou zhora nadol. Štandardne je šírka elementu na celú šírku okna prehliadača, resp. maximálna šírka v rámci elementu, v ktorom je vložený. Z toho vyplýva, že blokové elementy nemôžu byť vedľa seba v jednom riadku (obrázok 10.1).
- Riadkové elementy sa radia vodorovne vedľa seba zľava doprava. Šírka elementu je daná jeho obsahom (obrázok 10.2).



Obrázok 10.1 Radenie blokových elementov.



Obrázok 10.2 Radenie riadkových elementov.

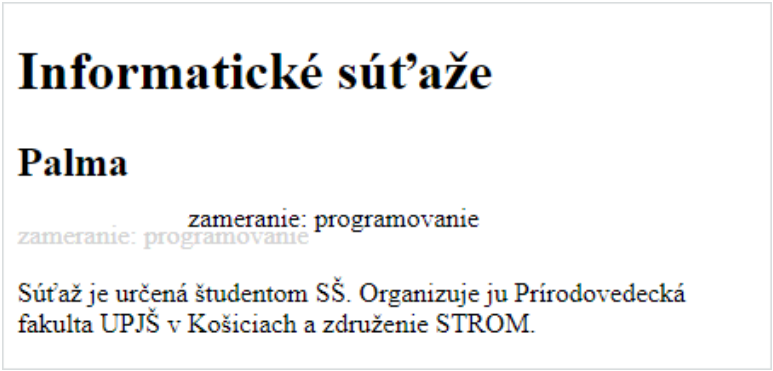
Relatívne umiestňovanie

Každý objekt (element) umiestnený normálnym spôsobom, môžeme relatívne posunúť vzhľadom k jeho základnej polohe (tzv. relatívne umiestňovanie). To urobíme tak, že elementu nastavíme vlastnosť `position` na hodnotu `relative` a zároveň pomocou niektorej (niektorých) z vlastností `top`, `right`, `bottom`, `left` určíme jeho posunutie.

PRÍKLAD 10.1

V editore JSFiddle vložme do HTML časti kód z obrázka 10.1 (súbor `10/relativne.txt`). Element `div` posunieme voči jeho **základnej polohe** o 100 bodov **zľava** a 10 bodov **zdola** (obrázok 10.3). V časti CSS definujeme štýl:

```
div {
  position: relative;
  left: 100px;
  bottom: 10px;
}
```



Obrázok 10.3 Relatívne posunutie elementu (sivý text naznačuje pôvodnú polohu elementu div).

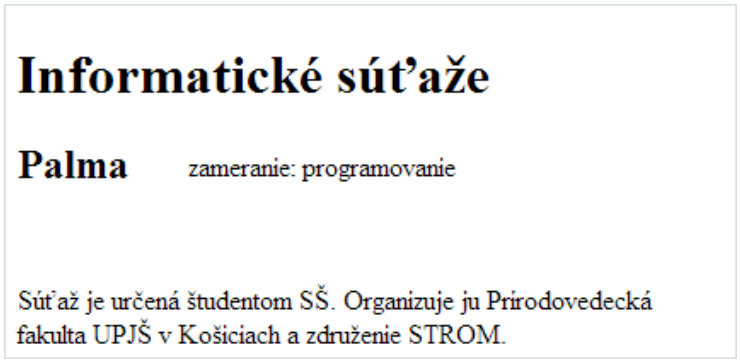
Pri relatívnom umiestňovaní majú vlastnosti `top`, `right`, `bottom` a `left` takýto význam:

- `top` určuje, o koľko posúvame objekt **zhora** (teda koľko miesta nad ním sa má vynechať),
- `right` určuje, o koľko posúvame objekt **sprava** (teda koľko miesta vpravo od neho sa má vynechať),
- `bottom` určuje, o koľko posúvame objekt **zdola** (teda koľko miesta pod ním sa má vynechať),
- `left` určuje, o koľko posúvame objekt **zľava** (teda koľko miesta vľavo od neho sa má vynechať).

ÚLOHA 10.2

V kóde z *príkladu 10.1*:

- experimentujte s hodnotou vlastnosti `left`, skúšajte napr. hodnoty `10px`, `50px`, `300px`, `-20px`, `1em`, `10%`, `50%`, …
- experimentujte s hodnotou vlastnosti `bottom`, skúšajte napr. `40px`, `80px`, `-20px`, `5%`, `1em`, …
- namiesto vlastnosti `left` nastavte `right`: `10px`, potom skúšajte iné hodnoty,
- namiesto vlastnosti `bottom` nastavte `top`: `1em`, potom skúšajte iné hodnoty,
- nastavte potrebné vlastnosti a ich hodnoty tak, aby element `div` bol vedľa nadpisu `h2` (obrázok 10.4).



Obrázok 10.4 Element div vedľa elementu h2.

Pri relatívnom umiestnení objektu prehliadač vyhradí priestor, v ktorom by bol objekt normálne zobrazený, no pomocou vlastností `top`, `left`, `right` či `bottom` môžeme objekt z tohto priestoru vysunúť. Posun objektu pomocou `position: relative` nemá žiaden vplyv na umiestnenie iných objektov. Môže sa teda stať, že relatívne umiestnený objekt bude prekrývať iné objekty. Relatívne umiestňovanie sa používa skôr na „dokreslenie“ objektov, a nie definovanie nejakého základného rozloženia (štruktúry) objektov.

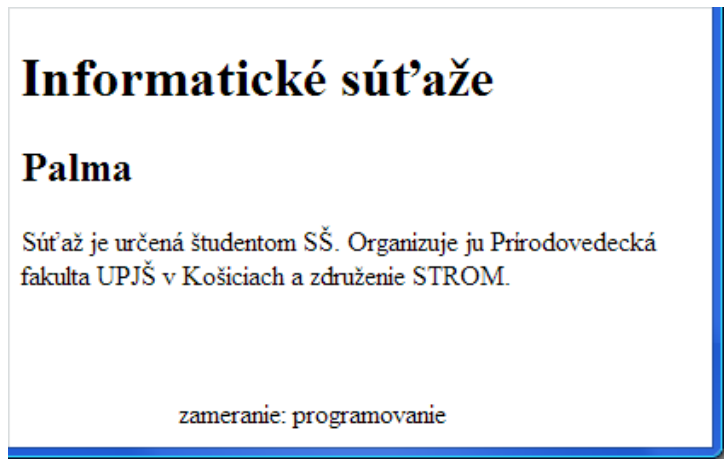
Absolútne umiestňovanie

PRÍKLAD 10.3

V CSS kóde z *príkladu 10.1* nahradíme hodnotu `relative` hodnotou `absolute`.

Na obrázku 10.5 vidíme výsledok - element `div` „odskočil” k dolnému okraju stránky.

```
div {
  position: absolute;
  left: 100px;
  bottom: 10px;
}
```



Obrázok 10.5 Absolútne umiestnenie elementu.

Skôr, ako si vysvetlíme, čo vlastne absolútna pozícia znamená, budeme experimentovať s predchádzajúcim kódom.

ÚLOHA 10.4

V kóde z *príkladu 10.3* riešte nasledujúce úlohy. Po každej zmene hodnoty meňte v editore JSFiddle veľkosť časti s výslednou stránkou (Result) - šírku aj výšku.

- experimentujte s hodnotami vlastnosti `left`,
- experimentujte s hodnotami vlastnosti `bottom`, vyskúšajte aj záporné hodnoty,
- namiesto vlastnosti `left` použite vlastnosť `right`, experimentujte s jej hodnotami,
- namiesto vlastnosti `bottom` použite vlastnosť `top`, experimentujte s jej hodnotami.

Na základe nášho skúmania by sa dalo predpokladať, že pri absolútnom umiestnení sa objekt umiestni na zadanú pozíciu (hodnotami `top`, `left`, `bottom`, `right`) v rámci stránky. Nasledujúci príklad nás presvedčí o tom, že to nie je úplne tak.

PRÍKLAD 10.5

V kóde z *príkladu 10.3* „zabalíme" všetky údaje o súťaži PALMA do elementu `<section>`. Elementu `section` v CSS nastavíme `position` na hodnotu `relative` a kvôli názornosti mu ešte pridáme orámovanie. Výsledná stránka je na obrázku 10.6.

```
<body>
  <h1>Informatické súťaže</h1>
  <section>
    <h2>Palma</h2>
    <div>zameranie:
programovanie</div>
    <p>Súťaž je určená študentom SŠ.
Organizuje ju Prírodovedecká fakulta
UPJŠ v Košiciach a združenie
STROM.</p>
  </section>
</body>
```

```
div {
  position: absolute;
  left: 100px;
  bottom: 10px;
}
section {
  border: 1px solid red;
  position: relative;
}
```

Obrázok 10.6 Absolútne umiestnenie elementu v rámci sekcie.

Vidíme, že element `div` už nie je umiestnený `10px` od spodného okraja stránky. Poďme znova skúmať jeho polohu.

ÚLOHA 10.6

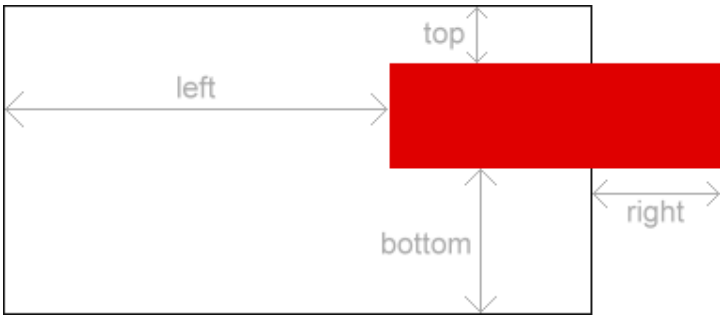
V CSS kóde z *príkladu 10.5* experimentujte so štýlom pre `div`:

- skúšajte rôzne hodnoty pre vlastnosť `bottom`,
- skúšajte rôzne hodnoty pre vlastnosť `left`,
- nahraďte vlastnosť `bottom` vlastnosťou `top` a skúšajte rôzne hodnoty,
- nahraďte vlastnosť `left` vlastnosťou `right` a skúšajte rôzne hodnoty,
- zrušte vlastnosť `border` pre `section` a nastavte potrebné vlastnosti tak, aby element `div` bol vedľa nadpisu `h2` (obrázok 10.4),
- pridajte informácie o ďalších súťažiach (môžete použiť kód zo súboru `10/sutaze.txt`).

Zdá sa, že vlastnosti `top`, `left`, `right` a `bottom` nám teraz určujú pozíciu v rámci elementu `section`.

ZAPAMÄTAJTE SI

- Umiestnený objekt je objekt, ktorý má nastavenú vlastnosť `position` na inú ako prednastavenú hodnotu (`static`).
- Absolútne umiestnený objekt (teda objekt, ktorému nastavíme `position: absolute`) sa posunie na danú pozíciu v rámci najbližšieho umiestneného rodičovského objektu. Ak absolútne umiestnený objekt nemá žiadneho umiestneného rodiča, umiestňuje sa v rámci elementu `body`.
- Pozíciu absolútne umiestneného objektu definujeme pomocou vlastností `top`, `right`, `bottom` a `left`, ktoré v tomto prípade majú takýto význam (pozri aj obrázok 10.7):
  - `top` určuje, o koľko je horný okraj elementu posunutý od horného okraja umiestneného rodiča, resp. elementu body (ďalej len rodiča),
  - `right` určuje, o koľko je pravý okraj elementu posunutý vľavo od pravého okraja rodiča,
  - `bottom` určuje, o koľko je dolný okraj elementu posunutý hore od dolného okraja rodiča,
  - `left` určuje, o koľko je ľavý okraj elementu posunutý vpravo od ľavého okraja rodiča.
- Absolútne umiestnený objekt nemá žiaden vplyv na umiestnenie ostatných objektov na jeho úrovni (vnútri umiestneného rodičovského objektu).
- Absolútne umiestnené objekty môžu prekrývať iné objekty na stránke.



Obrázok 10.7 Význam vlastností top, right, bottom a left pri absolútnom umiestňovaní. Červený objekt je absolútne umiestnený objekt v rámci umiestneného rodiča (objekt s čiernym rámikom). Hodnota right bude v tomto prípade záporná.

V *príklade 10.5* sme absolútne umiestnili element `div`. Jeho rodičom je element `section`, ktorý má definovanú vlastnosť `position: relative`, teda je umiestnený. Umiestnenie elementu `div` sa určuje vzhľadom na element `section`. V *príklade 10.3* element `div` žiadneho umiestneného rodiča nemal, preto sa jeho poloha určovala vzhľadom na element `body`.

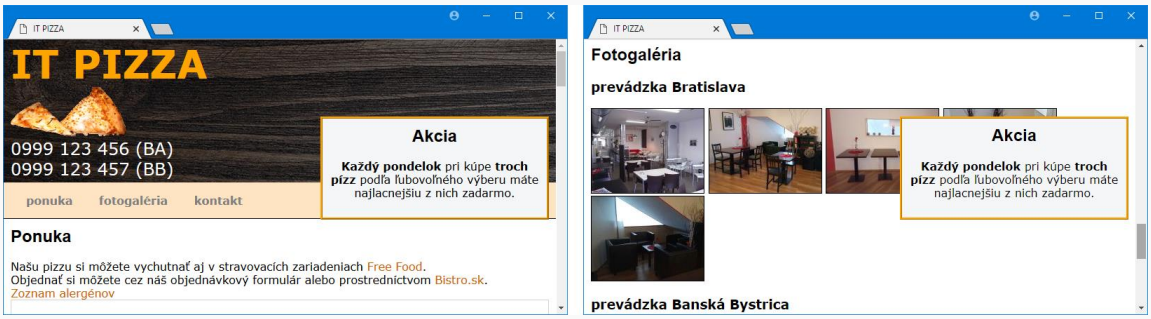
Fixné umiestňovanie

Okrem hodnôt `relative` a `absolute` môže mať vlastnosť `position` aj hodnotu `fixed`.

PRÍKLAD 10.7

Na stránke IT Pizza umiestnime časť *Akcia* tak, aby bola vždy v pravej hornej časti okna prehliadača, bez ohľadu na to, ktorú časť stránky práve vidíme (obrázok 10.8). Budeme vychádzať zo súboru `10/index.html`.

```
aside {
  ...
  position: fixed;
  right: 10px;
  top: 100px;
  width: 300px;
}
```



Obrázok 10.8 Fixovaná časť Akcie (vľavo hlavička a ponuka, vpravo fotogaléria).

Ak nejakému elementu nastavíme `position: fixed`, potom hodnotami `top`, `left`, `bottom` a `right` určujeme jeho pozíciu vzhľadom na okraje okna prehliadača. Táto pozícia zostáva fixná, aj keď stránku posúvame.

ÚLOHA 10.8

Skúmajte stránku IT Pizza z predchádzajúceho príkladu:

- posúvajte stránku hore a dolu a pozorujte umiestnenie časti *Akcia*,
- meňte veľkosť okna prehliadača a pozorujte umiestnenie časti *Akcia*,
- meňte hodnoty `top` a `right` a pozorujte umiestnenie časti *Akcia*,
- namiesto `top` a `right` vyskúšajte `bottom` a `left`.

Plávajúce umiestňovanie

Plávajúce objekty

V textových dokumentoch vieme umiestniť obrázok do textu tak, aby text obtekal okolo obrázka. Efekt obtekania obrázka textom vieme čiastočne vytvoriť aj na webových stránkach.

Obrázok 10.9 Stránka s obrázkom: text je nad a pod obrázkom.

PRÍKLAD 10.9

Na *obrázku 10.9* je stránka o vitamínoch a jej zdrojový kód (súbor `10/vitaminy1.html`). Vidíme, že na stránke sa nachádza obrázok medzi dvoma nadpismi, presnejšie jeden nadpis je nad obrázkom a druhý pod obrázkom, pričom vedľa obrázka vpravo je prázdny priestor. Upravíme stránku tak, aby obrázok bol vľavo a ostatné objekty ho obtekali sprava (*obrázok 10.10*). Obtekanie obrázka vyriešime definovaním štýlu pre element `img`.

```
<style>
  img {
    float: left;
  }
</style>
```

Obrázok 10.10 Plávajúce umiestnenie elementu img (obrázok).

Pomocou vlastnosti `float` sme z obrázka spravili tzv. **plávajúci objekt**. Ako sa taký plávajúci objekt správa, resp. ako sa správajú iné objekty na stránke, ak je medzi nimi plávajúci objekt, budeme skúmať v nasledujúcej úlohe.

ÚLOHA 10.10

V kóde z *príkladu 10.9*:

- meňte šírku okna a pozorujte, ktoré objekty obtekajú obrázok,
- nastavte šírku okna na najväčšiu možnú a určte, ktoré objekty obtekajú obrázok,
- zmeňte hodnotu vlastnosti `float` na `right` a zopakujte predchádzajúce úlohy,
- v HTML kóde pridajte za element `img` ďalší `img` element (môžete použiť aj ten istý) a pozorujte obtekanie.

Plávajúce objekty sú posunuté úplne vľavo alebo vpravo v rámci bloku, v ktorom sa nachádzajú (buď k okraju bloku alebo k inému plávajúcemu objektu). Okolo plávajúcich objektov môže (ale nemusí) obtekať ďalší obsah stránky. Každý plávajúci objekt má presne stanovenú šírku, buď explicitne určenú vlastnosťou `width`, alebo prirodzenú šírku (podľa obsahu). Pripomíname, že „neplávajúci“ blokový objekt zaberá celú šírku stránky alebo nadradeného elementu, aj keby jeho obsah bol menší.

Vlastnosť `float` môže nadobúdať hodnoty:

- `left` – objekt sa posunie k ľavému okraju,
- `right` – objekt sa posunie k pravému okraju,
- `none` – zrušenie plávania objektu.

PRÍKLAD 10.11

Upravíme CSS kód z *príkladu 10.9* tak, aby päta stránky (`footer`) bola vždy pod obrázkom, t.j. aby ho nemohla obtekať, alebo ešte inak povedané, aby obrázok nemohol plávať vľavo vedľa päty, čo pri doterajšom riešení mohol a pri istej šírke okna aj pláva.

```
<style>
  img {
    float: left;
  }
  footer {
    clear: left;
  }
</style>
```

Pomocou vlastnosti `clear` definujeme, na ktorej strane objektu nemôžu byť plávajúce objekty. Vlastnosť `clear` môže nadobúdať hodnoty:

- `left` – na ľavej strane objektu nemôže byť plávajúci objekt,
- `right` – na pravej strane objektu nemôže byť plávajúci objekt
- `both` – na ľavej ani pravej strane objektu nemôže byť plávajúci objekt,
- `none` – okolo objektu môžu byť plávajúce objekty.

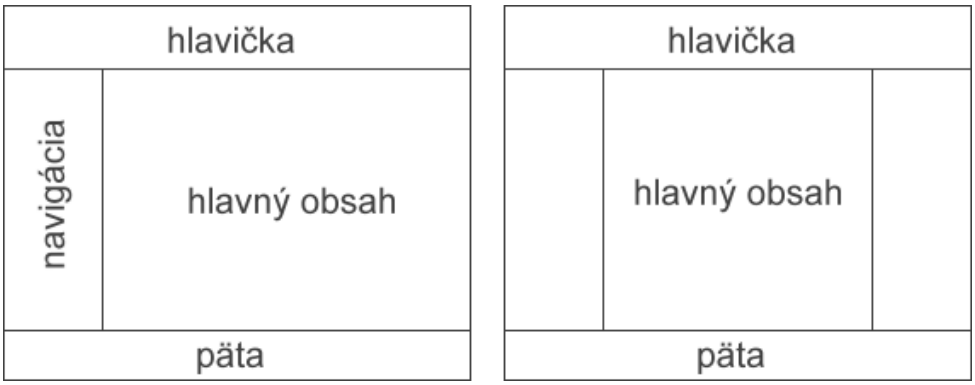
Vlastnosť `clear` zvyčajne nastavujeme objektu nasledujúcemu za plávajúcim objektom.

- Ak plávajúci objekt pláva vľavo, potom nasledujúcemu objektu nastavíme `clear: left` (ale môžeme aj `clear:both`).
- Ak plávajúci objekt pláva vpravo, použijeme `clear: right` (alebo `clear: both`).
- Ak máme viacero plávajúcich objektov s rôznymi nastaveniami pre `float`, použijeme `clear: both`.

Plávajúcim objektom môže byť ľubovoľný element, nielen obrázok.

Použitie plávajúcich objektov na vytvorenie viacstĺpcového dizajnu

Plávajúce objekty sa často využívajú na vytvorenie viacstĺpcového rozloženia objektov na stránke. Ide o také rozloženie objektov, kde celá stránka alebo jej časť je rozdelená na niekoľko stĺpcov (zvyčajne dva alebo tri). Schematické znázornenie dvoj- a trojstĺpcového rozloženia stránky si môžete pozrieť na obrázku 10.11.



Obrázok 10.11 Schéma dvojstĺcového a trojstĺcového rozloženia stránky.

ODPOVEDZTE

Akú schému majú stránky, ktoré najčastejšie navštevujete? Do koľkých stĺpcov sú rozložené a čo sa nachádza v jednotlivých stĺpcoch? Nájdite aspoň dve stránky s dvojstĺpcovým a aspoň dve stránky s trojstĺpcovým rozložením. Ako sa tieto stránky zobrazujú v mobile či tablete?

Stránka IT Pizza, tak ako sme ju navrhli v predchádzajúcich kapitolách (od prvej až po deviatu), má jednostĺpcové rozloženie objektov. Pri takomto rozložení sa stránka dobre zobrazí na obrazovkách rôznych zariadení, od mobilov až k počítačom.

V nasledujúcich príkladoch budeme postupne meniť rozloženie objektov na stránke IT Pizza tak, aby sa zobrazovali vo viacerých (v dvoch, v troch) stĺpcoch. Viacstĺpcové rozloženie objektov však už nie je vhodné pre zobrazovanie napr. v mobiloch či tabletoch. Preto ďalšie vlastnosti, ktoré jednotlivým elementom stránky IT Pizza od tohto momentu pridáme, nebudeme pridávať k už definovaným vlastnostiam, ale definujeme ich samostatne, za všetkými doteraz vytvorenými štýlmi. Ak napríklad chceme pridať spodný vnútorný okraj pre element `div` s telefónnymi číslami, tak to neurobíme takto:

```
header div {
  color: #FFF;
  font-size: 1.5em;
  padding-bottom: 1em;
}
```

ale takto:

```
<style>
  /* všetky doteraz definované štýly */
  ...
  header div {
    color: #FFF;
    font-size: 1.5em;
  }
  ...

  /* štýly súvisiace s viacstĺpcovým rozložením */
  header div {
    padding-bottom: 1em;
  }
</style>
```

PRÍKLAD 10.12

Na stránke IT Pizza rozdelíme fotky v galérii do dvoch stĺpcov: v ľavom stĺpci budú fotky z prevádzky v Bratislave, v pravom stĺpci fotky z prevádzky v Banskej Bystrici (*obrázok 10.12*). Vychádzame zo súboru `10/index-float.html`.

Obrázok 10.12 Galéria v dvoch stĺpcoch (vľavo Bratislava, vpravo Banská Bystrica).

Potrebujeme presne určiť, čo patrí do ľavého a čo do pravého stĺpca. Do ľavého stĺpca patrí nadpis `<h3>prevádzka Bratislava</h3>` a nasledujúcich päť `img` elementov. Zabalíme ich do jedného elementu `div`. Nadpis `<h3>prevádzka Banská Bystrica</h3>` a nasledujúcich päť `img` elementov vnoríme do ďalšieho elementu `div`. Sekcia s fotogalériou bude mať teda nasledujúci kód:

```
<section id="galeria">
  <h2>Fotogaléria</h2>
  <div>
    <h3>prevádzka Bratislava</h3>
    <img src="obrazky/foto-01m.jpg" alt="foto 1" ...>
    <img src="obrazky/foto-02m.jpg" alt="foto 2" ...>
    <img src="obrazky/foto-03m.jpg" alt="foto 3" ...>
    <img src="obrazky/foto-04m.jpg" alt="foto 4" ...>
    <img src="obrazky/foto-05m.jpg" alt="foto 5" ...>
  </div>
```

```
  <div>
    <h3>prevádzka Banská Bystrica</h3>
    <img src="obrazky/foto-06m.jpg" alt="foto 6" ...>
    <img src="obrazky/foto-07m.jpg" alt="foto 7" ...>
    <img src="obrazky/foto-08m.jpg" alt="foto 8" ...>
    <img src="obrazky/foto-09m.jpg" alt="foto 9" ...>
    <img src="obrazky/foto-10m.jpg" alt="foto 10" ...>
  </div>
</section>
```

Z oboch stĺpcov spravíme objekty plávajúce vľavo a nastavíme im šírku 50% (aby boli oba rovnaké a spoločne zaberali celú šírku stránky). Keďže oba elementy `div` budú mať rovnaké vlastnosti a iné elementy `div` v sekcii s identifikátorom `galeria` nemáme, môžeme definovať štýl `#galeria div`, t.j. štýl pre tie elementy `div`, ktoré sú v elemente s `id galeria`).

```
/* štýly súvisiace s viacstĺpcovým rozložením */
#galeria div {
  float: left;
  width: 50%;
}
```

Všimnite si, že žltá farba pozadia päty „pretiekla" aj do sekcie s galériou, ktorá mala pôvodne biele pozadie. Ešte by sme mali zabezpečiť, aby objekt, ktorý nasleduje za plávajúcimi `div` elementami, nemohol mať vedľa seba žiadne plávajúce objekty. Za sekciou galéria nasleduje päta stránky, preto vytvoríme štýl pre `footer` s nastavením `clear: left`.

```
/* štýly súvisiace s viacstĺpcovým rozložením */
#galeria div {
  float: left;
  width: 50%;
}
footer {
  clear: left;
}
```

ODPOVEDZTE

Čo by sa zmenilo, keby sme v *príklade 10.12* použili `float: right` a `clear: right`? Vyskúšajte.

ÚLOHA 10.13

Na stránke IT Pizza využite plávajúce objekty na vytvorenie trojstĺpcového rozloženia päty (obrázok 10.13). Pomôcka: Vytvorte div elementy pre každý zo stĺpcov (kontakt Bratislava, kontakt Banská Bystrica, kontakty na sociálne siete), nastavte im vhodné identifikátory a definujte potrebné štýly. Šírky stĺpcov rozumne prispôsobte (nemusia byť rovnako široké).

Obrázok 10.13 Rozloženie informácií v päte do troch stĺpcov.

ZAPAMÄTAJTE SI

Ak chceme plávajúce objekty použiť na vytvorenie viacstĺpcového vzhľadu stránky alebo jej časti:

- definujeme každý stĺpec ako samostatný element (ak stĺpec tvorí viacero elementov, vnoríme ich napr. do elementu `div`),
- z každého stĺpca spravíme objekt plávajúci vľavo pomocou `float: left`,
- objektu, ktorý v HTML kóde nasleduje bezprostredne po plávajúcich objektoch (stĺpcoch), zakážeme, aby vedľa neho vľavo bol nejaký plávajúci objekt pomocou `clear: left`, resp. `clear: both`,
- každému stĺpcu definujeme šírku tak, aby súčet šírok vrátane ľavých a pravých okrajov bol spolu 100%,
- ak treba, nastavíme iné potrebné vlastnosti (napr. výšku pre niektorý z plávajúcich objektov).

S akými problémami sa môžeme stretnúť pri použití plávajúcich objektov?

- Pri zmenšení okna prehliadača sa začnú jednotlivé stĺpce prekrývať a informácie v nich budú nečitateľné. Riešenie tohto problému si ukážeme v *kapitole 11*.
- Môže sa stať, že obsah objektu „vytečie" z boxu, ktorý je preň definovaný. Plávajúce objekty dobre fungujú vtedy, ak vieme garantovať, že výšky jednotlivých stĺpcov sú približne rovnaké, a to aj pri zmene veľkosti prehliadača. Inak je výhodnejšie použiť tzv. flexibilné boxy, ktoré dokážu prispôsobiť výšky jednotlivých stĺpcov podľa najvyššieho stĺpca. O flexibilných boxoch sa dozvieme neskôr v tejto kapitole.

Vlastnosť display

Vlastnosť display definuje spôsob zobrazenia elementu. Pomocou vlastnosti display môžeme riadkový element zmeniť na blokový alebo naopak blokový na riadkový, či dokonca element skryť. Môže nadobúdať množstvo hodnôt, my sa oboznámime s niektorými z nich.

ÚLOHA 10.14

V editore JSFiddle do časti HTML napíšte nasledujúci kód:

```
V tejto vete je použitý <em>riadkový</em> element em.
<p>Toto je odsek, štandardne blokový element. Mal by teda začínať
na samostatnom riadku.</p>
Toto je obyčajný text za odsekom.
```

V CSS časti postupne definujte nasledujúce štýly a pozorujte zobrazenie elementu `em` a `p`.

a)
```
em {background-color: lightgreen; }
```

b)
```
em {
  background-color: lightgreen;
  display: block;
}
```

c)
```
em {
  background-color: lightgreen;
  display: block;
  width: 150px;
}
```

d)
```
em {
  background-color: lightgreen;
  display: inline;
  width: 150px;
}
```

e)
```
em {
  background-color: lightgreen;
  display: inline-block;
  width: 150px;
}
```

f)
```
p {
  border: 1px solid red;
}
```

g)
```
p {
  border: 1px solid red;
  display: inline;
}
```

h)
```
p {
  border: 1px solid red;
  display: none;
}
```

Význam použitých hodnôt vlastnosti `display` popisujeme v tabuľke 10.1. Vlastnosť `display` môžeme využiť na rôzne zmeny vzhľadu navigácie.

Tabuľka 10.1 Popis vybraných hodnôt vlastnosti display.

hodnota	popis
`block`	element sa zobrazí ako blokový; začína na novom riadku, zaberá celú šírku nadradeného elementu; môžeme nastavovať šírku a výšku
`inline`	element sa zobrazí ako riadkový; nemôžeme nastavovať šírku ani výšku (tá závisí od obsahu elementu)
`inline-block`	element sa zobrazí ako riadkový, ale môžeme nastavovať šírku a výšku
`none`	element sa nezobrazí; stránka sa zobrazí tak, akoby element neexistoval

PRÍKLAD 10.15

V editore JSFiddle máme definovanú jednoduchú vodorovnú navigáciu (*obrázok 10.14*, súbor `navigacia.txt` – samostatne nakopírujeme HTML časť a CSS časť).

vitamín A | vitamín B - tiamín | vitamín C | vitamín D | vitamín E

Obrázok 10.14 Vodorovná navigácia s odkazmi rôznej šírky.

Zmeníme jednotlivé odkazy tak, aby mali všetky rovnakú šírku a text v nich bol centrovaný (obrázok 10.15).

vitamín A | vitamín B - tiamín | vitamín C | vitamín D | vitamín E

Obrázok 10.15 Vodorovná navigácia s odkazmi rovnakej šírky.

Element `a` je štandardne riadkový, teda vlastnosť `display` má nastavenú na hodnotu `inline`. Ak chceme, aby sa odkazy zobrazovali vedľa seba ako riadkové elementy, ale mohli sme im nastavovať šírku ako blokovým elementom, zmeníme hodnotu vlastnosti display na `inline-block`. Potom už zostáva len nastaviť vhodnú šírku a centrovanie textu.

```
<nav>
  <a href="#">vitamín A</a>
  <a href="#">vitamín B -
tiamín</a>
  <a href="#">vitamín C</a>
  <a href="#">vitamín D</a>
  <a href="#">vitamín E</a>
</nav>
```

```
nav a {
  background-color: #e6e6ff;
  padding: 10px;
  display: inline-block;
  width: 120px;
  text-align: center;
}

nav a:hover {
  background-color: #c2d6d6;
}
```

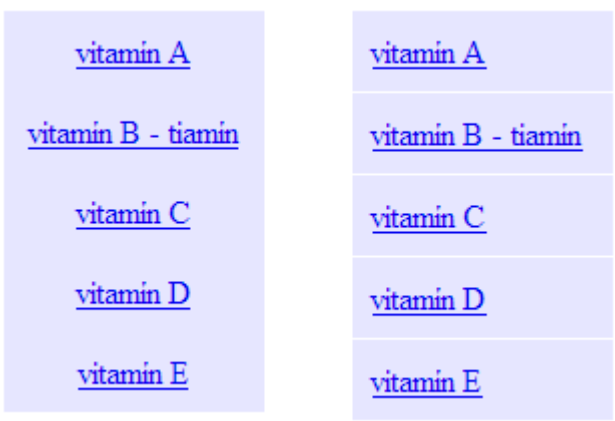
PRÍKLAD 10.16

Navigáciu z *príkladu 10.15* zmeníme na zvislú, t.j. jednotlivé odkazy sa budú zobrazovať pod sebou (*obrázok 10.16* vľavo). Umiestnenie odkazov pod sebou dosiahneme tým, že im zmeníme spôsob zobrazovania na taký, aký majú blokové elementy.

```
nav a {
  background-color: #e6e6ff;
  padding: 10px;
  display: block;
  width: 120px;
  text-align:center;
}
```

ÚLOHA 10.17

Zrušte centrovanie textu v odkazoch a vytvorte medzi nimi medzeru (*obrázok 10.16* vpravo).



Obrázok 10.16 Zvislá navigácia.

Flexibilné boxy

Aktuálne najnovšou technológiou na definovanie rozloženia objektov na webových stránkach sú tzv. **flexibilné boxy**. Ich výhodou je, že stránky vytvorené pomocou nich vieme ľahko prispôsobiť rôznym zobrazovacím zariadeniam (nielen obrazovky počítačov či notebookov, ale aj tablety, či mobily). Nevýhodou je, že nie sú podporované staršími verziami prehliadačov.

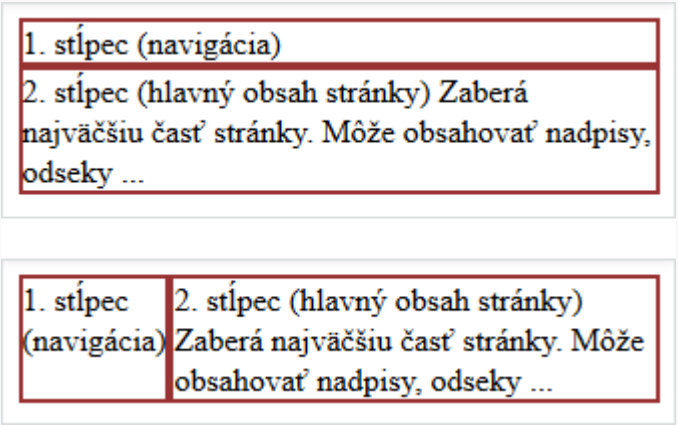
PRÍKLAD 10.18

Budeme experimentovať s flexibilnými boxami. V editore JSFiddle vytvoríme jeden `div` element s dvoma vnorenými `section` elementami (súbor `10/flex.html`). Pre sekcie definujeme v CSS jednoduché orámovanie, napr. `section {border: 2px solid #993333;}`. Meníme veľkosť okna s výslednou stránkou a pozorujeme (*obrázok 10.17* hore).

```
<div>
  <section>1. stĺpec (navigácia)</section>
  <section>2. stĺpec (hlavný obsah stránky) Zaberá najväčšiu časť
stránky. Môže obsahovať nadpisy, odseky ...
  </section>
</div>
```

V CSS nastavíme elementu `div` vlastnosť `display: flex`. Meníme veľkosť okna s výslednou stránkou a pozorujeme (obrázok 10.17 dolu).

```
section {
  border: 2px solid #993333;
}
div {
  display: flex;
}
```



Obrázok 10.17 Zobrazenie sekcií pred (hore) a po (dolu) nastavení display: flex.

Nepoužili sme žiaden druh umiestňovania (ani cez `position` ani cez `float`), ani sme sekciám nezmenili spôsob zobrazenia na `inline`, no napriek tomu sa sekcie zobrazili **vedľa** seba. Umožňuje to nastavenie `display: flex`. Toto nastavenie definujeme pre ten element, ktorý je bezprostredne nadradený elementom, ktoré chceme umiestniť vedľa seba, t.j. vytvára pre ne akýsi kontajner. Všimnite si, že nech akokoľvek zmeníme šírku okna, výška všetkých častí v rámci flexibilného kontajnera je rovnaká.

Element s nastavením `display: flex` budeme nazývať **flexibilný box.**

ÚLOHA 10.19

V HTML kóde z *príkladu 10.18* vykonajte postupne nasledujúce zmeny.

- Doplňte ďalšiu sekciu, napr. `<section>3. stĺpec (reklama, sponzori, akcie)</section>`.
- Obsah stredného stĺpca naformátujte tak, aby text *2. stĺpec (hlavný obsah stránky)* bol nadpis a zvyšný text bol odsek. Umiestnia sa nadpis a odsek vedľa seba alebo pod seba?
- Meňte šírku „prehliadača" a všímajte si nasledujúce vlastnosti.
  - Sú jednotlivé sekcie rovnako široké? Od čoho závisí ich šírka?
  - Zaberajú všetky sekcie celú šírku stránky alebo len časť? Ak len časť, sú zarovnané vľavo, vpravo, či centrované?
  - Sú jednotlivé sekcie (stĺpce) tesne vedľa seba alebo sú medzi nimi medzery?
  - Ak je stránka príliš úzka, sú jednotlivé sekcie stále vedľa seba alebo pod sebou?

ZAPAMÄTAJTE SI

Ak chceme niekoľko elementov umiestniť vedľa seba tak, aby mali garantovanú rovnakú **výšku**, vnoríme ich do flexibilného boxu, t.j. elementu (zvyčajne `div`), s vlastnosťou `display: flex`.

Vlastnosti flexibilných boxov

Flexibilným boxom môžeme nastavovať niekoľko vlastností, ktoré ovplyvňujú spôsob rozloženia prvkov v nich. Význam a použitie niektorých z nich si ukážeme na príkladoch a úlohách v tejto časti.

PRÍKLAD 10.20

Na stránke IT Pizza zmeníme hlavičku tak, aby nadpis, logo a telefónne čísla boli na jednom riadku: nadpis vľavo, logo v strede, telefónne čísla vpravo (*obrázok 10.18*).

Obrázok 10.18 Hlavička stránky IT Pizza v troch stĺpcoch.

Využijeme flexibilný box. Budeme vychádzať zo súboru `10/index-flex.html`. Opäť budeme všetky nové vlastnosti pridávať samostatne až za riadok `/* štýly súvisiace s viacstĺpcovým rozložením */`.

Z elementu `header` spravíme flexibilný box pomocou nastavenia `display: flex`. Jeho tri prvky (`h1`, `img` a `div`) chceme rozložiť do jedného riadku tak, aby dva boli na kraji a jeden v strede. Na to použijeme vlastnosť `justify-content` s hodnotou `space-between`. V HTML časti nie je potrebné nič meniť.

```
/* štýly súvisiace s viacstĺpcovým rozložením */
header {
  display: flex;
  justify-content: space-between;
}
```

ÚLOHA 10.21

V predchádzajúcom príklade:

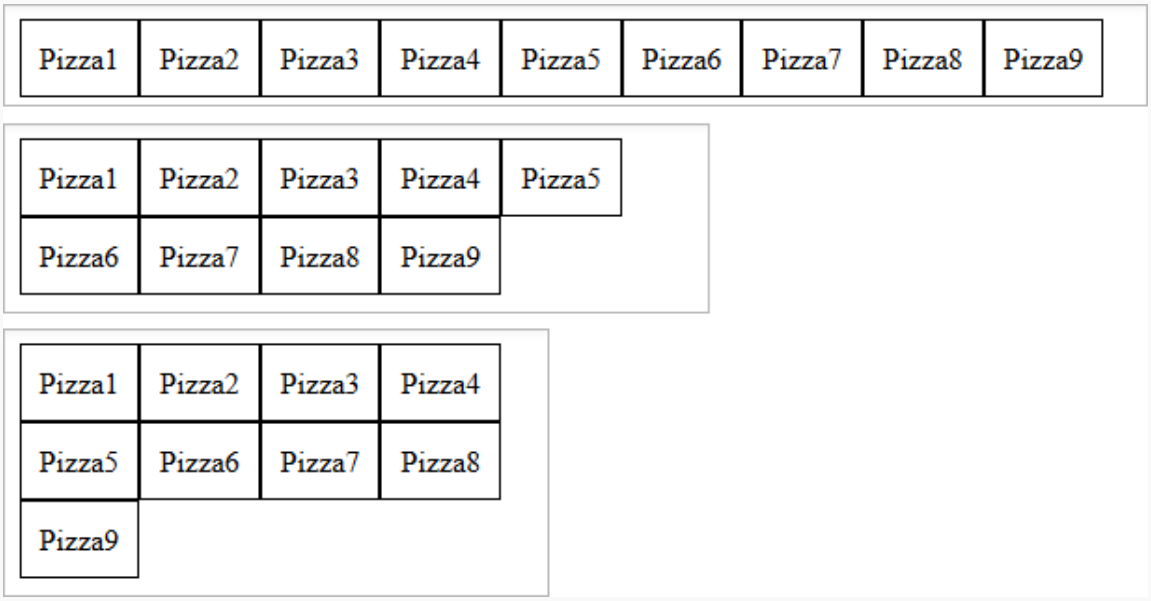
- skúšajte meniť šírku stránky a pozorujte rozloženie jednotlivých častí hlavičky,
- postupne skúšajte ďalšie hodnoty vlastnosti `justify-content`: `flex-start`, `flex-end`, `center` a `space-around`, pozorujte, ako sú rozmiestnené jednotlivé časti hlavičky pri rôznych šírkach stránky.

ÚLOHA 10.22

Zmeňte pätu na stránke IT Pizza z *príkladu 10.20* tak, aby adresy prevádzok v Bratislave, v Banskej Bystrici a kontakty na sociálne siete boli v jednom riadku, ako na *obrázku 10.13* (v *úlohe 10.13*). Využite flexibilný box a jeho vlastnosti.

PRÍKLAD 10.23

Upravíme ponuku pízz na stránke IT Pizza tak, aby sa pizze zobrazovali vedľa seba po riadkoch, pričom počet pízz v jednom riadku bude závisieť od šírky okna prehliadača (*obrázok 10.19*). Vychádzame z kódu z *príkladu 10.20*, resp. *úlohy 10.22*.

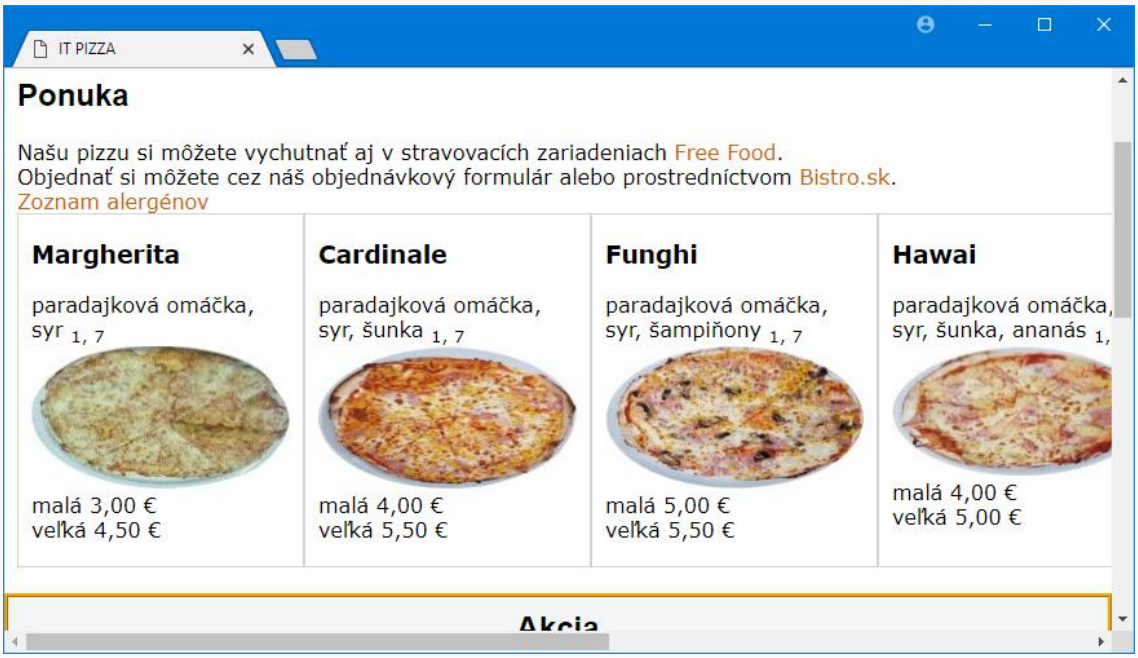


Obrázok 10.19 Schémy zobrazenia ponuky pízz pri rôznych šírkach stránky.

Najprv všetky pizze (teda všetky elementy `article` v sekcii *ponuka*) zabalíme do jedného elementu `div`, z ktorého spravíme flexibilný box. Po definovaní uvedeného štýlu vidíme (*obrázok 10.20*), že kartičky s pizzou sú rôznych šírok, zobrazujú sa všetky v jednom riadku a niektoré z nich ani nevidno.

```
/* štýly súvisiace s viacstĺpcovým rozložením */
...
#pizze {
  display: flex;
}
```

```
...
<section id="ponuka">
  <h2>Ponuka</h2>
  ...
  <div id="pizze">
    <article>
      <h3>Margherita</h3>paradajková omáčka, syr ...<br>
        <img src="obrazky/margerita.jpg" alt="Margherita"...><br>
        malá 3,00 &euro;<br>
      veľká 4,50 &euro;
      </article>
     ...
      <article>
        <h3>Tonno</h3>paradajková omáčka, mozarella, ...<br>
        <img src="obrazky/tonno.png" alt="Tonno"...><br>
        malá 4,00 &euro;<br>
        veľká 5,50 &euro;
    </article>
  </div>
</section>
```



Obrázok 10.20 Ponuka pízz: rôzne široké kartičky, všetky v jednom riadku

Potom v štýle `section article` nastavíme šírku tak, aby sa do nej zmestili všetky informácie o pizzi, a to pre ľubovoľnú pizzu (vhodnú šírku zistíme experimentovaním). Tak zabezpečíme rovnakú šírku kartičiek. Na to, aby sa kartičky zobrazovali vo viacerých riadkoch, ak sa do jedného riadku nezmestia, použijeme vlastnosť `flex-wrap` s hodnotou `wrap`.

```
/* štýly súvisiace s viacstĺpcovým rozložením */
...
#pizze {
  display: flex;
  flex-wrap: wrap;
}
section article {
  width: 380px;
}
```

ÚLOHA 10.24

So stránkou z predchádzajúceho príkladu robte postupne nasledujúce činnosti.

- Meňte šírku stránky a pozorujte, koľko pízz sa zmestí do jedného riadka.
- Vyskúšajte ďalšie hodnoty vlastnosti `flex-wrap`: `nowrap` a `wrap-reverse`. Pozorujte, ako sú rozmiestnené jednotlivé kartičky s pizzou. Skúmajte pri rôznych šírkach stránky.
- Nastavte `flex-wrap` na `wrap` a doplňte do štýlu `#pizze` vlastnosť `justify-content`. Skúšajte rôzne hodnoty `justify-content` a pozorujte rozloženie kartičiek s pizzou pri rôznych šírkach stránky.

ÚLOHA 10.25

V prehliadači zobrazte súbor `10/skumajflex.html`. Otvorte si tiež zdrojový kód súboru a prezrite si ho. V zdrojovom kóde:

- do štýlu pre `div` pridajte nastavenie `align-items`: `flex-start` a pozorujte zobrazenie jednotlivých častí flexibilného boxu, aj pri rôznych šírkach stránky,
- postupne priraďujte vlastnosti `align-items` hodnoty `flex-start`, `flex-end`, `center`, `stretch` a `baseline` a pozorujte zobrazenie častí flexibilného boxu, aj pri rôznych šírkach stránky,
- slovne popíšte, čo podľa vás vlastnosť `align-items` a jej jednotlivé hodnoty znamenajú.

ÚLOHA 10.26

V zdrojovom kóde z predchádzajúcej úlohy postupne vykonajte nasledujúce zmeny.

- Doplňte do štýlu pre flexibilný box nastavenie `flex-direction: row-reverse`. Čo sa zmenilo?
- Zmeňte hodnotu `flex-direction` na `column`.
- Zmeňte hodnotu `flex-direction` na `column-reverse`.
- Sformulujte, čo ovplyvňuje vlastnosť `flex-direction`.

Ak si chcete precvičiť vlastnosti flexibilných boxov, zahrajte sa hru na stránke flexboxfroggy.com.

Prehľad vybraných vlastností flexibilných boxov, ich hodnoty a popis uvádzame v *tabuľke 10.2*. Vždy prvá hodnota z uvedených je prednastavená.

Ukázali sme si niekoľko spôsobov ako rozložiť objekty na stránke do viacerých stĺpcov: absolútne umiestňovanie, plávajúce objekty a flexibilné boxy. Absolútne umiestňovanie je málo flexibilné a ťažšie sa dokáže prispôsobovať rôznym veľkostiam okna prehliadača, vhodnejšie je využiť plávajúce objekty či flexibilné boxy. Ďalšou možnosťou je využitie vhodného frameworku, napr.

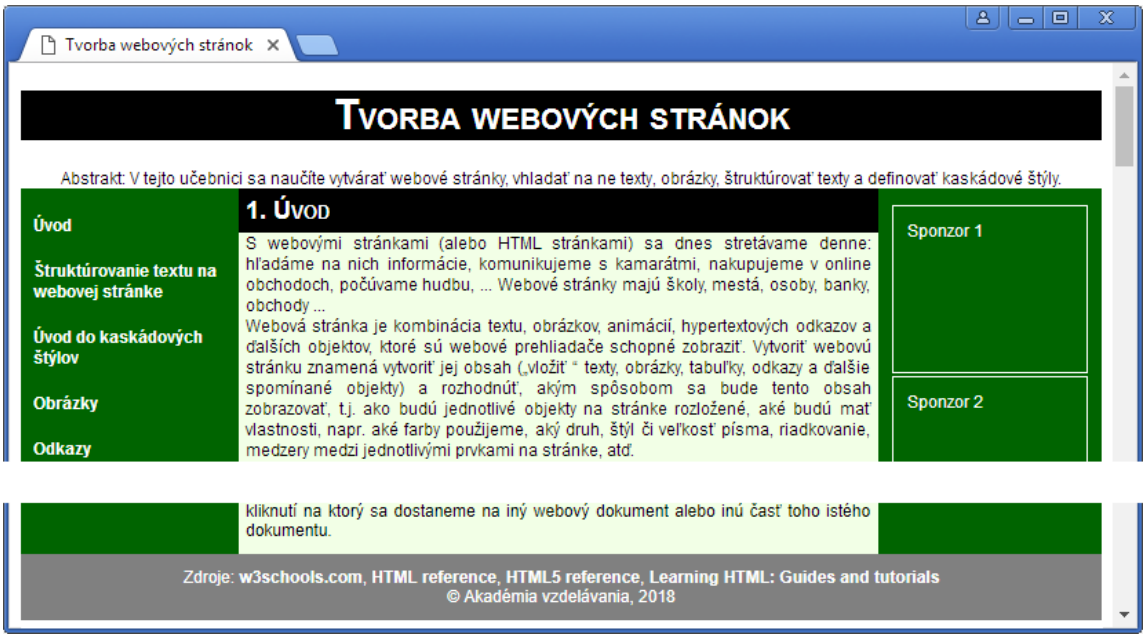
Bootstrap (https://getbootstrap.com/) alebo Foundation (https://foundation.zurb.com/). To je však už nad rámec našej učebnice.

Tabuľka 10.2 Vlastnosti flexibilných boxov.

vlastnosť	hodnoty	popis
`flex-direction`	row, row-reverse, column, column-reverse	definuje, či sa majú prvky flexibilného boxu radiť pod seba alebo vedľa seba; reverse je vždy v opačnom poradí ako poradie v HTML
`flex-wrap`	nowrap, wrap, wrap-reverse	definuje, či sa majú prvky flexibilného boxu zobrazovať v jednom riadku alebo vo viacerých riadkoch, ak sa nezmestia na šírku stránky; nemá zmysel, ak sú prvky flexibilného boxu v jednom stĺpci
`justify-content`	flex-start, flex-end, center, space-around, space-between	definuje spôsob vodorovného rozloženia prvkov flexibilného boxu
`align-items`	stretch, flex-start, flex-end, center, baseline	definuje zvislé zarovnanie prvkov flexibilného boxu

ÚLOHA 10.27

Pre stránku `10/ucebnica.html` vytvorte trojstĺpcový dizajn (*obrázok 10.21*): hore bude hlavička, pod ňou v ľavom stĺpci navigácia, v strede jednotlivé kapitoly a v pravom stĺpci logá sponzorov, dolu pod všetkým bude päta. Zvoľte si spôsob, ktorý považujete za najvhodnejší.



Obrázok 10.21 Trojstĺpcový vzhľad stránky s učebnicou.

Metodika pre učiteľa

CIEĽ

Cieľom je oboznámiť sa a vyskúšať si rôzne spôsoby umiestňovania objektov.

VÝKLAD

Pri riešení príkladov a úloh v tejto kapitole je vhodné vytvorené stránky skúšať s rôznymi rozmermi okna prehliadača (či rôznymi rozmermi časti Result v editore JSFiddle). Pre túto kapitolu je obzvlášť dôležité experimentovanie.

Umiestňovanie objektov je pomerne náročná a rozsiahla téma, mohla by byť sama o sebe témou pre celú učebnicu. My uvádzame len niekoľko príkladov a spôsobov, ako rozložiť objekty na stránke.

Nadväznosť na predchádzajúce kapitoly: Kapitola vyžaduje kapitoly 1-8. Táto kapitola je veľmi náročná, odporúčame jej zaradenie len pre pokročilých, alebo žiakov, ktorí majú hlbší záujem. Viaže sa na ňu len kapitola 11 (čiastočne aj kapitoly 16-18), ostatné nie. V prípade jej zaradenie do výučby, odporúčame kapitolu realizovať na minimálne 3 vyučovacích hodinách.

Absolútne umiestňovanie

Príklad 10.5: Pre element `section` môžeme nastaviť hodnotu `position` na ľubovoľnú okrem `static`, čo je prednastavená hodnota, t.j. nemusí to byť práve hodnota `relative`.

Plávajúce umiestňovanie

Úloha 10.9: Žiaci by mali prísť na to, že plávajúci objekt neobteká tie objekty, ktoré sú v HTML kóde definované pred ním, ale len tie, čo sú v HTML za ním.

Úloha 10.13 - popis riešenia

- Potrebujeme tri elementy `div`: jeden pre kontakt v Bratislave, jeden pre kontakt v Banskej Bystrici a jeden pre socialne kontakty.
- Prvý a druhý element `div` budú mať rovnaké vlastnosti (`float: left`, šírka napr. 40%), preto si pre ne vytvoríme triedu (napr. `.adresa`).
- Pravý `div` bude mať tiež `float: left` a šírku zvyšných 20%. Dáme mu identifikátor napr. `social` a vytvoríme štýl pomocou `#social`. Môžeme (nemusíme) pridať zarovnanie textu vpravo.
- Odseku s kopirajtom (`footer p`) nastavíme `clear: both`.

Box-sizing

Na zváženie učiteľa ešte nechávame spomenutie vlastnosti `box-sizing`. Ak pre celú stránku nastavíme `*{box-sizing: border-box;}`, potom sa vnútorné okraje (`padding`)

počítajú do šírky boxu, a teda ak má element nastavenú šírku 30%, tak zaberá na stránke 30% bez ohľadu na to, aké má vnútorné okraje.

Vlastnosť display

Vlastnosť `display` nemení typ elementu, len jeho zobrazenie. Napríklad, ak elementu `<em>` definujeme `display: block`, bude sa síce zobrazovať ako blokový, ale nemôžeme do neho vnárať iné blokové elementy.

Flexibilné boxy

Flexibilné boxy sú podporované až v novších verziách prehliadačov (v Chrome od verzie 29, v IE/Edge od verzie 11, v Mozille od verzie 22, v Safari od verzie 10 a v Opere od verzie 48).

Úloha 10.22 - popis riešenia: V HTML treba doplniť flexibilný `div` pre uvedené tri prvky, lebo ním nemôže byť `footer` (v elemente `footer` je ešte aj nadpis a posledný odsek, ktoré nemajú byť vedľa seba). Flexibilný `div` musí obaliť len tie elementy, ktoré majú byť vedľa seba, čiže začať mal za `<h2>Kontakt</h2>` a skončiť mal pred `<p>IT akadémia... </p>`. Preň zadefinujeme napr. triedu s vlastnosťami `display: flex` a `justify-content: space-around`.

Úlohu 10.27 možno využiť na opakovanie.

Užitočné zdroje

- https://www.w3schools.com/css/css_website_layout.asp
- https://www.w3schools.com/css/css3_box-sizing.asp
- https://www.w3schools.com/css/css3_flexbox.asp
- http://flexboxfroggy.com
- http://interval.cz/clanky/trisloupcovy-layout-s-hlavickou-a-patickou/
- http://interval.cz/clanky/trisloupcovy-layout-webu-pomoci-css/
- http://interval.cz/clanky/trisloupcovy-layout-svaty-gral/
- http://www.alistapart.com/articles/flexiblelayouts/
- http://www.csszengarden.com/
- Meyer E.: Eric Meyer o CSS – ovládněte kaskádové styly (projekt 9)
- Cederholm D.: Webdesign s webovými standardy
- Cederholm D.: Flexibilní web design

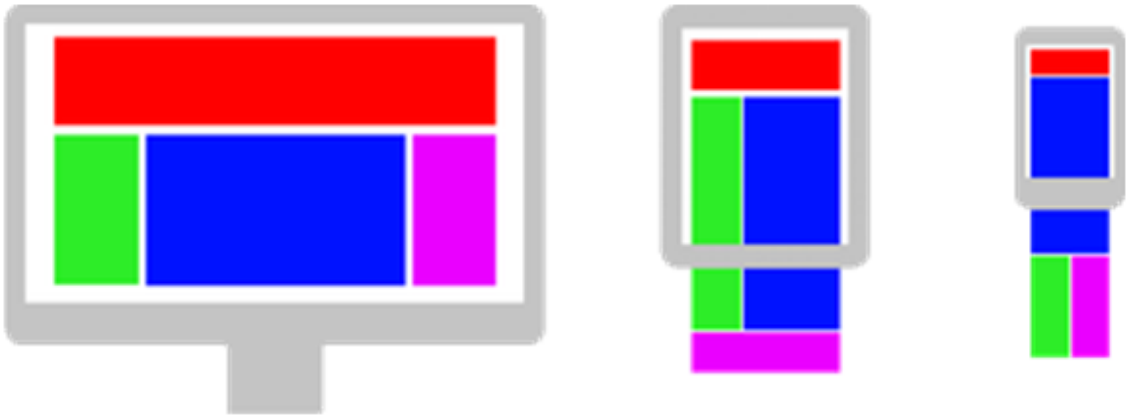
11 RESPONZÍVNY WEB A ŠTÝLY PRE RÔZNE ZOBRAZOVACIE ZARIADENIA, ŠTÝL PRE TLAČ

Webové stránky si pozeráme na rôznych zariadeniach: počítačoch, notebookoch, tabletoch či mobiloch. Každé z týchto zariadení má inú veľkosť zobrazovacej plochy - to, čo sa zmestí na šírku obrazovky počítača, sa v rovnakom tvare určite nezmestí na obrazovku mobilu. Preto by tvorcovia stránok mali stránky vytvárať tak, aby sa správne zobrazovali na všetkých zariadeniach. Nazývame to responzívny web (resp. responzívny dizajn) - t.j. taký web (dizajn), ktorý sa automaticky prispôsobí zariadeniu, na ktorom je zobrazený.[2]

ÚLOHA 11.1

Nájdite príklady stránok, ktoré sa inak zobrazujú na počítači, inak v tablete či v mobile.

V čom sa líši počítačová verzia stránky od tej mobilnej? Aké je rozloženie jednotlivých prvkov? Sú tieto verzie obsahovo rovnaké?



Obrázok 11.1 Zobrazenie obsahu webovej stránky na počítači, tablete a smartfóne.

Ako vytvárať responzívne stránky? Máme nasledujúce dve možnosti.

- Môžeme použiť niektorý z existujúcich CSS frameworkov, ktoré už majú predpripravené responzívne prvky a umožňujú škálovanie. Medzi najznámejšie patria Bootstrap, Skeleton, Foundation.
- Môžeme vytvárať rozloženie stránky „od nuly", t.j. písať vlastný kód, v ktorom využívame len HTML a CSS. Responzívnosť dosahujeme v podstate tým, že vytvárame viacero verzií štýlov: verziu pre mobilné zariadenia, verziu pre tablety, verziu pre počítače.

[2] Viete, že Google od roku 2015 umiestňuje responzívne webstránky vyššie vo vyhľadávaní?

Pri tvorbe webu by sme už od začiatku mali myslieť na to, aby bol náš web responzívny. V súčasnosti sa za optimálny prístup považuje tzv. **mobile first** prístup, teda že najskôr vytvoríme najjednoduchšiu verziu webu vhodnú pre mobily a postupne pridávame vlastnosti pre rôzne ostatné zariadenia.

Media queries

Pri vytváraní responzívneho webu využívame tzv. **media-queries**. Media queries sú niečo ako podmienky a pravidlá, pomocou ktorých môžeme určiť, čo a ako zobrazovať na konkrétnom zariadení v závislosti od jeho vlastností, napr. šírky, výšky.

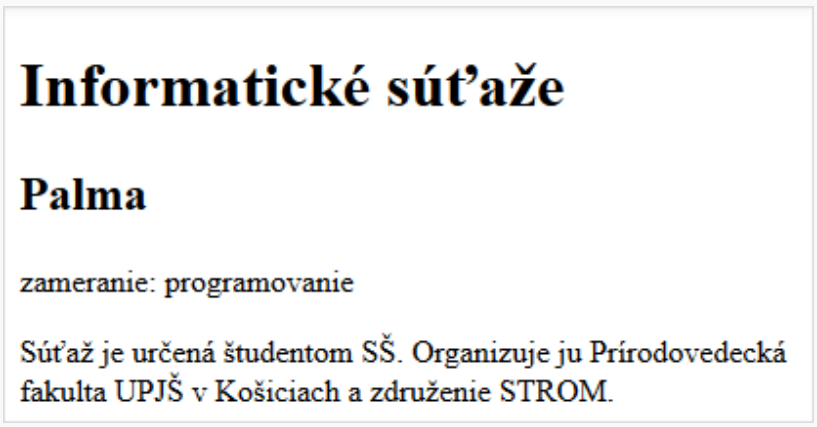
PRÍKLAD 11.2

V editore JSFiddle skopírujeme do časti JSFiddle obsah súboru `11/sutaze.txt`. Pomocou CSS upravíme stránku tak, že v prípade jej zobrazenia na obrazovke so šírkou minimálne 800px, sa zväčší font na 120% a zmení farba textu na tmavomodrú (*obrázok 11.1*), inak bude farba textu a veľkosť fontu štandardná (*obrázok 11.2*).

```
@media screen and (min-width: 800px) {
  body {
      font-size: 120%;
      color: darkblue;
  }
}
```

Otestujme zmenou veľkosti okna so stránkou.

Obrázok 11.2 Zobrazenie pri šírke aspoň 800px (zväčšená veľkosť písma, modrá farba textu).



Obrázok 11.3 Zobrazenie pri šírke menšej ako 800px (štandardná veľkosť písma, štandardná farba textu).

Všimnime si v CSS kóde riadok `@media screen and (min-width: 800px)`. To je **media query** – akási podmienka, ktorá ak je splnená, tak sa na stránku aplikujú štýly definované v zátvorkách {} za ňou, v našom príklade štýl pre element `body`. My budeme zväčša využívať media queries v tvare:

```
@media typ_média and vlastnosť_média
```

Význam jednotlivých častí je nasledovný:

- pomocou `@media` povieme, že ideme definovať media query,
- `typ_média` určuje typ média, pre ktorý chceme definovať štýly. Najčastejšie sa stretneme s hodnotou `screen` (obrazovka počítača, tabletu, smartfónu,...), ale môže mať aj hodnotu `print` (tlačiareň), `speech` (čítač obrazovky) alebo `all` (všetky uvedené),
- `vlastnosť_média` definuje nejakú vlastnosť média, pre ktorú sa má aplikovať daný štýl, napr.:
- `min-width`,
- `max-width`,
- `min-height`,
- `max-height`,
- `orientation` – môže mať hodnoty `portrait` (prednastavená) a `landscape`.

PRÍKLAD 11.3

Nastavme stránke z predchádzajúceho príkladu sivé pozadie v prípade, že je zobrazená na výšku a pozadie farby `#FFCCCC`, ak je zobrazená na šírku. Taktiež nastavme font pre text na `Verdana`, a to pri ľubovoľnej orientácii stránky.

Štandardná orientácia stránky je na výšku, preto pre nastavenie sivého pozadia nemusíme definovať žiaden media-query, stačí pridať klasický štýl pre element `body` s vlastnosťou `background-color: gray;`. Ak pri orientácii stránky na šírku chceme nastaviť inú farbu pozadia, musíme zadefinovať media-query s podmienkou `screen and (orientation: landscape)` a v rámci neho definovať štýl pre body s nastavením `background-color: #FFCCCC;`. Nastavenie fontu `Verdana` stačí zadefinovať v štýle pre `body` mimo media-query. Tento štýl sa totiž použije na všetkých zariadeniach, aj tých s orientáciou na výšku aj tých s orientáciou na šírku.

```
body {
  background-color: gray;
  font-family: Verdana;
}
@media screen and (min-width: 800px) {
  body {
      font-size: 120%;
      color: darkblue;
  }
}
@media screen and (orientation: landscape) {
  body {
    background-color: #FFCCCC;
  }
}
```

ÚLOHA 11.4

Doplňte CSS kód z predchádzajúceho príkladu tak, aby nadpisy úrovne 2 boli centrované a pri šírke okna aspoň 800px zarovnané vpravo. Svoj kód otestujte – meňte veľkosť okna tak, aby raz bola šírka väčšia ako výška, inokedy výška väčšia ako šírka.

Ak si stránky vytvorené v *príkladoch 11.2* až *11.4* prezeráme na počítači, vidíme, že na zmenu veľkosti okna reagujú tak, ako by sme očakávali. Ak by sme ich testovali na mobilnom telefóne, s najväčšou pravdepodobnosťou by sa zobrazili vo verzii pre šírku aspoň 800px. K verzii pod 800px by sme sa vôbec nedopracovali (možno na staršom type mobilu). Nebudeme sa zaoberať dôvodmi, vysvetlíme si radšej, ako zabezpečiť, aby sa stránky využívajúce media queries naozaj správne zobrazovali na všetkých obrazovkách pri akejkoľvek šírke.

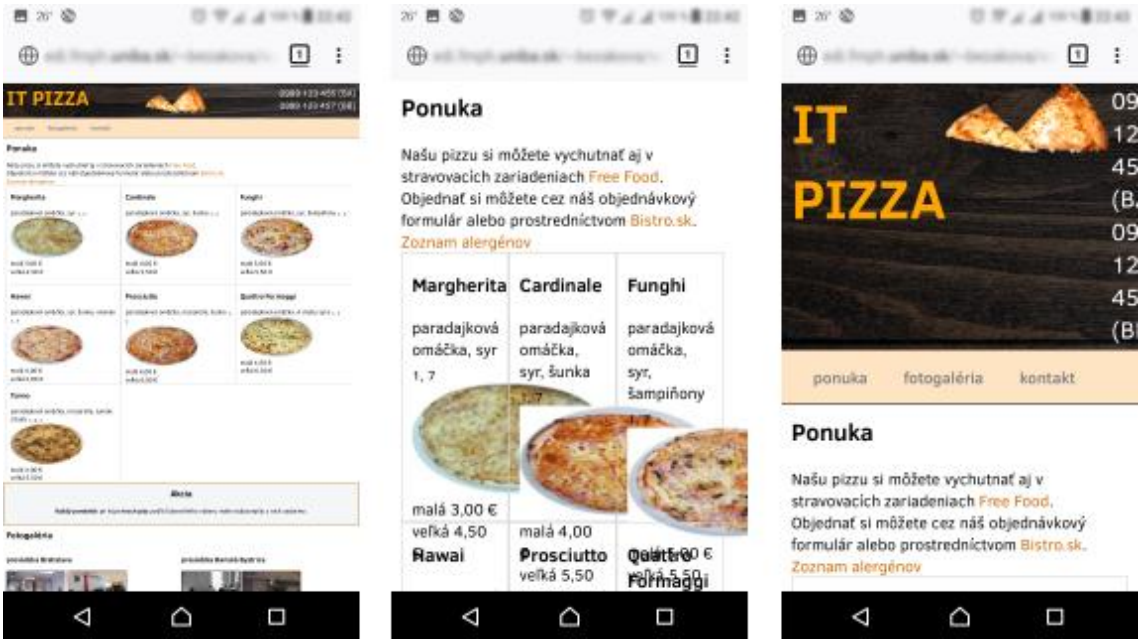
Nevyhnutnou súčasťou HTML kódu responzívnych stránok je element `<meta name="viewport" content="width=device-width, initial-scale=1.0">`, **ktorý pridávame do elementu** `head`. Tým povieme, že šírka stránky bude zodpovedať šírke zariadenia (nie rozlíšeniu!) a nastaví sa počiatočná úroveň škálovania (zoom-u). Bez tohto meta elementu nebude zobrazovanie stránky na mobilných zariadeniach správne fungovať, aj napriek použitiu vhodných media queries.

Responzívny dizajn stránky IT Pizza

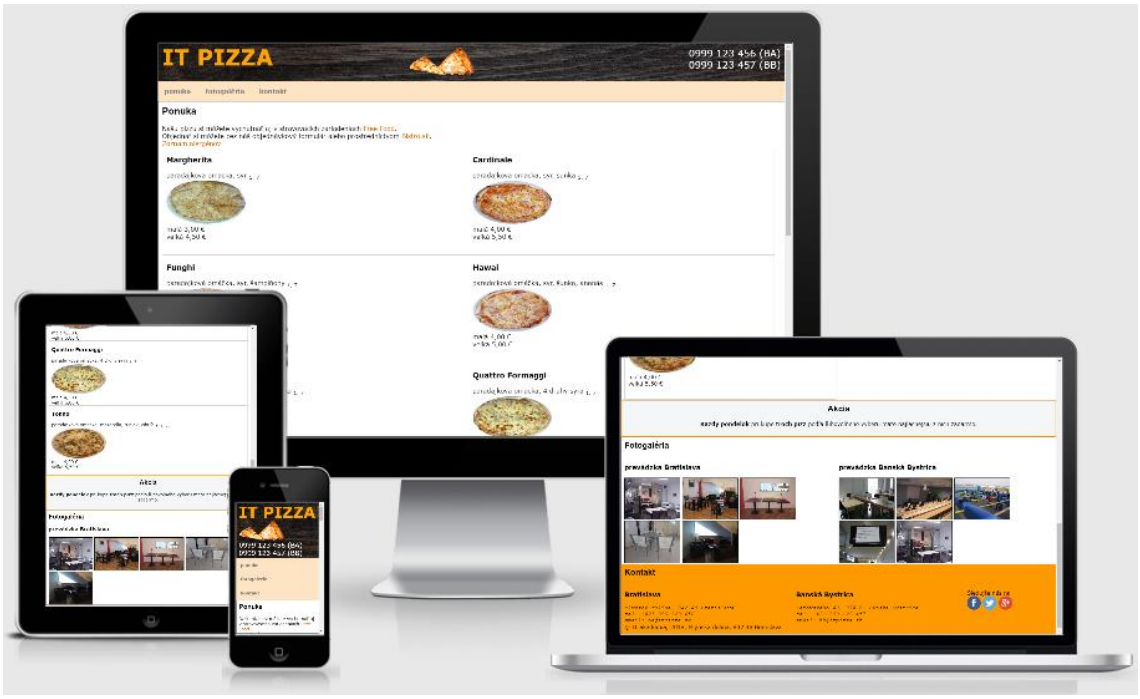
Až po 9. kapitolu (vrátane) mala stránka IT Pizza v podstate jednostĺpcový dizajn. Keby sme do tejto verzie stránky pridali uvedený `meta` element `viewport`, zobrazovala by sa na mobiloch správne. Dá sa teda povedať, že sme stránku vytvárali prístupom *mobile first*.

V 10. kapitole sme pre jednotlivé časti stránky vytvorili viacstĺpcové rozloženie objektov: hlavičku a pätu sme rozdelili do troch stĺpcov, galériu do dvoch stĺpcov, ponuku pízz sme spravili flexibilnú – počet kartičiek v riadku závisí od šírky stránky. Toto rozloženie sme realizovali prevažne pomocou flexibilných boxov (okrem galérie, ktorú sme realizovali pomocou plávajúcich objektov, ale nie je zložité prerobiť toto riešenie použitím flexibilných boxov). Toto riešenie, tak ako je zatiaľ vytvorené, nie je vhodné pre zobrazovanie stránky v mobile (*obrázok 11.4*). V lepšom prípade by prehliadač v mobile stránku zmenšil tak, že by texty boli nečitateľne malé (*obrázok 11.4 vľavo*). V horšom prípade by sa niektoré objekty prekrývali (*obrázok 11.4 v strede*) alebo by niektoré prvky nebolo vidno (*obrázok 11.4 vpravo*) – to sme mohli vidieť aj na počítači, ak sme príliš zúžili okno prehliadača.

Upravíme stránku IT Pizza tak, aby bola responzívna: na zariadeniach s menšou zobrazovacou plochou sa objekty zobrazia v jednom stĺpci, na obrazovke so šírkou aspoň 800px sa rozložia do dvoch, resp. troch stĺpcov (*obrázok 11.5*). Budeme upravovať verziu stránky, v ktorej sme použili flexibilné boxy.



Obrázok 11.4 Zlé zobrazenia stránky IT Pizza na mobile.



Obrázok 11.5 Požadované zobrazenie stránky IT Pizza na rôznych zariadeniach (obrázok bol vytvorený pomocou http://ami.responsivedesign.is)

Teraz oceníme, že sme všetky vlastnosti, ktoré súvisia s umiestňovaním objektov (`display:flex`, a s tým súvisiace vlastnosti, nastavenia rozmerov) definovali samostatne. Vďaka tomu nebudeme musieť nastavovať žiadne nové štýly či pridávať vlastnosti, stačí len definície týchto "umiestňovacích" nastavení „obaliť" vhodným media-query.

PRÍKLAD 11.5

Pomocou media queries upravíme stránku IT Pizza riešenú pomocou plávajúcich objektov (súbor `11/index.html`) tak, aby sa len pri šírke aspoň 800px zobrazovala vo viacerých stĺpcoch. Všetky definície vlastností súvisiace s umiestňovaním (tie, čo sme vkladali za riadok /* štýly súvisiace s viacstĺpcovým rozložením */) vložíme do media query @media screen and (min-width: 800px).

Najskôr pridajme do elementu `head` spomínaný dôležitý meta element:

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>IT PIZZA</title>
  ...
```

Potom v elemente `style` definujme media query:

```
/* štýly súvisiace s viacstĺpcovým rozložením */
@media screen and (min-width: 800px) {
  header {
    display: flex;
    justify-content: space-between;
  }

  .flex {
    display: flex;
    justify-content: space-around;
  }

  #pizze {
    display: flex;
    flex-wrap: wrap
  }

  section article {
    width: 380px;
  }
}
```

Ako overíme, či je stránka naozaj responzívna? Ak máme stránku len na lokálnom počítači, testujeme ju ako doteraz pri rôznych veľkostiach okna prehliadača. V prípade, že už máme stránku umiestnenú na webovom serveri, môžeme použiť niektorý dostupný testovač responzivity, napr. Am I Responsive? (http://ami.responsivedesign.is/) alebo Mobile Friendly Test (https://search.google.com/test/mobile-friendly).

ÚLOHA 11.6

Pomocou media queries upravte stránku zo súboru `11/vitaminy.html`, aby sa pri šírke aspoň 800px zobrazovala v dvoch stĺpcoch (v ľavom navigácia, v pravom text o vitamínoch), inak v jednom. Nezabudnite na meta element!

Všimnite si, že na stránke IT Pizza sa odkazy pri malej šírke nezobrazujú dobre (*obrázok 11.6*). Chceli by sme, aby sa zobrazovali pekne vedľa seba pri väčšej šírke a pod sebou pri menšej.

Obrázok 11.6 Zlé zobrazenie odkazov.

PRÍKLAD 11.7

Upravme navigáciu na stránke IT Pizza tak, aby sa odkazy zobrazovali pod sebou pri šírke do 800px a vedľa seba pri šírke aspoň 800px (*obrázok 11.7*). Budeme vychádzať zo súboru z *príkladu 11.5*.

Znova využijeme flexibilný box. Jeho prvkom totiž dokážeme jednoducho nastaviť, či sa majú zobrazovať pod sebou alebo vedľa seba – pomocou vlastnosti `flex-direction`. Flexibilným boxom bude navigácia (z elementu `nav`). Jej prvky, teda odkazy, sa budú štandardne zobrazovať pod sebou (`flex-direction: column`) a pri šírke aspoň 800px vedľa seba (`flex-direction: row`). Kvôli krajšiemu vzhľadu ešte zrušíme `padding` pre element `nav`.

```
nav {
 background-color: #FFE4C4;
 border-bottom: 1px solid black;
 padding: 1em;
 display: flex;
 flex-direction: column;
}
...
/* štýly súvisiace s viacstĺpcovým rozložením */
@media screen and (min-width: 800px) {
  ...
  nav {
    flex-direction: row;
  }
}
```

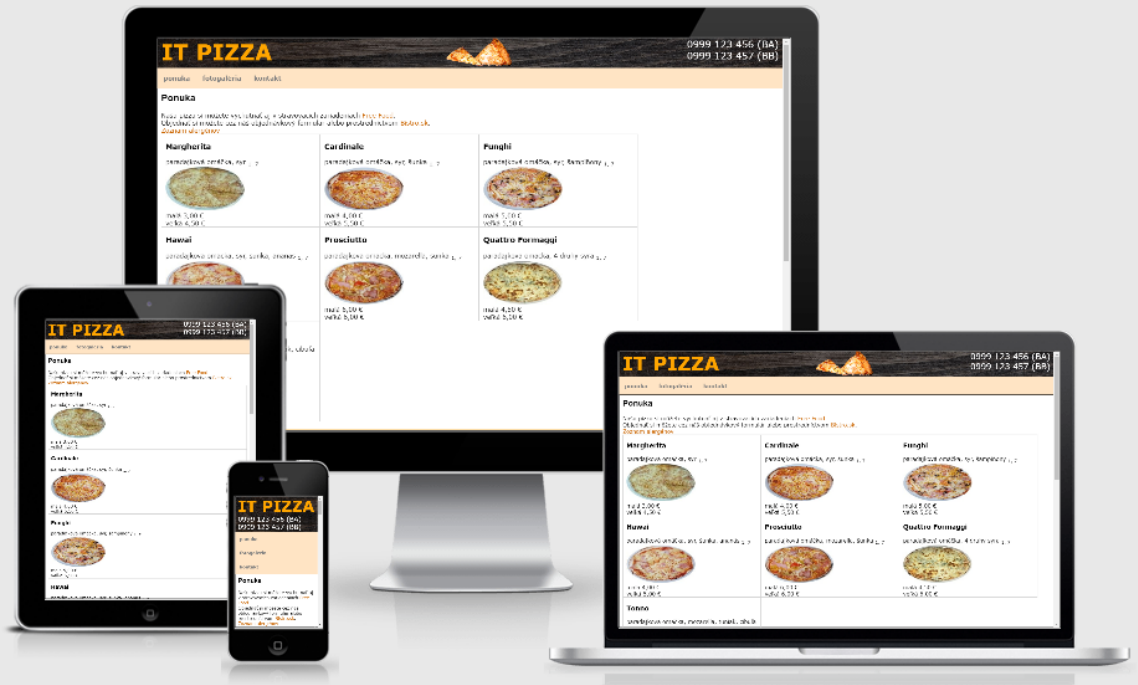
Obrázok 11.7 Zvislá alebo vodorovná navigácia v závislosti od šírky stránky.

Zatiaľ sme rozlišovali dve rôzne zobrazenia stránky IT Pizza, pri šírke do 800px a pri šírke aspoň 800px. Pomocou media queries však môžeme definovať aj viacero rôznych verzií, napr. úzku, strednú a širokú.

PRÍKLAD 11.8

Upravíme stránku IT Pizza z predchádzajúceho príkladu tak, aby:

- sa odkazy zobrazovali do šírky 400px pod sebou a od šírky 400px vedľa seba,
- sa logo stránky zobrazovalo až od šírky 800px,
- nadpis a telefónne čísla boli pod sebou pri šírke do 400px, v jednom riadku pri šírke aspoň 400px, pričom od šírky 400px sa medzi nimi zobrazí aj logo (*obrázok 11.8*).



Obrázok 11.8 Zobrazenie stránky IT Pizza pri rôznych šírkach.

Zhrnieme, ako sa majú zobrazovať časti stránky pri vymedzených šírkach. Pomôže nám to pri rozhodovaní, čo dať do ktorého media query a do ktorého štýlu.

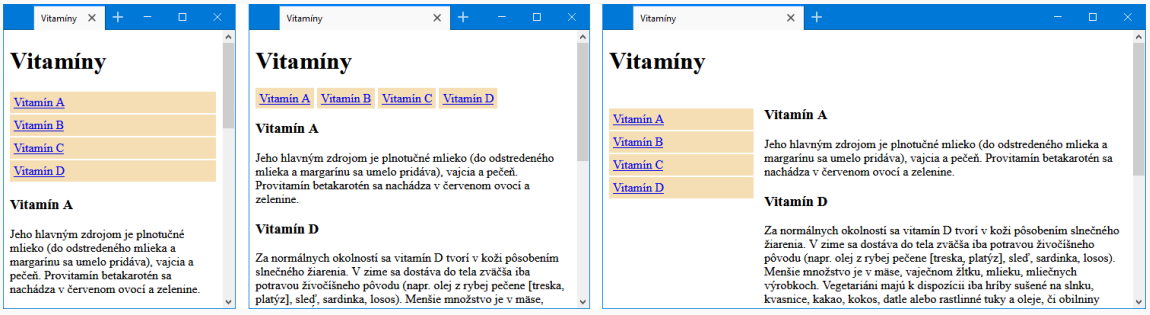
časť stránky/ šírka	**ľubovoľná šírka**	**šírka aspoň 400px**	**šírka aspoň 800px**
hlavička	bez loga, nadpis a telefónne čísla pod sebou	bez loga, nadpis a telefónne čísla v jednom riadku	nadpis, logo a telefónne čísla v jednom riadku
odkazy	pod sebou	vedľa seba	vedľa seba
zvyšok stránky	jednostĺpcový vzhľad	jednostĺpcový vzhľad	viacstĺpcový vzhľad

V časti s CSS kódom (v elemente `style`) budeme mať teda skupinu „základných štýlov" pre zobrazovanie stránky v ľubovoľnej šírke a ďalšie dve skupiny štýlov v dvoch media queries, jednu pre zobrazovanie stránky v šírke aspoň 400px a druhú pre zobrazovanie stránky v šírke aspoň 800px. Skrytie loga vyriešime pomocou nastavenia `display: none` a jeho zobrazenie pomocou `display: inline`, pretože obrázok je štandardne riadkový element. V nasledujúcom kóde si všimnite aj komentáre vpravo:

```
...
header img {
  display: none;     /* v základnom štýle obrázok skryjeme */
}
nav {
 background-color: #FFE4C4;
 border-bottom: 1px solid black;
 display: flex;              /* navigácia bude flexibilným boxom */
 flex-direction: column;  /* odkazy budú pod sebou v stĺpci */
}
...
/* štýly súvisiace s viacstĺpcovým rozložením */
@media screen and (min-width: 400px) {
  nav {
   flex-direction: row;      /* odkazy budú vedľa seba v riadku */
  }
  header {
    display: flex;           /* hlavička bude flexibilným boxom  */
    justify-content: space-between;
      /* časti hlavičky budú vedľa seba oddelené medzerou */
  }
}
@media screen and (min-width: 800px) {
  header img {
    display: inline;         /* zobrazíme obrázok */
  }
  .flex {
    display: flex;
    justify-content: space-around;
  }
  #pizze {
    display: flex;
    flex-wrap: wrap;
  }
  section article {
    width: 380px;
  }
}
```

ÚLOHA 11.9

Upravte stránku Vitamíny z *úlohy 11.6* tak, aby sa odkazy v navigácii zobrazovali vedľa seba pri šírke do 400px a aspoň 800px, inak vedľa seba. Pre rozlloženie na 1 resp. 2 stĺpce platia rovnaké podmienky ako v *úlohe 11.6*. Nepoužívajte ďalšie flexibilné boxy, len rozumne nastavte hodnotu vlastnosti `display`.



Obrázok 11.9 Zobrazenie stránky Vitamíny pri rôznych šírkach.

Štýl pre tlač

Webové stránky čítame zväčša priamo v mobile, či v počítači. Niekedy si však chceme stránku vytlačiť, napr. cestovný poriadok, recept na koláč, text piesne.

DISKUTUJTE

Už ste niekedy tlačili webovú stránku? Zhodovala sa tlačená verzia stránky s elektronickou? Myslíte, že je potrebné, aby sa stránka vytlačila v rovnakej podobe, ako vyzerá na počítači alebo v mobile? Ktoré časti stránky IT Pizza by podľa vás mali byť v jej tlačenej verzii?

V predchádzajúcej podkapitole sme definovali štýly pre obrazovky (screen) rôznych elektronických zariadení. Pomocou media queries dokážeme definovať aj tzv. **štýl pre tlač**, t.j vieme ovplyvniť, ako bude vyzerať stránka vytlačená na papieri.

Štýl pre tlač definujeme pomocou media-query: `@media print {}`, pričom v zátvorkách `{}` definujeme potrebné vlastnosti elementov. Treba si uvedomiť, že ak máme na stránke definovaný aj iný štýl, ktorý je platný pre všetky zariadenia, tak sa vlastnosti štýlov zlučujú, čo môže mať vplyv na konečné zobrazenie stránky.

PRÍKLAD 11.10

Vytvorme štýl pre tlač stránky IT Pizza z *príkladu 11.8*. Najdôležitejšími informáciami na stránke je ponuka pízz a kontakty. Pri tlači môžeme vynechať logo stránky, navigáciu, akciu, fotogalériu a kontakty na sociálne siete. Elementy prislúchajúce týmto častiam či prvkom teda v štýle pre tlač nezobrazíme, čo nám umožní nastavenie `display: none`. Textom v hlavičke (nadpis, telefónne čísla) nastavíme farbu na čiernu. Odkazom nastavíme farbu písma na sivú a podčiarkneme ich.

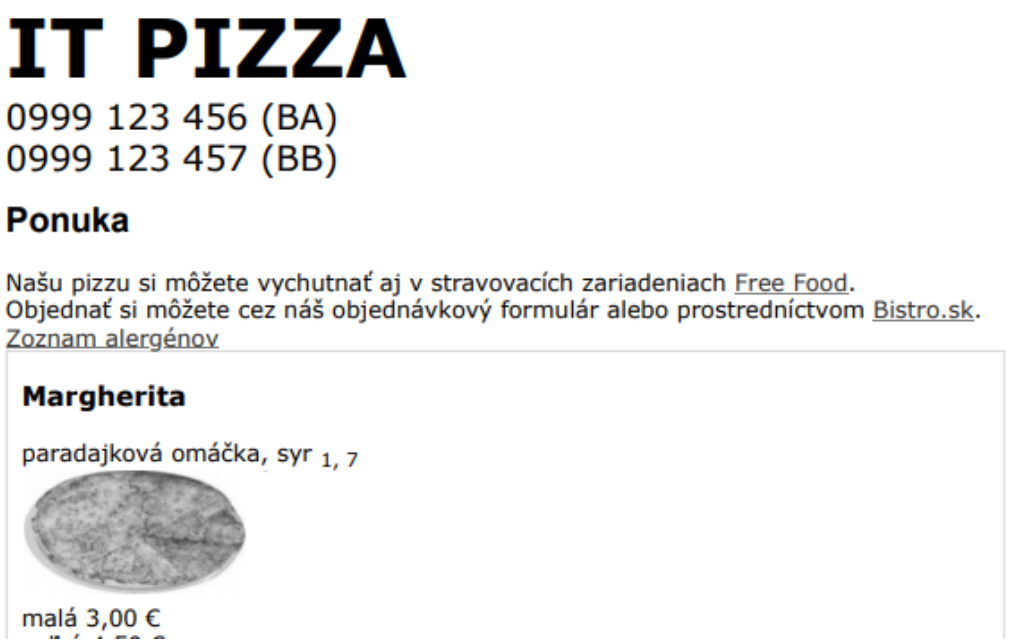
Do elementu `style` doplníme nasledujúci kód:

```
@media print {
  header img, nav, #galeria, aside, #social {
    display: none;
  }
  header h1, header div  {
    color: black;
  }
  a {
    color: gray;
    text-decoration: underline;
  }
}
```

Štýl pre tlač otestujeme tak, že si zobrazíme ukážku stránky pred tlačou (obrázok 11.10).

ÚLOHA 11.11

Pre stránku Vitamíny z *úlohy 11.9* vytvorte štýl pre tlač, v ktorom iba skryjete navigáciu. Pozrite si ukážku pred tlačou.



Obrázok 11.10 Stránka IT Pizza vo verzii pre tlač.

ZAPAMÄTAJTE SI

V štýle pre tlač zvykneme:

- skrývať objekty, ktoré pre tlač nie sú dôležité, napríklad navigáciu, niektoré obrázky dekoratívneho či doplňujúceho charakteru, rôzne reklamy, …
- nastaviť všetkým textom čiernu farbu písma,
- upraviť šírky potrebných elementov tak, aby sa pri tlači využil aj priestor, ktorý sa vytvoril skrytím iných elementov.

Externé štýly

Kaskádové štýly, ktoré sme zatiaľ definovali, by sme mohli označiť ako ***interné kaskádové štýly***. Interné preto, lebo sme ich definovali priamo v tom súbore, v ktorom sme ich aj použili. Tento spôsob je vhodný vtedy, ak definovanú skupinu kaskádových štýlov používa len jediná stránka, ako napríklad naša stránka IT Pizza.

V prípade, že webové sídlo tvorí viacero stránok, ktoré majú mať rovnaký vzhľad, je použitie interných štýlov neefektívne. Všetky zmeny, ktoré by sme spravili v štýloch na jednej stránke, by sme totiž museli kopírovať aj na ostatné stránky. Preto sa pri tvorbe webových sídiel s viacerými stránkami využívajú ***externé kaskádové štýly***. Externé znamená len to, že sú umiestnené v inom (externom) súbore, nie priamo v súbore s HTML kódom.

Externé kaskádové štýly môžeme písať v ľubovoľnom textovom editore, zväčša používame ten istý editor, v ktorom píšeme HTML kód. Definície externých štýlov, jednotlivé vlastnosti a ich

hodnoty, zapisujeme úplne rovnako ako pri interných štýloch. Súbor so štýlmi musíme uložiť s príponou `.css`. Tento súbor nesmie obsahovať žiadne HTML značky.

Súbor s kaskádovým štýlom musíme „pripojiť" ku všetkým HTML súborom, ktoré ho majú používať. To znamená, že v HTML kóde stránky musíme zadefinovať, že chceme, aby stránka používala daný súbor so štýlmi. Na to slúži element `<link>`, ktorý sa vkladá do elementu `<head>` v tvare: `<link href="subor.css" rel="stylesheet">`. Pomocou atribútu `href` určujeme, aký súbor chceme pripojiť k webovej stránke.

PRÍKLAD 11.12

Pre stránku so súťažami (`11/sutaze.html`) vytvoríme externý štýl `styl.css`, v ktorom nastavíme písmo pre celú stránku na `Arial` a farbu písma pre nadpisy úrovne 2 na tmavomodrú. Pripojíme štýl k stránke.

V editore, ktorý zvyčajne používame na písanie HTML kódu, vytvoríme nový súbor a napíšeme doň nasledujúci kód:

new 1

```
1  body {
2    font-family: Arial;
3  }
4
5  h2 {
6    color: darkblue;
7  }
```

Súbor uložíme ako `styl.css` do toho istého priečinka, v ktorom je umiestnený súbor `sutaze.html`. Do HTML kódu pridáme element `link` na pripojenie štýlu:

```
<head>
  <meta charset="utf-8">
  <title>Informatické súťaže</title>
  <link href="styl.css" rel="stylesheet">
</head>
}
```

ÚLOHA 11.13

V priečinku `11/ucebnica` máme pripravené stránky s piatimi kapitolami učebnice (`kap1.html`, `kap2.html`, …, `kap5.html`). a súbor `styly-uceb.css` so štýlmi pre učebnicu.

- Pripojte štýl zo súboru `styly-uceb.css` do všetkých kapitol učebnice.
- Skontrolujte, či všetky kapitoly učebnice vyzerajú rovnako.
- V súbore `styly-uceb.css` zmeňte nastavenie písma pre nadpisy na `Verdana` a súbor uložte.

Overte, či sa písmo zmenilo vo všetkých kapitolách učebnice.

V predchádzajúcich častiach sme sa naučili vytvárať štýly pre rôzne zariadenia. Aj tieto môžeme vytvoriť ako externé štýly. V elemente `link` môžeme totiž určiť, pre aké zariadenie sa má uvedený štýl použiť, a to pomocou atribútu `media`. Atribút `media` môže nadobúdať hodnoty:

- `all` – štýly pre všetky zariadenia (prednastavená hodnota, ak atribút neuvedieme),
- `screen` – štýly pre obrazovku,
- `projection`, `tv` – štýly pre projekciu a televízor,
- `print` – štýly pre tlač.

Napríklad:

- štýl pre ľubovoľnú obrazovku pripojíme pomocou elementu `<link href="styl.css" rel="stylesheet" media="screen">`,
- štýl pre tlač pripojíme pomocou elementu `<link href="styl.css" rel="stylesheet" media="print">`.

Okrem typu zariadenia môžeme v atribúte `media` definovať aj jeho vlastnosti, podobne ako v media queries. Napríklad:

- štýl pre obrazovky so šírkou aspoň 800px pripojíme elementom `<link href="styl.css" rel="stylesheet" media="screen and (min-width: 800px)">`
- štýl pre obrazovky so šírkou nanajvýš 1000px pripojíme elementom `<link href="styl.css" rel="stylesheet" media="screen and (max-width: 1000px)">`

Ak používame na stránke viacero druhov štýlov pre rôzne zariadenia s rôznymi vlastnosťami, ukladáme každý do samostatného súboru a k HTML súboru ich pripájame pomocou viacerých elementov `link`.

Vráťme sa k stránke IT Pizza. Jej zdrojový kód je už pomerne dlhý a začína byť neprehľadný. V nasledujúcich úlohách a príkladoch oddelíme CSS kód od HTML kódu, samostatne uložíme štýly pre jednotlivé zariadenia a šírky a pripojíme ich k HTML súboru.

ÚLOHA 11.14

Štýly (t.j. obsah elementu `style`) zo stránky IT Pizza z *príkladu 11.10* rozdeľte a uložte do štyroch samostatných súborov:

- základný štýl (to, čo nie je v žiadnom media query) uložte do súboru `styly.css`, do toho istého priečinka, v ktorom je súbor `index.html`,
- štýl pre obrazovky so šírkou aspoň 400px uložte do súboru `styly-min400.css`,
- štýl pre obrazovky so šírkou aspoň 800px uložte do súboru `styly-min800.css`,
- štýl pre tlač uložte do súboru `styly-print.css`.
- V súbore `index`.html vymažte element `<style> </style>` s celým jeho obsahom.

PRÍKLAD 11.15

V súbore `index.html` pripojíme `css` súbory vytvorené v *úlohe 11.14*.

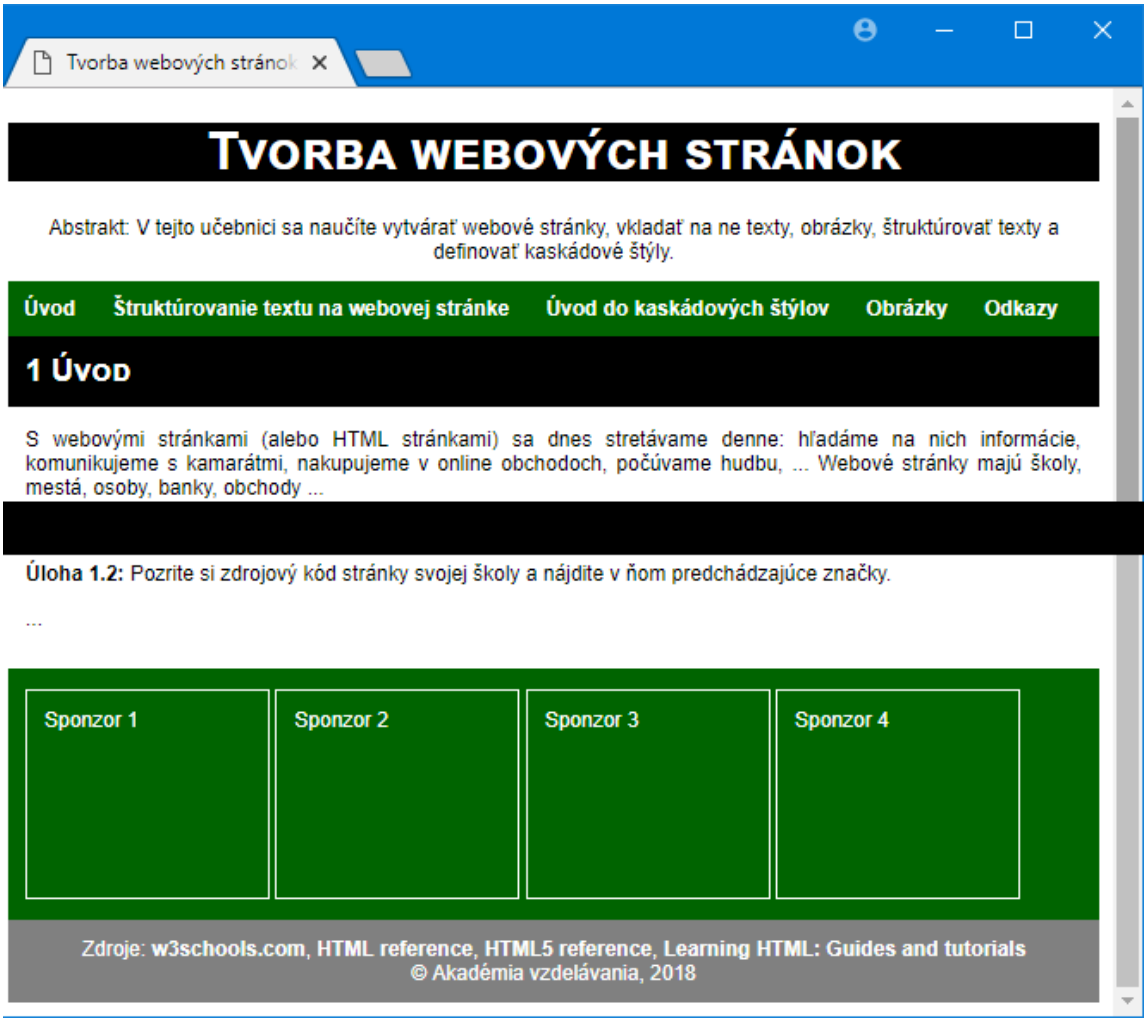
```
<head>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-
scale=1.0">
 <link href="styly.css" rel="stylesheet" media="screen">
 <link href="styly-min400.css" rel="stylesheet" media="screen and
(min-width: 400px)">
<link href="styly-min800.css" rel="stylesheet" media="screen and
(min-width: 800px)">
 <link href="styly-print.css" rel="stylesheet" media="print">
 <title>IT PIZZA</title>
</head>
```

ÚLOHA 11.16

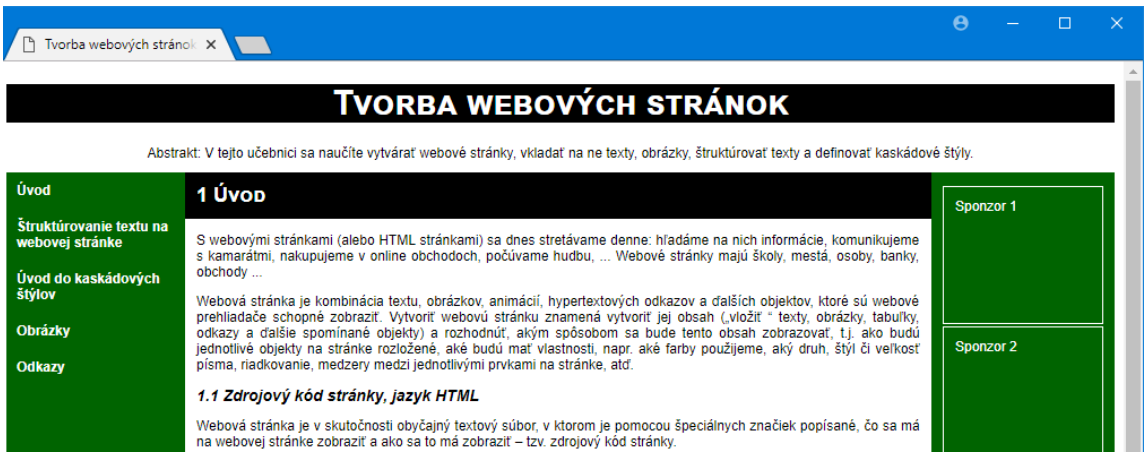
Upravte stránku s učebnicou (súbory v priečinku `11/ucebnica`) tak, aby bola responzívna. Definujte tiež štýl pre tlač. Všetky štýly definujte ako externé súbory.

- základné zobrazenie bude v jednom stĺpci, odkazy v navigácii budú pod sebou, sponzori vedľa seba koľko sa zmestí do riadka (*obrázok 11.11*),
- pri šírke aspoň 650px: vzhľad je stále jednostĺpcový, sponzori aj odkazy sa zobrazujú vedľa seba (*obrázok 11.12*),
- pri šírke aspoň 1000px: vzhľad je trojstĺpcový, vľavo sú odkazy pod sebou, v strede je text kapitoly, vpravo sú sponzori pod sebou (*obrázok 11.13*),
- tlačiť sa bude len hlavný nadpis, text kapitoly a päta. Nadpisy a texty v päte budú čierne, odkazy sivé, podčiarknuté a nie tučné (*obrázok 11.14*).

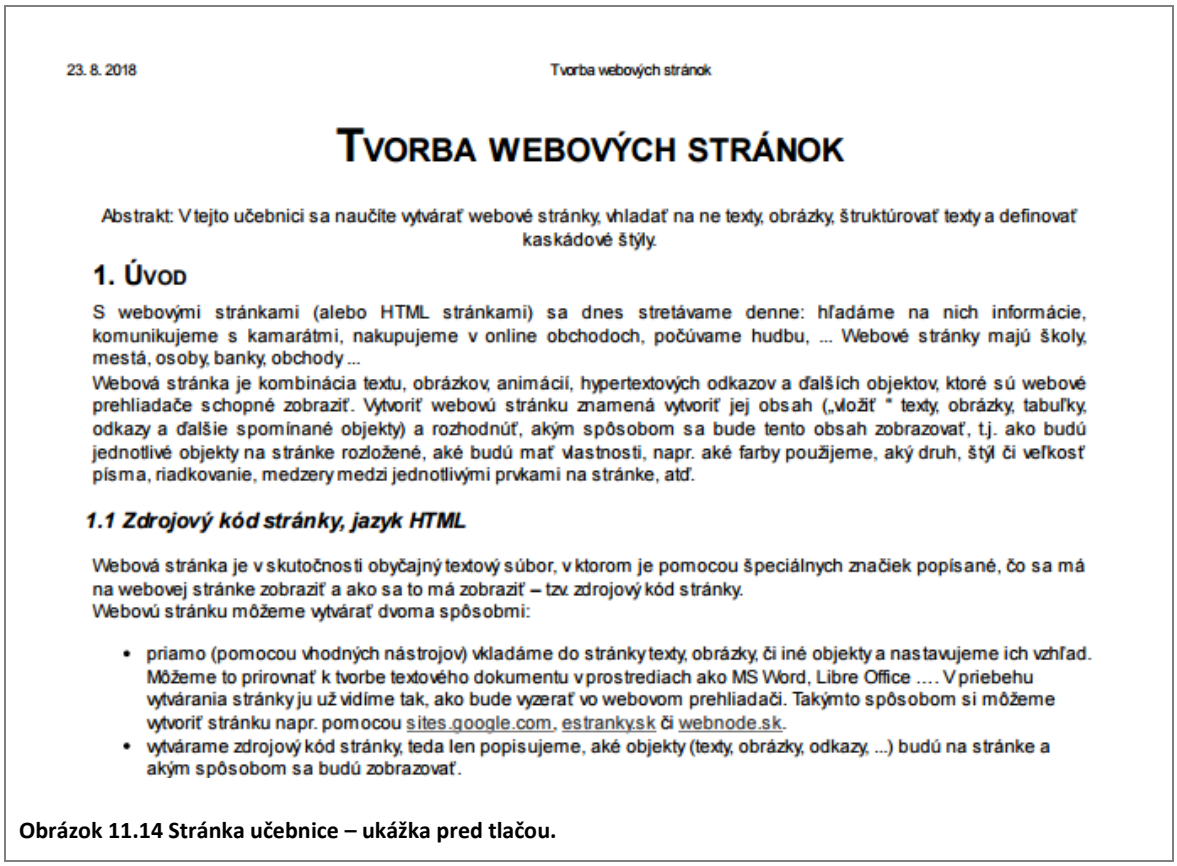
Obrázok 11.11 Stránka učebnice – zobrazenie pre všetky zariadenia.



Obrázok 11.12 Stránka učebnice – zobrazenie pre obrazovky pri šírke aspoň 650px.



Obrázok 11.13 Stránka učebnice - zobrazenie pre obrazovky pri šírke aspoň 1000px.



23. 8. 2018 Tvorba webových stránok

TVORBA WEBOVÝCH STRÁNOK

Abstrakt: V tejto učebnici sa naučíte vytvárať webové stránky, vhladať na ne texty, obrázky, štruktúrovať texty a definovať kaskádové štýly.

1. ÚVOD

S webovými stránkami (alebo HTML stránkami) sa dnes stretávame denne: hľadáme na nich informácie, komunikujeme s kamarátmi, nakupujeme v online obchodoch, počúvame hudbu, ... Webové stránky majú školy, mestá, osoby, banky, obchody ...
Webová stránka je kombinácia textu, obrázkov, animácií, hypertextových odkazov a ďalších objektov, ktoré sú webové prehliadače schopné zobraziť. Vytvoriť webovú stránku znamená vytvoriť jej obsah („vložiť " texty, obrázky, tabuľky, odkazy a ďalšie spomínané objekty) a rozhodnúť, akým spôsobom sa bude tento obsah zobrazovať, t.j. ako budú jednotlivé objekty na stránke rozložené, aké budú mať vlastnosti, napr. aké farby použijeme, aký druh, štýl či veľkosť písma, riadkovanie, medzery medzi jednotlivými prvkami na stránke, atď.

1.1 Zdrojový kód stránky, jazyk HTML

Webová stránka je v skutočnosti obyčajný textový súbor, v ktorom je pomocou špeciálnych značiek popísané, čo sa má na webovej stránke zobraziť a ako sa to má zobraziť – tzv. zdrojový kód stránky.
Webovú stránku môžeme vytvárať dvoma spôsobmi:

- priamo (pomocou vhodných nástrojov) vkladáme do stránky texty, obrázky, či iné objekty a nastavujeme ich vzhľad. Môžeme to prirovnať k tvorbe textového dokumentu v prostrediach ako MS Word, Libre Office .... V priebehu vytvárania stránky ju už vidíme tak, ako bude vyzerať vo webovom prehliadači. Takýmto spôsobom si môžeme vytvoriť stránku napr. pomocou sites.google.com, estranky.sk či webnode.sk.
- vytvárame zdrojový kód stránky, teda len popisujeme, aké objekty (texty, obrázky, odkazy, ...) budú na stránke a akým spôsobom sa budú zobrazovať.

Obrázok 11.14 Stránka učebnice – ukážka pred tlačou.

Metodika pre učiteľa

CIEĽ
Cieľom je pochopiť pojem responzívny web a oboznámiť sa so spôsobmi, ako ho tvoriť.

Nadväznosť na predchádzajúce kapitoly: Kapitola vyžaduje kapitoly 1-8 a 10.

Je to pomerne náročná kapitola. Isté princípy sa dajú ukázať na príkladoch 1 až 4 a časti 11.2, ktoré sú jednoduchšie a nie sú viazané na kap. 10. Zvyšok odporúčame skôr len pre pokročilých.

K syntaxi media queries: Syntax media queries môže byť aj zložitejšia, pozrite na https://www.w3schools.com/cssref/css3_pr_mediaquery.asp.

Úloha 11.9: Riešenie: Vo všeobecných štýloch ponecháme zobrazovanie odkazov v navigácii ako bloky (`nav a {display: block}`), v media query pre šíriku minimálne 400px z nich spravíme inline-bloky (`nav a {display: inline-block}`) a pre šírku minimálne 800px zase bloky (`nav a {display: block}`).

K testovaniu responzivity:

Pre potreby učebnice sme využívali testovače:

- Am I responsive - (http://ami.responsivedesign.is) - výhodou je, že vie testovať aj lokálne stránky, ale len cez tzv. Localhost (webový server na lokálnom počítači),
- Mobile Friendly Test (https://search.google.com/test/mobile-friendly).

Prácu s týmito testovačmi v učebnici nepopisujeme, je intuitívna a jednoduchá.

Zoznam ďalších aplikácií na testovanie responzivity nájdete na stránke: https://www.webdesignerdepot.com/2017/10/7-free-tools-for-testing-responsive-layouts/.

Úloha 11.10: V úlohe sme riešili iba skrytie niektorých objektov. Môžete sa ešte so žiakmi porozprávať, či je lepšie tlačiť úzke rozloženie stránky (ako pre mobily) alebo široké. Prípadne experimentovať a pokúsiť sa vytvoriť také rozloženie objektov na stránke IT Pizza, aby ste spotrebovali čo najmenej papiera, ale aby sa potrebné informácie na papier zmestili.

Príklad 11.12: Je potrebné upozorniť, že ak budú teraz chcieť žiaci kopírovať stránku na iný počítač, musia skopírovať všetky s ňou súvisiace súbory, teda nielen html, ale aj css.

Úloha 11.16 je opakovacia. Pravidlá, čo a ako zobrazovať pri jednotlivých šírkach, si môžu žiaci vymyslieť aj sami.

Zdroje k tejto kapitole:

- W3 schools - časť HTML responsive - pre plávajúce objekty
- W3 schools - časť CSS Flexbox (na konci) - pre flexový dizajn
- https://www.alejtech.sk/sk/blog-o-webdizajne/responzivny-dizajn-standard-pri-tvorbe-novych-webov.html

- https://www.websupport.sk/blog/2013/11/aky-je-to-responzivny-web/
- http://www.run.sk/responzivny-dizajn

12 FORMULÁRE

S formulármi na webových stránkach sa stretávame praktický dennodenne. Veď už len prihlásenie do e-mailového účtu znamená vyplnenie formulára s dvoma položkami: prihlasovacie meno a heslo. Takisto pri vyhľadávaní zadávame hľadaný výraz do textového políčka nejakého formulára. Aj mnohé úradné, predtým len papierové formuláre, už dnes majú svoju elektronickú podobu.

DISKUTUJTE

S akými formulármi ste sa stretli na internete? Na akých stránkach? Aké formuláre vypĺňate pravidelne? Už ste sa niekedy rozhodovali medzi papierovou a webovou verziou formulára? Pri akej príležitosti?

Formuláre využívame všade tam, kde potrebujeme od používateľa získať nejaké vstupy, údaje. Tie sa potom nejakým spôsobom spracujú. Pod spracovaním formulára si môžete predstaviť napr. kontrolu údajov, ktoré používateľ zadal, uloženie údajov zväčša do databázy, spätnú väzbu pre používateľa (napr. potvrdenie objednávky, rekapitulácia údajov, ktoré zadal, alebo výpis chýb, ak niečo vo formulári vyplnil nesprávne).

V tejto kapitole sa naučíme **vytvárať** formuláre s rôznymi prvkami. Nebudeme sa však zaoberať ich spracovaním. Formulár takmer vždy finálne spracováva nejaký skript (program) na strane servera, napr. PHP, ASP, Perl a pod. (nie JavaScript). Len pomocou jazyka HTML formuláre spracovávať nevieme.

Definovanie formulára

Definovanie formulára a jeho dôležité atribúty si ukážeme na príklade jednoduchého vyhľadávacieho formulára (*obrázok 12.1*).

Obrázok 12.1 Jednoduchý vyhľadávací formulár.

PRÍKLAD 12.1

Vytvoríme vyhľadávací formulár s textovým poľom na zadanie hľadaného textu a tlačidlom *Hľadaj*.

Formulár vytvoríme v novom súbore, takže najskôr musíme definovať všetky základné elementy (`doctype`, `html`, `head`, `body`), nastaviť titulok a kódovanie (pozri koniec *1. kapitoly*). Súbor uložíme ako `hladaj.html`. Do tela stránky (`body`) napíšeme nasledujúci kód:

```
<body>
  <form>
    <input type="text">
    <input type="submit" value="Hľadaj">
  </form>
</body>
```

Formulár definujeme pomocou elementu `<form>`. Obsahom elementu `form` sú jednotlivé prvky formulára a ich popisy. Náš formulár obsahuje dva prvky, textové pole a odosielacie tlačidlo.

Textové pole definujeme pomocou elementu `input` s atribútom `type` nastaveným na hodnotu `text`, teda `<input type="text">`. Elementom `input` sa definujú rôzne typy prvkov formulára, pomocou atribútu `type` určujeme, o aký typ prvku formulára ide.

Odosielacie tlačidlo je špeciálne tlačidlo, ktorým odosielame údaje z formulára na spracovanie. Je nevyhnutnou súčasťou každého formulára, inak nedokážeme preniesť, a teda ani spracovať, hodnoty zadané používateľom. Odosielacie tlačidlo definujeme pomocou elementu `<input type="submit">`. V atribúte `value` uvádzame text, ktorý sa vypíše na tlačidlo.

ÚLOHA 12.2

Experimentujte s formulárom z *príkladu 12.1*.

- zmeňte hodnotu atribútu `value` pre tlačidlo na *Search*,
- zrušte atribút `value` pre tlačidlo.

Element `form` môže mať niekoľko atribútov. Uvedieme dva z nich: `action` a `method`. Hodnotou atribútu `action` je adresa súboru, ktorý má údaje z formulára spracovať, t.j. súboru, ktorý obsahuje program vo vhodnom jazyku, ktorý nejakým spôsobom spracuje hodnoty odoslané z formulára. Napríklad, ak by sme formulár z príkladu 12.1 zadefinovali elementom `<form action="vyhladaj.php">`, tak po odoslaní (stlačení tlačidla *Hľadaj*) by sa výraz zadaný do textového poľa poslal súboru `vyhladaj.php`. Ak atribút `action` neuvedieme, bude formulár spracovávať ten súbor, v ktorom je formulár uložený – v našom prípade súbor `hladaj.html`.

Atribút `method` určuje spôsob odosielania údajov z formulára. Môže mať hodnotu `get` alebo `post`.

- pri odosielaní metódou `get` sa všetky údaje zadané do formulára prenesú do adresy prehliadača. Znamená to, že ich môže vidieť niekto, kto stojí pri používateľovi, resp. si ich môže neskôr pozrieť v histórii prehliadača. Pomocou metódy `get` môžeme preniesť len obmedzený objem údajov. Metóda sa používa na prenos údajov, ktoré nie sú citlivé, ako napr. vyhľadávaný výraz (vyhľadávače), alebo číslo strany pri stránkovaných webových dokumentoch,
- pri odosielaní metódou `post` sa hodnoty z formulára posielajú "skryte", teda nie sú prenášané cez adresu prehliadača, ani ich nevieme zistiť z histórie prehliadača. Z toho vyplýva, že je o niečo bezpečnejšia ako metóda `get`. Používame ju na prenos citlivých a osobných údajov. Pomocou metódy `post` môžeme preniesť väčší objem údajov, aj uploadovať súbory.

Formulár, ktorý sme vytvorili v *príklade 12.1,* síce vyzerá, ako sme chceli, ale nesplnil by svoju funkciu, pretože zadané údaje by sa vôbec neodoslali.

Každý prvok formulára musí mať definovaný atribút `name`. Pomocou jeho hodnoty vieme pri spracovaní formulára pristupovať k údajom zadaným v jednotlivých položkách formulára. Až na

isté výnimky, ktoré si ukážeme neskôr, platí, že hodnota atribútu `name` musí byť pre každý prvok formulára unikátna. Mená formulárových prvkov sa nesmú začínať číslom, nesmú obsahovať medzeru a neodporúčame ani diakritiku.

PRÍKLAD 12.3

Upravme formulár z *príkladu 12.1*. Definujme každému prvku formulára atribút `name` a nastavme spôsob odoslania formulára na metódu `post`.

```
<form method="post">
  <input type="text" name="retazec">
  <input type="submit" value="Hľadaj" name="odosli">
</form>
```

ÚLOHA 12.4

Vo formulári z *príkladu 12.3* zadajte do textového poľa nejaký text a odošlite ho. Zmeňte metódu odoslania údajov na `get`. Stránku s formulárom načítajte znova (nie pomocou F5, ale kliknite na adresu stránky a stlačte Enter!). Zadajte do textového poľa nejaký text a odošlite. Všimite si adresu v prehliadači.

Prvky formulára

Vo formulároch zväčša buď niečo dopĺňame (kratší text, dlhší text, čísla) alebo vyberáme z ponuky možností. Pomocou jazyka HTML vieme zadefinovať niekoľko rôznych typov prvkov určených na dopĺňanie textu aj na výber z ponuky.

ÚLOHA 12.5

Nájdite na internete nejaký formulár s aspoň piatimi položkami. Aké prvky sa v ňom nachádzajú? Skúste nájsť formulár, v ktorom je čo najviac rôznych typov prvkov.

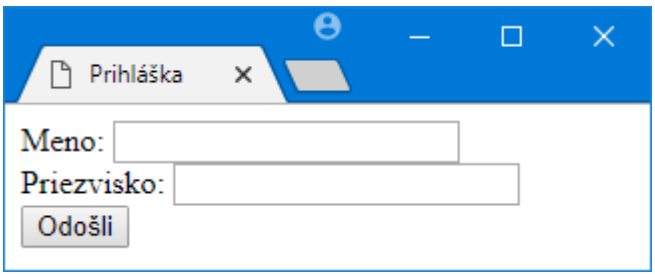
V nasledujúcej časti budeme postupne vytvárať formulár pre uchádzača o prácu v pizzerii IT Pizza. Ukážeme si na ňom, aké rôzne typy formulárových prvkov môžeme používať a ako sa definujú.

Jednoriadkové textové pole

Textové pole je jeden z najčastejšie používaných formulárových prvkov. Už vieme, že ho definujeme elementom `<input type="text">`, aj to, že každému takému prvku musíme dať atribút `name`.

ÚLOHA 12.6

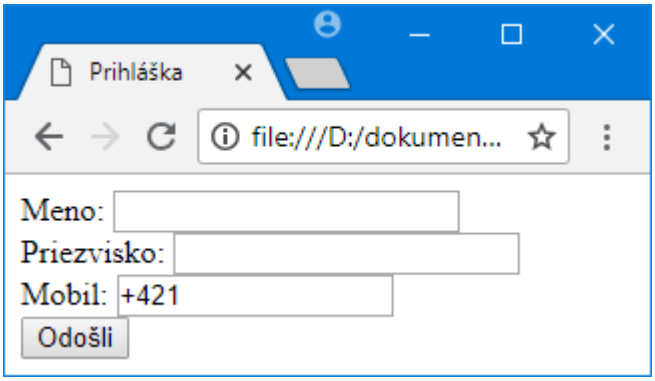
Vytvorte novú stránku `prihlaska.html` a v nej definujte formulár s dvoma textovými poľami, jedno na zadanie mena, druhé na zadanie priezviska (*obrázok 12.2*). Pre element `form` nenastavujte žiadne atribúty. Pre elementy `input` nastavte všetky dôležité atribúty. Doplňte aj popisy k formulárovým prvkom *Meno*, *Priezvisko* – obyčajné texty, v zdrojovom kóde pred príslušným formulárovým elementom. Na formátovanie do viacerých riadkov môžete využiť napr. element `<br>`.



Obrázok 12.2 Formulár na zadanie mena a priezviska.

Pre textové pole môžeme nastaviť šírku pomocou atribútu `width`. Hodnotou atribútu je číslo, ktoré definuje šírku textového poľa v znakoch. Atribút nelimituje počet znakov, ktoré dokážeme do poľa zapísať, iba to, koľko ich môžeme naraz vidieť. Ak zapíšeme do poľa viac znakov, začne sa obsah poľa posúvať.

V prípade, že chceme pre textové pole nastaviť nejakú počiatočnú hodnotu, ktorá sa v ňom zobrazí už pri načítaní stránky, použijeme atribút `value`, ktorého hodnotou môže byť ľubovoľný reťazec (*obrázok 12.3*).



Obrázok 12.3 Formulár: položka Mobil má prednastavenú hodnotu +421.

PRÍKLAD 12.7

Do formulára z úlohy 12.6 pridáme textové pole na zadanie telefónneho čísla. Jeho veľkosť nastavíme na 15 znakov a počiatočnú hodnotu na +421 (*obrázok 12.3*).

```
<form>
   Meno: ...
   Priezvisko: ...
   Mobil: <input type="text" name="mobil" size="15"
value="+421"><br>
   <input type="submit" value="Odošli" name="odosli">
</form>
```

ÚLOHA 12.8

Do formulára z *príkladu 12.7* doplňte textové pole *E-mail* s prednastavenou hodnotou @. Experimentujte s hodnotou atribútu `width` pre položku *Meno*. Vyskúšajte 5, 10, 20, 30,...

Prepínač (radiobutton)

Prepínače (radiobuttony) vo formulári používame vtedy, ak chceme, aby používateľ zvolil práve jednu z istého (menšieho) počtu možností.

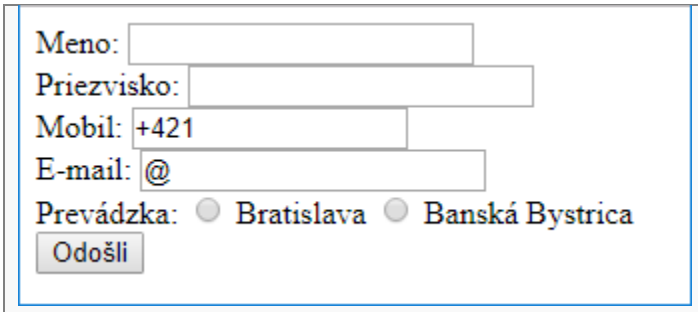
Prepínač definujeme pomocou elementu `<input type="radio">`. Každý prepínač by mal mať nastavené atribúty `name` a `value`. Ak chceme definovať skupinu prepínačov, z ktorých sa má dať vybrať len jedna možnosť, tak **všetky položky skupiny musia mať rovnaké meno** (atribút `name`), ale budú sa líšiť hodnotami (atribút `value`). Vybraná hodnota bude po odoslaní formulára hodnotou celej skupiny (elementu).

PRÍKLAD 12.9

Do formulára z *úlohy 12.8* pridáme dva prepínače na výber prevádzky, do ktorej chce uchádzač nastúpiť, jeden pre Bratislavu a druhý pre Banskú Bystricu.

```
<form>
  ...
  E-mail: ...
  Prevádzka:
  <input type="radio" name="mesto" value="BA"> Bratislava
  <input type="radio" name="mesto" value="BB"> Banská Bystrica
  <br>
  <input type="submit" value="Odošli" name="odosli">
</form>
```

Atribútom `value` definujeme hodnotu, ktorá sa po odoslaní formulára pošle na spracovanie (pošle sa len hodnota toho prepínača, ktorý je zvolený). Popis, ktorý sa pri prepínači zobrazí, nie je súčasťou elementu `input`, je to obyčajný text, ktorý umiestňujeme zväčša za prepínač. Môže, ale nemusí sa zhodovať s hodnotou (`value`) prepínača.

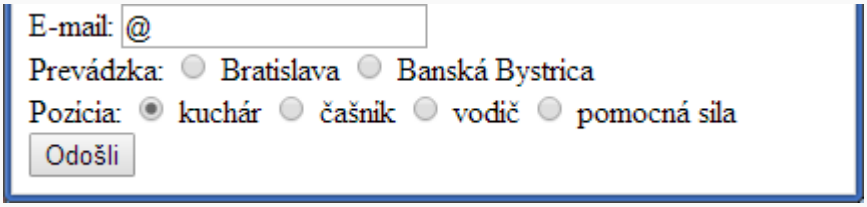


Obrázok 12.4 Formulár s prepínačmi pre výber prevádzky.

Ak chceme mať niektorý prepínač označený už pri načítaní stránky, pridáme do jeho definície atribút `checked`, napríklad `<input type="radio" name="mesto" value="BB" checked>`. Atribút `checked` patrí medzi tzv. booleovské atribúty, ktoré nemajú hodnotu.

ÚLOHA 12.10

Do formulára z *príkladu 12.9* pridajte ďalšiu skupinu prepínačov na zadanie pozície, na ktorú chce uchádzač nastúpiť. Pozíciu *kuchár* zvoľte ako prednastavenú (pozri *obrázok 12.5*).



Obrázok 12.5 Prepínač pre výber pracovnej pozície.

Začiarkávacie políčko (checkbox)

Začiarkávacie políčka zvyčajne používame vtedy, ak používateľ nemusí zvoliť žiadnu z ponúkaných možností, alebo ich môže označiť viacero. Pri začiarkávacích políčkach, na rozdiel od prepínačov, môžeme každú možnosť označiť aj odznačiť.

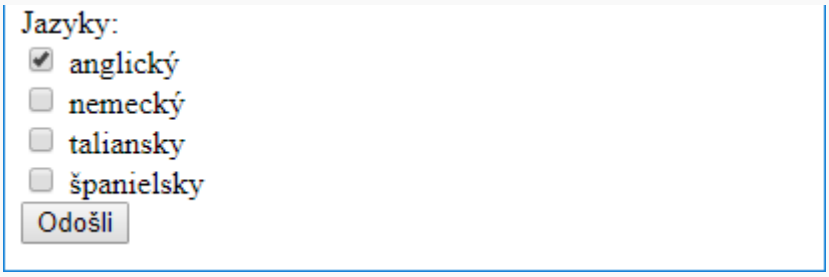
Na stránku môžeme vkladať začiarkávacie políčka samostatne alebo môžeme vytvoriť skupinu (podobne ako pri prepínačoch). Tvorba skupiny začiarkávacích políčok však nie je taká jednoduchá ako pri prepínačoch, pričom tiež záleží, ako budeme formulár spracovávať. Preto sa nebudeme zaoberať možnosťou tvorby skupiny začiarkávacích políčok, ale len samostatnými začiarkávacími políčkami.

Začiarkávacie políčko definujeme pomocou elementu `<input type="checkbox">`. Každé začiarkávacie políčko musí mať nastavené atribúty `name` a `value`. Ak chceme mať niektoré začiarkávacie políčko označené už pri načítaní stránky, pridáme mu atribút `checked`.

PRÍKLAD 12.11

Vo formulári chceme zisťovať, aké jazyky uchádzač o prácu ovláda. Do formulára z úlohy 12.10 pridáme štyri začiarkávacie políčka s popismi *anglický*, *nemecký*, *taliansky* a *španielsky* (obrázok 12.6). Ako mená a hodnoty použijeme skratky, napr. aj, nj, tj a sj. Anglický jazyk bude začiarknutý už pri načítaní stránky.

```
<form>
  ...
  Jazyky:<br>
  <input type="checkbox" name="aj" value="aj" checked> anglický
  <br>
  <input type="checkbox" name="nj" value="nj"> nemecký<br>
  <input type="checkbox" name="tj" value="tj"> taliansky<br>
  <input type="checkbox" name="sj" value="sj"> španielsky
  <br>
  <input type="submit" value="Odošli" name="odosli">
</form>
```

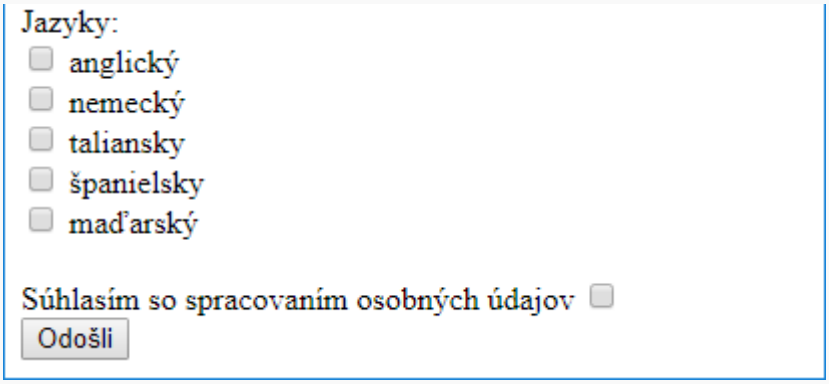


Obrázok 12.6 Formulár so začiarkávacími políčkami.

ÚLOHA 12.12

Do formulára z *príkladu 12.11*:

- pridajte začiarkávacie políčko s nejakým ďalším jazykom – zvoľte vhodne atribút `name`, tak, aby bol unikátny,
- zrušte začiarknutie anglického jazyka,
- pred odosielacie tlačidlo pridajte začiarkávacie políčko s menom `suhlas`, hodnotou `suhlasim` a popisom *Súhlasím so spracovaním osobných údajov* (*obrázok 12.7*).



Obrázok 12.7 Začiarkávacie políčka.

Výberová ponuka SELECT

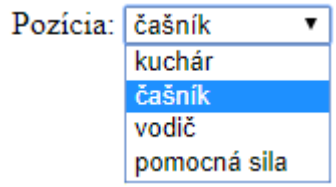
Výberová ponuka (pozri Obrázok 12.8) umožňuje používateľovi vybrať jednu z viacerých ponúkaným možností. Možnosti však nie sú viditeľné hneď pri načítaní stránky, ale až po rozbalení ponuky, preto sa niekedy nazýva aj rozbaľovacia ponuka (v angličtine *drop down list*). Používa sa zväčša vtedy, keď je v ponuke veľa možností, takže riešenie pomocou prepínačov by zabralo príliš veľa miesta, napr. na zadanie dňa v mesiaci (31 možností).

Výberovú ponuku definujeme pomocou elementu `<select></select>`. Obsahom elementu sú jednotlivé prvky ponuky, ktoré definujeme pomocou elementu `<option></option>`. Popis jednotlivých prvkov sa vkladá medzi značky `<option></option>`. Element `select` musí mať nastavený atribút `name`. Element `option` nemá atribút `name`, ale mal by mať nastavený atribút `value`. Atribút `value` zvolenej možnosti sa posiela po odoslaní formulára ako hodnota celého elementu `select`. Štandardne je nastavený (a viditeľný) prvý prvok ponuky. Ak chceme prednastaviť nejakú inú možnosť, urobíme tak v príslušnom elemente `option` pomocou atribútu `selected`. Vo výberovej ponuke je teda vždy niektorá z hodnôt prednastavená, kým pri prepínačoch a začiarkávacích políčkach nie.

PRÍKLAD 12.13

Vo formulári z predchádzajúcej úlohy nahradíme prepínače na výber pozície výberovou ponukou. Pozícia *čašník* bude prednastavená (*obrázok 12.8*).

```
...
  Pozícia:
  <select name="pozicia">
    <option value="kuchár">kuchár</option>
    <option value="čašník" selected>čašník</option>
    <option value="vodič">vodič</option>
    <option value="pomocná sila">pomocná sila</option>
  </select>
  <br>
  Jazyky:<br>
  ...
```



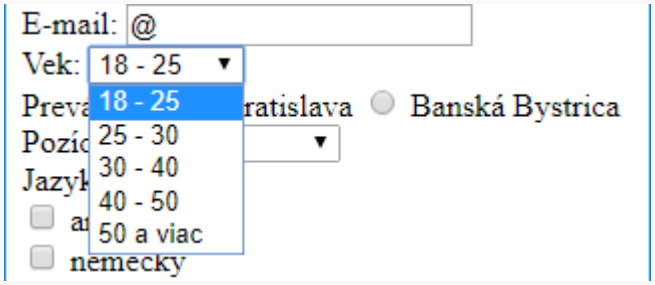
Obrázok 12.8 Výberová ponuka.

ÚLOHA 12.14

Do výberovej ponuky z predchádzaúcej úlohy pridajte možnosť: *roznášač*.

ÚLOHA 12.15

Do formulára z predchádzajúcej úlohy za položku *E-mail* pridajte výberovú ponuku na zadanie vekovej kategórie. Vytvorte možnosti podľa *obrázka 12.9*.

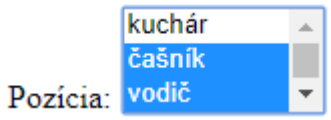


Obrázok 12.9 Výberová ponuka.

Element `select` môže mať navyše dva atribúty:

- `size`, ktorým určíme počet viditeľných možností (štandardne je jedna)
- a `multiple`, ktorým umožníme používateľovi zvoliť viac ako jednu hodnotu.

Ak by sme element `<select name="pozicia">` z nášho formulára zmenili na `<select name="pozicia" size="3" multiple>`, zobrazila by sa ponuka po načítaní stránky tak, ako na *obrázku 12.10*.



Obrázok 12.10 Výberová ponuka s možnosťou výberu viacerých prvkov.

Viacriadkové textové pole TEXTAREA

Viacriadkové textové pole, ako už názov hovorí, slúži na zadanie dlhšieho textu, ktorý obsahuje viac riadkov (čiže pri jeho písaní môžeme použiť aj kláves Enter).

Obrázok 12.11 Viacriadkové textové pole.

Viacriadkové textové pole definujeme pomocou elementu `<textarea></textarea>`. Element musí mať nastavený atribút `name`.

Element `textarea` nemá atribút `value` ako ostatné formulárové elementy. Ak chceme, aby vo viacriadkovom textovom poli bola pri načítaní stránky nejaká prednastavená hodnota (text), uvedieme ju medzi počiatočnú a koncovú značku elementu v zdrojovom kóde, napr. `<textarea>základná škola</textarea>`.

PRÍKLAD 12.16

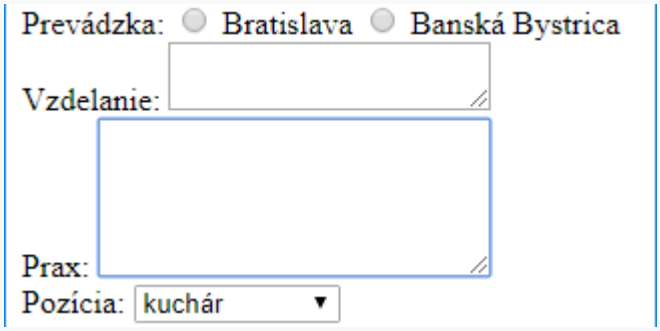
Do formulára s prihláškou pridáme viacriadkové textové pole s popisom *Vzdelanie* (*obrázok 12.11*). Umiestnime ho za prepínače na výber prevádzky.

```
...
  Prevádzka:
  <input type="radio" name="mesto" value="BA"> Bratislava
  <input type="radio" name="mesto" value="BB"> Banská Bystrica
  <br>
  Vzdelanie: <textarea name="vzdelanie"></textarea>
  <br>
  Pozícia:
  ...
```

Veľkosť viacriadkového textového poľa môžeme nastaviť pomocou atribútov `cols` (počet stĺpcov) a `rows` (počet riadkov), napr. `<textarea name="text" cols="25" rows="5"></textarea>`. Podobne ako pri jednoriadkovom textovom poli, ide len o šírku a výšku "viditeľnej časti", **neobmedzuje** sa tým počet riadkov, ktoré môžeme do viacriadkového textového poľa zapísať (ak ich zapíšeme viac, zobrazí sa posúvač).

ÚLOHA 12.17

Do formulára z predchádzajúceho príkladu pridajte viacriadkové textové pole *Prax*, veľkosti 25 stĺpcov a 5 riadkov, bez prednastavenej hodnoty. Umiestnite ho za viacriadkové pole *Vzdelanie* (*obrázok 12.12*).



Obrázok 12.12 Viacriadkové textové polia *Vzdelanie* a *Prax*.

Tlačidlo

V úvode kapitoly sme si ukázali, ako pomocou elementu `<input type="submit">` definujeme odosielacie tlačidlo. Ak zmeníme hodnotu `submit` na hodnotu `reset`, t.j. `<input type="reset">`, definujeme tzv. resetovacie tlačidlo – tlačidlo, ktoré "zmaže" všetky údaje zadané používateľom a nastaví preddefinované hodnoty.

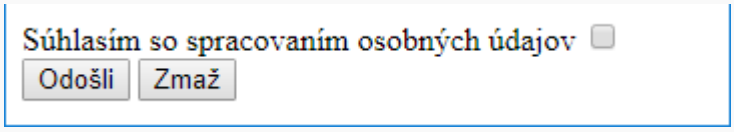
Tlačidlá môžeme definovať aj pomocou elementu `<button></button>`. Na rozdiel od elementu `<input>`, element `button` ponúka oveľa viac možností (pomocou CSS). Na tlačidle môže byť zobrazený nielen text, ale napríklad aj obrázok. Element je párový, jeho obsahom je text, ktorý sa zobrazí na tlačidle. Elementu môžeme definovať atribúty `name`, `value` a `type`. Pomocou atribútu `type` určujeme, o aký typ tlačidla ide. Hodnota `submit` znamená odosielacie

tlačidlo, hodnota `reset` resetovacie tlačidlo a hodnota `button`, obyčajné tlačidlo (zvykne sa používať v spojení s JavaScriptom na vykonanie nejakého JS kódu.)

PRÍKLAD 12.18

Vo formulári s prihláškou zmeníme definíciu odosielacieho tlačidla pomocou elementu `button`. Taktiež pridáme tlačidlo na zmazanie údajov.

```
...
  <br>
  <button type="submit" value="odosli" name="odosli">Odošli
  </button>
  <button type="reset" value="reset" name="reset">Zmaž</button>
</form>
```



Obrázok 12.13 Tlačidlá na odoslanie formulára a zmazanie údajov.

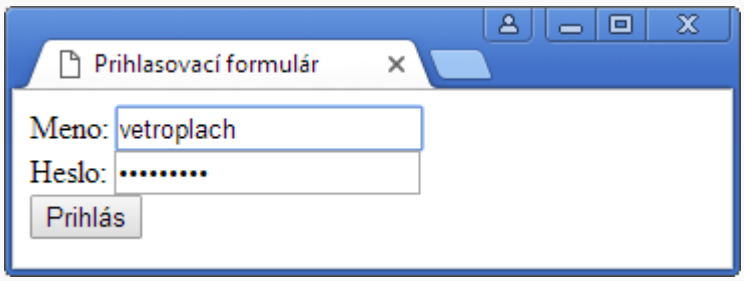
Jednoriadkové textové pole na zadanie hesla

Textové pole na zadanie hesla je v podstate rovnaké ako obyčajné jednoriadkové textové pole s tým rozdielom, že text, ktorý do neho používateľ píše, sa nezobrazuje, ale nahrádza hviezdičkami. Ak máme vo formulári tento prvok, určite by sme ho mali odosielať metódou `post`.

Textové pole na zadanie hesla definujeme pomocou elementu `<input type="password">`. Ako všetky ostatné formulárové elementy, musí mať definovaný atribút `name`. Môžeme mu nastaviť veľkosť pomocou atribútu `size` ako obyčajnému textovému poľu.

ÚLOHA 12.19

Vytvorte novú stránku `login.html` s prihlasovacím formulárom (*obrázok 12.14*). Formulár bude obsahovať jedno obyčajné textové pole, jedno textové pole na zadanie hesla a tlačidlo na odoslanie. Tlačidlo realizujte pomocou elementu `button`.



Obrázok 12.14 Prihlasovací formulár.

Nové formulárové prvky a atribúty v HTML5

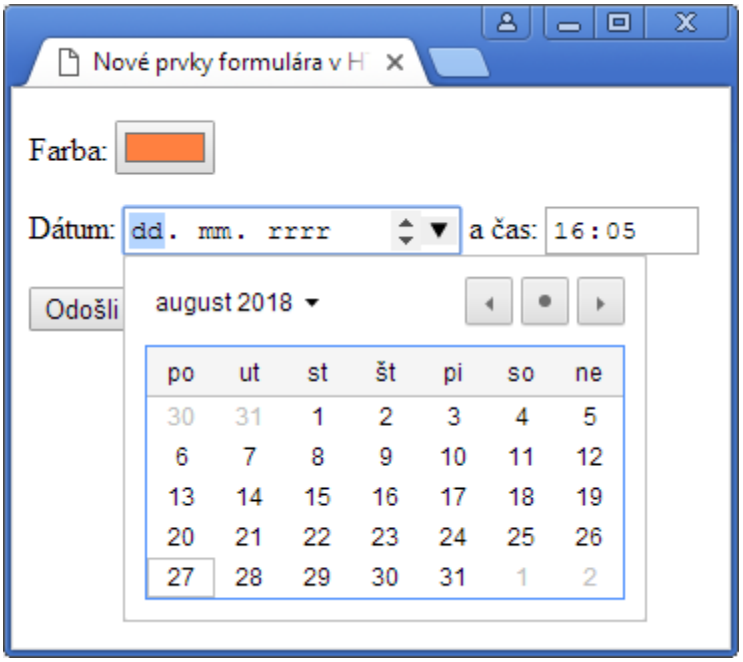
Štandard HTML5 zaviedol niekoľko nových prvkov formulára. Všetky definujeme pomocou elementu `input`, typ prvku určujeme pomocou atribútu `type`. Za všetky spomenieme nasledujúce tri:

- prvok na výber **farby** – element `<input type="color">`,
- prvok na zadanie či výber **dátumu** – element `<input type="date">`,
- prvok na zadanie **času** – element `<input type="time">`.

Vo všetkých troch prípadoch ide o komplexnejšie prvky, ktoré sú sami o sebe malými formulármi. Pri ich použití si musíme byť vedomí, že **nie sú podporované staršími prehliadačmi**. Staršie prehliadač zobrazia nové typy prvkov ako jednoriadkové textové pole.

ÚLOHA 12.20

Vyskúšajte formulár v súbore `12/nove.html`. Zobrazujú sa vo vašom prehliadači podobne ako na *obrázku 12.15*, alebo vidíte len obyčajné textové polia?



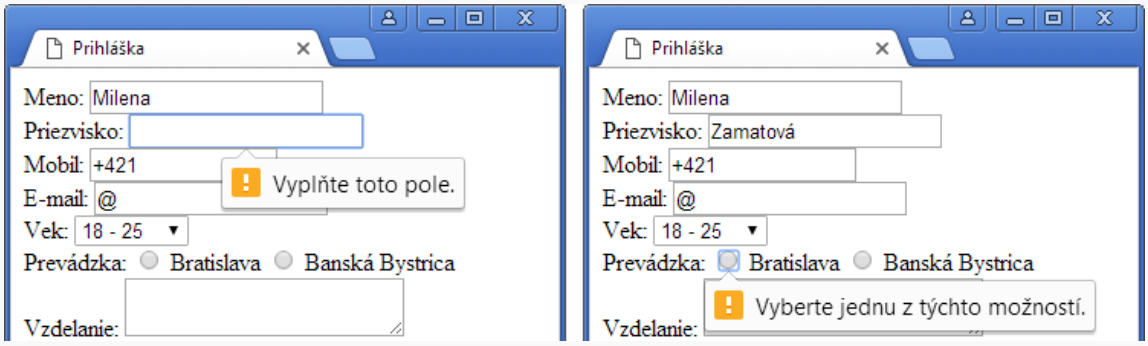
Obrázok 12.15 Formulár s novými prvkami na výber farby, zadanie dátumu a času.

Formulárovým prvkom definovaným pomocou elementu `input` môžeme pridať atribút `required`, ktorým určíme, že daná položka formulára je povinná. V prípade, že používateľ stlačí odosielacie tlačidlo a niektorý z údajov s atribútom `required` nebude vyplnený, formulár sa neodošle a prehliadač používateľa upozorní na "chybu" (*obrázok 12.16*).

PRÍKLAD 12.21

Položky *Meno*, *Priezvisko* a *Prevádzka* vo formulári `prihlaska.html` z *úlohy 12.18* definujeme ako povinné.

```
<form method="get">
  Meno: <input type="text" name="meno" size="20" required><br>
  Priezvisko: <input type="text" name="priezvisko" required><br>
  ...
  ...
  Prevádzka:
  <input type="radio" name="mesto" value="BA" required>
  Bratislava
  <input type="radio" name="mesto" value="BB" required>
  Banská Bystrica
  ...
```

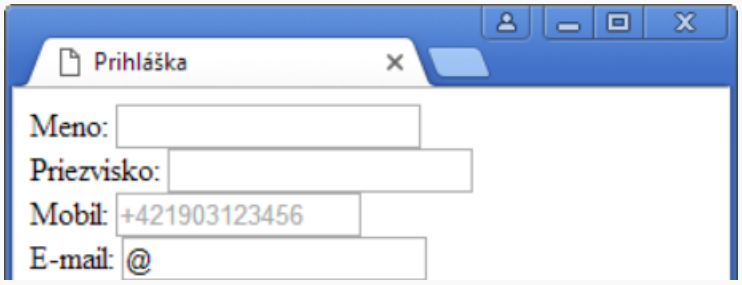


Obrázok 12.16 Reakcie na nezadanie povinných údajov.

Pre elementy `input` typu `text` a `password` môžeme definovať atribút `placeholder`. Slúži na definovanie pomocného textu, ktorý popisuje očakávanú hodnotu v príslušnom prvku formulára. Môže ísť o vysvetlenie očakávaného formátu položky, príklad vyplnenia, alebo nejaké obmedzenia na položku (napr. *len čísla*). Pomocný text sa zobrazuje priamo v položke, kým do nej používateľ nezadá nejakú hodnotu (obrázok 12.17).

PRÍKLAD 12.22

Vo formulári z predchádzajúcej úlohy zrušíme prednastavenú hodnotu položke *Mobil* a namiesto nej definujeme `placeholder` pre položku *Mobil* (*obrázok 12.17*).



Obrázok 12.17 Placeholder pre položku Mobil.

```
  ...
  Mobil: <input type="text" name="mobil" size="15"
   placeholder="+421903123456"><br>
  ...
```

Keďže atribúty `required` a `placeholder` sú definované až v štandarde HTML5, nemajú podporu v starších prehliadačoch. Pri tvorbe a spracovaní formulára na to musíme myslieť.

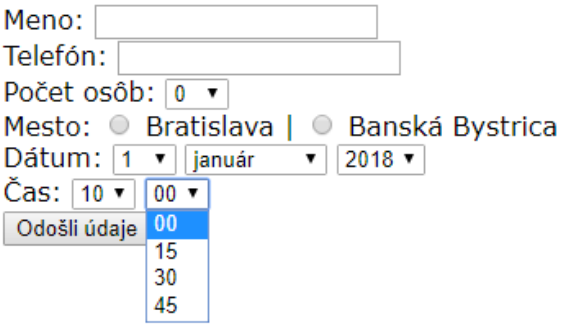
ÚLOHA 12.23

Vo formulári z *príkladu 12.22*:

- položky *Mobil*, *E-mail* a *Súhlas* so spracovaním údajov zmeňte na povinné,
- zrušte prednastavenú hodnotu pre položku *E-mail*,
- pre položku *E-mail* definujte rozumný `placeholder`.

ÚLOHA 12.24

Vytvorte formulár `rezervacia.html` na rezerváciu stola v pizzérii podľa *obrázka 12.18*. Stôl bude možné rezervovať v prevádzkach Bratislava alebo Banská Bystrica pre 0 až 10 osôb v čase od 10:00 do 19:45, len na rok 2018 alebo 2019. Zvážte, ktoré položky majú byť povinné.



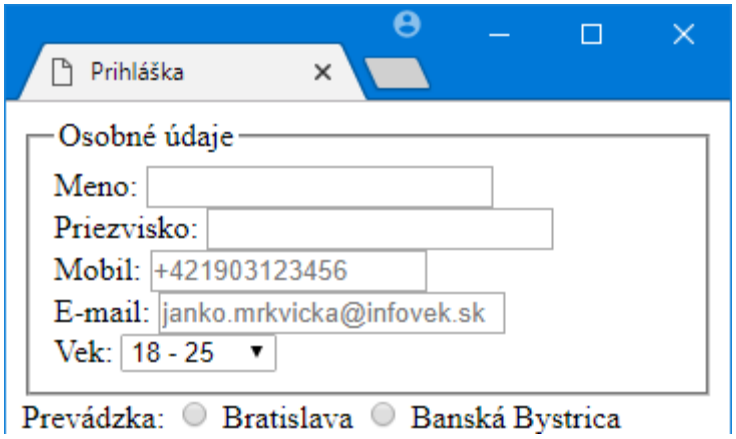
Obrázok 12.18 Rezervačný formulár.

Štruktúrovanie formulára

Ak formulár obsahuje väčšie množstvo prvkov, môže sa stať neprehľadným. Jedna z možností, ako formulár sprehľadniť, je použiť blokové elementy alebo tabuľky. Špeciálne na použitie vo formulároch je určený blokový element `<fieldset></fieldset>`.

Element `fieldset` umožňuje logické i optické zoskupenie prvkov formulára – zobrazí orámovanie okolo elementov, ktoré sú jeho obsahom (*obrázok 12.19*). Každý element `fieldset` môže mať svoju legendu, ktorú definujeme pomocou elementu `<legend></legend>`. Element `legend` musí byť definovaný vnútri elementu `fieldset`.

Elementy `fieldset` a `legend` nemajú žiaden vplyv na funkčnosť formulára.



Obrázok 12.19 Orámovanie prvkov formulára a legenda.

PRÍKLAD 12.25

Vo formulári `prihlaska.html` z *úlohy 12.23* zoskupíme osobné údaje (meno, priezvisko, mobil, email, vek) a definujeme pre ne legendu (*obrázok 12.19*).

```
<form>
  <fieldset>
    <legend>Osobné údaje</legend>
    Meno: <input type="text" name="meno" size="20" required"><br>
    Priezvisko: <input type="text" name="priezvisko" required>
    <br>
    Mobil: <input type="text" name="mobil" size="15"
      placeholder="+421903123456"><br>
    E-mail: <input type="text" name="email"
      placeholder="janko.mrkvicka@infovek.sk"><br>
    Vek:
    <select name="vek">
      <option value="18-25">18 - 25</option>
      <option value="25-30">25 - 30</option>
      <option value="30-40">30 - 40</option>
      <option value="40-50">40 - 50</option>
      <option value="50-">50 a viac</option>
    </select>
  </fieldset>
  ...
```

ÚLOHA 12.26

Vo formulári z predchádzajúceho príkladu zoskupte pomocou elementu `fieldset` položky *Prevádzka*, *Vzdelanie*, *Prax*, *Pozícia* a *Jazyky*. Legendu uvádzať nemusíte.

Popisy prvkov formulára

Najjednoduchší spôsob popisovania formulárových prvkov je intuitívny: pomocou textu pred, resp. za elementom. Nevýhodou tohto spôsobu je, že príslušný formulárový element a jeho popis nie sú logicky nijako prepojené. Pre nevidiacich je takýto spôsob popisovania formulára nevhodný, pretože si nedokážu vytvoriť spojitosť medzi prvkom formulára a okolitým textom, najmä ak je text za prvkom formulára, prípadne je to dlhší text.

Pre popis prvku formulára, ktorý s ním bude jednoznačne spojený, slúži element `<label></label>`. Jeho použitie si ukážeme na príklade.

PRÍKLAD 12.27

Vo formulári z predchádzajúcej úlohy definujeme popis položky *Meno* pomocou elementu `label`:

- Pre element `input`, ktorý definuje prvok formulára na zadanie mena, definujeme jednoznačný identifikátor `id="meno"`.
- Pôvodný popis prvku, text *Meno*, obalíme elementom `label`, ktorému pridáme atribút `for="meno"`, čím vytvoríme spojenie popisu s formulárovýmm prvkom.

```
...
<legend>Osobné údaje</legend>
<label for="meno">Meno:</label> <input type="text" name="meno"
id="meno" size="20" required"><br>
...
```

Popis prvku teda definujeme ako `<label for="id_prvku">popis</label>`, kde `id_prvku` je hodnotou `id` atribútu toho prvku formulára, ku ktorému patrí popis. Hodnota atribútu `id` môže byť rovnaká ako hodnota atribútu `name`. Vo všeobecnosti to odporúčame.

PRÍKLAD 12.28

Pomocou elementu `label` vytvorme popis výberovej ponuky *Vek* a prepínačov v časti *Prevádzka*:

```
<label for="vek">Vek:</label>
  <select name="vek" id="vek">
    <option value="18-25">18 - 25</option>
    <option value="25-30">25 - 30</option>
    <option value="30-40">30 - 40</option>
    <option value="40-50">40 - 50</option>
    <option value="50-">50 a viac</option>
  </select>
</fieldset>
<br>
<fieldset>
  Prevádzka:
  <input type="radio" name="mesto" id="ba" value="BA" required>
  <label for="ba">Bratislava</label>
  <input type="radio" name="mesto" id="bb" value="BB" required>
  <label for="bb">Banská Bystrica</label>
  <br>
```

Všimnime si, že v časti *Prevádzka* nie je popisom text *Prevádzka*, ale text pri konkrétnych prepínačoch, teda Bratislava a Banská Bystrica, pretože každý prepínač je samostatný element formulára (na rozdiel od výberovej ponuky, ktorú celú zastrešuje jeden element `select`). Tiež si musíme uvedomiť, že hoci prepínače patriace do jednej skupiny majú rovnakú hodnotu atribútu `name`, hodnota atribútu `id` musí byť pre každý prepínač iná.

Element `label` nijako nemení vzhľad formulára, jeho prvkov, ani textov, ktoré tvoria popisy. Použitím elementu `label` tiež mierne uľahčíme prácu s jednotlivými prvkami formulára. Nemusíme už totiž klikať do konkrétneho prvku formulára (na prepínač, do textového poľa atď.), ale stačí kliknúť na popis prvku a efekt bude rovnaký, ako keby sme klikli na prvok formulára.

ÚLOHA 12.29

Upravte popisy pomocou elementu `label` aj pre všetky ostatné prvky formulára (okrem tlačidiel) v súbore `prihlaska.html`.

ZAPAMÄTAJTE SI

- aby sme mohli odoslať formulár, musí v ňom byť tlačidlo typu `submit`,
- každý prvok formulára musí mať definovaný atribút `name`,
- hodnota atribútu `name` prvkov formulára musí byť unikátna s výnimkou skupiny prepínačov (prepínače v rámci skupiny majú rovnakú hodnotu atribútu `name`),
- popisy formulárových elementov je vhodné definovať pomocou elementu `label`.

Formuláre a kaskádové štýly

Pomocou kaskádových štýlov môžeme ovplyvniť aj vzhľad formulárov, resp. jednotlivých formulárových elementov. Rovnako, ako pre iné elementy, môžeme definovať štýly pre elementy `input`, `select`, `textarea`, `button`, `form`, `fieldset`, `legend`, `label`. Môžeme im nastavovať farbu textu či pozadia, orámovanie, vonkajšie či vnútorné okraje, šírku, ... Môžeme tiež definovať štýl pre konkrétny element s využitím jeho atribútu `id`.

PRÍKLAD 12.30

Na stránke `prihlaska.html` zmeníme orámovanie všetkých textových polí (jednoriadkových aj viacriadkových) a výberových ponúk na oranžovú (`#ff9900`). Tiež im nastavíme horný a dolný vonkajší okraj, čím jednotlivé prvky od seba odsunieme. Tlačidlám zmeníme farbu pozadia na `#ff9900` a taktiež pridáme horný a dolný vonkajší okraj (*obrázok 12.20 vľavo*). Do hlavičky stránky (elementu `head`) pridáme nasledujúcu definíciu štýlov:

```
<style>
 input[type=text], select, textarea {
   border: 1px solid #ff9900;
   margin: 5px 0;
 }
 button {
   background-color: #ff9900;
   margin: 5px 0;
   border: none;        /* zruší preddefinované orámovanie */
 }
</style>
```

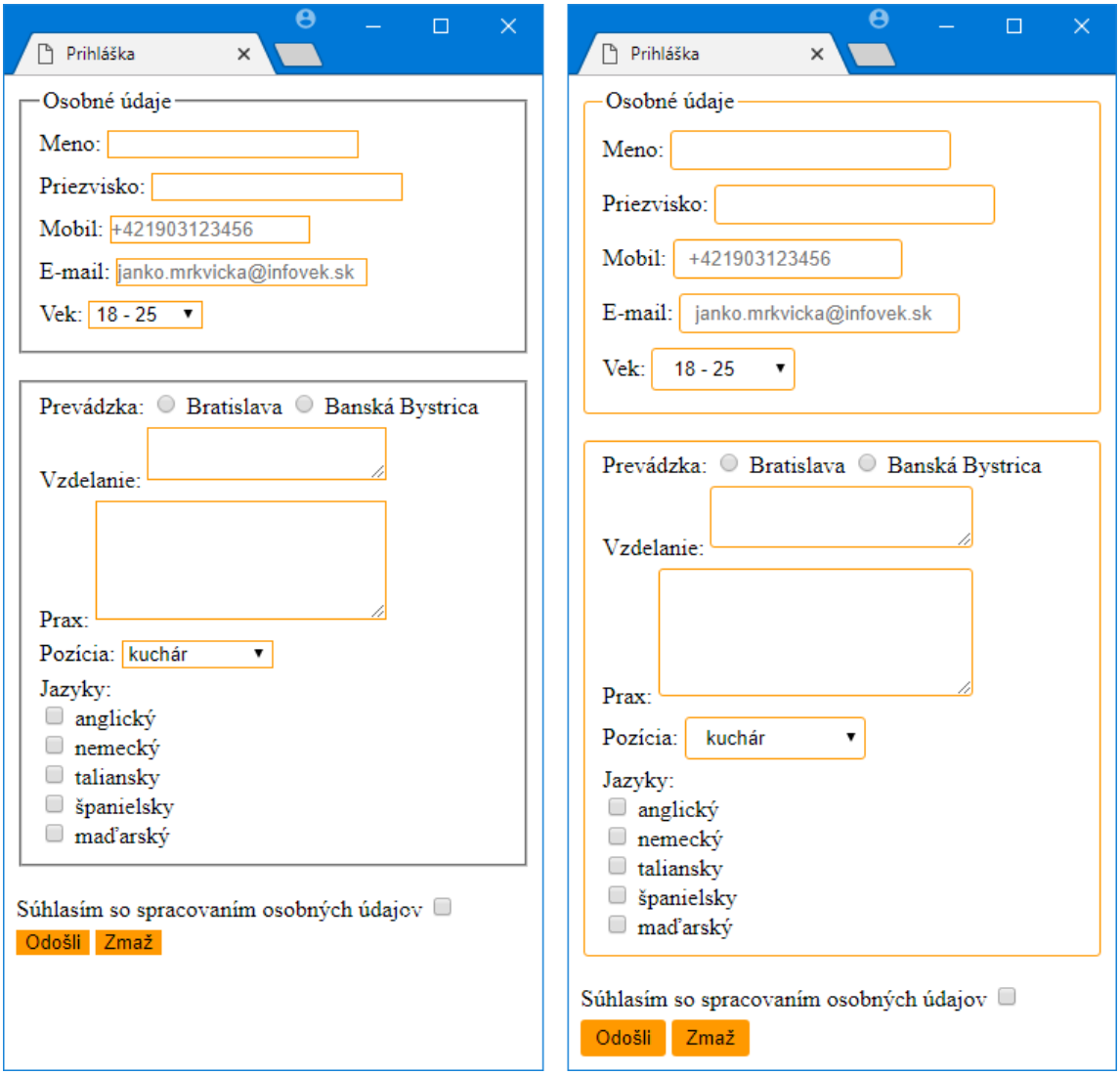
Kódom `input[type=text]` hovoríme, že chceme definovať štýl len pre tie elementy `input`, ktoré majú atribút `type` nastavený na hodnotu `text`, teda pre jednoriadkové textové polia. Podobne vieme definovať štýly len pre:

- odosielacie tlačidlá `input[type=submit]`,
- resetovacie tlačidlá `input[type=reset]`,
- obyčajné tlačidlá `input[type=button]`.

ÚLOHA 12.31

Do štýlov z predchádzajúceho príkladu:

- nastavte všetkým textovým poliam, výberovej ponuke, aj tlačidlám vnútorné okraje napr. na hodnotu `5px 10px`,
- nastavte rovnaké orámovanie, ako majú textové polia, aj pre element `fieldset`,
- experimentujte s hodnotami už definovaných vlastností,
- experimentujte s vlastnosťou `border-radius` pre element `button`, resp. pre iné elementy,
- experimentujte s inými nastaveniami, ktoré už poznáte.



Obrázok 12.20 Vľavo prihláška po aplikácii štýlov z *príkladu 12.30*, vpravo prihláška po pridaní niektorých vlastností z *úlohy 12.31*.

ÚLOHA 12.32

Vo formulári `rezervacia.html` upravte popisy všetkých formulárových prvkov pomocou elementu `label` a oštýlujte ju podľa vlastnej fantázie.

Metodika pre učiteľa

CIEĽ

Cieľom je oboznámiť sa s rôznymi typmi formulárových prvkov a ich spôsobom vkladania do HTML kódu.

VÝKLAD

V príkladoch a úlohách, v ktorých sa tvorí formulár, sme nepoužívali (ale vysvetlili) atribút `method`. Ak by ste chceli testovať, aké údaje sa posielajú, môžete použiť metódu GET a vždy po odoslaní formulára sa pozrieť do riadka s adresou. Pri dlhšom formulári to však už bude dosť neprehľadné.

K diskusii v úvode: vyhľadávanie, prihlasovanie, registrácia, objednávanie/nákup, testy, …

Úloha 12.5: Ide o to, aby si žiaci zistili, aké rôzne formulárové prvky sa vo formulároch môžu používať (textové políčka, prepínače, začiarkávacie políčka, výberová ponuka, rôzne druhy tlačidiel). Nezáleží na tom, ako žiaci tieto prvky pomenujú.

Úloha 12.10: Prepínače v druhej skupine musia mať iný atribút `name`, ako prepínače v prvej.

Nadväznosť na predchádzajúce kapitoly: Kapitola vyžaduje prebratie kapitol 1 – 8. Ak ste vôbec nevyučovali CSS, môžete vynechať časť 12.5.

Atribút required a placeholder

Kým neexistoval atribút `required`, musel takéto kontroly robiť vývojár buď ešte pred odoslaním formulára na strane klienta pomocou JavaScriptu, alebo až po odoslaní formulára na strane servera, v skripte, ktorý spracováva formulár.

`Placeholder` možno definovať aj k typom search, url, tel, email (z HTML5), ale tie sme v učebnici neuvádzali.

Obrázok 12.17: Reakcie sa graficky líšia v závislosti od prehliadača.

Doplňujúce úlohy

Úloha: K stránkam učebnice (z úlohy 11.16) pridajte stránku s objednávkovým formulárom na učebnicu. Premyslite, čo by takýto formulár mal obsahovať. Doplňte odkaz na stránku s formulárom aj do navigácie.

Úloha: Rozdeľte stránku IT Pizza na dve podstránky, v jednej bude ponuka a v druhej galéria. Ostatné časti (hlavička, navigácia, päta s kontaktmi) by mali zostať na oboch stránkach. Upravte navigáciu tak, aby obsahovala odkazy na stránky ponuka, galéria, prihlaska.html a rezervacia.html. Stránky prihlaska.html a rezervacia.html upravte tak, aby mali rovnaký vzhľad ako stránky IT Pizza, t.j. pridajte hlavičku, navigáciu a pätu, pripojte potrebné štýly.

13 VALIDITA

Z programovania (napr. v jazyku Python, Pascal, či v ľubovoľnom inom) viete, že písanie programov má isté pravidlá. Vedeli by ste si na nejaké spomenúť? Napríklad môžeme používať iba príkazy, ktorým príslušný programovací jazyk rozumie (ktoré pozná), začiatok a koniec nejakého bloku programu musíme označiť (v Pascale sa na to používajú rezervované slová begin a end, v Pythone odsadenie, v C++ či Jave zátvorky {}), atď.

Aj pri tvorbe zdrojového kódu webovej stránky platia isté pravidlá a zásady, ktoré by sme mali dodržiavať, ak chceme, aby prehliadače našu stránku zobrazovali správne.

Kontrolovať korektnosť zdrojového kódu (syntax) nie je jednoduché, resp. je zdĺhavé, najmä ak je stránka dlhšia. Môžu nám pomôcť tzv. **validátory**, napr. validátory konzorcia W3C (http://validator.w3.org).

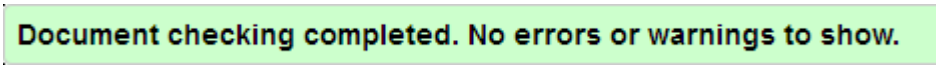
HTML validátor

PRÍKLAD 13.1

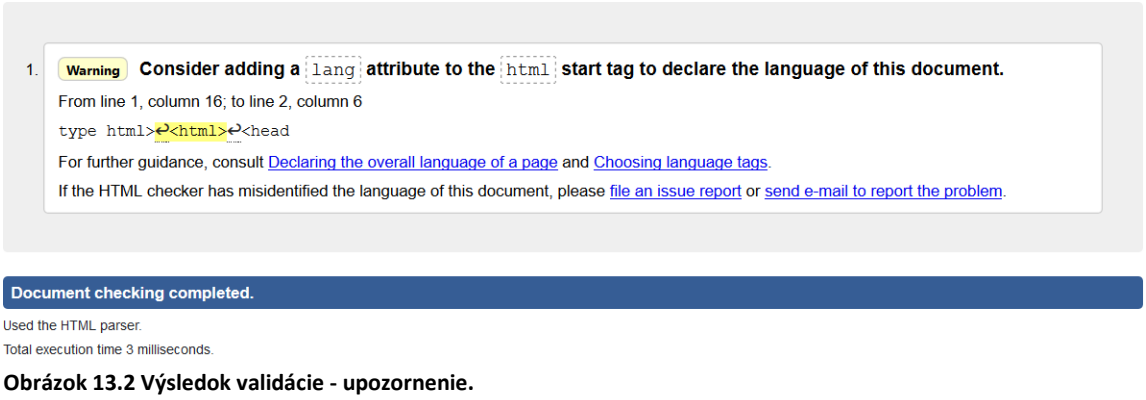
Skontrolujeme (budeme hovoriť, že **zvalidujeme**) staršiu verziu stránky IT Pizza (súbor `13/index.html`).

1. Na stránke https://validator.w3.org/ zvolíme záložku **Validate by File Upload**.
2. Stlačíme tlačidlo Browse a nájdeme súbor so stránkou IT Pizza (`13/index.html`), prípadne inou, ktorú chceme zvalidovať.
3. Stlačíme tlačidlo **Check**.

Pokiaľ sme dostali správu ako na *obrázku 13.1* alebo *13.2*, je naša stránka korektná, bez chýb. Hovoríme tiež, že je validná.



Obrázok 13.1 Výsledok validácie - žiadne chyby.



Obrázok 13.2 Výsledok validácie - upozornenie.

Na *obrázku 13.2* nám síce validátor „nepovedal", že naša stránka je bez chýb, zobrazil však len upozornenie (**Warning**) – konkrétne, že máme zvážiť použitie atribútu `lang` v elemente `html`). Upozornenia neznamenajú, že máme v kóde chyby a v učebnici sa im nebudeme venovať. Aj v tomto prípade budeme stránku považovať za validnú.

Spôsobom popísaným v *príklade 13.1* zvalidujeme stránku, ktorá sa nachádza lokálne na našom počítači, t.j. nie je zatiaľ verejne prístupná na internete.

PRÍKLAD 13.2

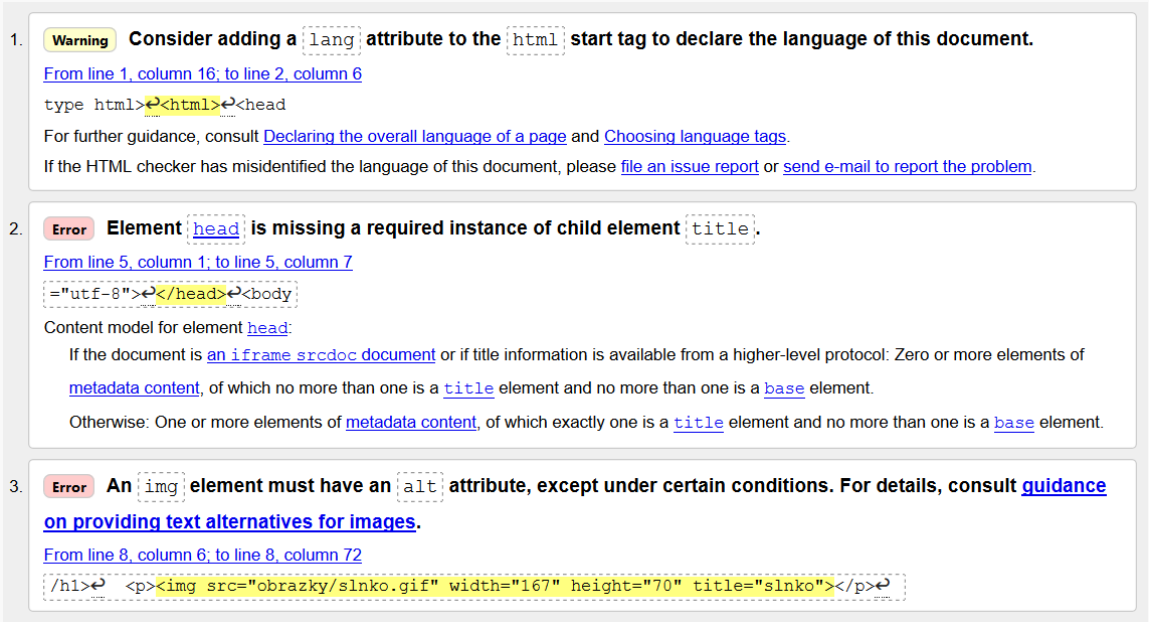
Zvalidujeme stránku s nasledujúcim zdrojovým kódom.

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
  <h1>Môj dnešný deň</h1>
  <img src="obrazky/slnko.gif" width="50" height="50">
  <p>Dnes o siedmej som bol vyvenčiť svojho psíka.<br>
  Vonku svietilo slniečko. Kúpil som si zmrzlinu.</p>
</body>
</html>
```

Jedna možnosť je, že si uvedený kód uložíme do html súboru a potom použijeme postup z *príkladu 13.1*. Môžeme však postupovať aj takto:

1. Na stránke https://validator.w3.org/ zvolíme záložku **Validate by Direct Input**.
2. Do veľkého textového poľa nakopírujeme zdrojový kód stránky, ktorú chceme zvalidovať. Musí to však byť kód celej stránky, vrátane definovania DOCTYPE a `html` elementu. My nakopírujeme kód zo súboru `13/priklad2.txt`.
3. Stlačíme tlačidlo **Check**.

Dostali sme výsledok podobný tomu na *obrázku 13.3*.



Obrázok 13.3 Výsledok validácie – zoznam chýb.

Tentokrát náš zdrojový kód nebol úplne korektný, validátor nám okrem upozornenia zobrazil aj nejaké chyby (*Error*). O každej chybe dostaneme niekoľko informácií: stručný popis chyby (príčina), na ktorom riadku a stĺpci sa chyba nachádza a prípadne ešte nejaký dodatočný popis. Dôležité je dobre si prečítať a skúsiť porozumieť úvodnému popisu chyby (prvý tučným písmom zvýraznený riadok). V našom zdrojovom kóde sa nachádzali tieto dve chyby:

Error **Element `head` is missing a required instance of child element `title`.**

čo v preklade znamená, že *V elemente `head` chýba element `title`* a

Error **An `img` element must have an `alt` attribute, except under certain conditions.**

t.j. *Element `img` musí mať atribút `alt`*.

ÚLOHA 13.3

Odstráňte chyby v zdrojovom kóde z *príkladu 13.2*, t.j. pridajte element `title` do elementu `head` a doplňte atribút `alt` do elementu `img`. Takto opravený kód znovu zvalidujte. Ak ste *úlohu 13.3* vyriešili správne, mali by ste po validácii dostať už nanajvýš upozornenie ako na *obrázku 13.2*.

PRÍKLAD 13.4

Ukážeme si ešte niekoľko ďalších chýb, ktoré sa môžu vyskytnúť pri kódovaní v HTML. Po validácii nasledujúceho zdrojového kódu

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Môj denník</title>
</head>
<body>
  <h1>Môj dnešný deň, <p>ktorý dopadol celkom inak</p></h1>
  <img src="obrazky/slnko.gif"alt="slnko" width="50" height="50">
  <p>Dnes o siedmej som bol vyvenčiť svojho psíka.<br>
  Vonku svietilo slniečko. Kúpil som si zmrzlinu.</p>
  <img scr="obrazky/pes.gif" alt="pes" width="100" height="66">
</body>
</html>
```

dostaneme štyri správy o chybách.

Spáva Error **Element `p` not allowed as child of element `h1` in this context.**

súvisí s riadkom kódu `<h1>Môj dnešný deň, <p>ktorý dopadol celkom inak</p></h1>` a hovorí nám, že element `p` nemôžeme použiť v rámci elementu `h1` (`p` nemôže byť "dieťaťom" `h1`).

Správa Error No space between attributes. sa týka kódu `<img src="obrazky/slnko.gif"alt="slnko" width="50" height="50">` a hovorí, že chýba medzera medzi atribútmi. Skutočne, atribút `alt` sme od predchádzajúceho zabudli oddeliť medzerou.

Nasledujúce dve chyby spolu súvisia a obe sa týkajú riadku: `<img scr="obrazky/pes.gif" alt="pes" width="100" height="66">`.

Správa Error Attribute `scr` not allowed on element `img` at this point. nás informuje o tom, že v elemente `img` nie je povolený atribút `scr` a správa

Error Element `img` is missing required attribute `src`. zase o tom, že v elemente `img` nemáme atribút `src`. V tomto prípade ide o obyčajný preklep, namiesto `scr` sme mali napísať `src`. Na tomto príklade teda vidíme, že niekedy môže s jednou chybou súvisieť aj viac chybových správ.

ÚLOHA 13.5

Opravte chyby v kóde z *príkladu 13.4* (nájdete ho v súbore `13/priklad4.html`) a znovu ho zvalidujte. Opakujte, až kým kód nebude validný.

ÚLOHA 13.6

Zvalidujte nasledujúci zdrojový kód (súbor `13/uloha6.html`) – odporúčame validáciu pomocou **Validate by Direct Input**. Prečítajte si výsledok validácie a skúste porozumieť jednotlivým správam o chybách. Chyby postupne opravte a zvalidujte kód znovu. Opakujte, až kým kód nebude validný.

```
<!doctype html>
<html>
<head>
  <meta charset="utf-9">
  <title>Úloha 5.6</title>
</head>
<body>
  <img src="obrazky/pizza.png"alt="pizza" width="167" height="7O">
  <h1>Pizzeria Emanuel
  <h2>pizza, na ktorú nezabudnete</h2>
  </h1>
  <em>Vyberte si z našej <h3>ponuky </h3>:</em> margerita,
cardinale, funghi, havai
  <h2Kontakt</h2>
  <address>Mlynská dolina, 842 48 Bratislava</address>
</body>
</html>
```

ÚLOHA 13.7

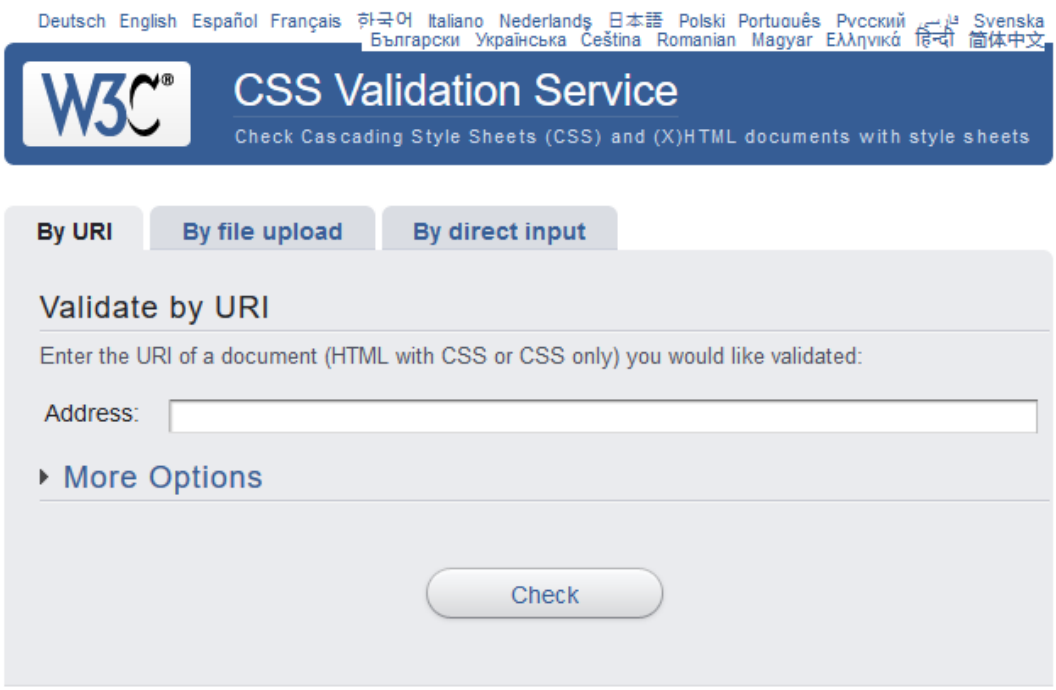
Zvalidujte stránku vašej školy, stránku vášho mesta alebo iné stránky, ktoré často navštevujete. Využite záložku **Validate by URI** na stránke validator.w3.org – do položky Address zadajte adresu validovanej stránky, napr. www.w3schools.com, www.bratislava.sk.

ÚLOHA 13.8

Zvalidujte zdrojový kód ľubovoľnej stránky, ktorú ste vytvorili v rámci nejakej úlohy z tejto učebnice (teda nie stránku, ktorú ste dostali ako vstupnú). Zistil validátor nejaké nedostatky? Ak áno, prediskutujte ich so spolužiakmi či učiteľom a odstráňte.

CSS validátor

Na validovanie CSS kódu môžeme použiť online nástroj konzorcia W3C na stránke https://jigsaw.w3.org/css-validator/



Obrázok 13.4 CSS validátor konzorcia W3C.

Dizajnovo CSS validátor vyzerá podobne ako HTML validátor, s ktorým sme pracovali v predchádzajúcej podkapitole. Opäť si môžeme zvoliť z troch možností:

- validovanie CSS kódu stránky, ktorá je na webe – záložka **By URI**,
- validovanie CSS súboru, ktorý máme v počítači – záložka **By file upload**,
- validovanie skopírovaného CSS kódu – záložka **By direct input**.

PRÍKLAD 13.9

Zvalidujme nasledujúci CSS kód z *príkladu 7.3*.

```
h1 {
  font-family: Verdana;
}
h2 {
  font-style: italic;
  font-weight: normal;
}
```

```
a {
  font-weight: bold;
  color: blue;
  text-decoration: none;
}
```

Postupujeme nasledovne.

1. Na stránke https://jigsaw.w3.org/css-validator/ zvolíme záložku **By direct input**.
2. Do veľkého textového poľa nakopírujeme vyššie uvedený CSS kód, ktorý chceme zvalidovať.
3. Stlačíme tlačidlo **Check**.

Ak sa zobrazí správa ako na *obrázku 13.5*, náš kód je validný.

Obrázok 13.5 Výsledok validácie – žiadne chyby.

PRÍKLAD 13.10

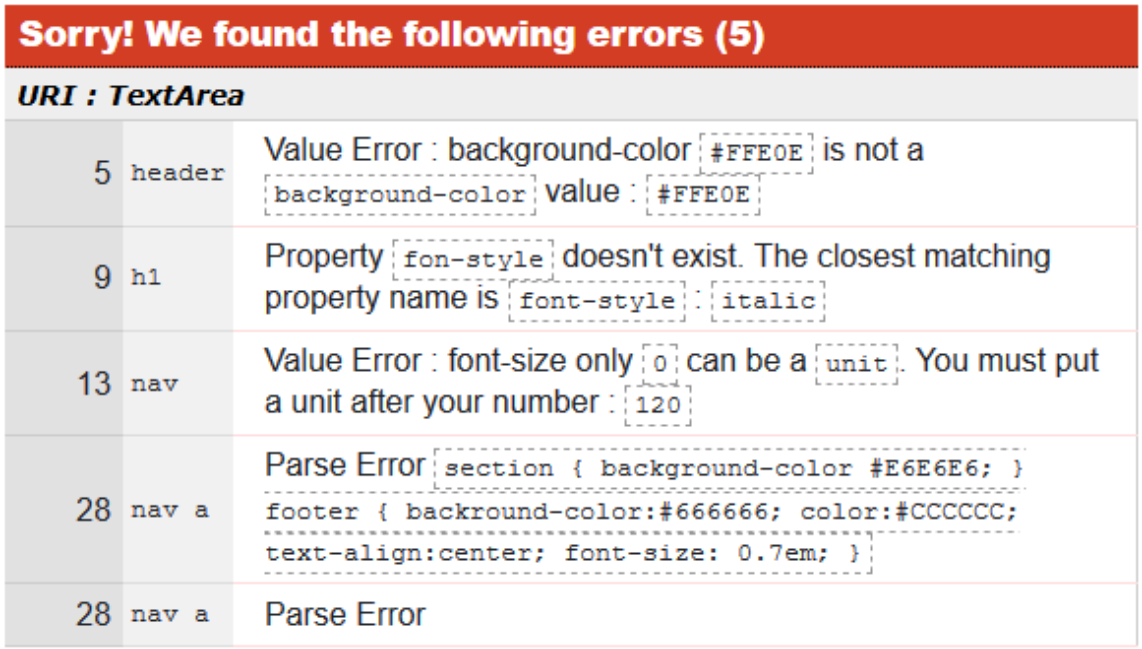
Zvalidujeme nasledujúci CSS kód.

```
body {
    font-family:Verdana, Geneva, Arial, Helvetica, sans-serif;
}
header {
    background-color:#FFE0E;
}
h1 {
    color: #9F0000;
    fon-style: italic;
}
nav {
    background-color: #666666;
    font-size: 120;
}
nav a {
    color: #FFFFFF;
    text-decoration: none;
    font-weight:bold;

section {
    background-color #E6E6E6;
}
footer {
    backround-color:#666666;
    color:#CCCCCC;
    text-align:center;
    font-size: 0.7em;
}
```

Tentokrát si vyššie uvedený kód skopírujeme do CSS súboru a ten si vo validátore vyvoláme cez záložku **By file upload**, kde stlačíme tlačidlo **Browse**.

Po kliknutí na tlačidlo **Check** sa nám zobrazí výsledok validácie, ktorý s veľkou pravdepodobnosťou vyzerá nasledovne.



Sorry! We found the following errors (5)

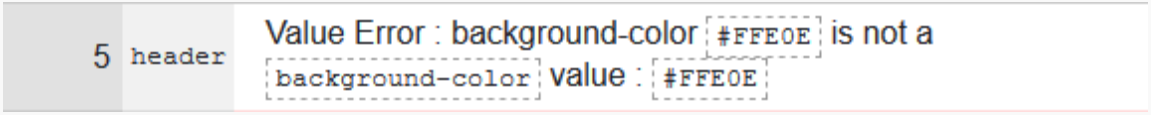
URI : TextArea

5	`header`	Value Error : background-color `#FFE0E` is not a `background-color` value : `#FFE0E`
9	`h1`	Property `fon-style` doesn't exist. The closest matching property name is `font-style` : `italic`
13	`nav`	Value Error : font-size only `0` can be a `unit`. You must put a unit after your number : `120`
28	`nav a`	Parse Error `section { background-color #E6E6E6; } footer { backround-color:#666666; color:#CCCCCC; text-align:center; font-size: 0.7em; }`
28	`nav a`	Parse Error

Obrázok 13.6 Výsledok validácie – zoznam chýb.

Validátor našiel päť chýb, ktoré si teraz popíšeme podrobnejšie:

Správa

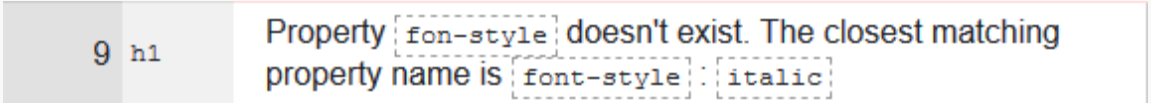


5	`header`	Value Error : background-color `#FFE0E` is not a `background-color` value : `#FFE0E`

súvisí s riadkom číslo 5, v ktorom je `background-color:#FFE0E;`

Value Error znamená, že nastavujeme chybnú hodnotu. Konkrétne pre element `header` nastavujeme farbu pozadia na hodnotu `#FFE0E`, čo nie je šesťciferné hexadecimálne číslo, ale len päťciferné. Je potrebné ešte jednu cifru doplniť.

Správa

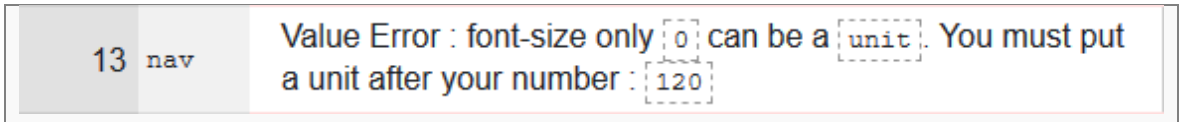


9	`h1`	Property `fon-style` doesn't exist. The closest matching property name is `font-style` : `italic`

súvisí s riadkom číslo 9, v ktorom je `fon-style: italic;`

Validátor nás upozorňuje, že vlastnosť `fon-style` neexistuje a pravdepodobne tam má byť `font-style`.

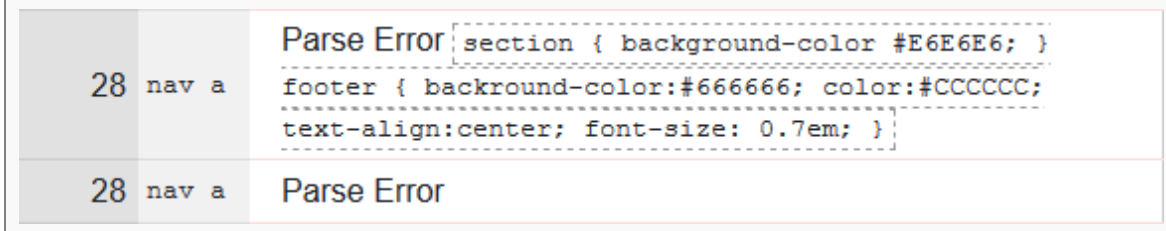
Správa



súvisí s riadkom 13, v ktorom je `font-size: 120;`

Validátor nás upozorňuje, že za číslom 120 nie je špecifikovaná jednotka, potrebné je doplniť znak `%`.

Správy



súvisia s elementom `nav a`. **Parse Error** väčšinou indikuje chýbajúcu zátvorku alebo dvojbodku. Pre uvedený element naozaj chýba pravá zložená zátvorka. Číslo 28 však nie je číslom riadku, do ktorého treba zátvorku doplniť, ale číslo posledného riadku kódu.

Všimnime si, že pod zoznamom chýb validátor zobrazil aj validnú časť kódu, ako je na *obrázku 13.7*. Zaujímavé je, že okrem častí, v ktorých validátor našiel chyby, tam chýba celá časť kódu počnúc elementom `nav a`.

Obrázok 13.7 Výsledok validácie – validný kód.

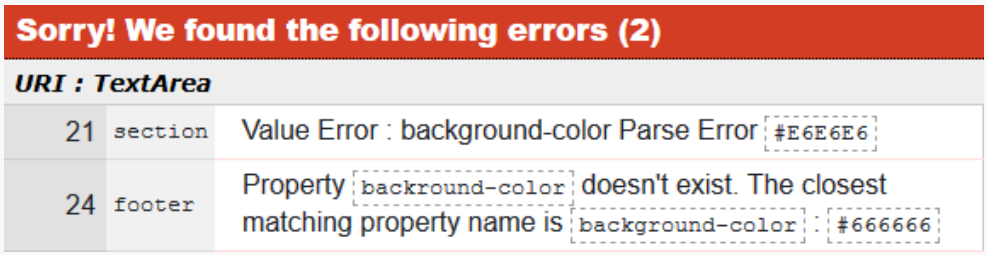
PRÍKLAD 13.11

Opravme chyby, ktoré v predchádzajúcom príklade zistil validátor a kód opäť zvalidujme. Opravené časti kódu sú zvýraznené.

```
body {
    font-family:Verdana, Geneva, Arial, Helvetica, sans-serif;
}
header {
    background-color:#FFE0E0;
}
h1 {
```

```
    color: #9F0000;
    font-style: italic;
}
nav {
    background-color: #666666;
    font-size: 120%;
}
nav a {
    color: #FFFFFF;
    text-decoration: none;
    font-weight:bold;
}
section {
    background-color #E6E6E6;
}
footer {
    backround-color:#666666;
    color:#CCCCCC;
    text-align:center;
    font-size: 0.7em;
}
```

Na naše prekvapenie, validátor našiel ďalšie chyby, ktoré neodhalil pri predchádzajúcom validovaní. Zobrazí sa nám výsledok validácie ako na obrázku

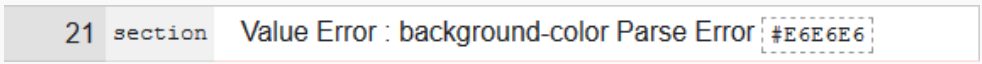


Obrázok 13.8 Výsledok validácie – zoznam ďalších chýb.

Tieto chyby sa nám predtým nezobrazili z toho dôvodu, že si validátor pravdepodobne nedokázal poradiť s kontrolou kódu, ktorý nasledoval za chýbajúcou zátvorkou. Keď sme ju doplnili a spustili validáciu, potom validátor fungoval.

Rozoberme tieto novo nájdené chyby.

Správa



súvisí s kódom

```
section {
    background-color #E6E6E6;
}
```

Za vlastnosťou `background-color` chýba dvojbodka.

Správa

24	`footer`	Property `backround-color` doesn't exist. The closest matching property name is `background-color` : `#666666`

súvisí s kódom `backround-color:#666666;` v riadku 24. Je tam chybne napísaná vlastnosť `backround-color` namiesto `background-color`.

Keď chyby v kóde opravíme, musíme kód opäť zvalidovať a v prípade, že validátor nájde ďalšie chyby, je potrebné tento postup opakovať, až kým validátor zobrazí správu, že nenašiel žiadne chyby, ako na *obrázku 13.5*.

ÚLOHA 13.12

Opravte vyššie zistené chyby a opäť kód zvalidujte. V prípade potreby postup opakujte, až kým bude kód validný.

ÚLOHA 13.13

Zvalidujte CSS kód v súboroch pizzerie, ktoré ste vytvárali v rámci predchádzajúcich kapitol. Ak validátor nájde chyby, odstráňte ich.

ÚLOHA 13.14

Zvalidujte CSS kód stránky vašej školy, prípadne vášho mesta, alebo iné stránky, ktoré často navštevujete. Využite záložku **By URI** a do položky **Address** zadajte adresu validovanej stránky.

ZAPAMÄTAJTE SI

- Validovanie nám pomôže odhaliť, prečo sa stránka nezobrazuje tak, ako by sme chceli.
- Stránka s validným kódom sa s veľkou pravdepodobnosťou zobrazí správne aj v iných prehliadačoch, ako je ten náš.
- Validita je dôležitá aj kvôli prístupnosti informácii, ktorej sa budeme venovať v *kapitolách 15 až 18*.

Metodika pre učiteľa

CIEĽ

Cieľom je oboznámiť sa so spôsobmi validovania HTML a CSS kódu a uvedomiť si dôležitosť a význam validácie.

Nadväznosť na predchádzajúce kapitoly: Kapitola vyžaduje kapitoly 1-7. Časť o HTML validácii sa dá zaradiť už po 4. kapitole, časť o CSS validácii odporúčame najskôr po 7. kapitole.

Po absolvovaní tejto kapitoly odporúčame, aby žiaci vypracovali projekt **Webové sídlo súťaže**. Zadanie je v súbore *projekty*-pre_ziakov.zip. Žiaci môžu vytvárať webové sídlo podľa vlastného výberu, prípadne môžu vytvárať stránky podľa predlohy na obrázkoch. V takom prípade môžu použiť poskytnuté dátové súbory. Učitelia môžu pri kontrole použiť výslednú stránku v súbore *projekty*-pre_ucitelov.zip. Podľa motivačných preferencií žiakov možno použiť aj ďalšie námety na projekty, ktoré sú tiež súčasťou uvedených súborov.

14 PUBLIKOVANIE A SPRÁVA OBSAHU WEBOVEJ STRÁNKY

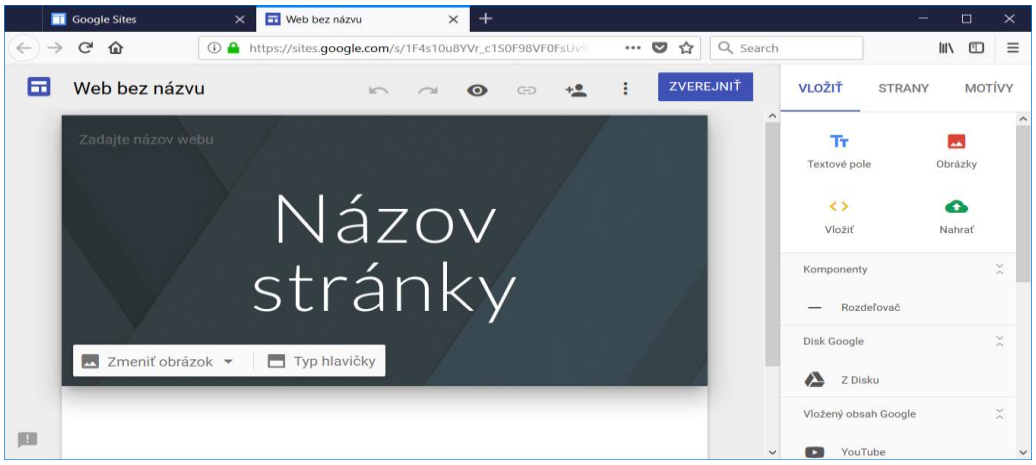
Webové stránky pizzerie, ktoré sme v predchádzajúcich kapitolách vytvárali, máme zatiaľ uložené na lokálnom disku počítača, prípadne na svojom študentskom konte, ktoré máme vyhradené na školskom serveri. Zatiaľ však pravdepodobne nie sú zverejnené na internete.

Možnosti publikovania na webe

Aby si naše stránky mohol prezerať ktokoľvek cez internet, musíme ich umiestniť na webový server.

Webový server môžeme použiť vlastný, alebo si ho môžeme prenajať. Prenájom webového servera môže byť realizovaný rôznymi spôsobmi.

- **Webhosting** – prenájom webového priestoru na serveri. Môže byť zdarma – freehosting, alebo spoplatnený – komerčný hosting (zvyčajne poskytuje výhody naviac, napríklad podstatne viac priestoru). V oboch prípadoch využívame služby webového servera s vlastnosťami, ktoré nevieme ovplyvniť a sú určené prenajímateľovou inštaláciou. Pre freehosting prenajímateľ často poskytuje aj systémy správy obsahu (CMS - Content Management System). Medzi najznámejšie patria Joomla (www.joomla.org), Google webové stránky (sites.google.com), WordPress (sk.wordpress.org), eStranky (www.estranky.sk).
- **Serverhosting** – prenájom fyzického servera (hardvéru), ktorý si môžeme sami nainštalovať. Celá réžia servera je v našich rukách a je len na nás, čo a ako si nainštalujeme (vrátane operačného systému, aplikácií, ...). Inštaláciu servera môže realizovať aj prenajímateľ na základe našich požiadaviek.
- **Server housing** – umiestnenie vlastného fyzického servera u poskytovateľa internetového pripojenia, ktorý má zvyčajne lepšie pripojenie a zabezpečenie serverovne ako my. Môže mať napríklad lepšiu priepustnosť siete, klimatizovanú serverovňu, záložné zdroje a prepracovanejší systém zálohovania.



Obrázok 14.1 Snímka systému správy obsahu na sites.google.com.

ÚLOHA 14.1

Zistite, aké možnosti publikovania na webe ponúka pre svojich študentov vaša škola.

Nahrávanie súborov

Keď máme k dispozícii prístup na webový server, môžeme naň umiestniť nami vytvorené súbory. Nahrávanie súborov na server sa tiež často označuje anglickým slovom **upload**. Sú dve možnosti pre nahrávanie súborov – cez WWW alebo cez FTP.

Nahrávanie cez WWW

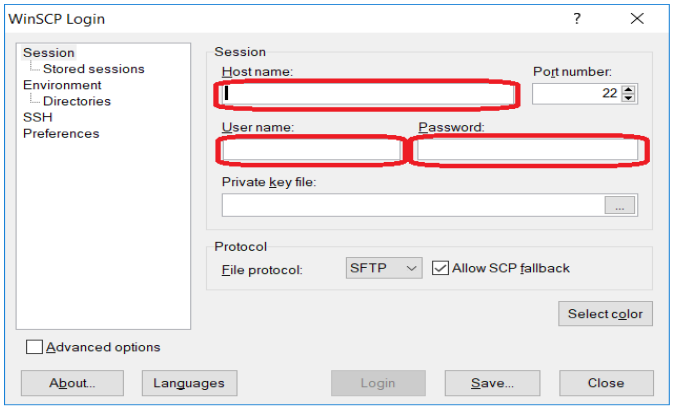
Túto možnosť poskytujú najčastejšie poskytovatelia služby webhosting. Po prihlásení na webový server sa nastavíme na príslušnú stránku a vyberieme súbory, ktoré si prajeme nahrať. Konkrétny postup je však u každého poskytovateľa iný.

Nahrávanie cez FTP

Táto možnosť sa používa v prípade, že máme možnosť na serveri používať FTP službu na nahrávanie (upload) alebo sťahovanie (download) súborov. Pre využívanie FTP služieb potrebujeme klientský program. Klientský program komunikuje so serverom za pomoci protokolu FTP (File Transfer Protocol). Pri používaní štandardného FTP protokolu sa posielajú prihlasovacie údaje v čitateľnej podobe, a teda hrozí, že odchytením komunikácie môže neoprávnená osoba získať prístup k našim súborom. V takom prípade je vhodnejšie použiť zabezpečené pripojenie k serveru. Najčastejšie sa používa protokol SFTP (SSH File Transfer Protocol). Protokol SFTP pre zabezpečenie komunikácie využíva protokol SSH-2 (Secure Shell), ktorý komunikuje na porte 22. Protokol SFTP podporujú napríklad klienti FileZilla (www.filezilla-project.org) a WinSCP (winscp.net), taktiež ho podporuje aj súborový manažér Total Commander (www.ghisler.com) a FAR manager (www.farmanager.com) s modulom (plugin) WinSCP.

Pri prihlasovaní na server je potrebné zadať adresu servera a prihlasovacie údaje. Tieto informácie spravidla poskytne správca servera.

V názvoch súborov nepoužívame špeciálne znaky (diakritiku a podobne) a medzery, pretože sa po nahratí na server môžu tieto znaky nahradiť inými znakmi a adresy stránok nebudú fungovať správne. Súbory potrebné pre zobrazenie webovej stránky sa väčšinou nahrávajú do priečinku **public_html** a súbor so zdrojovým kódom úvodnej stránky by mal byť nazvaný ako **index.html**. Toto však nemusí byť pravidlom. Informácie o tom, ako to funguje na konkrétnom webovom serveri, si treba zistiť u správcu alebo učiteľa informatiky. Oni vám tiež prezradia, aká bude adresa vašich webových stránok.



Obrázok 14.2 Snímka prihlasovacieho okna programu WinSCP.

ÚLOHA 14.2

Nahrajte stránky pizzerie, ktoré ste vytvárali na predchádzajúcich hodinách, na dostupný webový server.

Sociálne a právne aspekty publikovania na webe

Akonáhle webovú stránku sprístupníme pre širokú verejnosť, musíme si byť vedomí všetkých rizík, ktoré s tým súvisia. Je preto potrebné pamätať najmä na nasledujúce aspekty:

- validita zdrojového kódu,
- prístupnosť a použiteľnosť,
- dodržiavanie etických a právnych noriem.

Validita zdrojového kódu

Zdrojový kód webovej stránky musí byť korektný, aby ju dokázal správne zobraziť akýkoľvek webový prehliadač. Problematike zabezpečenia validity sa venujeme v *kapitole 13*.

Prístupnosť a použiteľnosť

Malo by byť v našom záujme, aby nami publikované informácie boli prístupné čo najväčšiemu okruhu návštevníkov bez ohľadu na ich zdravotný stav, znalosti, skúsenosti a zobrazovacie možnosti. Okrem zabezpečenia prístupnosti je dôležité, aby získali z nášho webového sídla príjemný dojem a radi sa naň vracali. Problematike zabezpečenia prístupnosti a použiteľnosti sa venujeme v *kapitolách 15 až 18*.

Dodržiavanie etických a právnych noriem

Ako pokročilí používatelia internetu určite ovládate pravidlá netikety. Treba ich dodržiavať nielen pri komunikácií cez internet, ale aj pri publikovaní informácií na webe. V tejto časti spomenieme tri pravidlá netikety, ktoré treba brať do úvahy.

Ako prvé uvedieme nasledujúce pravidlo.

Majte ohľad k druhým. Nie každý má rýchle internetové pripojenie ako vy.

Pri publikovaní na webe môžeme toto pravidlo aplikovať nasledujúcim spôsobom.

- Dbajme na to, aby súbory s obrázkami neboli príliš veľké. Potrebnú veľkosť obrázka nastavme v aplikácii určenej na prácu s grafikou a nie až v zdrojovom kóde.
- Používajme stručné a jasné texty. Nezahlcujme čitateľa zbytočnými informáciami.
- Ak je cieľom odkazu nejaký rozsiahly súbor, uveďme na stránke informáciu o jeho veľkosti.

Pri publikovaní na webe by sme sa mali zaujímať o autorské práva získavaných a následne nami publikovaných informácií. Hovorí sa o tom aj v nasledujúcich pravidlách netikety.

Rešpektujte autorské práva iných. Nepublikujte cudzí text pod svojim menom, vždy uvádzajte meno pravého autora a zdroj, odkiaľ je text prevzatý.

Nevydávajte za svoju prácu niekoho iného. Obrázky, texty a rôzne iné súbory sa z internetu dajú ľahko stiahnuť. Akoby sa vám páčilo, keby niekto iný vydával vaše dielo za svoje? Ak využijete prácu iných, mali by ste spomenúť ich autorstvo.

Nami vytvorené webové stránky sú našou vizitkou, preto by sme mali na nich spravidla používať text s diakritikou, ktorý je po jazykovej a gramatickej stránke korektný a slušný. Vyhýbajme sa tiež akémukoľvek hanlivému obsahu. Týmto aspektom sa venuje nasledujúce pravidlo.

Nebuďte grobianom! Snažte sa o správny pravopis. Publikovať nepravdivé informácie, alebo niekoho ohovárať tiež nie je vhodné.

Na hodnovernosť publikovaných informácií má pozitívny vplyv, ak na stránke uvedieme meno ich autora a dátum poslednej aktualizácie.

Okrem dodržiavania pravidiel netikety netreba zabúdať na **ochranu osobných údajov** (meno, priezvisko, rodné číslo, adresa, ...) a v žiadnom prípade nezverejňovať citlivé údaje (údaje týkajúce sa vierovyznania, zdravia, politiky, rasovej príslušnosti, ...) o sebe a iných osobách.

ÚLOHA 14.3

Skontrolujte stránky pizzerie, ktoré ste vytvorili, či sú na nich dodržané etické a právne normy uvedené v tejto kapitole. Odstráňte prípadné nedostatky.

DISKUTUJTE

Aké máte skúsenosti s dodržiavaním nariadenia GDPR pri publikovaní informácií na webe?

Metodika pre učiteľa

CIEĽ

Cieľom je poznať základné možnosti publikovania webových stránok a využívania hostingových služieb. Okrem toho si majú študenti precvičiť umiestňovanie vytvoreného webového sídla na webový server a zvažovať sociálne, etické a právne aspekty publikovania na webe.

MOTIVÁCIA

Študenti možno nebudú mať potrebu publikovať webové sídlo fiktívnej pizzerie. V úvode hodiny môže prebehnúť diskusia o tom, aký obsah by chceli publikovať na webe a aké sú prípadné výhody a riziká s tým spojené.

VÝKLAD

Je možné, že študenti už v minulosti publikovali nejaký obsah na webe. Mohli by informovať spolužiakov o systémoch, ktoré použili a svojich skúsenostiach s nimi.

Pre úspešný priebeh hodiny odporúčame, aby škola umožnila študentom prístup a ukladanie vytvorených súborov na webový server. Nemusí to byť nutne vzdialený prístup cez SFTP, postačí aj prístup na sieťový disk v rámci školskej siete.

Pre plynulý priebeh záverečnej diskusie odporúčame, aby si učiteľ vopred pripravil niekoľko ukážok porušenia GDPR.

Nadväznosť na predchádzajúce kapitoly: Táto kapitola sa môže zaradiť už skôr (napríklad po kapitole 9). Je na zvážení učiteľa, či a kedy bude chcieť, aby žiaci svoje stránky publikovali.

ZHRNUTIE

V závere hodiny môže prebehnúť diskusia o tom, či študenti prehodnotili svoj názor na obsah, ktorý by chceli publikovať na webe.

15 OSOBY SO ŠPECIFICKÝMI POTREBAMI A WEB

Na predchádzajúcich hodinách sme vytvorili webové stránky pizzerie. Budú však tieto stránky vhodné naozaj pre všetkých používateľov webu? Gymnazista Gabo nevidí. Bude si vedieť zistiť, aké pizze sú v ponuke a dokáže si niektorú z nich objednať? Jeho dedko má tiež rád pizzu, ale trpí farbosleposťou. Dokáže si prečítať všetky informácie?

Na svete je asi 15% osôb s nejakým zdravotným postihnutím

ODPOVEDZTE

- Ako pracujú s počítačom nevidiaci?
- Aké ďalšie osoby, okrem osôb so zrakovým postihnutím, by mohli mať problémy pri práci s informáciami na webe?

Používatelia webu so špecifickými potrebami

Keď uvažujeme o tom, ktorí používatelia by mohli mať s používaním webu problémy, zamyslime sa nad tým, aké schopnosti si vyžaduje práca s webom a s počítačom samotným. Uvedomíme si, že nevyhnutné sú nasledujúce schopnosti.

- Musíme byť schopní zadávať vstupy pomocou klávesnice alebo myši.
- Musíme vidieť na obrazovku.
- Musíme rozumieť tomu, čo vidíme na obrazovke.
- Musíme byť schopní zmysluplne interagovať s počítačom (napríklad musíme vedieť, kde kliknúť, ako rolovať obsah okna a podobne).

Z uvedeného vyplýva, že problémy s používaním počítača by mohli mať nasledujúce skupiny používateľov.

- Osoby s poruchami jemnej motoriky.
- Osoby s nejakým druhom zrakového postihnutia.
- Osoby, ktoré majú problémy porozumieť textu.
- Osoby s poruchami pamäti a pozornosti.

Aby sme dokázali pochopiť problémy spomínaných používateľov, skúsme chvíľu pracovať s počítačom a vnímať okolitý svet tak ako oni.

Nevidiaci používatelia

Nevidiace osoby prijímajú všetky informácie z okolia len pomocou sluchu, hmatu, čuchu a chuti. Pri práci s počítačom zadávajú vstup iba pomocou klávesnice. Myš nepoužívajú, pretože by nedokázali kontrolovať nastavenie kurzora myši na obrazovke. Musia si teda pamätať množstvo klávesových príkazov.

Nevidiaci tiež nepoužívajú monitor, ale obsah obrazovky im postupne číta špeciálny program – čítač obrazovky. Najznámejším čítačom obrazovky je program NVDA, ktorý je voľne dostupný.

ÚLOHA 15.1

Ovládajte počítač iba pomocou klávesnice.

a) Spustite nejaký program z pracovnej plochy.
b) Minimalizujte okno programu.
c) Spustite ďalší program zo štartovacej ponuky.
d) Prepnite sa do predchádzajúceho programu.
e) Zavrite postupne okná bežiacich programov.

ÚLOHA 15.2

15/opravovacky.doc

Spustite čítač obrazovky NVDA. Tento voľne dostupný program si môžete stiahnuť zo stránky https://www.nvaccess.org/download/.

a) Otvorte dokument `opravovacky.docx`. Pracujte iba pomocou klávesnice a s vypnutou obrazovkou. Nájdite a opravte chyby v texte. (Po riadkoch textu sa pohybujte šípkou hore a dole, po slovách pomocou Ctrl+šípka vľavo alebo vpravo, po znakoch pomocou šípky vľavo alebo vpravo.)
b) Otvorte si webovú stránku www.gymls.edu.sk/indexsk.htm. Pomocou tabulátora sa pohybujte po textoch, ktoré sú odkazmi na iné stránky. Čo hovorí čítač obrazovky? Sú tieto texty zrozumiteľné a výstižné? Dostáva nevidiaci používateľ rovnakú informáciu ako vidiaci? (Zoznam odkazov môžete vyvolať aj pomocou klávesovej skratky Insert+F7.)
c) Otvorte si webovú stránku www.unss.sk. Čo hovorí čítač po nastavení sa na obrázok? Dostáva nevidiaci používateľ rovnakú informáciu ako vidiaci? (Pomocou klávesu G sa pohybujte po jednotlivých obrázkoch.)
d) Otvorte si webovú stránku http://www.uvn.sk/pre-pacientov. Pracujte iba pomocou klávesnice a s vypnutou obrazovkou. Zistite, o čom sa v dokumente píše a akú má štruktúru. Použite klávesové príkazy na pohyb po nadpisoch (kláves H, prípadne Shift+H).

Slabozrakí používatelia

Slabozrakí používatelia zadávajú vstup pomocou myši a aj pomocou klávesnice, ako aj iní používatelia. Aby však videli, čo je na obrazovke, používajú zväčšovací softvér. Takýto softvér je aj súčasťou operačného systému.

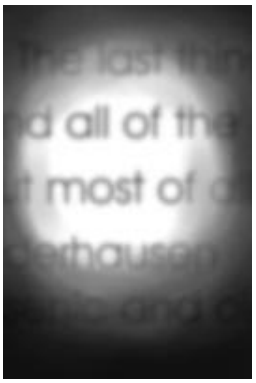
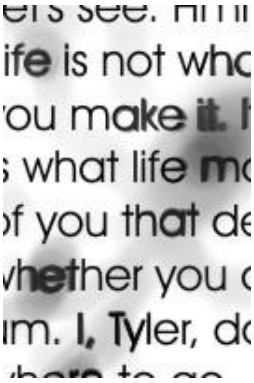
ÚLOHA 15.3

Spustite si program **Zväčšovacie sklo**. Nastavte si v ňom zväčšenie 400%.

a) V dokumente http://volby.statistics.sk/nrsr/nrsr2012/sr/tab2okr.jsp@lang=sk.htm zistite, aký bol počet voličov z obce Námestovo, ktorí hlasovali osobne. Akú stratégiu ste použili?
b) V dokumente https://publi.cz/books/160/04.html si pozrite jednotlivé časti zariadenia na obrázku 3 a ich popisy v texte. Dá sa sledovať oboje, obrázok aj popis? Je vhodnejšie popisy pridávať rovno do obrázka, ako je to na obrázkoch 1 a 2?

Slabozrakosť sa vyskytuje v najrôznejších formách. Zväčšovací softvér nie je vhodným riešením pre každú z nich. Napríklad pri narušenom centrálnom alebo periférnom videní je jeho použitie nevhodné, pretože postihnutá osoba vidí po zväčšení menej objektov.

Tabuľka 15.1 Simulácie videnia pri rôznych poruchách zraku [20].

			
Porucha periférneho videnia	**Porucha centrálneho videnia**	**Poškodenie sietnice**	**Šedý zákal**

Osoby s poruchami farebného videnia

Asi 10% osôb má problémy s vnímaním farieb. Za bezchybnú schopnosť rozlišovania farieb sú zodpovedné čapíky v oku. Čapíky môžu ako zmyslový vnem vnímať svetelné lúče s vlnovou dĺžkou približne medzi 760 nm (červené) a 380 nm (modré).

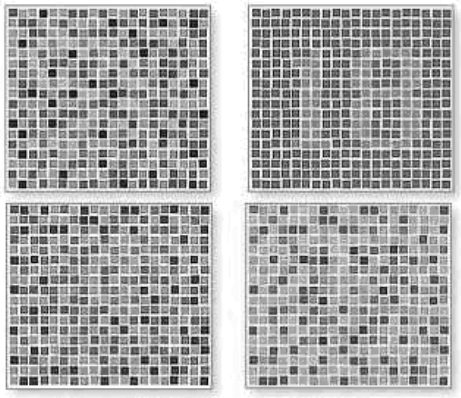
V čapíkoch sú tri rôzne zrakové substancie, ktoré sú podľa vlnovej dĺžky dopadajúceho svetla rôznym spôsobom stimulované a ktoré vedú k rozličným farebným vnemom v mozgu. Zmiešaním troch základných farieb – **červenej**, **zelenej** a **modrej** vznikajú v mozgu všetky farebné odtiene spektra.

Existujú rôzne druhy a rôzny rozsah porúch farebného videnia. Rozlišuje sa:

- slabozrakosť až farbosleposť na červenú,
- slabozrakosť až farbosleposť na zelenú,
- slabozrakosť až farbosleposť na modrú,
- totálna farbosleposť.

V nasledujúcej tabuľke uvádzame ilustráciu, ako vidí obrázok osoba so zdravým zrakom a osoba s totálnou farbosleposťou.

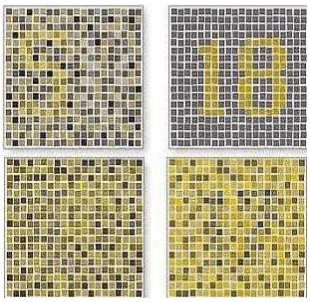
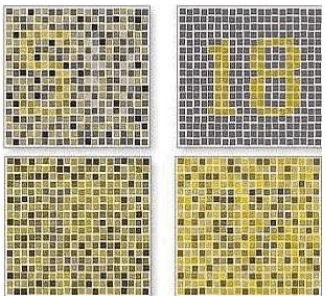
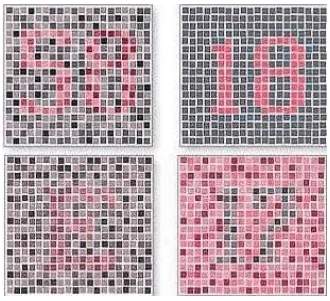
Tabuľka 15.2 Simulácie videnia pri totálnej farbosleposti [9].

 **Videnie so zdravým zrakom**	 **Videnie s totálnou farbosleposťou**

Všimnime si, že osoba s totálnou farbosleposťou dokáže rozpoznať len číslo 18 v pravej hornej časti obrázka. Číslo 58 v ľavej hornej časti, písmeno E v ľavej spodnej časti a číslo 17 v pravej dolnej časti osoba s totálnou farbosleposťou nevidí.

V nasledujúcej tabuľke je ilustrácia videnia osôb s farbosleposťou na zelenú, červenú a modrú farbu. Všimnime si, že osoby s farbosleposťou na zelenú a červenú farbu nedokážu rozpoznať niektoré informácie na obrázkoch.

Tabuľka 15.3 Simulácie videnia pri rôznych druhoch farbosleposti vytvorené pomocou programu Color Contrast Analyser.

 **Videnie pri farbosleposti na červenú - protanopia**	 **Videnie pri farbosleposti na zelenú - deuteranopia**	 **Videnie pri farbosleposti na modrú - tritanopia**

ÚLOHA 15.4

Použite test farebného videnia na webe.

a) Vyskúšajte test farebného videnia na stránke https://colormax.org/color-blind-test/.
b) Spustite si program Color Contrast Analyser (nájdete ho na stránke https://developer.paciellogroup.com/resources/contrastanalyser/). Postupne nasimulujte, ako by stránku videli osoby s farbosleposťou na červenú, zelenú farbu a osoby s totálnou farbosleposťou.
c) Nájdite obrázok so semaforom a pomocou programu Color Contrast Analyser nasimulujte, ako by ho videli osoby s farbosleposťou na červenú, zelenú farbu a osoby s totálnou farbosleposťou.

Osoby s pohybovým postihnutím

Určite všetci poznáte fyzika – kozmológa Stephena Hawkinga. Tento známy vedec bol dlhé roky pripútaný k invalidnému vozíku a so svetom komunikoval len pomocou počítača, ktorý ovládal pohybom očí.

Obrázok 15.1 Stephen Hawking [15].

ÚLOHA 15.5

Pozrite si video o zariadení Tobii na stránke https://www.specialnepomocky.sk/tobii-dynavox-zariadenia/tobii-pceye-mini-ocna-navigacia/.

ÚLOHA 15.6

Vyskúšajte ovládať počítač pomocou webovej kamery sledujúcej pohyb nejakého miesta na vašom tele – napríklad špičky nosa. Potrebujete k tomu program CameraMouse (http://www.cameramouse.org).

Spustite si aj program **Klávesnica** na obrazovke a textový editor. Napíšte niečo pomocou týchto nástrojov.

Ďalšou známou osobnosťou pripútanou na invalidný vozík bol známy herec Christopher Reeve – predstaviteľ Supermana. Christopher ostal nehybný po páde z koňa, ale aj potom žil aktívnym životom. Jeho veľkým pomocníkom bol počítač, ktorý ovládal rečou. V televíznom filme Raer Window stvárnil pohybovo postihnutého muža, ktorý sa z dlhej chvíle často pozeral z okna sledujúc dianie v susedných domoch. Jedného dňa si všimol podozrivé správanie muža v náprotivnom dome a vďaka počítaču sa mu podarilo odhaliť a usvedčiť zločin. Hlavný hrdina v tomto filme ovládal počítač ústami.

Obrázok 15.2 Christopher Reeve vo filme Raer Window [6].

ÚLOHA 15.7

Pozrite si video o zariadení IntegraMouse na stránke http://www.specialnepomocky.sk/integramouse-plus-mys/integramouse-plus-ustna-mys/.

Pohybových postihnutí je široké spektrum. Prácu s počítačom komplikujú predovšetkým nasledujúce postihnutia:

- príliš malé alebo veľké ruky pre klávesnicu,
- tras – problém zamerať objekt,
- artritída (zápal kĺbov) – obmedzená pohyblivosť ruky,
- ochrnutie – nepohyblivé ruky,
- strata končatiny – absencia ruky.

V súčasnosti je k dispozícii široká ponuka pomôcok pre osoby s pohybovým postihnutím. Tu je len malá ukážka niektorých z nich.

Tabuľka 15.4 Pomôcky pre osoby s pohybovým postihnutím [16].

 **Jednoručná klávesnica**	 **Zameriavač klávesnice pre osoby s neovládateľným trasom**	 **Trackball a trackpad pre osoby s poruchami jemnej motoriky**

Osoby so špecifickými poruchami učenia

Existuje mnoho druhov špecifických porúch učenia. K najznámejším patria poruchy pozornosti, poruchy pamäti a poruchy spracovania informácií v textovej podobe – dyslexia.

V európskych krajinách je okolo 4 - 8% žiakov ZŠ s dyslexiou.

Dyslexia je znížená schopnosť správne čítať – napriek primeranej inteligencii, normálnemu sociokultúrnemu zázemiu a bežným vyučovacím metódam. Osoby s dyslexiou majú problém so správnym dekódovaním písaného textu – prečítajú slovo inak, preskočia ho, čítajú veľmi pomaly, nie sú schopné pochopiť význam prečítaného.

ÚLOHA 15.8

Pozrite si nasledujúce simulácie vnímania textu osobami s dyslexiou:

- http://geon.github.io/programming/2016/03/03/dsxyliea
- http://www.edu.fmph.uniba.sk/~jaskova1/IKTH/tema05/tema05.html

ÚLOHA 15.9

Pozrite si simuláciu vnímania okolitého sveta osobami s autizmom. Na stránke http://www.sposa.sk/simulator-autizmu/ si zvoľte Simulátor autizmu – škola.

Metodika pre učiteľa

CIEĽ

Cieľom je, aby žiaci spoznali spôsob vnímania okolitého sveta osobami s rôznymi typmi zdravotného postihnutia a so špecifickými poruchami učenia. Zameriavame sa predovšetkým na prácu s informáciami pomocou počítača. Študenti majú získať vlastné skúsenosti pomocou zážitkového učenia.

MOTIVÁCIA

V úvode hodiny môže prebehnúť diskusia o známych osobnostiach s postihnutím. Ak je to možné, žiaci môžu ísť na exkurziu do centra pre podporu študentov so špecifickými potrebami na niektorej univerzite, ktoré je vybavené špeciálnymi pomôckami.

VÝKLAD

Návody k úlohám.

15.1 Pri riešení úlohy je potrebné poznať klávesové príkazy, ktoré sa dajú zistiť buď v pomocníkovi systému Windows alebo sa dajú vyhľadať na webe.

a. **Windows+M** minimalizuje všetky okná. Na zástupcu aplikácie, ktorú chceme spustiť, sa nastavíme stláčaním prvého písmena názvu zástupcu.
b. **Alt+Medzerník** zobrazí ponuku, v ktorej sa šípkou nastavíme na príkaz **Minimalizovať** a stlačíme **Enter**.
c. **Windows** vyvolá štartovaciu ponuku, v ktorej zvolíme aplikáciu šípkami a spustíme ju klávesom **Enter**.
d. **Alt+Tabulátor** – umožňuje prepínanie medzi aplikáciami.
e. **Alt+F4** – zavrie okno aktuálnej aplikácie.

15.2 Pri riešení úlohy je potrebné použiť klávesové príkazy programu NVDA, ktoré sa dajú zistiť v pomocníkovi tohto programu. Potrebné klávesové príkazy sú uvedené v zadaní úlohy.

15.4 V prípade, že sa nepodarí stiahnuť a nainštalovať program Color Contrast Analyser, dá sa použiť niektorý z online nástrojov: http://www.vischeck.com/ alebo https://www.color-blindness.com/coblis-color-blindness-simulator/.

Nadväznosť na predchádzajúce kapitoly: Pre túto kapitolu nie je nevyhnutné, aby študenti prebrali predchádzajúce kapitoly.

ZHRNUTIE

V závere hodiny môže prebehnúť krátka diskusia o tom, ktorá skupina používateľov je v bežných podmienkach najviac znevýhodnená.

16 PRÍSTUPNOSŤ INFORMÁCIÍ NA WEBE

V predchádzajúcej kapitole sme nahliadli do sveta osôb s rôznymi druhmi postihnutia. V tejto kapitole sa podrobnejšie zameriame na problémy týchto osôb s prístupnosťou informácií na webe.

Prístupnosť webového miesta

Prístupné webové miesto je použiteľné pre každého používateľa webu, bez ohľadu na jeho:

- zdravotný stav,
- znalosti,
- schopnosti,
- zobrazovacie možnosti.

ODPOVEDZTE

- Prečo majú byť webové stránky prístupné pre všetkých?
- Predstavte si webové stránky, ktoré často navštevujete. Koľko percent z nich nie je prístupných pre ľudí s rôznymi postihnutiami?

Prístupnosť pre nevidiacich

Nevidiace osoby nepoužívajú monitor, obsah obrazovky im sprostredkúva čítač obrazovky. Z toho pramenia nasledujúce problémy.

- Prístupné sú iba informácie v textovej podobe. Obrázky, video, animácie sú neprístupné. Treba k nim poskytnúť aj textovú alternatívu.
- Chýba náhľad na celú obrazovku, takže ak prvky v usporiadaní riadkov zhora nadol a v rámci riadkov zľava doprava nedávajú zmysel, informácie sú nezrozumiteľné.
- Odlišný význam prvku (napr. nadpis, položka zoznamu, tabuľka) je potrebné vyznačiť vizuálne aj sémanticky, príslušnou HTML značkou.
- Informácie sprostredkované iba pomocou farieb sú neprístupné, treba ich vyjadriť aj inak ako farbou (napríklad povinné polia formulára nestačí odlíšiť farebne, ale treba ich vyznačiť hviezdičkou).
- Zdanlivo nezmyselný text (napr. dlhé webové adresy, dlhé cudzie slová) je ťažko zrozumiteľný a použiteľný, treba ho nahradiť zrozumiteľným ekvivalentom (napríklad pomocou atribútu `title`).
- Pri čítaní obsahu bunky nejakej tabuľky je ťažké pochopiť jeho význam. V tabuľkách je potrebné vyznačiť väzbu medzi dátovou bunkou a bunkou so záhlavím riadku alebo stĺpca. Informácie v tabuľkách musia dávať zmysel pri čítaní riadkov zhora nadol a v rámci riadkov zľava doprava.
- Pri práci s formulárovými poľami bez priradeného popisu je ťažké pochopiť, aké vstupné údaje sa očakávajú. Každé formulárové pole musí mať priradený popis pomocou značky `label`.

Nevidiace osoby nepoužívajú myš, vstup zadávajú len pomocou klávesnice. Pri ovládaní stránky musí byť splnená nasledujúca podmienka:

- Všetko, čo sa dá vykonávať pomocou myši, musí sa dať vykonať aj pomocou klávesnice.

ÚLOHA 16.1

Spustite čítač obrazovky NVDA.

a) Zobrazte si stránku www.gymza.sk. Dokážu si nevidiaci prečítať text na titulnom obrázku (Gymnázium, Hlinská 29, Žilina) na vrchu stránky? (Po obrázkoch sa dá pohybovať pomocou klávesu G a Shift+G.)
b) Zobrazte si stránku www.bratislava.sk. Dokážu si nevidiaci prečítať text BRATISLAVA na titulnom logu na vrchu stránky?
c) Porovnajte zdrojové kódy obidvoch stránok. V čom sa líšia značky na vloženie obrázkov?

ÚLOHA 16.2

Spustite čítač obrazovky NVDA.

a) Pozrite si stránku https://www.nissan.sk/. Pomocou tabulátora sa nevidiaci môžu pohybovať po odkazoch na stránke, alebo si pomocou klávesovej skratky Insert+F7 môžu zobraziť zoznam odkazov. Dokážu nevidiaci len z textov odkazov **VIAC INFORMÁCIÍ** pochopiť, kam smerujú?
b) Pozrite si stránku http://www.vpslm.sk/. Pomocou tabulátora sa nevidiaci môžu pohybovať po odkazoch na stránke, prípadne pomocou Insert+F7 si môžu zobraziť ich zoznam. Dokážu nevidiaci len z textov odkazov **viac informácií čítajte tu** pochopiť, kam smerujú?
c) Porovnajte zdrojové kódy obidvoch stránok. V čom sa líšia značky na vloženie odkazov?

ÚLOHA 16.3

Spustite čítač obrazovky NVDA .

a) Pozrite si stránku https://gvpt.sk s nadpisovými štýlmi. Dá sa pomocou klávesu H a Shift+H pohybovať po nadpisoch?
b) Pozrite si stránku http://www.gymgolnr.sk. Texty OZNAMY a AKTIVITY sú nadpismi jednotlivých častí. Dá sa po nich pohybovať pomocou klávesu H a Shift+H?
c) Porovnajte zdrojové kódy obidvoch stránok. V čom sa líšia použité značky pre vloženie nadpisov?

ÚLOHA 16.4

Spustite čítač obrazovky NVDA.

a) Zobrazte si stránku http://www.skolafelix.sk/register/?what=form. Pomocou F a Shift+F sa môžete pohybovať po formulárových poliach. Hovorí čítač, čo treba vyplniť do jednotlivých formulárových polí?
b) Zobrazte si stránku https://nadaciatatrabanky.egrant.sk/register. Pomocou F a Shift+F sa môžete pohybovať po formulárových poliach. Hovorí čítač, čo treba vyplniť do jednotlivých formulárových polí?

c) Porovnajte zdrojové kódy obidvoch stránok. V čom sa líšia značky pre formulárové polia?

ÚLOHA 16.5

Stretli ste sa s ďalšími stránkami, ktoré sú problematické pre nevidiacich? Uveďte príklady.

Prístupnosť pre slabozrakých

Niektoré stránky s málo kontrastnými farbami textu a pozadia môžu byť ťažko čitateľné pre osoby so **slabozrakosťou**.

Napríklad modrá farba pre text odkazov na čiernom pozadí, alebo červený text na zelenom pozadí alebo iné kombinácie farieb, ktoré sú ťažko čitateľné pre kohokoľvek, ale najväčšie problémy majú s nimi slabozrakí čitatelia.

Niektorí slabozrakí používatelia používajú vlastné nastavenia (farieb, veľkosti textu) v operačnom systéme a/alebo vo webovom prehliadači, aby dosiahli optimálny kontrast a veľkosť objektov. Obľúbené sú nastavenia bieleho textu na čiernom pozadí alebo žltého textu na čiernom pozadí, ale môžu byť aj iné.

Obrázok 16.1 Žltý text na čiernom pozadí v porovnaní s čiernym textom na bielom pozadí.

Ak je potrebné obsah okna horizontálne rolovať aj napriek tomu, že je maximalizovaná jeho veľkosť na celú obrazovku, je to nepohodlné aj pre ľudí s perfektným zrakom. Ešte väčšie problémy to však spôsobí používateľom zväčšovacích programov, pretože tí musia rolovať o to viac vo vnútri malého zväčšeného priestoru okna.

Pre ťažko slabozrakých používateľov **môže byť použitie vizuálnych efektov** (napr. blikanie textu, automatické zmeny zobrazenia textu, animácie textu atď.) veľmi nevhodné a podstatne sťažujúce alebo znemožňujúce čítanie danej informácie.

ÚLOHA 16.6

Zobrazte si stránku `index.html` v priečinku `Uloha16_06`.

- Je hlavná navigácia dostupná, aj keď si v prehliadači nastavíte zväčšenie 240%?
- Je formulár dostupný, aj keď si v prehliadači nastavíte zväčšenie 240%?

16/Uloha16_06

- Pozrite si stránku http://vin.edu.fmph.uniba.sk/ pri nastavenom zväčšení 240%. Sú dostupné všetky jej časti?

ÚLOHA 16.7

Zobrazte stránku http://ilumina.sk/. Čo si myslíte o jej prístupnosti pre osoby so slabozrakosťou?

ÚLOHA 16.8

Stretli ste sa s ďalšími stránkami, ktoré sú problematické pre ľudí so slabozrakosťou? Uveďte príklady.

Prístupnosť pre osoby s poruchami farebného videnia

Väčšina porúch farebného vnímania sa prejavuje v oblasti červenej a zelenej farby. Ťažké môže byť hlavne rozlišovanie červenej a zelenej farby. Čisto zelenú vnímajú tieto osoby ako sivastú. Červenú a zelenú môžu vidieť ako žltú alebo pomarančovú. Pri veľmi zriedkavej forme úplnej farbosleposti nemožno vnímať žiadne farby, ale len rozdiely v jase.

Pri vytváraní stránok pre ľudí s poruchami farebného videnia netreba konvertovať všetky obrázky na čiernobiele alebo ich úplne odstraňovať zo stránky. Obrázky netreba modifikovať vôbec, stačí k nim poskytnúť alternatívu v podobe textu.

16/Uloha16_09

ÚLOHA 16.9

Spustite si program Color Contrast Analyser. Postupne nasimulujte, ako by stránku `index.html` v priečinku `Uloha16_09` videli osoby so slabozrakosťou na červenú, zelenú, modrú farbu a osoby s totálnou farbosleposťou. Ktoré objekty by boli pre jednotlivé skupiny používateľov ťažko čitateľné?

ÚLOHA 16.10

Pozrite si stránku http://cvicebnice.ujep.cz/cvicebnice/FRVS1973F5d/data/Snimek25_text.html. Dokáže osoba s totálnou farbosleposťou rozlíšiť, ktoré odpovede sú správne a ktoré nie?

ÚLOHA 16.11

Pozrite si stránku https://www.cspsd.cz/storage/files/TEST_1_odpovedi.pdf. Dokáže osoba s totálnou farbosleposťou rozlíšiť, ktoré odpovede sú správne a ktoré nie?

ÚLOHA 16.12

Stretli ste sa s ďalšími stránkami, ktoré sú problematické pre ľudí s poruchami farebného videnia? Uveďte príklady.

Prístupnosť pre osoby s pohybovým postihnutím

Pohybovo postihnutí používatelia môžu mať problém do dostatočnej miery ovládať klávesnicu a myš.Pohybovo postihnutí používatelia môžu, podobne ako zrakovo postihnutí používatelia, používať softvér aktivujúci zvuk (reč).

Používanie podporných technológií (napr. hlavou ovládaná palička, ústami ovládaná palička a pod.) **je fyzicky náročné a postihnuté osoby sa môžu ľahko unaviť.**

ÚLOHA 16.13
Zobrazte si stránku http://www.liptov.zasahy.sk/prihlaska/podnet. Je stránka plne ovládateľná aj pre osoby, ktoré nemôžu používať myš?
ÚLOHA 16.14
Zobrazte si stránku https://www.topky.sk/. Je stránka pohodlná aj pre osoby, ktoré nemôžu používať myš, ale po odkazoch sa môžu presúvať len pomocou tabulátora?
ÚLOHA 16.15
Stretli ste sa s ďalšími stránkami, ktoré sú problematické pre ľudí s pohybovým postihnutím? Uveďte príklady.

Prístupnosť pre osoby so špecifickými poruchami učenia

Osoby s poruchami učenia môžu mať problémy:

- so správnym spracovávaním informácií,
- s vnímaním textu a súvislosti medzi jednotlivými informáciami,
- s orientáciou v texte,
- s udržaním pozornosti.

Pre osoby s poruchou pozornosti môžu rušivo pôsobiť pohybujúce sa objekty na stránke. Osoby s poruchami pamäti sa ťažko orientujú v navigácii. Môžu tiež ľahko stratiť orientáciu na stránke s veľkým množstvom informácií.

ÚLOHA 16.16
Pozrite si webovú stránku http://www.zsturzovka.edu.sk/historia.htm. Je vhodná pre osoby s dyslexiou? Ako by ju bolo potrebné zmeniť, aby bola vhodná pre osoby s dyslexiou?
ÚLOHA 16.17
Pozrite si stránku https://jayj.dk/grim/. Posúďte jej prístupnosť pre osoby s autizmom.

ÚLOHA 16.18
Je navigácia na stránke https://zsbeladulice.edupage.org/ prehľadná a ľahko zapamätateľná aj pre osoby s poruchami pamäti? Ako by ste ju navrhovali zmeniť?

ÚLOHA 16.19
Stretli ste sa s ďalšími stránkami, ktoré sú problematické pre ľudí s poruchami učenia? Uveďte príklady.

Legislatívne riešenia prístupnosti webu

Prístupnosť webových stránok je v mnohých štátoch riešená aj legislatívne. Vznikajú **špecifické právne normy** upravujúce mieru prístupnosti webových stránok a jasne deklarujúce podmienky ich bezbariérovosti. Spomenieme teraz najznámejšie z nich.

Web Content Accessibility Guidelines

Jedná sa o pravidlá pre tvorbu bezbariérového webu, ktoré v máji 1999 zostavila skupina Web Accessibility Initiative (WAI) vytvorená v rámci konzorcia W3C. Nie sú to zákony, ale pravidlá, ktoré majú funkciu odporúčaní. Na tieto pravidlá nadviazali všetky ostatné metodiky, ktoré vznikli po roku 1999.

Žiadna krajina však neaplikovala do svojich noriem, upravujúcich prístupný web, WCAG 1.0 ako celok, vždy boli vybrané iba niektoré body a niektoré zase doplnené. V súčasnosti je aktuálna verzia, ktorá je známa ako Web Content Accessibility Guidelines 2.1. Podrobnejšie informácie sú na https://www.w3.org/TR/WCAG21/.

Section 508

Section 508 je časťou číslo 508 amerického zákona Rehabilitation Act. Táto časť bola do zákona pridaná v roku 1998 a stanovuje federálnym orgánom USA povinnosť poskytovať informácie prístupným spôsobom. Konkrétne pravidlá, ktoré musia federálne orgány dodržiavať, pripravil v decembri roku 2000 americký úrad US Access Board pod názvom **Electronic and Information Technology Accessibility Standard**. Vzhľadom na zameranie zákona sú tieto pravidlá poňaté dosť široko a týkajú sa napr. aj hardvéru, telekomunikačných zariadení, aplikačného softvéru a pod. Časť, ktorá sa venuje internetovým a intranetovým aplikáciám, obsahuje 16 bodov, ktoré sa značne prekrývajú s bodmi pravidiel WCAG 1.0 (v 11 bodoch sa zhodujú). Podrobnejšie informácie sú na https://www.section508.gov/.

Blind Friendly Web

Zásady prístupnosti webových stránok pre ťažko zrakovo postihnutých používateľov vznikli v roku 2000 v rámci projektu Sjednocené organizace nevidomých a slabozrakých. V mnohom rešpektujú WCAG 1.0. Projekt Blind Friendly Web má vďaka Občianskym iniciatívam eSlovensko a Únii nevidiacich a slabozrakých Slovenska pôsobnosť aj v Slovenskej republike. Podrobnejšie informácie sú na http://blindfriendly.cz/ a http://blindfriendly.sk/.

Zákon o informačných systémoch verejnej správy

Na Slovensku platí **Zákon č. 275/2006 Z.z. o informačných systémoch verejnej správy,** ktorý stanovuje povinnosť poskytovania informácií prístupnou formou. Vzťahuje sa na subjekty verejnej správy a samosprávy v SR. Miera prístupnosti je špecifikovaná **Výnosom o štandardoch pre informačné systémy verejnej správy** (nadobudol účinnosť od 1. augusta 2006). Pracovná komisia sa nechala inšpirovať najmä pravidlami s najvyššou prioritou z metodiky WCAG 1.0 a metodikou Blind Friendly Web. Posledné zmeny boli vykonané v súlade s metodikou WCAG 2.0 vo **Výnose z 13. marca 2014 o štandardoch pre informačné systémy verejnej správy**. Podrobnejšie informácie sú na http://www.informatizacia.sk/ext_dok-harmonizacia_wcag_2-0/6611c.

POZNÁMKA

Webové sídla, ktorých tvorcovia dodržiavajú pravidlá konkrétnej metodiky, obyčajne obsahujú prehlásenie o prístupnosti s konkretizáciou aplikovaných pravidiel a logom aplikovanej metodiky.

Tabuľka 16.1 Logá niektorých metodík.

Logo konzorcia W3C	**Section 508 - logo**	**Blind Friendly – logo**

Metodika pre učiteľa

CIEĽ

Cieľom, je aby sa študenti naučili vnímať webové stránky z pozície osôb s rôznymi typmi postihnutia.

MOTIVÁCIA

Hodina môže začať diskusiou o stránkach, ktoré študenti často navštevujú. Na predchádzajúcej hodine študenti spoznali problémy osôb s rôznymi typmi postihnutia. Ktoré stránky by boli pre nich problematické?

VÝKLAD

Výklad obsahuje pre každý typ postihnutia popis problematických prvkov a vlastností webových stránok. Potom nasledujú úlohy ilustrujúce príklady zlej a dobrej praxe.

V závere sú uvedené najznámejšie normy riešiace prístupnosť informácií na webe, ale aj vo všeobecnosti. Študenti by mali vedieť o ich existencii, ale podrobne sa s nimi na hodine nie je potrebné zaoberať.

Návody k úlohám.

16.1 Študenti majú prísť na to, že ak obrázok obsahuje text, treba tento text uviesť v atribúte **alt**, aby ho povedal čítač obrazovky.

16.2 Študenti majú prísť na to, že ak sa na jednej stránke nachádza viac odkazov s rovnakým textom, treba v atribúte **title** uviesť, kam odkaz vedie.

16.3 Študenti majú prísť na to, že čítač vie rozoznať nadpisy, iba ak sa v HTML kóde použijú správne značky (<h1></h1>, <h2></h2>, …). Nestačí nastaviť veľkosť písma.

16.4 Študenti majú prísť na to, že čítač vie povedať správny popis k formulárovému poľu iba vtedy, ak je k nemu správne priradený pomocou značky **label** (s atribútom **for**).

16.6 Študenti majú prísť na to, že je možné vytvoriť stránku, na ktorej budú všetky objekty viditeľné aj pri nastavení ľubovoľného zväčšenia. Je potrebné správne vyriešiť rozmiestňovanie objektov v CSS a tiež vytvoriť rôzne štýly pre rôznu veľkosť okna prehliadača. Pri zväčšení obsahu okna prehliadača sa zvolia štýly priradené k menšej šírke okna.

16.7 Študenti majú prísť na to, že osoby so slabozrakosťou môžu mať so stránkou problémy v nasledujúcich prípadoch – na stránke je pohyb objektov, nedostatočný kontrast textu a pozadia, ...

16.9 Študenti majú prísť na to, že najkritickejšia je kombinácia zeleného textu na červenom pozadí, ktorá sa používa v navigácii a v pätičke.

16.10 Študenti majú prísť na to, že ak sú správne odpovede písané zelenou farbou a nesprávne červenou, osoba s totálnou farbosleposťou ich nebude vedieť rozlíšiť.

16.11 V tomto prípade sú správne odpovede podčiarknuté a nesprávne nie sú. Takže osoba s totálnou farbosleposťou ich bude vedieť rozlíšiť.

16.13 Bez myši (len pomocou klávesnice) sa nedá označiť miesto na mape, ku ktorému sa viaže podnet.

16.14 Na stránke je príliš veľa odkazov a pri stláčaní tabulátora sa ťažko sleduje, ktorý odkaz je zameraný.

16.16 Text na vzorovanom pozadí je veľmi ťažko čitateľný aj pre bežného človeka.

16.17 Na stránke je príliš veľa rušivých prvkov – pohyblivé objekty, vzorované pozadie, krikľavé farby.

16.18 Navigácia obsahuje veľa položiek a nachádza sa hore a aj dvakrát vľavo. Keď sa zvolí položka **O škole**, objaví sa 17 ďalších podpoložiek.

Nadväznosť na predchádzajúce kapitoly: Táto kapitola nadväzuje na kapitoly 1-12 a kapitolu 15.

ZHRNUTIE

V závere hodiny môže prebehnúť krátka diskusia o tom, ktorá skupina používateľov to má najťažšie z hľadiska prístupu k informáciám na webe.

17 TVORBA PRÍSTUPNÝCH WEBOVÝCH STRÁNOK

V predchádzajúcej kapitole sme si ukázali problematické prvky a vlastnosti webových stránok. V tejto kapitole si povieme, ako na webových stránkach zabezpečíme prístupnosť pre jednotlivé skupiny používateľov.

Zabezpečenie prístupnosti pre nevidiacich

Grafika

Ku grafike je potrebné poskytnúť textové alternatívy informácií v atribúte `alt` alebo `title`, alebo dlhšie textové vysvetlenia buď ako samostatné textové prvky na tej istej, prípadne na samostatnej stránke.

PRÍKLAD 17.1

Nasledujúci obrázok by sme na webovú stránku vložili pomocou nasledujúcej značky.

```
<img src="Koloseum.JPG" title="Koloseum v Pule" alt="Pula - koloseum">
```

Nadpisy

Texty tvoriace nadpisy a zoznamy majú byť korektne vyznačené v zdrojovom kóde pomocou príslušnej HTML značky. Prvky, ktoré netvoria nadpisy alebo zoznamy, nemajú byť takto vyznačené.

PRÍKLAD 17.2

Nasledujúci text by sme na webovú stránku vložili pomocou nasledujúceho kódu.

Zrakové postihnutie

- Úvod
- Kto je človek so zrakovým postihnutím?
- Zrakové poruchy

Úvod

Život so zrakovým postihnutím je spojený s mnohými ťažkosťami. Najvypuklejšími sú pre nevidiacich a slabozrakých ľudí problémy so samostatným pohybom a orientáciou, so získavaním informácií a s vykonávaním činností v domácnosti , akými sú varenie, upratovanie či nakupovanie.

```
<h1>Zrakové postihnutie</h1>

<ul>
    <li><a href="#uvod">Úvod</a></li>
    <li><a href="#clovek-zp">kto je človek so zrakovým
postihnutím?</a></li>
    <li><a href="#zrakove-poruchy">Zrakové poruchy</a></li>
</ul>

<h2><a name="uvod">Úvod</a></h2>
<p>
Život so zrakovým postihnutím je spojený s mnohými ťažkosťami.
Najvypuklejšími sú pre nevidiacich a slabozrakých ľudí problémy so
samostatným pohybom a orientáciou, so získavaním informácií a s
vykonávaním činností v domácnosti , akými sú varenie, upratovanie
či nakupovanie.
</p>
```

Zoznamy

Používateľom, ktorí nemôžu používať myš, treba umožniť preskočenie navigačných ponúk, dlhých zoznamov položiek a iných objektov, ktoré by mohli byť pre čítač obrazovky náročné alebo zbytočné.

PRÍKLAD 17.3

Odkaz na hlavný obsah by mal byť na začiatku stránky, buď hneď za značkou `<body>`, alebo hneď za značkou `<nav>`.

```
<div id="skocit"><a href="#content">skočiť na obsah</a></div>
<nav>
...
</nav>
<section>
<h1><a name="content"></a> hlavný nadpis</h1>
<p>toto je prvý odsek.</p>
```

Keďže odkaz **Skočiť na obsah** je pre vidiacich používateľov zbytočný, zvykne sa skrývať a zobrazí sa, až keď sa naň používateľ nastaví tabulátorom. Je preto potrebné do prilinkovaného CSS súboru pridať nasledujúci kód, ktorý pridá štýl pre prvok `#skocit a` (aby bol skrytý).

```
#skocit a {
  position:absolute;
  left:-10000px;
  top:auto;
  width:1px;
  height:1px;
  overflow:hidden;
}
```

Pre prvok `#skocit a:focus` sa nastavia nasledujúce vlastnosti, aby bol viditeľný, iba keď sa naň používateľ nastaví tabulátorom.

```
#skocit a:focus{
position:static;
width:auto;
height:auto;
}
```

Odkazy

Text odkazov musí dávať zmysel aj mimo kontextu (napr. odkaz "kliknite tu" je problematický). Pomocou atribútu `title` treba upresniť cieľ odkazu.

PRÍKLAD 17.4

```
Informácie o úradných hodinách nájdete
<a href="uradneHodiny.html" title="Úradné hodiny">tu.</a>
```

Používatelia očakávajú, že po aktivácii odkazu ostanú v tom istom okne. V odkazoch by sa nemali používať skripty, ktoré zobrazia stránku v novom okne, pokiaľ to nie je nevyhnutné. Ak sa takéto skripty na stránke použijú, je potrebné o tom používateľa vopred informovať.

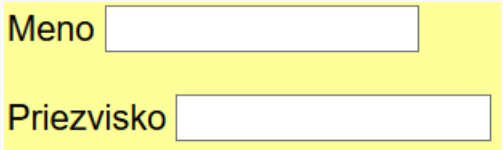
Obsah webovej stránky sa mení iba keď používateľ aktivuje nejaký prvok. Automatická zmena obsahu stránky môže byť pre zrakovo postihnutých používateľov veľkou prekážkou v prístupnosti.

Formuláre

Každému formulárovému prvku musí byť priradený výstižný popis alebo názov pomocou značky `label` alebo pomocou atribútu `title`.

PRÍKLAD 17.5

Nasledujúce dve textové polia slúžia na zadanie mena a priezviska. Prvé textové pole má popis **Meno** a druhé textové pole má popis **Priezvisko**.



Správne vytvorený zdrojový kód používajúci značku `<label>` vyzerá nasledovne.

```
<label for="meno">Meno</label>
<input type="text" name="meno1" id="meno">

<label for="priezvisko">Priezvisko</label>
<input type="text" name="meno2" id="priezvisko">
```

Uvedený formulár sa dá reprezentovať aj nasledujúcim zdrojovým kódom.

```
Meno <input type="text" name="meno1" id="meno" title="Meno">
Priezvisko <input type="text" name="meno2" id="priezvisko"
title="Priezvisko">
```

Formulár vyzerá presne tak ako predchádzajúci, akurát sa po nájazde myši nad formulárový prvok zobrazí nad prvkom informácia, ktorá je v atribúte `title`.

Tabuľky

Tabuľku treba opatriť stĺpcovými a riadkovými nadpismi pomocou značiek `<th>` a `</th>`.

PRÍKLAD 17.6

Nasledujúca tabuľka má len stĺpcové nadpisy v prvom riadku.

Typ vstupenky	Cena
Deti	0,- €
Študenti	5,- €
Dospelí	12,- €

Na bunky prvého riadku treba použiť značky `<th>` a `</th>`. Zdrojový kód bude vyzerať nasledovne.

```
<table>
   <tr>
     <th>Typ vstupenky</th>
     <th>Cena</th>
   </tr>
   <tr>
     <td>Deti</td>
     <td>0,- &euro;</td>
   </tr>
   <tr>
     <td>Študenti</td>
     <td>5,- &euro;</td>
   </tr>
   <tr>
     <td>Dospelí</td>
     <td>12,- &euro;</td>
   </tr>
</table>
```

Informácie v bunkách tabuľky (hlavne ak sú zlúčené) musia dávať zmysel pri čítaní riadkov zhora dole a pri čítaní riadkov zľava doprava.

Nasledujúca tabuľka v grafickom prehliadači dáva zmysel, ale v lineárnej interpretácii čítača obrazovky nedáva zmysel.

Meno	Výška	Váha	Vek
Ján Bobor	134	32	6
Júlia Čierna	125	23	5
Milan Zachar	132	30	6

Čítač obrazovky bude informácie v tabuľke čítať v nasledujúcom poradí.

Meno Výška Váha Vek
Ján Bobor Júlia Čierna Milan Zachar
134 125 132

32 23 30
6 5 6

Aby mala tabuľka správnu linearitu, musí vyzerať nasledovne.

Meno	Výška	Váha	Vek
Ján Bobor	134	32	6
Júlia Čierna	125	23	5
Milan Zachar	132	30	6

Čítač obrazovky bude informácie v tabuľke čítať v nasledujúcom poradí.

Meno Výška Váha Vek
Ján Bobor 134 32 6
Júlia Čierna 125 23 5
Milan Zachar 132 30 6

Farby

Informácia sprostredkovaná farbou musí byť sprostredkovaná aj iným spôsobom, napríklad označená hviezdičkou, prípadne v zátvorke je uvedený význam farby.

Ovládanie pomocou klávesnice

Nepoužívať skripty, ktoré si vyžadujú použitie myši, prípadne treba poskytnúť aj alternatívne riešenie pomocou klávesnice. O skriptoch sa naučíte v rámci predmetu Programovanie webových stránok.

ÚLOHA 17.7
Upravte stránku zoologickej záhrady v priečinku `Uloha17`, aby bola prístupná pre nevidiacich.

17/Uloha17

Zabezpečenie prístupnosti pre slabozrakých

Farby

- Dôležité je zabezpečiť čo najväčší kontrast pozadia a objektov v popredí (text, grafika, formuláre a podobne).
- Dôležité je, aby si používatelia mohli prispôsobiť kontrast textu a pozadia podľa vlastných potrieb. Z toho dôvodu je potrebné používať radšej textový ako grafický formát na zobrazenie textu (nepoužívať text v obrázkoch).

Pohyblivé objekty

Na webovej stránke by nič nemalo kmitať rýchlejšie ako raz za sekundu.

Rozmiestňovanie objektov

- Dôležité je, aby bolo všetko konfigurovateľné. Takže text musí byť textom, aby si ho používateľ mohol zväčšiť, zmeniť jeho farbu a farbu pozadia.
- Ak je veľkosť objektov zadaná v percentách, obrazovka sa môže rozširovať a zužovať podľa potrieb používateľa.

ÚLOHA 17.8

Upravte stránku zoologickej záhrady v priečinku `Uloha17`, aby bola prístupná pre slabozrakých.

Zabezpečenie prístupnosti pre osoby s poruchami farebného videnia

Farby

- Dôležité je, aby sa farby nepoužívali ako jediný spôsob sprostredkovania dôležitej informácie. Teda treba poskytnúť aj iný spôsob (najlepšie pomocou textu).
- Dôležité je zabezpečiť čo najväčší kontrast pozadia a objektov v popredí (text, grafika, formuláre a podobne).
- Dôležité je vyhýbať sa kombinácii zelenej a červenej farby.

ÚLOHA 17.9

Upravte stránku zoologickej záhrady v priečinku `Uloha17`, aby bola prístupná pre osoby s poruchami farebného videnia.

Zabezpečenie prístupnosti pre sluchovo postihnutých

Zvukové signály

Dôležité zvukové signály (napr. výstražné pípnutie) sa musia doplniť vizuálnymi signálmi (napr. bliknutie).

Titulky

Dôležité je otitulkovanie – vizuálny textový pohľad na audio, ktoré je vo zvukovom súbore alebo video klipe.

Formulácia textu

- Používať stručný, výstižný a jednoduchý jazyk.
- Nepoužívať novotvary, zloženiny, odborné výrazy a cudzie slová.
- Použiť doplnkový vizuálny obsah – ilustračné obrázky, vizuálne odrážky, animácie, fotografie.
- Písať štruktúrovane.

ÚLOHA 17.10
Upravte stránku zoologickej záhrady v priečinku `Uloha17`, aby bola prístupná pre nepočujúcich.

17/Uloha17

Zabezpečenie prístupnosti pre pohybovo postihnutých

Plná funkčnosť pomocou klávesnice

Treba zabezpečiť plnú funkčnosť pomocou klávesnice (musí sa dať presúvať medzi odkazmi pomocou tabulátora, tlačidlá a formulárové polia musia byť ovládateľné a funkčné pomocou klávesnice, atď.).

Odolnosť voči chybám používateľov

Stránky by mali byť odolné voči chybám používateľov (napr. treba zaradiť kontrolné otázky typu "Ste si istý, že chcete vymazať tento súbor?".

Odkazy

Treba sa vyhnúť vytváraniu odkazov z malých alebo pohybujúcich sa objektov.

Radenie informácií

- Informácie na stránke treba usporiadať podľa dôležitosti (najdôležitejšie na začiatku).
- Treba poskytnúť možnosť preskočiť dlhé zoznamy odkazov a dlhý súvislý text.

ÚLOHA 17.11
Upravte stránku zoologickej záhrady v priečinku `Uloha17`, aby bola prístupná pre osoby s pohybovým postihnutím.

17/Uloha17

Zabezpečenie prístupnosti pre osoby so špecifickými poruchami učenia

Mohlo by sa zdať, že vytvoriť prístupnú stránku pre človeka s ťažkým kognitívnym postihnutím nie je možné. Istá skupina používateľov bude mať vždy problém pochopiť nejaký obsah. Tomu sa určite nedá vyhnúť. Napriek tomu, tvorcovia stránok môžu uľahčiť prístup k informáciám aspoň ľuďom, ktorí nemajú až také ťažké postihnutie. Niektoré princípy uvádzame nižšie.

Navigačné mechanizmy

- Dôležité je poskytnúť jasnú a prehľadnú navigáciu, používať v nej jednoduché a jednoznačné slová.
- Navigácia by mala byť vizuálne vyčlenená z bežného obsahu a mala by byť veľmi výrazná.
- Vhodné je poskytnúť mapu webového sídla a plnotextové vyhľadávanie.

Zvýraznenie štruktúry

Musí byť jasné, čo je text, nadpis, navigácia, hlavička, pätička, atď. Štruktúru je potrebné vyjadriť:

- **vizuálne** – použiť v CSS rôzne veľkosti písma, rôzne fonty, odsadenie obsahových blokov, odrážky, obrázky a podobne,
- **sémanticky** – použiť správne HTML značky.

Doplnkový vizuálny obsah

Vhodné je použiť:

- ilustračné obrázky (jednak upútajú pozornosť, jednak lepšie znázornia význam obsahu ako text),
- vizuálne odrážky,
- fotografie,
- animácie a podobne.

Práca s textom

- Používať jasný, jednoduchý, zrozumiteľný jazyk.
- Používať krátke slová, krátke vety, krátke odseky.
- Písať štruktúrovane - používať podnadpisy, zoznamy a krátke odseky.
- Základná veľkosť písma by mala byť väčšia, používať voľný priestor okolo textu, aby sa nikde nedotýkal iných obsahových blokov.

ÚLOHA 17.12

Upravte stránku zoologickej záhrady v priečinku `Uloha17`, aby bola prístupná pre osoby s poruchami učenia.

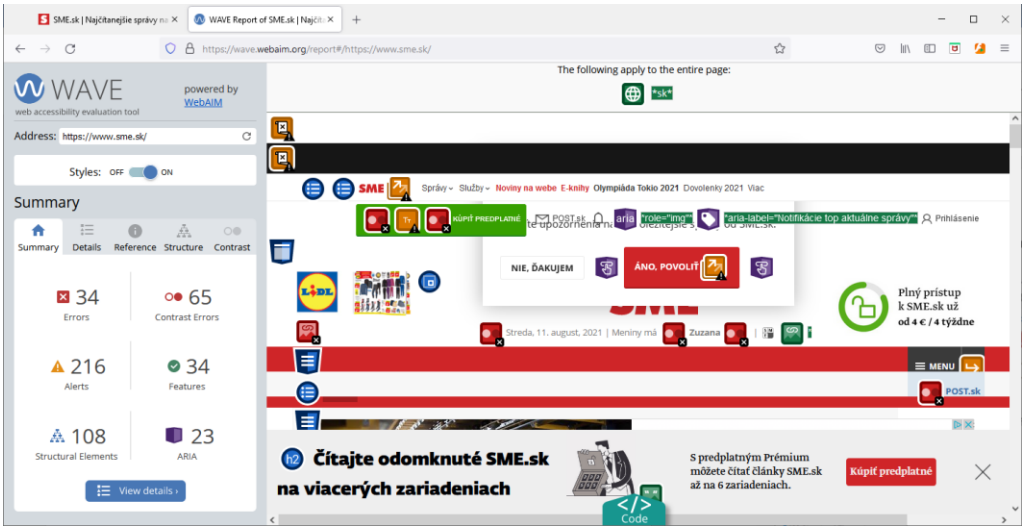
Testovanie prístupnosti

Prístupnosť informácií na webovej stránke môžeme otestovať aj pomocou automatického nástroja dostupného online. Takýchto nástrojov existuje viac. Pravdepodobne najpríjemnejší z hľadiska použitia je Wave Web Accessibility Tool (http://wave.webaim.org/). Výstupná informácia má prehľadný obrázkový charakter. Pri použití tohto nástroja si treba uvedomiť, že ide o automatický nástroj a všetky jeho výstupy treba prehodnotiť.

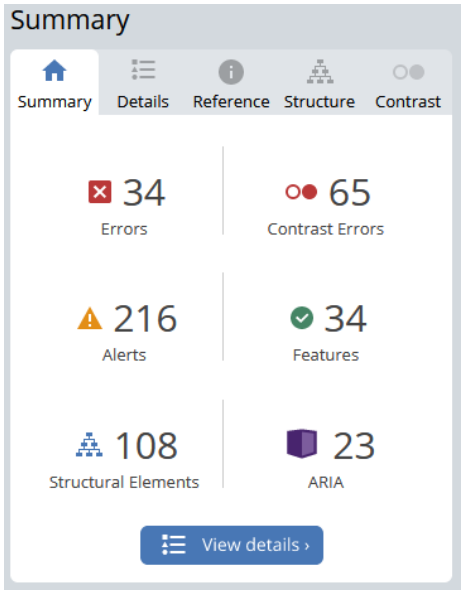
Postup pri testovaní stránky

1. Do políčka s textom **Web page address:** zadáme, prípadne skopírujeme adresu stránky, ktorú chceme testovať a stlačíme **Enter**.

2. Po chvíli sa v pravej časti okna prehliadača zobrazí testovaná webová stránka aj s vyznačenými komentármi týkajúcimi sa jej prístupnosti. V ľavej časti stránky je zobrazený sumárny zoznam komentárov.



3. Sumárny zoznam komentárov v ľavej časti obsahuje informácie o počte komentárov.

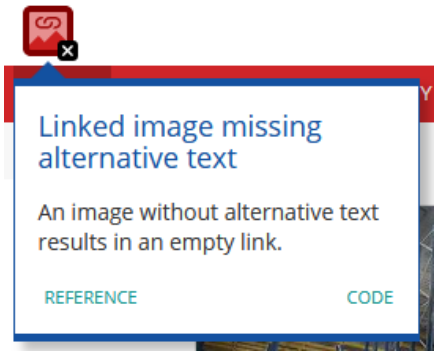


Jednotlivé typy komentárov sú odlíšené tvarom a farbou.

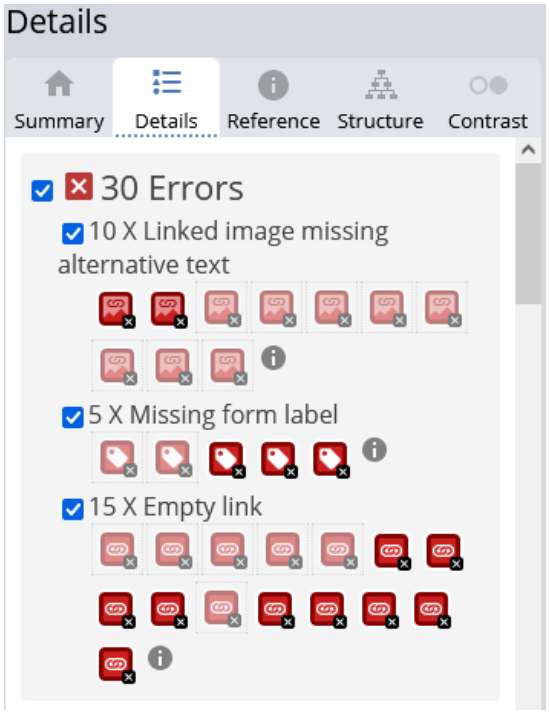
- Červený štvorec – chyby (Errors) – upozornenia na vlastnosti, ktoré sú porušením pravidiel prístupnosti (najčastejšie to býva chýbajúci alternatívny text k obrázku alebo chýbajúci popis ku formulárovému poľu, chýbajúci text k odkazu a podobne).
- Žltý trojuholník – výstrahy (Allerts) – upozornenia na vlastnosti, ktoré by sa dali zlepšiť (napríklad chýbajúci nadpis prvej úrovne na začiatku stránky, príliš dlhý alternatívny text, veľmi malý text a podobne).
- Zelený krúžok – vlastnosti riešiace prístupnosť (Features) –prvky, ktoré sa týkajú prístupnosti (doplnený alternatívny text, popisy k formulárovým poliam a podobne).
- Modrá ikona grafu – štrukturálne prvky (Structural Elements) – značky, ktoré sa používajú na štrukturovanie textu (nadpisy, zoznamy, tabuľky, zoznamy a podobne).

- Fialová kocka – ARIA – značky, ktoré sa používajú na vyjadrenie významu bloku (hlavička, pätička, ponuka a podobne) a vlastnosti, ktoré uľahčujú prístupnosť dynamických prvkov (formulárov, skriptov a podobne).
- Červený a biely krúžok (Contrast Errors) – nedostatočný kontrast farieb.

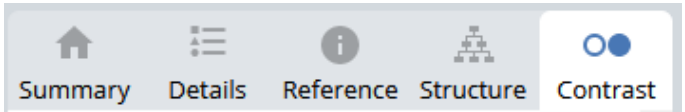
4. V prvom rade by sme sa mali zamerať na kontrolu chýb vyznačených červenou farbou. Podrobnejšie informácie o jednotlivých chybách získame, ak klikneme na jednotlivé červené ikonky označujúce chyby.



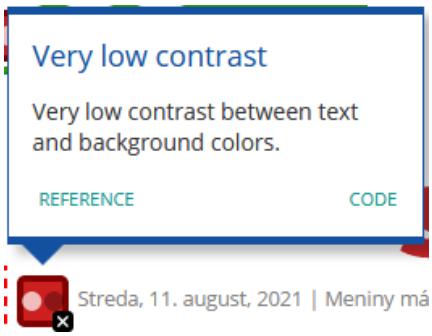
Aby sa lepšie našli všetky chyby, možno použiť ich zoznam, ktorý sa zobrazí po kliknutí na záložku **Details**. Ak v tomto zozname klikáme na jednotlivé ikonky, dostaneme sa v pravej časti na konkrétne miesta s chybami.



5. Dôležité je odstrániť aj chyby v kontraste farieb. V ľavej časti prepneme na záložku **Contrast**.



6. Prvky s nedostatočným kontrastom sú vyznačené ikonkou  . Keď na túto ikonku klikneme, zobrazí sa podrobnejšia informácia o probléme.



ÚLOHA 17.13

Otestujte pomocou nástroja Wave prístupnosť na stránke pizzerie, ktorú ste vytvárali na predchádzajúcich hodinách.

a) Aké chyby nástroj zahlásil?
b) Ktoré z nich sú opodstatnené?
c) Odstráňte tieto chyby.

Metodika pre učiteľa

CIEĽ

Cieľom je, aby sa študenti naučili, ako pri vytváraní webových stránok zabezpečiť prístupnosť informácií a plnú funkčnosť pre jednotlivé skupiny znevýhodnených používateľov.

MOTIVÁCIA

Vhodnou motiváciou môže byť úprava hotového webového sídla zoologickej záhrady, ktorú vytvoril neznámy študent.

VÝKLAD

Výklad obsahuje pre každý typ postihnutia popis konkrétnych riešení a návodov na zabezpečenie prístupnosti. V rámci úloh majú študenti aplikovať prezentované riešenia.

V závere kapitoly popisujeme prácu s automatickým nástrojom **WAVE** na testovanie prístupnosti webovej stránky. Študenti majú tento nástroj použiť na testovanie prístupnosti webového sídla, ktoré vytvárali na predchádzajúcich hodinách.

Návody k úlohám.

17.7 Aby bolo webové sídlo prístupné pre nevidiacich, je potrebné vykonať nasledujúce zmeny.

- Pridať zmysluplný alternatívny text k obrázkom na stránke v súbore `index.html`.
- Pridať zmysluplný titulok k odkazom s textom **Čítať viac** na stránke v súbore `index.html`.
- Použiť nadpisové štýly na texty nadpisov na stránke v súbore `zvierata.html`.
- Pre zoznam s nadpisom **Naše zvieratá** na stránke v súbore `zvierata.html` použiť značky určené pre odrážkový zoznam.
- V tabuľke na stránke v súbore `vstupne.html` použiť značky `<th>` a `</th>` namiesto značiek `<td>` a `</td>`.
- K formulárovým poliam na stránke v súbore `objednavka.html` priradiť popisy. Povinné polia treba odlíšiť aj znakom hviezdička. Nestačí, aby boli popisy červenou farbou.
- Pridať zmysluplný alternatívny text k obrázku s mapkou na stránke v súbore `kontakt.html`.

17.8 Aby bolo webové sídlo prístupné pre slabozrakých, je potrebné vykonať nasledujúce zmeny.

- Zmeniť pohyblivý obrázok (logo) v hlavičke za statický obrázok na stránke v súbore `index.html` (viď iné stránky).
- Zmeniť farbu hlavného nadpisu **ZOO – Veselá savana** (tmavozelená so svetlozelenou nevytvára dostatočný kontrast).

- Odstrániť vzorované pozadie na stránke v súbore `zvierata.html` (v štýle `.zvieratko`).
- Zmeniť obrázok s mapkou na stránke v súbore `kontakt.html`, aby bola farba textov dostatočne kontrastná.
- Použiť iný štýl pre užšie okno (`styles.css`) prehliadača a iný pre širšie okno (`styly_max.css`), aby sa texty jednotlivých objektov po zväčšení obsahu okna neprekrývali.

17.9 Aby bolo webové sídlo prístupné pre osoby s poruchami farebného videnia, je potrebné vykonať nasledujúce zmeny.

- Zmeniť farbu hlavného nadpisu **ZOO – Veselá savana** (tmavozelená so svetlozelenou nevytvára dostatočný kontrast).
- Odstrániť vzorované pozadie na stránke v súbore zvierata.html (v štýle `.zvieratko`).
- Pre formulárové polia na stránke v súbore `objednavka.html` treba povinné polia odlíšiť aj znakom hviezdička. Nestačí, aby boli popisy červenou farbou.
- Zmeniť obrázok s mapkou na stránke v súbore `kontakt.html`, aby bola farba textov dostatočne kontrastná.
- Zmeniť farbu textu alebo pozadia v navigácii (kombinácia červenej a zelenej je veľmi problematická).

17.10 Webové sídlo je vhodné pre nepočujúcich, pretože neobsahuje video súbory, ktoré by bolo potrebné otitulkovať, ani žiadne dôležité zvukové signály. Text je dostatočne jednoduchý a prehľadne členený. Keďže sme odstránili vzorované pozadie na stránke v súbore `zvierata.html`, text je dobre čitateľný.

17.11 Webové sídlo je vhodné pre osoby s pohybovým postihnutím. Je zabezpečená plná funkčnosť pomocou klávesnice aj myši. Navigácia nie je príliš rozsiahla, ale možno by sa stránka mohla doplniť o možnosť Skočiť na obsah.

17.12 Aby bolo webové sídlo vhodné pre osoby so špecifickými poruchami učenia, bolo potrebné vykonať nasledujúce zmeny.

- Zmeniť pohyblivý obrázok (logo) v hlavičke za statický obrázok na stránke v súbore `index.html` (viď iné stránky).
- Odstrániť vzorované pozadie na stránke v súbore `zvierata.html` (v štýle `.zvieratko`).

Inak je stránka vhodná. Text je dostatočne jednoduchý a prehľadne členený. Navigácia nie je komplikovaná. Informácie sú radené podľa dôležitosti.

Prístupná verzia webového sídla **ZOO – Veselá savana** je v dátových súboroch určených pre učiteľa.

Nadväznosť na predchádzajúce kapitoly: Táto kapitola nadväzuje na kapitoly 1-13 a 15-16.

ZHRNUTIE

V závere hodiny môže prebehnúť krátka diskusia o tom, pre ktorú skupinu používateľov je najnáročnejšie zabezpečiť prístupnosť webového sídla.

Študenti by si mali uvedomiť, že počiatočná námaha, ktorú budú musieť vynaložiť na to, aby sa naučili vytvárať prístupné webové sídla, sa neskôr vráti v podobe vyššieho počtu spokojných návštevníkov nimi vytvorených stránok.

Prístupný web je aj lepšie použiteľný a okrem toho sa zobrazí omnoho vyššie vo výsledkoch vyhľadávania.

18 POUŽITEĽNOSŤ WEBU

Kým prístupnosť sa zameriava predovšetkým na používateľov s nejakým obmedzením, použiteľnosť sa snaží o ústretovosť voči všetkým používateľom webu. Treba si však uvedomiť, že prístupnosť a použiteľnosť spolu súvisia a nie sú striktne oddelené.

Približne 40 % návštevníkov sa na webové sídlo nikdy nevráti, ak ich prvá návšteva viedla k negatívnej skúsenosti.

DISKUTUJTE

- Kedy je podľa vás nejaká vec použiteľná?
- Aké vlastnosti musí mať napríklad použiteľný mobilný telefón? Stačí, aby nebol pokazený?
- Diskutujte o webových sídlach, ktoré navštevujete neradi. Uveďte aspoň tri dôvody, prečo tomu tak je.
- Diskutujte o webových sídlach, ktoré ste navštívili len raz a povedali ste si „Sem už nikdy viac!“. Uveďte aspoň tri dôvody, prečo tomu tak bolo.

Použiteľný web

Použiteľnosť (usability) webovej stránky určuje:

- jej ovládanie,
- rýchlosť orientovania sa používateľov,
- jej pochopenie,
- jednoduchosť a nenáročnosť.

Použiteľný web sa vyznačuje nasledujúcimi vlastnosťami.

- Používateľ rýchlo pochopí a dokáže používať web, na ktorý prišiel po prvý krát.
- Používateľ ľahko a rýchlo dokáže dosiahnuť svoj cieľ.
- Používateľ si usporiadanie a ovládanie webu zapamätá a rýchlo sa mu vybaví, keď sa po určitom čase naň vráti.
- Používateľ robí minimum závažných chýb a z každej chyby sa rýchlo spamätá.
- Používanie webu prináša používateľovi príjemný zážitok.

Na zle použiteľnom webe používatelia:

- zmätkujú,
- nevedia dosiahnuť cieľ, pre ktorý na webové sídlo prišli,
- často robia chyby a z chýb sa nevedia spamätať,
- často zablúdia,
- keď sa vyskytne prekážka, opúšťajú stránku so zlým pocitom.

ÚLOHA 18.1

Analyzujte použiteľnosť nasledujúcich webových sídiel.

a) Je navigácia na webovom sídle https://trencin.sk/ konzistentná (na všetkých stránkach rovnaká) a prehľadná?
b) Je jasné zameranie a určenie stránky http://www.centrummemory.sk/?
c) Má text na stránke https://kratkosrstemorca.webnode.sk/privykanie/ vhodnú štruktúru? Dá sa v ňom dobre orientovať?
d) Sú odkazy na stránke http://www.vsetkoomorcatach.wbl.sk/zdrave-morca.html jasne vizuálne oddelené?

ÚLOHA 18.2

Nájdite webovú stránku, ktorá podľa vás nespĺňa vlastnosti dobre použiteľného webu. Uveďte dôvody svojho výberu.

Princípy použiteľnosti webu

Teraz uvedieme sedem základných princípov [22], ktoré treba dodržiavať, aby bol web dobre použiteľný.

Princíp 7±2

Podľa G. A. Millera si bežný človek udrží v krátkodobej pamäti päť až deväť vecí. Navigácia by preto nemala obsahovať viac ako 5 až 9 položiek.

Princíp dvoch sekúnd

Zo skúseností vieme, že čím menej musíme pri práci s webovou stránkou čakať, tým radšej s ňou pracujeme. Používatelia by na odozvu systému nemali čakať viac ako 2 sekundy.

Princíp troch kliknutí

Používatelia prestanú webové sídlo používať, ak na nájdenie akejkoľvek informácie potrebujú viac ako tri kliknutia. Webové sídlo by preto malo mať jednoduchú a intuitívnu navigáciu.

Princíp 80/20

Návštevníci webu sa potrebujú dostať s minimálnou námahou k maximálnemu množstvu informácií. Tiež možno povedať, že na najnavštevovanejších stránkach stačí používateľom venovať 20% času a námahy na to, aby získali 80% informácií.

Princíp dobrej interakcie s používateľom

Ben Shneiderman navrhol nasledujúce pravidlá pre interakciu používateľa s počítačovými systémami vo všeobecnosti.

- Usilujte sa o konzistenciu.

- Umožnite používanie klávesových príkazov.
- Ponúkajte informatívnu spätnú väzbu.
- Ponúkajte jednoduché spracovanie chýb.
- Podporujte pocit získanej kontroly.
- Umožnite jednoduché navrátenie akcií.

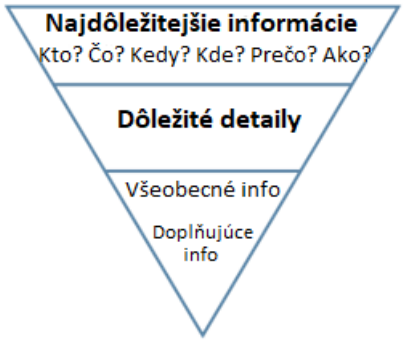
Fittsov princíp

V roku 1954 Paul Fitts zistil, že ľudské smerovanie k cieľu závisí od vzdialenosti od cieľa a od jeho veľkosti. Takže dosiahnutie cieľov, ktoré sú menšie a vzdialenejšie, je časovo náročnejšie.

To znamená, že prvky, na ktoré sa bude často klikať, treba umiestniť vo vrchnej časti stránky alebo na postrannom paneli a nie v spodnej časti stránky.

Princíp obrátenej pyramídy

Najprv treba uviesť súhrnné informácie a až potom podrobnejšie detaily o téme.



Obrázok 18.1 Obrátená pyramída známa aj z oblasti žurnalistiky. Ide o názornú ilustráciu metódy štruktúrovania a radenia informácií [22].

ÚLOHA 18.3

Nájdite stránky, ktoré nespĺňajú niektoré z uvedených princípov použiteľnosti webu. Uveďte ktoré princípy nie sú splnené. Zdôvodnite svoje tvrdenie.

Testovanie použiteľnosti webu

Test použiteľnosti je metóda overovania, či a akým spôsobom je schopný návštevník používať webovú stránku a plniť na nej úlohy, pre ktoré bola vytvorená. Tento test sa vykonáva pri vytváraní webu a zavádzaní nových funkčných prvkov. Cieľom je overiť, či bežný používateľ používa funkcie webovej stránky tak, ako to autor predpokladal.

Na **testovanie použiteľnosti** sa zväčša vyberá skupina ľudí, ktorí by mali byť typickými návštevníkmi webu. Títo používatelia dostávajú niekoľko úloh, ktoré by mali v prostredí webovej stránky vyriešiť.

Typicky sú to úlohy rôznej zložitosti od vyhľadania konkrétnej informácie až po komplikovanejšie úlohy, napríklad uzatvorenia objednávky na nejaký tovar v testovanom eshope.

Pri riešení úloh sú testovaní používatelia sledovaní špecialistami na **testy použiteľnosti** a zistené fakty o ich správaní v prostredí webovej stránky sú zaznamenávané, vyhodnocované a následne potom zohľadnené pri ďalšom vývoji webovej stránky.

ÚLOHA 18.4

Otestujte použiteľnosť niektorej stránky, ktorú často navštevujete. Požiadajte o pomoc troch nezávislých používateľov, najlepšie počítačových nováčikov. Pripravte pre nich napríklad nasledujúce otázky, na ktoré by mali počas testovania odpovedať.

- Na akú webovú stránku som sa dostal?
- Na čo slúži táto webová stránka?
- Kde je navigácia a viem sa pomocou nej dostať, kam chcem?
- Kde sú kontaktné informácie?
- Ako sa mi číta text na stránke? Je dostatočný kontrast?
- Ako na mňa stránka pôsobí vizuálne?

Ak webové sídlo obsahuje e-shop, možno doplniť nasledujúce otázky.

- Chcem spraviť objednávku, viem ako na to?
- Ako sa mi na stránke vyhľadáva?
- Je ponuka služieb/produktov prehľadná?
- Ako si vytvorím zákaznícky účet?
- Ako stornujem/zopakujem/doplním objednávku?
- Kde si prezriem históriu mojich objednávok?

Testovaných používateľov webu je potrebné sledovať počas toho, ako používajú stránku a treba si robiť poznámky (čo o stránke hovoria, ako sa správajú, čo im robí problémy, atď.).

ÚLOHA 18.5

Otestujte použiteľnosť vašej školskej webovej stránky. Návrhy na zlepšenie prediskutujte so spolužiakmi.

Metodika pre učiteľa

CIEĽ

Cieľom je, aby študenti:

- oboznámili sa s vlastnosťami používateľsky prítulného webového sídla,
- vedeli otestovať použiteľnosť webového sídla,
- vedeli vytvoriť používateľsky prívetivé webové sídlo.

MOTIVÁCIA

V úvode hodiny môžu študenti diskutovať o webových sídlach, ktoré navštevujú radi a o tých, ktoré navštevujú neradi a o vlastnostiach týchto sídiel.

VÝKLAD

Výkladová časť obsahuje informácie o princípoch použiteľnosti webu a návod na testovanie použiteľnosti.

Návody k úlohám.

18.1 Pri analýze webových sídiel by si mali študenti uvedomiť nasledujúce.

a) Navigácia na webovom sídle https://trencin.sk/ pôsobí veľmi zmätočným dojmom. Na úvodnej stránke je umiestnená inde ako na ďalších stránkach. Po voľbe položky KULTÚRA A TURIZMUS sa používateľ dokonca dostane na stránku s nekonzistentným vzhľadom a nevie sa vrátiť. Navigácia s rovnakými položkami je aj vo vrchnej časti a aj v ľavej časti stránok.
b) Podľa úvodných informácií na stránke http://www.centrummemory.sk/ nie je jasné, či sa jedná o stránku Slovenskej Alzheimerovej spoločnosti, či Centra Memory, alebo Neurologického ústavu SAV.
c) Text na stránke https://kratkosrstemorca.webnode.sk/privykanie/ nemá takmer žiadnu štruktúru. Písmo má v rámci celého textu rovnaké vlastnosti – veľkosť, rez, font. Používateľ musí čítať všetko zaradom. Ak ho v čítaní niečo vyruší, nevie, kde prestal čítať. V texte chýbajú orientačné body. Ani obrázky nie sú zobrazené.
d) Odkazy na stránke http://www.vsetkoomorcatach.wbl.sk/zdrave-morca.html nie sú podčiarknuté, ako to býva zvykom. Zmätočne pôsobí text, ktorý je podčiarknutý, ale nie je to text odkazov.

Nadväznosť na predchádzajúce kapitoly: Táto kapitola nadväzuje na kapitoly 1-13 a 15-17.

ZHRNUTIE

V závere by mala opäť nasledovať krátka diskusia. V rámci nej by si mali študenti uvedomiť, že súčasťou tvorby webu je aj jeho testovanie s reálnymi nezávislými používateľmi a následné zlepšovanie jeho vlastností.

MULTIMÉDIÁ

DANA HORVÁTHOVÁ A PATRIK VOŠTINÁR

19 MULTIMÉDIÁ

V predchádzajúcich kapitolách sme sa naučili vytvárať jednoduché webové stránky, vkladať do nich rôzne objekty (hlavičky, nadpisy, zoznamy, tabuľky, obrázky, odkazy a pod.) a prispôsobiť ich rôznym používateľom (napr. aj osobám so špecifickými potrebami).

V nasledujúcich kapitolách si rozšírime poznatky z oblasti multimédií a naučíme sa ako ich vhodne využiť pri tvorbe webových stránok a práci s informáciami na internete. Osvojíme si niektoré pokročilejšie techniky na prácu s vektorovou grafikou, zvukom, animáciou a videom a vyriešime spoločne niekoľko praktických úloh, v ktorých budeme využívať tieto mediálne elementy.

DISKUTUJTE

- Diskutujte o vašich skúsenostiach s multimédiami, o výhodách a nevýhodách využitia multimédií vo všeobecnosti, resp. na vašej konkrétnej webstránke.
- Aké zaujímavé webstránky s multimediálnym obsahom poznáte, resp. používate?

Dôležité pojmy v multimédiách

DISKUTUJTE

- Pokúste sa sformulovať vysvetlenie jednotlivých pojmov, ako mediálny element, mediálna forma, multimediálna aplikácia, používateľské rozhranie (tzv. interfejs), interaktivita.

Pojem **mediálny element** reprezentuje danú **mediálnu formu**, akou je text, obraz, zvuk, animácia a video. Multimédiá tieto mediálne formy integrujú do výsledného produktu, ktorý nazývame **multimediálna aplikácia**. Tá prostredníctvom **používateľského rozhrania** komunikuje s používateľom. Práve schopnosť prispôsobiť sa čo najviac potrebám používateľa, ponúknuť mu pohodlný výber z viacerých možností, umožniť mu ovplyvniť chod deja, poskytnúť mu voľbu času, následnosti, rýchlosti a formy prezentácie informácií, je jednou z najdôležitejších vlastností multimédií a volá sa **interaktivita**.

Internet nám dnes prezentuje informácie interaktívne a v rôznych mediálnych formách, čo ho robí zaujímavým, názorným a príťažlivým médiom. Práve multimédiá sú zjednocujúcim faktorom, ktorý využíva možnosti nielen **statických mediálnych elementov**, ako sú **text** a **obraz**, resp. rôzne **druhy grafiky**, ale najmä **dynamických mediálnych elementov**, akými sú **animácia**, **zvuk** a **video**. Tie naposledy menované prinášajú informácie od ich tvorcu k používateľovi rýchlejšie, názornejšie a efektívnejšie. Preto sa aj webové stránky stávajú vďaka ich dynamike atraktívnejšími.

Úlohou nasledujúcich kapitol bude vysvetliť a priblížiť nielen rôzne možnosti tvorby grafiky, ale najmä postupy práce s rôznymi dynamickými mediálnymi elementami. Zameriame sa na

spôsoby nahrávania zvuku a videa na webové stránky, resp. ich sťahovania z internetu, vysielania naživo (tzv. streamovanie) a pozrieme sa aj na programovanie multimédií prostredníctvom programovacích nástrojov akými sú HTML5 a jazyk Python.

Text

Text je špeciálny druh informácie, ktorý v jazykovej forme (usporiadaná množina zrozumiteľných znakov a formátovacích informácií) vyjadruje myšlienky autora a sprostredkúva ich čitateľovi. Je to historicky najstaršia forma komunikácie medzi človekom a počítačom.

Text má vysokú informačnú hodnotu, uľahčuje orientáciu a komunikáciu a zvyšuje zrozumiteľnosť. V digitálnom svete aplikácií a webstránok je jeho funkcia obsahová (prináša informácie o predmetnej oblasti) a komunikačná (umožňuje „rozhovor“ medzi používateľom a aplikáciou, či webstránkou).

DISKUTUJTE

Aká je najdôležitejšia vlastnosť textu? Aké ďalšie vlastnosti má mať text použitý na webovej stránke? Prečo je dôležité formátovanie textu?

Obraz a počítačová grafika

V predchádzajúcich kapitolách tejto učebnice sme sa už stretli s pojmami ako fotografie, obrázky, ikonky, symboly, logá a pod. Všetky tieto pojmy reprezentujú obrazovú informáciu. Z hľadiska multimédií je **obraz** charakterizovaný ako množina farebných plôch uzatvorená zvyčajne v obdĺžnikovom ráme. Slovo obraz označuje zobrazenie reality na projekčnom platne, hárku papiera, obrazovke počítača a pod., bez toho, aby sme vedeli o spôsobe jeho vzniku. Ak sa jedná o vznik alebo spracovanie obrazu pomocou počítača, hovoríme o počítačovej grafike.

Počítačová grafika[3] je medziodborová disciplína zahŕňajúca časť informatiky, fyziky, biológie, matematiky, designu a umenia. Sú to všetky vizualizované informácie vytvorené alebo upravované pomocou počítača, prípadne ďalších (napr. mobilných) zariadení.

Počítačovú grafiku rozdeľujeme do 4 základných kategórií:

- znaková grafika – grafika zostavená z ASCII znakov,
- rastrová grafika – zložená z elementárnych farebných plôšok (tzv. pixelov),
- vektorová grafika – zostavená z objektov ako sú body, línie a plochy,
- metagrafika – obsahuje prvky vektorovej aj rastrovej grafiky súčasne.

[3] **Pojem** “Computer Graphics” zaviedol William Fetter v r.1960

Znaková grafika mala najmä v počiatkoch počítačovej grafiky svoj význam, no dnes sa už používa minimálne.

Rastrová grafika má svoje dôležité miesto pri zachytávaní a úprave nasnímaných informácií z reálneho sveta. Jej prednosťou je rýchlosť získavania a spracovania zložitých obrazových informácií.

Vektorová grafika sa používa hlavne pri vytváraní ilustrácií, diagramov, v počítačovej animácií, vo virtuálnej realite a pod. Umožňuje pracovať s každým objektom v obraze zvlášť, zväčšovať a zmenšovať bez straty kvality, čo podstatne zjednodušuje editáciu.

Na prácu s počítačovou grafikou existuje veľké množstvo softvérov, ktoré reprezentujú najmä nasledovní zástupcovia:

- Adobe Photoshop, Microsoft Paint, GIMP – rastrové editory,
- Adobe Illustrator, CorelDraw, Inkscape – vektorové editory.

S rastrovou grafikou už máte pravdepodobne väčšie skúsenosti, preto sme do nasledujúcej kapitoly tejto učebnice zaradili najmä praktickú tvorbu vektorovej grafiky, príp. metagrafiky a zameriame sa na editor Inkscape[4], ktorý je multiplatformový, otvorený a slobodný softvér, uvoľnený pod licenciou GNU a lokalizovaný je aj do slovenčiny.

DISKUTUJTE
■ S akou grafikou (vektorovou, rastrovou,...) ste sa stretli na webových stránkach? Aké kategórie grafiky ste v minulosti vytvárali, spracovávali a aké nástroje ste pri tom použili?

Zvuk

Zvuk je z fyzikálneho hľadiska mechanické vlnenie hmotných častíc, ktoré sa šíria všetkými smermi, ale len v hmotnom prostredí. Teda napríklad vo vákuu sa zvuk šíriť nebude. Z fyziologického hľadiska je zvuk každý akustický podnet, ktorý je schopný vyvolať sluchový vnem. Teda zvuk je všetko, čo počujeme.

Zvuk má v digitálnom svete aplikácií a webstránok niekoľko funkcií. Najčastejšie býva nositeľom informácií, správ pre používateľa, hlasového komentára a pod. (informačná funkcia). Môže však aj motivovať k pozornosti (motivačná funkcia), spríjemňovať estetický zážitok z prijímania informácií (estetická funkcia), prípadne zachytávať atmosféru okolia (sprievodná funkcia).

[4]**Inkscape základy + videotutoriály:** https://inkscape.org/sk/doc/tutorials/basic/tutorial-basic.html + https://inkscape.org/learn/

V tejto učebnici sa venujeme hlavne použitiu zvuku vo videu, možnostiam práce so zvukom na webe a metodike sťahovania/nahrávania zvukových súborov v počítačovej sieti.

DISKUTUJTE

Stretli ste sa so zvukom na webových stránkach? Aké to boli stránky? Akú funkciu malo na stránke použitie zvuku? Máte vlastné skúsenosti so spracovaním zvuku na počítači? Aké nástroje ste použili?

Animácia

Slovo **animácia** pochádza z latinského slova anima (duša, oživenie), čo znamená uvádzanie do pohybu niečo, čo je nehybné. Teda ide len o vytvorenie akejsi ilúzie pohybu. Základnou myšlienkou je rozdelenie pohybu na jednotlivé fázy, pričom každej fáze zodpovedá jedna snímka. Pri premietaní týchto snímok s dostatočnou obnovovacou frekvenciou (cca 25 snímok za sekundu) vnímame pohyb ako plynulý.

Počítačová animácia je teda rýchle striedanie statických, počítačom vygenerovaných obrázkov, ktoré vyvolávajú dojem pohybu.

Využitie animácie je širokospektrálne. Od jednoduchých 2D animácií upútavajúcich pozornosť používateľa, zviditeľnenie dynamických dejov najmä v mikro a makrosvete, až po náročné simulácie pohybov 3D objektov v realisticky vyzerajúcich scénach, napr. virtuálnej reality. Dnešné aplikácie virtuálnej, rozšírenej, či zmiešanej reality, ktoré si bez animácií ťažko predstaviť, majú obrovský potenciál meniť nielen svet filmu a zábavy, ale aj spôsob nakupovania, komunikácie, podnikania a celkovo nášho prežívania.

DISKUTUJTE

Aké rôzne animácie ste videli na webových stránkach? Aká bola ich funkcia? Kedy je podľa vás vhodné použiť animáciu na webovej stránke a kedy nie?

Cieľom tejto učebnice však nie je náročná tvorba animácií (to by si vyžadovalo oveľa väčší priestor), ale len poukázanie na možnosti vloženia jednoduchej animácie do videa, keď treba video doplniť o pohyblivé objekty, oživiť ho o pohybujúci sa text, obrázky, alebo dokonca ďalšie video.

Video

Video je audiovizuálny systém využívajúci spojenie obrazovej a zvukovej informácie do jedného celku. Na rozdiel od animácie, väčšinou video zachytáva dianie reálneho sveta, aj keď v procese post-produkcie (t.j. pri jeho strihu) je možné do výsledného videa pridať akékoľvek mediálne elementy, čomu sa venujú neskoršie kapitoly v tejto učebnici. Okrem **reálneho videa** existuje aj **video virtuálne**, ktorého úlohou je zachytiť dianie na obrazovke monitora s možnosťou pridania hlasového komentára. Táto schopnosť niektorých softvérov sa využíva napr. pri tvorbe video tutoriálov, populárnych najmä vo vzdelávaní.

Video je v dnešnej dobe najrýchlejší a najefektívnejší spôsob, akým môžu digitálne médiá pôsobiť na diváka. Keďže je obľúbené najmä medzi mládežou, zaradili sme do učebnice pomerne rozsiahlu kapitolu o tvorbe a spracovaní videa vo video editore VSDC, o sťahovaní, nahrávaní a prenose videa na webe, o formátoch, kodekoch, ako aj o vysielaní videa naživo.

DISKUTUJTE

- Aké výhody má video oproti statickým mediálnym elementom na webových stránkach? Aké skúsenosti máte s tvorbou vlastného videa? Aké nástroje ste na to použili?

ÚLOHA 19.1

- Zamyslite sa nad tým, ktoré mediálne elementy je najvhodnejšie využiť, na ktorý účel. Začiarknite v tabuľke ďalšie bunky a svoje rozhodnutie zdôvodnite.

	Text	Grafika	Zvuk	Animácia	Video
Zachytenie atmosféry reštaurácie			✓		✓
Simulácia pohybu planét v slnečnej sústave					
Ukážka postupu prípravy pizze					
Zobrazenie ťažiska trojuholníka					
Názov kapitoly, webstránky, popis obrázku					
Vysvetlenie postupu činnosti pre zrakovo postihnutých					

DISKUTUJTE

- Ako by ste mohli využiť multimédiá na svojej stránke, čím by ste mohli zaujať pozornosť a čím naopak odradiť návštevníka stránky?

Metodika pre učiteľa

CIEĽ

Cieľom tejto kapitoly je priblížiť základný pojmový aparát v oblasti multimédií, aby študenti porozumeli základným pojmom a informáciám o jednotlivých mediálnych elementoch a aby sa o ne mohli oprieť v nasledujúcich kapitolách.

MOTIVÁCIA

Na začiatok tejto kapitoly odporúčame rozvinúť diskusiu o skúsenostiach študentov s multimédiami, o výhodách a nevýhodách využitia multimédií vo všeobecnosti, resp. na ich konkrétnej webstránke. Treba poukázať na potrebu primeranosti a striedmosti pri použití rôznych mediálnych foriem. Niekedy „menej je viac“. Je dobré zistiť, aké zaujímavé webstránky s multimediálnym obsahom študenti poznajú, resp. používajú.

Ukážky webstránok, ktoré využívajú multimédiá a interaktivitu:

http://www.metoo.sk/tvstream/index (multimediálna televízia)

http://www.climber.io/ (interaktívna produkčná spoločnosť)

https://yourplanyourplanet.sustainability.google/ (obehová ekonomika)

https://opensourcerover.jpl.nasa.gov (NASA - laboratórium s prúdovým pohonom)

http://www.mojevideo.sk/video/2133c/ako_vidi_adobe_buducnost_spracovania_videa_a_grafiky.html (reklamná stránka spoločnosti ADOBE)

VÝKLAD

Táto kapitola je hlavne teoretická, pričom sa predpokladá, že študenti spĺňajú štandardy stanovené pre tematickú oblasť Reprezentácie a nástroje v Rámcovom vzdelávacom programe pre gymnáziá. Odporúčame teda v rámci diskusie zistiť, na akej úrovni sú ich poznatky z tejto oblasti a ozrejmiť::

- čo sú multimédiá,
- aká je ich najdôležitejšia vlastnosť,
- ktoré mediálne elementy sú statické a ktoré dynamické,
- ako vysvetliť význam pojmov text, obraz, zvuk, animácia a video.

Nadväznosť na predchádzajúce kapitoly: Žiadna z predchádzajúcich kapitol nie je nevyhnutná na preberanie tejto kapitoly. V prípade, že študenti do dostatočnej miery neovládajú učivo o multimédiách, odporúčame najprv odstrániť tento nedostatok. Na doplnenie možno použiť zdroje [23], [24], [25], [26], [27].

20 TVORBA VEKTOROVEJ GRAFIKY A METAGRAFIKY V EDITORE INKSCAPE

Keďže súčasný web je plný multimediálneho obsahu (infografiky, fotografií, animácií, videí a pod.), zameriame svoju pozornosť v učebnici hlavne na tie mediálne elementy a na také postupy, s ktorými ste sa v rámci informatiky nemali možnosť stretnúť.

V predchádzajúcich kapitolách sme sa naučili vytvoriť statickú webovú stránku, vyhovujúcu aktuálnym štandardom W3C (HTML 5 a CSS 3). Zmienili sme sa o tom, že bežnou súčasťou webových stránok sú aj obrázky. Ukázali sme si, ako vkladať obrázky do webového dokumentu, oboznámili sme sa s ich dôležitými nastaveniami, ako aj so spôsobom, ako do webovej stránky umiestňovať rôzne ikonky a symboly.

V tejto kapitole učebnice sa zoznámime s možnosťou, ako si vytvoriť vlastný obrázok. Aj keď najjednoduchšie je zaručene zobrať fotoaparát, alebo mobilný telefón a odfotografovať daný objekt a fotografiu potom vložiť na stránku. Čo však v prípade, ak potrebujeme na stránku vložiť logo alebo ikonku, teda obrázok pozostávajúci z geometrických útvarov? Pre tento účel, sme si vybrali práve vektorovú grafiku (s ktorou, ako predpokladáme, majú študenti menej skúseností). **Vektorová grafika** označuje spôsob definovania obrazových informácií pomocou základných geometrických primitív, akými sú **bod, úsečka (vektor), priamka, krivka, mnohouholník**, ktoré sa dajú vyjadriť matematickými rovnicami. Prácu s týmito základnými objektami si ukážeme prostredníctvom voľne dostupného softvéru Inkscape (https://inkscape.org/).

Po inštalácii editora **Inkscape** si otvoríme nový dokument `Súbor>Nový`, alebo stlačíme `Ctrl+N`. Inkscape používa pre svoje súbory formát SVG (Scalable Vector Graphics). SVG je otvorený štandard, podporovaný viacerými grafickými aplikáciami. SVG súbory sú založené na jazyku XML (Extensible Markup Language, rozšíriteľnom značkovacom jazyku) a je ich možné upravovať akýmkoľvek textovým alebo XML editorom. Okrem formátu SVG program Inkscape dokáže importovať a exportovať niekoľko ďalších formátov, ako napr. EPS, PNG a ďalšie.

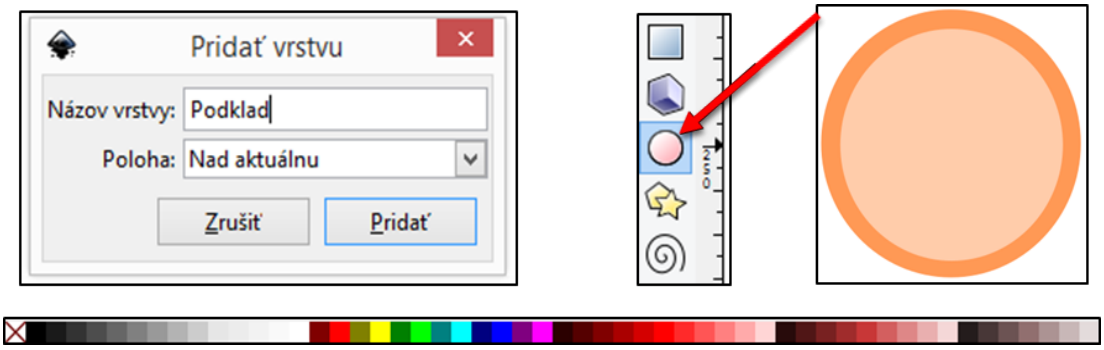
Práca s vrstvami

Umiestnenie objektov do rôznych vrstiev umožňuje nielen vymedziť priestor, zobraziť alebo skryť jednotlivé objekty, ale aj zabrániť náhodným zmenám v procese kreslenia.

PRÍKLAD 20.1

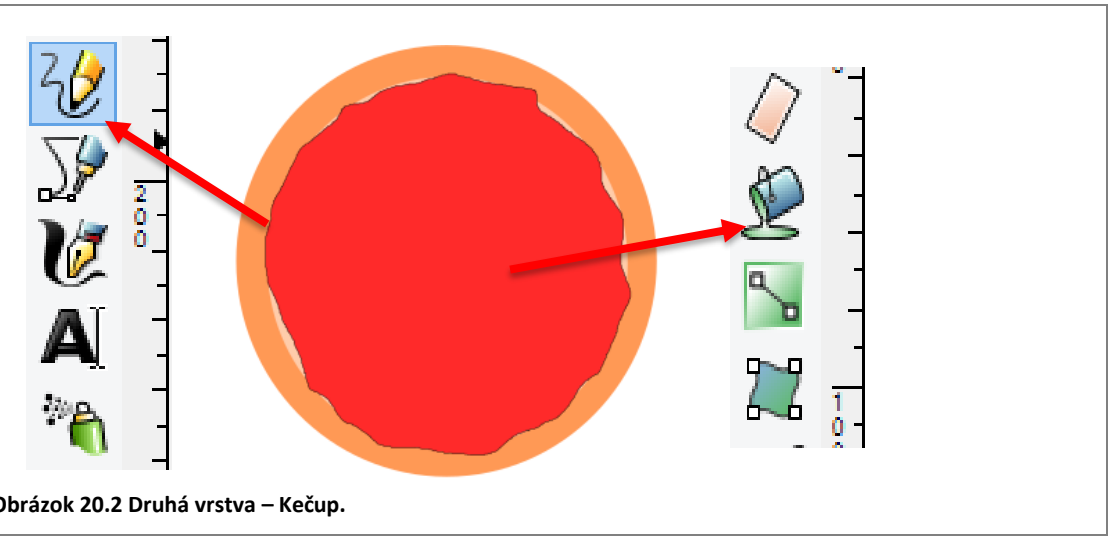
Vytvorme obrázok pizze, ktorý použijeme na stránke pizzerie ako základ pre ikonky rôznych pízz. Na povrch pizze umiestnime rôzne druhy oblohy (šunka, paradajky, kukurica, olivy, brokolica, syr). Využijeme pestré nástroje tvarov, farieb, ako aj import obrázkov a ich následné orezanie, škálovanie a otáčanie. Rôzne druhy oblohy vložíme do osobitných vrstiev, aby bolo možné vytvárať rôzne druhy pizze. Výsledný obrázok uložíme vo vektorovom formáte SVG a exportujeme ho aj do rastrového formátu PNG.

Na vytvorenie obrázku pizze použijeme niekoľko vrstiev, aby sme mohli ľubovoľne voliť zobrazovanie jednotlivých druhov oblohy. Najprv si zvolíme príkaz `Vrstva>nová vrstva` a zadáme jej meno, napr. `Podklad.` Z ponuky nástrojov rôznych tvarov vyberieme nástroj `Vytvorenie kruhov, elíps a oblúkov` a do tejto vrstvy vložíme dva sústredné kruhy. Vnútornú farbu kruhov zvolíme v paneli farieb, ktorý sa nachádza dole nad stavovým panelom.



Obrázok 20.1 Prvá vrstva – Podklad.

Ďalšiu novú vrstvu nazveme `Kecup` a vytvoríme ju pomocou nástroja `Kreslenie voľnou rukou`. Tvar tejto vrstvy kreslíme popri obvode vnútorného kruhu, pričom vnútro vzniknutého útvaru vyplníme nástrojom `Vyplniť ohraničené oblasti (Shift+F7)` a takto ho vyfarbíme červenou farbou z farebnej palety.

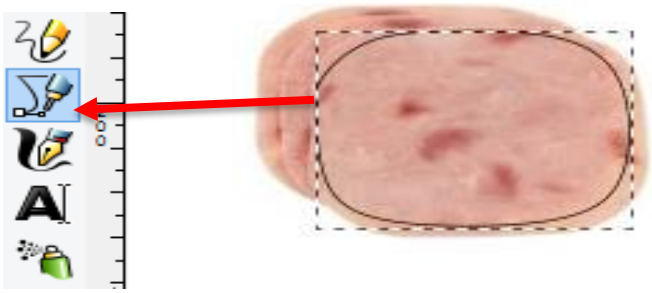


Obrázok 20.2 Druhá vrstva – Kečup.

Ďalšie vrstvy nazveme podľa druhu suroviny, ktorú umiestnime na pizzu, napr. `Sunka, Paradajky, Brokolica` atď. Práve posledné tri menované suroviny vytvoríme pomocou importu obrázku šunky `(Súbor>Importovať)`, paradajky a brokolice, ktoré si napríklad stiahneme z internetu a vystrihneme ich pomocou nástrojov na orezávanie podľa nasledujúceho postupu. Tieto obrázky sú fotografie daných objektov, vytvorené pomocou fotoaparátu, mobilu a pod. (najčastejšie vo formáte JPG, alebo PNG). Po importe týchto objektov vzniká v našom obrázku kombinácia vektorovej a rastrovej grafiky, teda metagrafika.

Šunku obkreslíme pomocou nástroja na `Kreslenie bézierovych a priamych čiar (Shift+F6)` a označíme celý objekt aj výrez súčasne kliknutím a súčasným stlačením klávesu

`Shift`. Následne z ponuky, ktorá sa zobrazí po kliknutí na pravé tlačidlo myši, vyberieme príkaz `Nastaviť orezanie.`



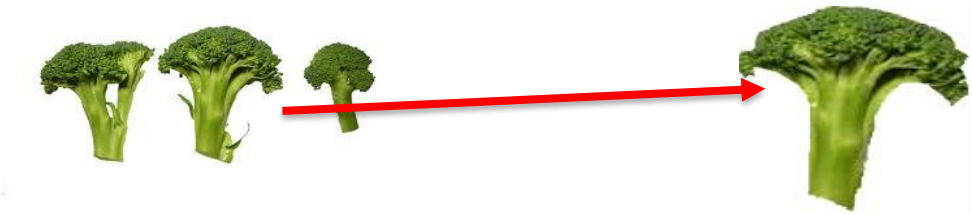
Obrázok 20.3 Tretia vrstva – Šunka.

Ďalšiu vrstvu vytvoríme z obrázku paradajky, ktorý si stiahneme z internetu, prípadne odfotíme rozkrojenú paradajku a následne ju orežeme.

V editore Inkscape môžeme orezávať pomocou nástroja `Vytvorenie kruhov, elíps a oblúkov (F5)` a súčasne označiť objekt a výrez (pomocou klávesu `Shift`) a vybrať `Nastaviť orezanie` z ponuky, ktorá sa zobrazí po stlačení pravého tlačidla myši. Takto získaný orezaný kruhový tvar paradajky môžeme potom kopírovať, zväčšovať, otáčať a tak vkladať do vrstvy `Paradajky`.

Obrázok 20.4 Štvrtá vrstva – Paradajky.

Zložitejší proces nás čaká pri tvorbe komplikovanejších tvarov, ako je napr. brokolica, ktorú po vložení ľubovoľného obrázku brokolice musíme obkresliť a orezať.



Obrázok 20.5 Piata vrstva – Brokolica.

Tvarovanie robíme pomocou nástroja `Kreslenie voľnou rukou`. Výrez brokolice docielime veľmi podobne, ako v prípade šunky, či paradajky. Kurzorom klikneme do začiatočného bodu, kde vznikne malá štvorcová kotva. Každým ďalším kliknutím tvoríme ďalšie kotvy, ktorými postupne obkresľujeme, resp. orezávame objekt. Tieto kotvy umožňujú buď pokračovať v ceste,

alebo ju uzavrieť v koncovom bode pomocou klávesu `Enter`. Ak nie sme s vyrezaním objektu spokojní, celú cestu môžeme zrušiť klavesom `ESC`.

Vyrezanú brokolicu môžeme zväčšovať, zmenšovať, otáčať a podobne. Pri rozkladaní brokolice na pizzu postupujeme rovnako ako pri predchádzajúcich surovinách, akurát objekty rozkladáme do vrstvy `Brokolica`.

Každú ďalšiu surovinu, ktorú by sme chceli mať na povrchu pizze, vytvoríme jedným zo spomenutých spôsobov a uložime do osobitnej vrstvy. Tak sme napríklad vytvorili vrstvy `Kukurica` a `Olivy`.

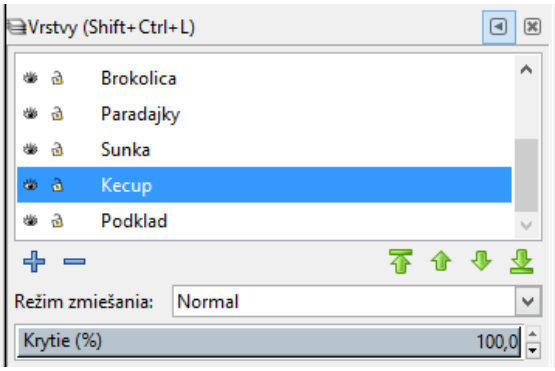
Obrázok 20.6 Ďalšie vrstvy – Kukurica a Olivy.

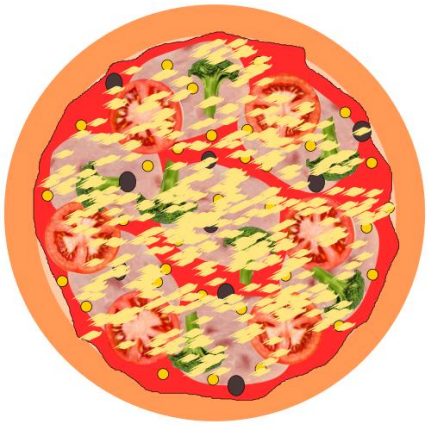
Poslednú vrstvu vytvoríme zo strúhaného syra a zoznámime sa tak s dvomi ďalšími funkciami z palety nástrojov. Pomocou nástroja `Kresliť kaligrafický ťah alebo ťah štetca (CTRL+F6)` si vytvoríme malý kúsok strúhaného syra, ktorý následne pomocou nástroja `Sprejovať objekty sochárstvom alebo maľovaním (Shift+F3)` náhodne rozprašujeme po povrchu pizze a ukladáme do vrstvy `Syr`.

Obrázok 20.7 Posledná vrstva – Syr.

Vrstvy vieme medzi sebou presúvať, pridávať ich, mazať, prípadne ich zobrazovať, skrývať a uzamykať. Výsledok môže byť napríklad podobný ako na Obrázku 20.8.

V priebehu práce je užitočné ukladať čiastkové stavy do súboru vektorového formátu SVG, v ktorom máme možnosť pracovať s jednotlivými vrstvami a meniť tak výsledné zobrazenie pizze. Finálny obrázok pizze môžeme uložiť aj do rastrového formátu, najčastejšie PNG. Export súboru PNG sa robí prostredníctvom dialógového okna `Exportovať bitmapu`. V tomto formáte však už možnosť ďalšej editácie pizze mať nebudeme, ale môžeme ho použiť na webovej stránke.





Obrázok 20.8 Výsledný obrázok pizze a použité vrstvy.

ÚLOHA 20.2

■ Vytvorte obrázky rôznych druhov pízz pomocou zapínania a vypínania vrstiev s príslušnými surovinami. Ak máte záujem na pizzu uložiť ešte ďalšie suroviny, smelo skúšajte, experimentujte a hrajte sa s možnosťami editora Inkscape.

Logické operácie s objektami

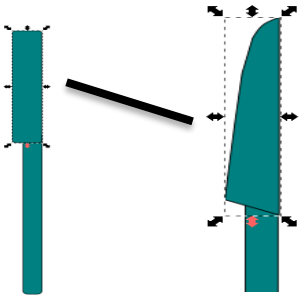
Logické operácie tvoria základ činnosti počítačov a umožňujú riešiť mnohé, nielen logické, ale aj matematické úlohy a problémy.

Medzi **základné logické operácie** patria logický súčet, logický súčin a negácia. V nasledujúcich príkladoch si ukážeme, ako je možné využiť logické operácie pri tvorbe vektorovej grafiky.

PRÍKLAD 20.3

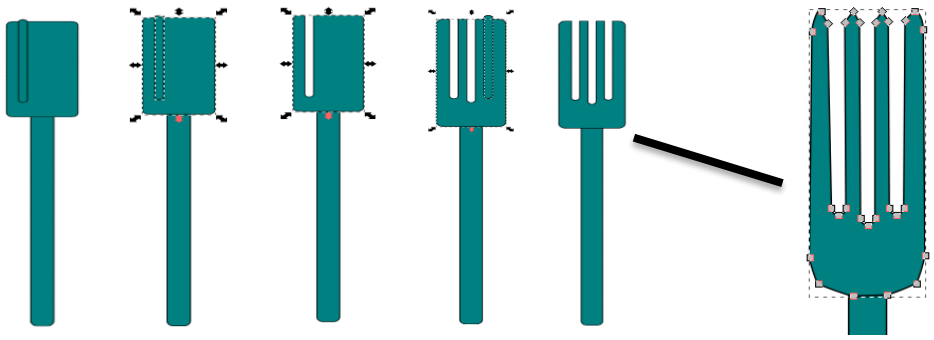
Inšpirujeme sa obrázkami z internetu a vytvoríme si vlastný piktogram vidličky a noža, pričom na tvorbu využijeme logické operácie s objektami, ako napr. zjednotenie, rozdiel, vylúčenie a pod. Na docielenie plynulejšieho tvaru kriviek použijeme prácu s uzlovými bodmi. Výsledný obrázok uložíme vo vektorovom formáte SVG a vyexportujeme ho aj do rastrového formátu PNG.

Najprv pomocou nástroja `Vytvorenie obdĺžnikov a štvorcov` nakreslíme hrubé obrysy noža, ktoré ďalej tvarovo upravujeme. Jednak používame nástroje na prácu s uzlovými bodmi a v niektorých momentoch sa nám hodia rôzne logické operácie, ktoré nám umožňujú kombinovať dva alebo viac objektov pomocou príkazov v ponuke `Cesta`.

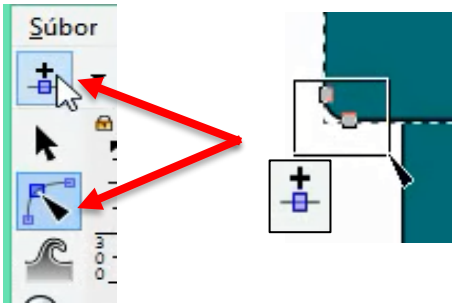


Obrázok 20.9 Vytvorenie čepele noža.

Podobne, ako sme vytvárali nôž, pomocou rovnakého nástroja na `Vytvorenie obdĺžnikov a štvorcov`, nakreslíme hrubé obrysy vidličky, do ktorej postupne zarezávame výrezy. V prípade hrotov vidličky použijeme logickú operáciu odčítania pomocou nástroja `Cesta>Rozdiel` a postupne vykrajujeme jednotlivé hroty vidličky, ktoré ešte neskôr zaoblíme a zaostríme.



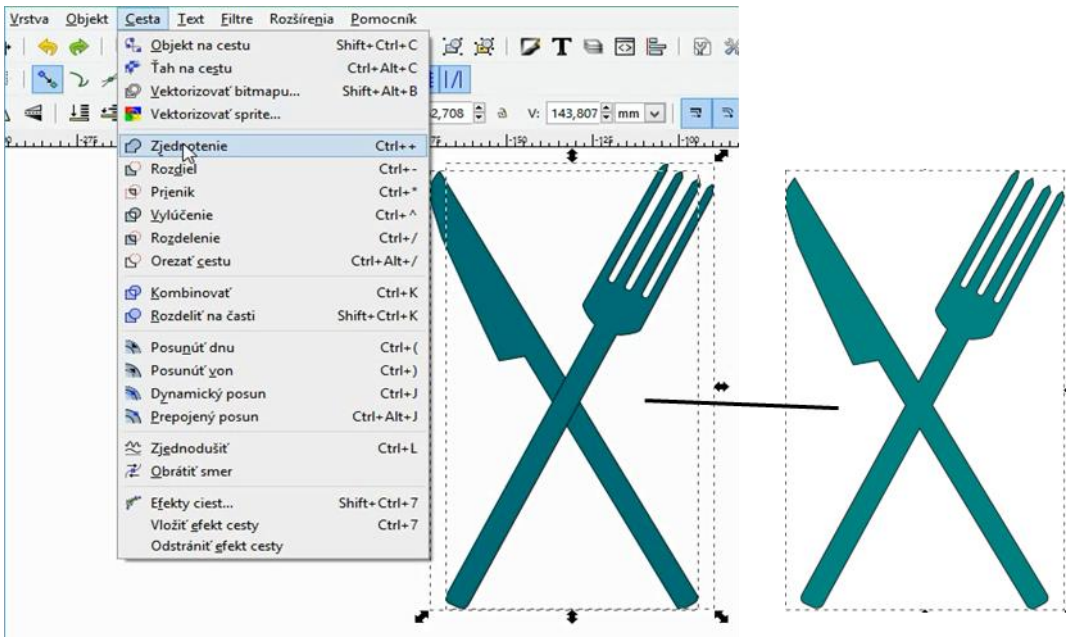
Obrázok 20.10 Postupné vyrezávanie a tvarovanie hrotov vidličky.



Obrázok 20.11 Vyhladzovanie hrán použitím nových uzlových bodov.

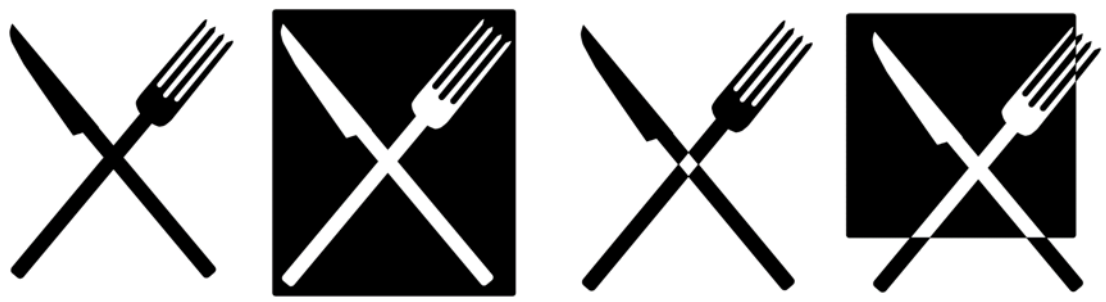
Ak chceme docieliť plynulejší tvar kriviek, môžeme použiť v režime `Upraviť uzly cesty (F2)` ikonu `Vložiť nové uzly do vybraných segmentov`.

V obrázku vyberieme dva uzly, medzi ktorými má vzniknúť nový uzol a klikneme na spomínanú ikonku. Môžeme tak urobiť opakovane viackrát, podľa potrebného počtu uzlov. Čím viac uzlových bodov umiestnime na krivku, tým plynulejší tvar môžeme docieliť.



Obrázok 20.12 Logické operácie – zjednotenie.

Následne upravujeme tvar krivky posúvaním jednotlivých uzlových bodov, až kým nedocielime požadované zaoblenie. Takýmto spôsobom vytvoríme výslednú čepeľ noža a hroty vidličky. Hotový nôž a vidličku potom umiestnime do požadovanej skríženej pozície a pomocou nástroja `Cesta>Zjednotenie` ich spojíme do jedného obrázku. Nakoniec môžeme hotové príbory prefarbiť a rôzne kombinovať s využitím logických operácií, ako vidieť na Obrázku 20.13.



Obrázok 20.13 Ukážky rôznych logických operácií – zjednotenie, rozdiel, prienik a kombinácie.

V prípade, že ukladáme postup našej tvorby priebežne do súboru vo formáte SVG, môžeme sa spätne vrátiť k jednotlivým krokom postupu prostredníctvom `Upraviť História vrátení (Shift+Ctrl+H)`. Ak vyexportujeme finálny obrázok do formátu PNG, ďalšia editácia objektov v obrázku už nie je možná.

ÚLOHA 20.4

- Pokúste sa vytvoriť za pomoci editora Inkscape iné logo, piktogram, alebo obrázok, ktorý sa viaže k reštauračným službám a ktorý by ste mohli využiť na stránke vašej pizzérie.

Metodika pre učiteľa

CIEĽ

Cieľom tejto kapitoly je naučiť študentov používať pokročilejšie techniky na tvorbu vektorovej grafiky a metagrafiky v grafickom editore Inkscape. Základom príkladov a úloh je práca s vrstvami, pričom sa študenti naučia používať niekoľko nástrojov na kreslenie, vypĺňanie, orezávanie, sprejovanie a pod. Ďalšia časť tejto kapitoly je venovaná logickým operáciám s objektami, kde sa študenti naučia využívať nástroje z ponuky príkazov Cesta (zjednotenie, rozdiel, prienik, vylúčenie a ich rôzne kombinácie).

MOTIVÁCIA

Hodinu navrhujeme začať diskusiou na tému vektorová versus rastrová grafika (skúsenosti študentov, náročnosť získavania, resp. tvorby obidvoch grafík, rozdiely v spracovaní, výhody a nevýhody,...). Pravdepodobne z diskusie vyplynie, že získavanie digitálnej fotografie je pre nich samozrejmé. Dokonca aj úprava, resp. jej ďalšie spracovanie je v mnohých editoroch názorné, jednoduché a intuitívne. S touto časťou počítačovej grafiky sa stretli v rámci predchádzajúcich hodín informatiky. Preto je učebnica zameraná hlavne na prácu s vektorovou grafikou, s ktorou majú študenti menšie skúsenosti. Odporúčame teda obrátiť pozornosť na možnosti tvorby a využitia vektorovej grafiky.

VÝKLAD

Táto kapitola je hlavne praktická. Predpokladáme, že študenti už ovládajú prácu s grafickým editorom na základnej úrovni. Okrem toho, že môžu študenti diskutovať o výhodách a nevýhodách vektorovej grafiky oproti rastrovej grafike, získajú praktické zručnosti v práci s:

- vrstvami vo vektorovej grafike,
- logickými operáciami s objektami.

Nadväznosť na predchádzajúce kapitoly: Táto kapitola nadväzuje na kapitolu 19.

ZHRNUTIE

V tejto časti učebnice sa venujeme tvorbe vektorovej grafiky a metagrafiky, práci s vrstvami a logickým operáciám s objektami.

V závere by mohla prebehnúť diskusia o tom, ako študentom pomohla, resp. mohla pomôcť vektorová grafika pri tvorbe ich vlastnej stránky.

21 TVORBA VIDEA

Video sa zdá byť novým fenoménom internetovej doby. YouTube sa stal najmä v posledných rokoch nielen obľúbenou alternatívou televíznej zábavy, ale aj miestom, na ktorom je odchovaná nová generácia. Miestom, kde vznikajú nové profesie, resp. hviezdy. Obrovskú popularitu si takto získali mnohí komici, hráči videohier, youtuberi či vlogeri (videoblogeri).

DISKUTUJTE

- Akú funkciu plní v súčasnosti video na webe, aké skúsenosti máte so snímaním a strihom videa, príp. nahrávaním a spracovaním zvuku. Aké trendy v tejto oblasti pozorujete (vlogy, youtuberi, podcasty a pod.)

Keďže dnešní študenti vyrastajú na videách z internetu, je pre mnohých z nich video samozrejmou súčasťou ich života. V rámci projektu IT Akadémia došlo k inovácií matematiky, informatiky, prírodovedných a odborných predmetov na ZŠ a SŠ, kde sa základom práce s videom venuje tematický celok Spracovanie videa. Preto by bolo plytvaním miesta v tejto učebnici, zoznamovať Vás so základnou tvorbou videa. V tejto kapitole si ukážeme, ako možno zobraziť viaceré videá naraz, resp. vložiť jedno do druhého, ako odfiltrovať z pôvodného videa zvukovú stopu a ako pridať do videa vlastný zvuk, animovaný objekt a titulky.

PRÍKLAD 21.1

Vytvoríme krátke video, ktoré bude na stránke našej pizzérie plniť niektorú z jeho dôležitých funkcií, napr. upútavať pozornosť, prezentovať spoločnosť, prípadne približovať postup nejakej kuchárskej činnosti.

21/01_video.html

Potrebný video materiál môžeme nasnímať napríklad pomocou mobilného telefónu. Okrem mobilného telefónu potrebujeme aj nástroj, ktorý umožňuje pracovať nielen s textom, titulkami, obrazovými súbormi a geometrickými tvarmi, ale aj s ďalšími objektmi ako sú zvuk, animácia a samozrejme video. Využijeme pritom pestré nástroje editora VSDC Free Video Editor na strih a tvorbu videa, ktorý je voľne dostupný a spĺňa väčšinu požadovaných kritérií.

Začíname s video editorom VSDC

VSDC Free Video Editor je intuitívny a výkonný softvér na úpravu videa, zameraný na vytváranie videí s rôznymi vizuálnymi a zvukovými efektmi a editáciu video súborov populárnych formátov. Navyše umožňuje rýchlu konverziu video a audio súborov z jedného formátu do druhého. V neposlednom rade pomáha používateľom pri zdieľaní svojich videí na sociálnych sieťach a miestach na zdieľanie videa, ako sú Facebook, Twitter, Youtube a Vimeo. VSDC Free Video Editor pochádza zo série softvérových produktov uverejnených spoločnosťou Flash-Integro LLC a je určený len pre domáce a vzdelávacie účely.

Budeme sa pohybovať v anglickom prostredí, preto všetky názvy budeme uvádzať v pôvodnom jazyku. K programu je prístupný podrobný manuál v angličtine doplnený o videotutoriály na stránke: http://www.videosoftdev.com/how-to-use-free-video-editor

Po pomerne jednoduchej inštalácií VSDC Free Video Editora a jeho spustení, máme niekoľko možností, ako využiť VSDC editor. My sa však zameriame len na niektoré a to: video vo videu, alebo viac videí vedľa seba, animácia, zvuk a titulky vo videu a import a export videa.

Obrázok 21.1 Prostredie VSDC Free Video Editora.

Najprv však treba vytvoriť svoj **vlastný projekt** - kliknutím na ikonku `New project` v ponuke príkazov `Projects`, resp. `Blank project` v rýchlom paneli. Neskôr budeme môcť predtým vytvorený projekt otvoriť pomocou tlačidla `Open project`, resp. `Recent files.`

Import videa a ďalších objektov

Po nastavení parametrov nového projektu (základné informácie o názve, o autorovi, rozlíšenie videoformátu, frekvencia obrázkov a pod.) načítame pomocou príkazu `Editor>Add object` všetky potrebné objekty, s ktorými budeme pracovať (Animation, Image, Audio, Video,...). Ďalšia možnosť **pridávania súborov** je výberom jednej z ikoniek na zvislom paneli nástrojov.

Obrázok 21.2 Vloženie objektu.

Po výbere súboru sa otvorí okno, kde si môžeme nakonfigurovať polohu objektu v čase a určiť jeho umiestnenie a veľkosť. Je dobré pripomenúť, že po každej dôležitej fáze tvorby je treba projekt priebežne ukladať, aby sme nestratili vytvorené dielo.

Video vo videu

Teraz si bližšie popíšeme prípad, keď chceme na obrazovku z nejakých dôvodov umiestniť dve videá (príp. aj viac), ktoré by sa prehrávali simultánne vedľa seba (príp. nad sebou, alebo jedno v druhom). Takéto súčasné prehrávanie **viacerých videí** sa stáva populárnym, takže si v nasledujúcom príklade ukážeme, ako to urobiť v editore VSDC.

PRÍKLAD 21.2

Importujme do VSDC Video Editora aspoň dve videá a experimentujme s ich vzájomnou polohou (vedľa seba, nad sebou a pod.), upravujme ich veľkosť zobrazenia, prípadne prekrytia jedného videa cez časť druhého. Po prehraní videí sledujeme, ako sa vzájomne ovplyvňujú, resp. miešajú zvukové stopy obidvoch videí.

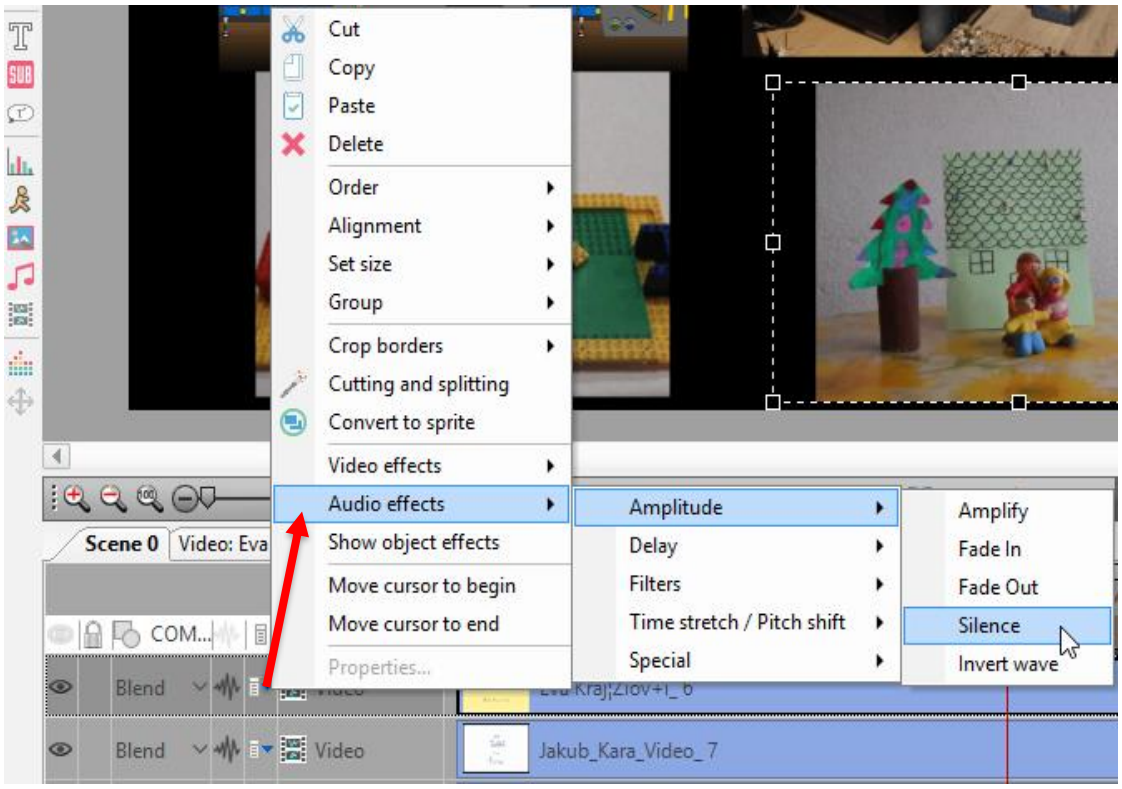
21/02_video.html

Všetky videá, ktoré chceme spolu kombinovať na jednej obrazovke, načítame rovnako, len musíme presne určiť ich vzájomnú polohu a prispôsobiť ich rozmery oknu. Prekážkou v takomto časovom prekrývaní videí môže byť nevhodné miešanie zvukových stôp pri prehrávaní viacerých videí súčasne. Preto, ak chceme potlačiť zvuk vo videu, musíme pre každú videostopu kliknúť na tlačidlo `Show menu`, v ponuke vybrať príkaz `Audio effects>Amplitude>Silence` a aplikovať ho na celú stopu videa.

Potom môžeme k zobrazeniu videí zvoliť **vlastný zvuk** tak, že v ponuke zvolíme príkaz `Editor>Add object>Audio` a vyberieme si z vopred pripravených zvukových súborov (na obr. 21.4 je to napríklad zvukový súbor Track 02_1).

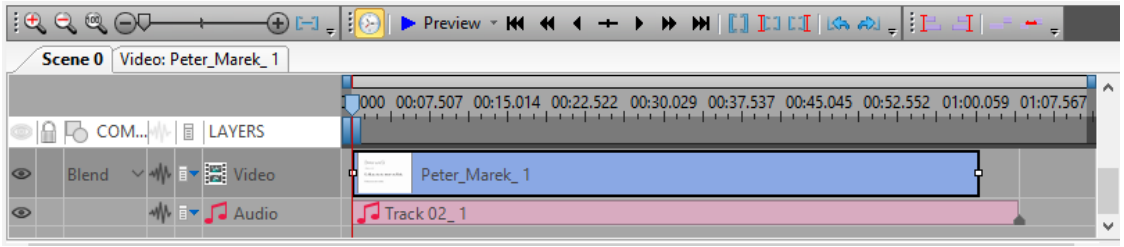
POZNÁMKA

Zvukovú stopu z videa odstránime/potlačíme len v prípade, ak sa nevhodne mieša so zvukom iného videa. Niekedy môže byť vhodné skombinovať obidva zvuky do nového videa.



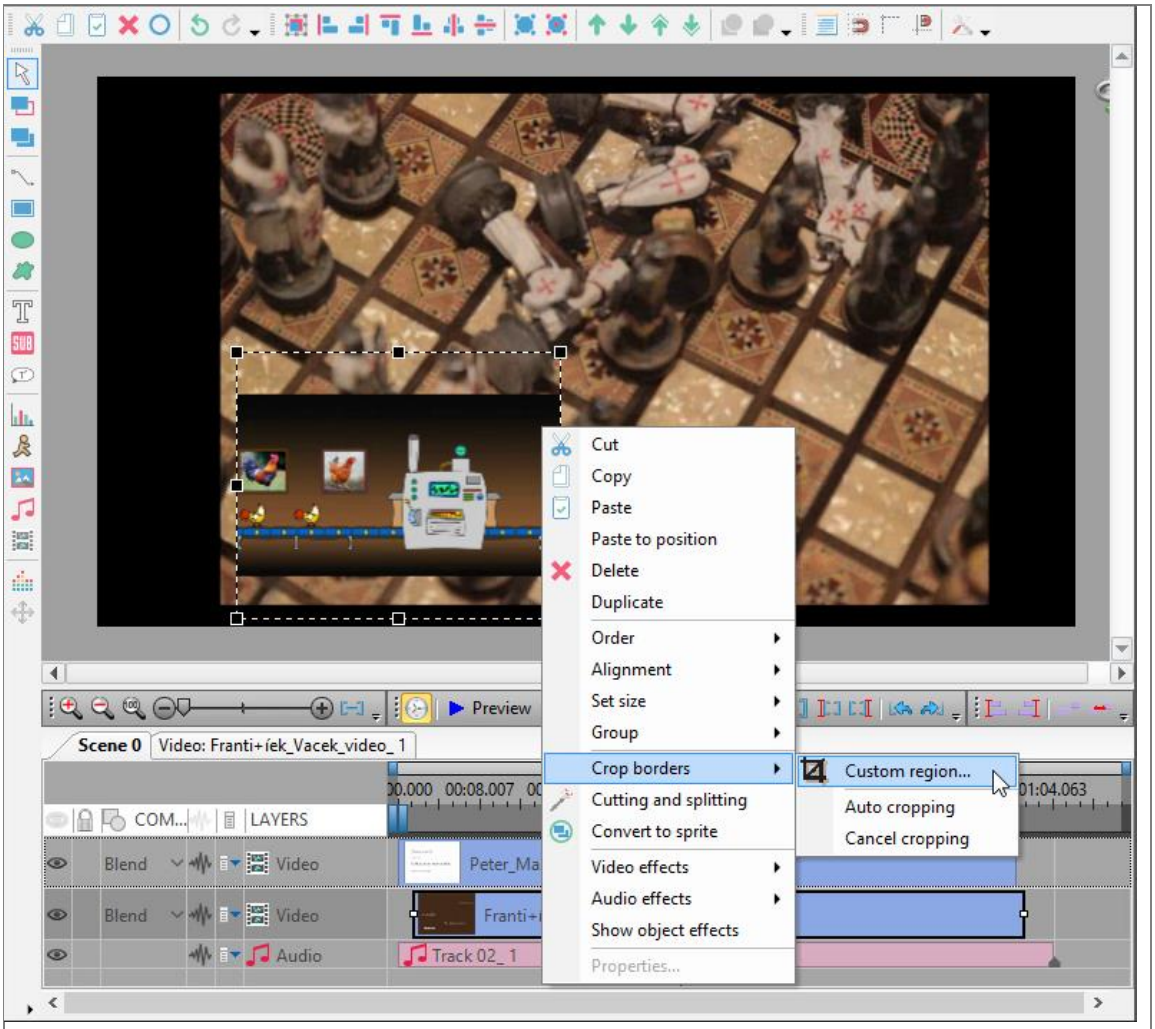
Obrázok 21.3 Odstránenie zvukovej stopy z videa.

V prípade, ak by sme chceli vložiť **jedno video do druhého** (ako obraz v obraze), použijeme podobný postup, len videá prekryjeme cez seba. Napríklad jedno video vložíme nad druhé a nastavíme im optimálnu veľkosť a polohu, ktorú budú mať počas celého prehrávania videí.



Obrázok 21.4 Pridanie zvukovej stopy k videu.

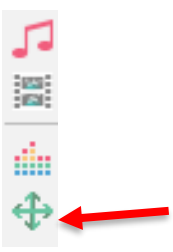
Môžeme dokonca urobiť **výrez videa** voľbou Crop borders>Custom region v ponuke, ktorá sa zobrazí po stlačení pravého tlačidla myši (ako na obrázku 21.5.). Ak by sme napríklad chceli video vložiť do oblasti s konkrétnymi rozmermi, napr. do obrázku televízora, či monitora a pod.



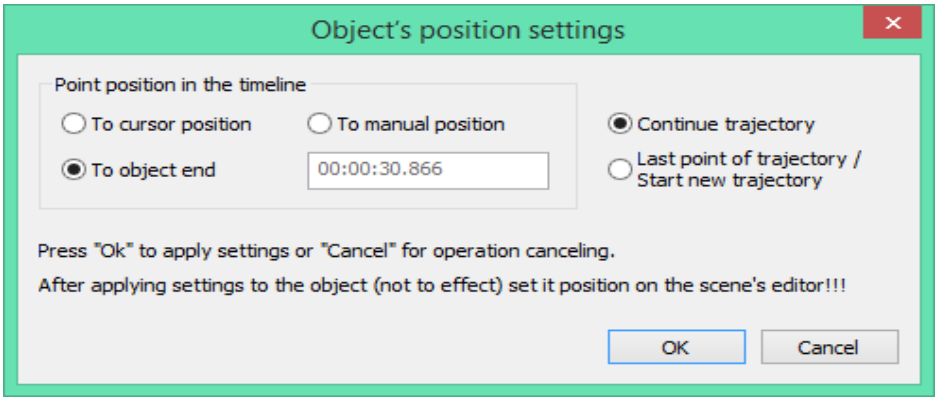
Obrázok 21.5 Vloženie videa do videa (presné umiestnenie).

Animácia vo videu

Niekedy je potrebné doplniť video o **pohyblivé objekty**, napríklad ho oživiť o pohybujúci sa text, obrázky, alebo dokonca ďalšie video. Pomocou efektu `Movement` môžeme túto úlohou ľahko zvládnuť. Treba si zvoliť objekt, ktorý sa bude pohybovať po scéne. Po kliknutí (dvojklik) naň sa vytvorí osobitná záložka na časovej osi. Potom voľbou ikonky `Movement` určíme jeho načasovanie a trajektóriu pohybu prostredníctvom okna `Object`s position settings`.

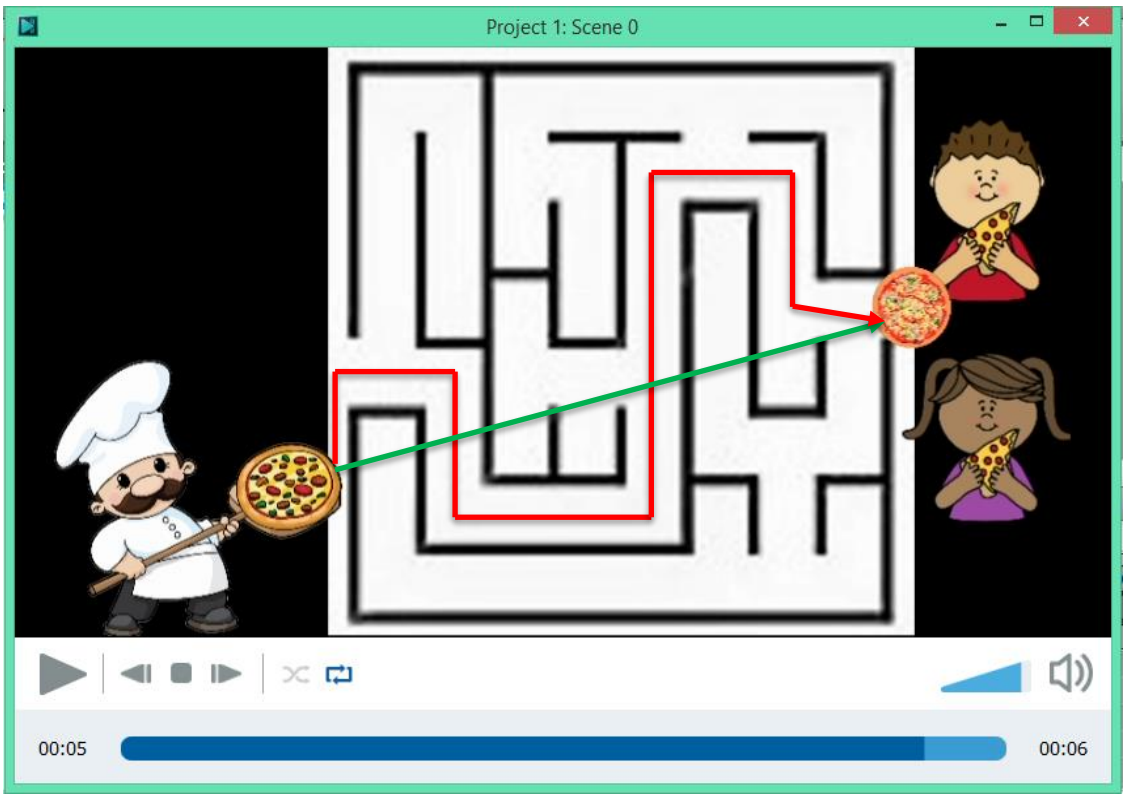


Obrázok 21.6 Animácia pomocou efektu Movement.



Obrázok 21.7 Nastavenie parametrov pohybu objektu.

Ak si chceme zvoliť zložitejšiu **trajektóriu** (červená čiara na obr.21.8), napr. ako v prípade, keď sa nami vytvorená pizza pohybuje po labyrinte až k dvom hladným deťom, potrebujeme na export daného videa platenú verziu programu VSDC Video Editor Pro, ktorá sa však za cca 20 Euro oplatí. V prípade použitia bezplatnej verzie je možný len priamočiary pohyb (zelená čiara na obr. 21.8) z jedného bodu do druhého bodu.



Obrázok 21.8 Jednoduchá trajektória (zelená) vo VSDC Free Video Editore a zložitejšia trajektória (červená) vo VSDC Video Editore Pro.

ÚLOHA 21.3

- Vytvorte kombináciu videa s ľubovoľným pohybujúcim sa objektom (textom, obrázkom, alebo videom), kde použijete funkciu `Movement`.

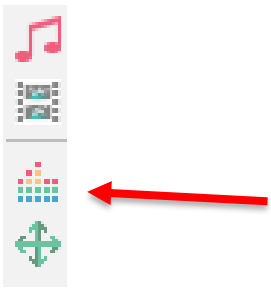
Zvuk vo videu

Do videa je možné vkladať **zvuky zo súborov** rôznych zvukových formátov, ako aj nahrať zvuk z rôznych audio portov, resp. **z mikrofónu** priamo pomocou `VSDC Free Audio Voice Recorder`. Zvuk môžeme pridať na požadované miesto vo videu a konvertovať ho do ľubovoľného formátu (MP3, WAV, WMA, OGG, AAC, M4A, AMR, AU, AIFF a pod.). VSDC Free Video Editor umožňuje dokonca aj redukovať prirodzený šum v nahranom zvuku, príp. znížiť intenzitu nechcených externých zvukov.

ÚLOHA 21.4

- Vytvorte krátke video, ktoré natočíte pomocou mobilu niekde v prírode a pridajte do neho vhodnú hudbu, znejúcu na pozadí.

Zaujímavé **efekty** sa dajú vytvárať, resp. pridávať do videa, pomocou nástrojov `Add object>Audio visualization>Spectrum (Shift+W)`, alebo `Audio abstraction (Shift+Alt+W)`, ktoré pomáhajú vytvárať jedinečné hudobné videá, propagovať albumy alebo upútať oko, či ucho aj náročnejšieho diváka. Viac o postupe a názorných ukážkach môžete nájsť na stránke http://www.videosoftdev.com/how-add-audio-spectrum-visualizer



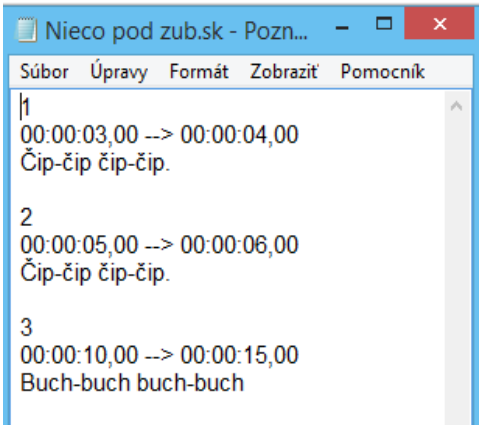
Obrázok 21.9 Vkladanie rôznych zvukových efektov.

Titulky vo videu

Niekedy je potrebné **video otitulkovať**. V tejto kapitole si povieme, ako môžeme vytvoriť vlastné titulky a následne ich pridať do videa.

Súbory s titulkami vo formáte SRT (SubRip Subtitle) sú súbory obyčajného textu, ktoré obsahujú informácie o titulkoch, ako je čas začiatku a ukončenia zobrazenia textu titulkov vo videu a samotný text titulkov. Tieto súbory zabezpečia zobrazenie titulkov v správnom okamihu, vo vybranej farbe a na vybranom mieste.

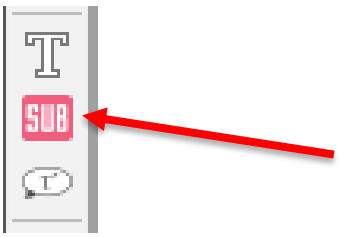
Jednou z možností, ako takýto súbor SRT vytvoriť, je pomocou akéhokoľvek textového editora, napr. Poznámkového bloku. Potrebné je napísať poradové číslo titulkov, presné časové umiestnenie začiatku aj konca zobrazenia textu vo videu v nasledovnom formáte `[hodiny]:[minúty]:[sekundy],[milisekundy]` a samotný text.



Obrázok 21.10 Titulky v editore Poznámkový blok.

Keď sú titulky hotové, je potrebné súbor uložiť ako obyčajný text, ale namiesto formátu TXT treba použiť formát SRT. Následne vo VSDC editore vytvoríme projekt a pomocou príkazu `Editor>Add object` načítame video, do ktorého chceme vložiť pripravené titulky. V tej istej ponuke sú aj príkazy na prácu s textom (vloženie textu, počítadla, titulkov a bublín).

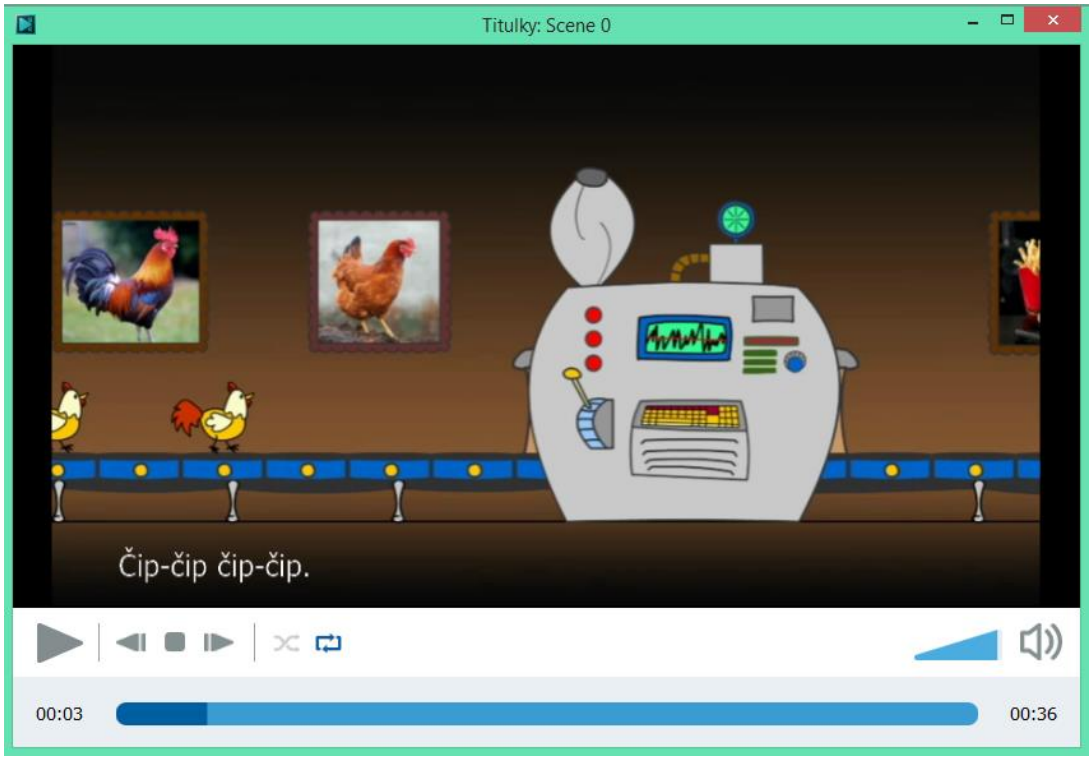
Pre vloženie titulkov zvolíme ikonku `SUB`, vyberieme pozíciu začiatku na časovej osi (`From scene begin`) a načítame predpripravený súbor. Následne kurzorom určíme jeho pozíciu v okne videa a v `Properties window` zvolíme font a farbu textu, prípadne pozadia, aby bol text dobre čitateľný.



Obrázok 21.11 Použitie funkcie Subtitles.

ÚLOHA 21.5

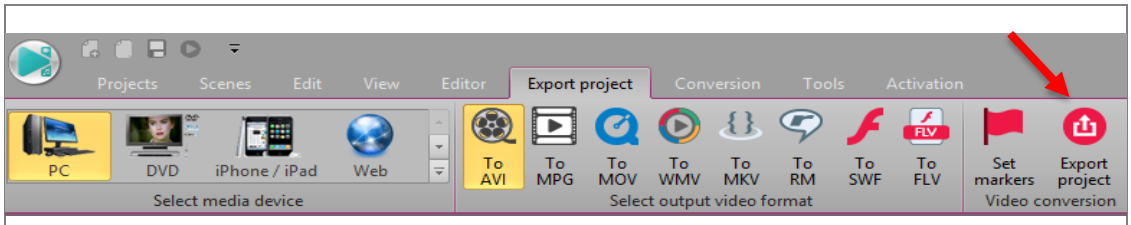
Nahrajte na video nejaký svoj monológ, prípadne krátky rozhovor dvoch priateľov. Aby ste sprístupnili vaše video pre nepočujúcich, pridajte doň titulky, ktoré prepisujú zvuky, príp. hovorené slovo, do textu. Titulky zobrazte na najvhodnejšom mieste videa tak, aby boli dobre čitateľné. Použite funkciu `Subtitles`.



Obrázok 21.12 Výsledné video aj s titulkami.

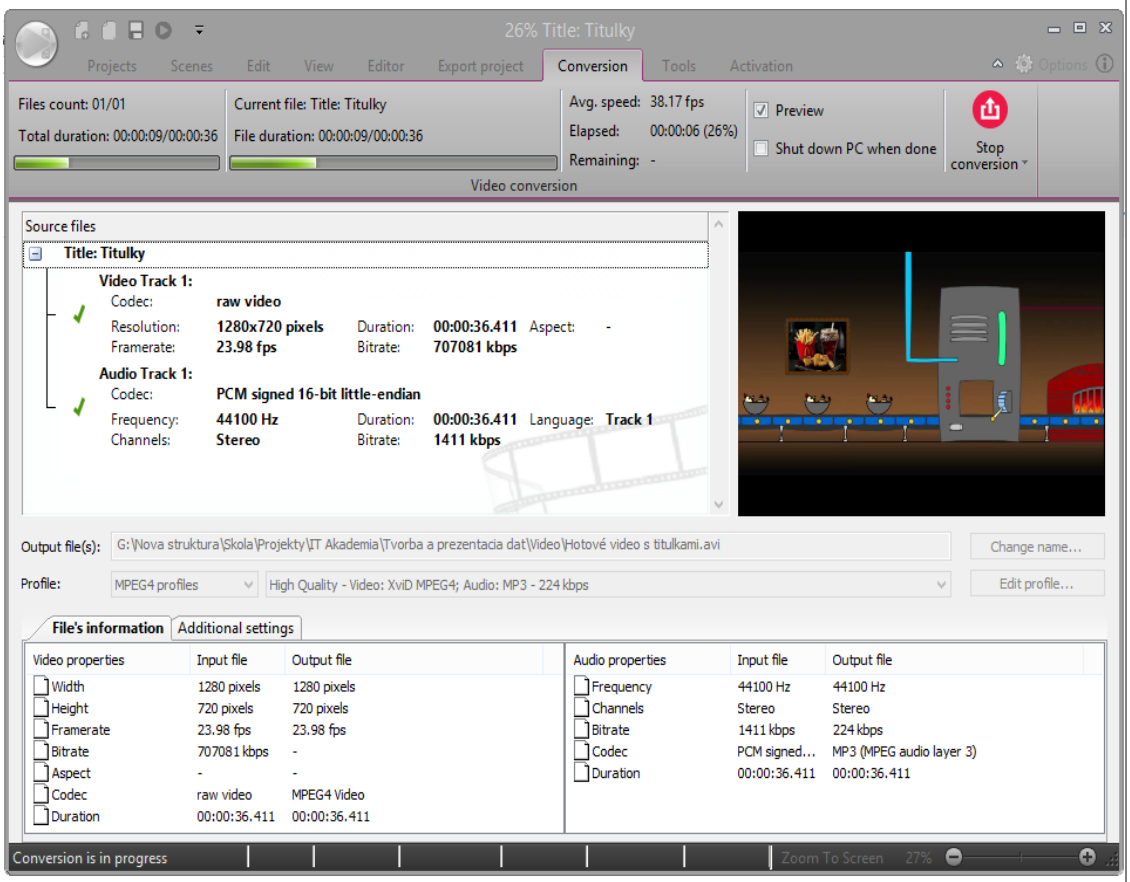
Export videa

Takto otitulkované video uložíme pomocou ponuky `Export project` do požadovaného výstupného formátu `AVI, MPG, MOV, WMV, MKV, RM, SWF, FLV` na vybrané zariadenie `PC, DVD, iPhone/iPad, Web, Mobile, Android, DVD, Xbox, BlackBerry, MP3/MP4 atď.` a dáme ho vyexportovať.



Obrázok 21.13 Export videa na vybrané zariadenie a do požadovaného formátu.

Konverzia do požadovaného formátu tzv. **rendering videa** chvíľku trvá, čo máme možnosť priebežne sledovať na obrazovke monitora. V zobrazovanom okne vpravo je viditeľná aj finálna verzia videa.



Obrázok 21.14 Konverzia videa.

ÚLOHA 21.6

■ Uložte každé vaše video do osobitného projektu, pre prípad, že by ste sa niekedy chceli vrátiť k jeho editácii, resp. strihu. Zvoľte pri exporte aj iné zariadenia (`Select media device`) ako PC, napr. DVD, iPhone/iPad, ale hlavne web. Vyexportujte vaše video do rôznych formátov a porovnávajte ich veľkosti, kvalitu, príp. čas renderovania a diskutujte spolu o tom, ktorý formát vám bude v budúcnosti najlepšie vyhovovať a prečo.

Metodika pre učiteľa

CIEĽ

Cieľom tejto kapitoly je naučiť študentov používať náročnejšie techniky tvorby videa, ako je video vo videu, resp. kombinácia viacerých videí, potlačenie starého, resp. vloženie nového zvuku do videa, animácia vo videu, resp. rozpohybovanie rôznych objektov vo videu, titulky vo videu a export videa do rôznych formátov.

MOTIVÁCIA

Počas diskusie o funkcií videa na webe môžeme študentom ukázať video o donáške pizze na prednášku (treba ich upozorniť, aby takúto situáciu nepokladali za inšpiráciu na narušenie procesu vzdelávania, ale ako technickú motiváciu) a v nasledujúcom videu poukázať na zaujímavú kombináciu dvoch videí.

https://www.youtube.com/watch?v=SsoSk_lp0Xs

https://www.youtube.com/watch?v=wPLTzjagdEA

Na záver odporúčame rozvinúť diskusiu o tom, prečo by bolo vhodné doplniť na stránku pizzerie video, akú má mať funkciu, aké má byť dlhé, aké veľké, v akom formáte a pod.

VÝKLAD

Táto kapitola je hlavne praktická. Študentov treba naučiť:

- ako vo video editore VSDC skombinovať dve a viac videí,
- odstrániť/potlačiť zvuk v zvukovej stope, príp. vložiť nový zvuk,
- rozpohybovať (rozanimovať) rôzne objekty vo videu,
- otitulkovať video,
- nájsť vhodný spôsob exportu videa pre požadované účely.

Nadväznosť na predchádzajúce kapitoly: Táto kapitola nadväzuje hlavne na kapitolu 19.

Pred absolvovaním kapitoly sa predpokladá, že sa študenti v rámci inovácií matematiky, informatiky, prírodovedných a odborných predmetov na ZŠ a SŠ už zoznámili so základmi práce s videom v tematickom celku Spracovanie videa.

Preto táto kapitola nadväzuje na tieto základy a venuje sa nadstavbovej práci s videom, ako je zobrazenie viacerých videí naraz, resp. vloženie jedného videa do druhého, odfiltrovaniu zvukovej stopy z pôvodného videa, ako aj pridávaniu vlastného zvuku, animovaného objektu a titulkov do videa.

ZHRNUTIE

Pri exporte videa je potrebné sa zamyslieť, čomu dať prednosť (kvalita verzus veľkosť) a optimalizovať výstupný súbor. Už pri strihu je potrebné odstrániť prázdne a nepotrebné časti a video zefektívniť ešte pred exportom. Výstupný formát v sebe zahŕňa komprimačné algoritmy a kodeky (budú vysvetlené neskôr), ktoré zmenšujú veľkosť pôvodného videa a ukladajú ho do prehrávateľnej formy.

22 MULTIMÉDIÁ NA WEBE

Prvé webové stránky obsahovali iba text a hypertextové odkazy, tzv. hypertext. Až neskôr sa na stránkach začali objavovať obrázky, zvukové efekty a ďalšie elementy, ktorými sa tvorcovia stránok snažili zatraktívniť web. S postupným zrýchľovaním internetového pripojenia sa kvalita týchto mediálnych elementov zlepšovala, čo umožnilo prenášať aj väčšie súbory – napríklad videá. Nevýhodou použitia mediálnych elementov v minulosti bol problém s ich spúšťaním vo webových prehliadačoch. Riešenie týchto problémov prišlo až vo forme zásuvných modulov (z angl. plugin), ktoré bolo potrebné doinštalovať do jednotlivých prehliadačov.

Zásuvný modul je externý program pre webové prehliadače, ktorý umožní prehrať multimediálny obsah. V prípade, že webová stránka obsahuje napr. video, ktoré vyžaduje zásuvný modul nainštalovaný v prehliadači, zobrazí sa upozornenie s odkazom na možnosť stiahnutia takéhoto zásuvného modulu. Najväčšou nevýhodou použitia zásuvného modulu je pomalá rýchlosť načítania stránok – stránky sú pomalšie, nakoľko webový prehliadač prenecháva prehrávanie obsahu na zásuvný modul. S príchodom značkovacieho jazyka HTML5 je možné používať multimediálne prvky priamo na webovej stránke a zásuvné moduly sa používajú už zriedkavejšie, resp. veľmi výnimočne.

V súčasnosti už nie je problém vložiť na webovú stránku obrovské multimediálne súbory, pričom sa už nemusíme obmedzovať vo veľkosti takýchto súborov. Pri vkladaní multimediálnych súborov na webové stránky máme na výber dve možnosti.

- Ak súbor nie je veľmi veľký, môžeme ho vložiť priamo na našu stránku. Túto možnosť preferujeme v prípade, ak nechceme byť závislí od iných stránok (vysielanie naživo - napr. YouTube).
- Ak je multimediálny súbor väčší a potrebujeme šetriť miesto kvôli obmedzenej veľkosti priestoru na serveri, môžeme využiť stránky tzv. tretích strán, na ktoré tieto súbory nahráme a na našich stránkach ich potom budeme prehrávať. Vtedy hovoríme o tzv. vysielaní naživo (streaming, z anglického výrazu stream – prúd) videa a zvuku (resp. audia).

Vysielanie naživo je technológia na prenášanie videa a zvuku medzi zdrojom a koncovým používateľom. Prenos môže prebiehať:

- on-line – priame vysielanie v reálnom čase (napr. YouTube videá, internetová televízia a rádio),
- off-line – prostredníctvom systému VoD (Video on Demand) – video na požiadanie, ktoré umožňuje používateľovi väčšinou za poplatok prehrávať video/audio súbory, akoby z virtuálnej videopožičovne, hocikedy bez obmedzenia.

V prípade vysielania naživo video/audio súborov je potrebné tieto súbory nahrať na špecializovaný server. Medzi najpopulárnejšie a najväčšie hostiteľské platformy, špecializované na video a audio súbory, patria YouTube a Vimeo.

Nahrávanie videa (z angl. upload), je prenos súboru z jedného počítačového systému (napr. používateľský počítač) do druhého systému, resp. webového servera. Druhý systém musí umožňovať prijímať súbory, pričom môže obsahovať určité obmedzenia, ako sú napríklad typ formátu, veľkosť súboru, prípadne dĺžka video/audio súborov. Obmedzenia sa dajú na väčšine webových serverov zrušiť zaplatením poplatku podľa služby, ktorú potrebujeme využívať.

Protikladom nahrávania videa je tzv. **sťahovanie** videa (z angl. download**)** sťahovanie súborov, resp. prenos dát zo vzdialeného systému (napr. z YouTube) na iný systém (napr. používateľský počítač). Z webu je možné stiahnuť rôzne druhy súborov, ako napr. dokumenty, obrázky, hudbu, videá, aplikácie a ďalšie doplnky pre prehliadač (pri sťahovaní videa však musíme dodržať licenčné podmienky). Všetky súbory stiahnuté z webu sa automaticky ukladajú do priečinka `Stiahnuté súbory` prostredníctvom tzv. `Správcu sťahovania.`

Pri prenose multimediálnych súborov na webe sa môžeme stretnúť ešte s jedným pojmom, ktorému by sme mali rozumieť. Webové stránky často obsahujú multimediálne elementy v rôznych typoch a formátoch. Pre prehrávanie videosúborov je nutné mať nainštalovaný kodek**.** **Kodek** (odvodené od kóder/dekóder) je vlastne algoritmus, ktorý dokáže video zakódovať do určitého formátu a pomocou dekóderu aj prehrať. Úlohou kodeku je komprimovať a dekomprimovať digitálny prúd dát tak, aby bola zachovaná, čo najväčšia kvalita obrazu a zvuku videa pri čo najmenšej veľkosti súboru. Existuje množstvo kodekov, ktoré súperia medzi sebou v čo najlepšom prispôsobení sa potrebám zákazníka. V súčasnosti webové prehliadače už majú nainštalované najpoužívanejšie kodeky a teda používateľ už nemusí nič sťahovať a inštalovať.

Obrázok 22.1 Logá najpoužívanejších kodekov.

DISKUTUJTE

Aké máte skúsenosti s používaním multimédií na webe? Na aké problémy ste narazili pri sledovaní tzv. vysielania naživo (z angl. streaming videa, audia) na webe? Máte už nejaké vlastné video, nahraté na webe?

Aké video a audio súbory sa najčastejšie vysielajú naživo? Aké multimediálne súbory by ste nemali nahrávať na web?

Prečo ľudia nahrávajú multimédia na web (napr. na YouTube)? Majú z toho nejaký profit?

Zvuk na webe

Web ponúka v súčasnosti rôzne možnosti využitia zvuku. Zvuk môžeme zakomponovať na webové stránky ako súbor na stiahnutie, prehrať ho na pozadí, alebo ho nechať zaznieť pri rôznych udalostiach. Avšak použitie zvuku je treba robiť uvážene, aby sme neobťažovali používateľa zbytočným počúvaním niečoho, na čo nemá náladu. Najčastejšie sa môžeme stretnúť s **prehrávaním hudby a podcastov,** čo sú zvukové záznamy na webe, ktoré sa dajú ľahko stiahnuť na smartfón či iné zariadenie. Podcasty často fungujú ako pravidelné rozhlasové, televízne, či internetové relácie.

POZNÁMKA
Slovo podcast vzniklo v roku 2004, spojením názvu prehrávača iPod od firmy Apple a anglického slova broadcasting (vysielanie).

Ďalšou veľmi užitočnou voľbou je predčítavanie obsahu stránok ľuďom, ktorí popri počúvaní robia inú činnosť, alebo aj nevidiacim. Umelá produkcia reči sa v súčasnosti stáva dôležitou zložkou komunikácie medzi človekom a počítačom, čo sa odzrkadľuje aj na webových rozhraniach.

Zvukové súbory existujú v rôznych zvukových formátoch. Každý formát má svoje výhody a nevýhody. Niektoré formáty preferujú veľkosť súboru pred kvalitou, prípadne naopak.

Audio formáty

Pri vytváraní audio súborov máme k dispozícii množstvo audio formátov. Každý formát má svoje výhody a nevýhody. V tejto kapitole uvedieme najpoužívanejšie audio formáty, s ktorými sa môžeme stretnúť nie len na webe.

MIDI (.mid)*

Skratka MIDI označuje digitálne rozhranie hudobnej elektroniky (Musical Instrument Digital Interface). Je to medzinárodný štandard na prepojenie elektronických hudobných nástrojov, počítačov a iných prístrojov. Tento formát je určený pre profesionálnych hudobníkov a slúži na generovanie zvukov rôznych nástrojov, na komunikáciu medzi nástrojmi a počítačom a na prevod notového zápisu do zvukovej podoby.

RealAudio (.ra, *.rm, *.ram)*

Patrí medzi najstaršie formáty, pričom je to prvý formát, ktorý sa použil na vysielanie zvukového súboru naživo. Má vysoký kompresný pomer, ktorý je ale vyvážený nízkou kvalitou zvuku.

WMA (.wma)*

Windows Media Audio je komprimovaný zvukový formát pôvodne vyvinutý ako súčasť prehrávača Windows Media Player. V súčasnosti sa už skoro nepoužíva.

WAV (.wav)*

Waveform Audio, alebo tiež WAVE je zvukový formát vytvorený spoločnosťami Microsoft a IBM na ukladanie zvuku v počítačoch. Tento zvukový formát patrí medzi najrozšírenejšie zvukové formáty. Nevýhodou tohto formátu je jeho veľkosť, nakoľko sa údaje zaznamenávajú v „plnej" kvalite. Formát WAV sa často používa na záznam zvuku, ktorý sa bude ešte ďalej spracovávať.

Ogg (.ogg)*

Ogg je formát súboru pre ukladanie zvukových záznamov (resp. aj multimediálnych dát) založený na stratovej kompresii, ktorá spočíva vo vypúšťaní signálov s vyššími frekvenciami zo záznamu. Medzi hlavné výhody použitia tohto formátu patrí bezplatná licencia a široká podpora v prehliadačoch.

MP3 (.mp3)*

MP3 je digitálny formát stratovej kompresie dát (v celom znení MPEG Layer 3 – Moving Pictures Experts Group), ktorý umožňuje značne zmenšiť veľkosť hudobných súborov pri zanedbateľnom, či dokonca nepostrehnuteľnom poklese kvality. Okrem zvuku samotného môžu byť do súboru MP3 vložené aj sprievodné informácie. Pri prehrávaní sa tak môžeme dozvedieť napr. názov skladby, meno interpreta, rok vydania a pod.

MP4 (.mp4)*

MP4 je formát digitálneho súboru, ktorý vzniká kompresiou dát na menšiu veľkosť a slúži na uchovávanie a prehrávanie audio a video súborov, môže obsahovať ponuku (menu), viacej titulkov, zvukových stôp, obrázkov, dokonca aj 3D objekty. Je to najpoužívanejší formát pre vysielanie naživo na internete.

Audio formáty v HTML5

Zvukové súbory môžeme na webových stránkach použiť v rôznych formátoch. HTML5 štandardne podporuje formáty MP3, WAV a OGG.

Tabuľka 22.1 Prehľad podporovaných audio formátov v jednotlivých prehliadačoch.

	MP3	WAV	Ogg
Internet Explorer	*áno*	*nie*	*nie*
Chrome	*áno*	*áno*	*áno*
Firefox	*áno*	*áno*	*áno*
Safari	*áno*	*áno*	*nie*
Opera	*áno*	*áno*	*áno*

Element audio

Audio element vkladáme na webovú stránku v jazyku HTML5 pomocou párovej značky `<audio></audio>`. Zdrojový súbor audio elementov uvedieme v atribúte `src`.

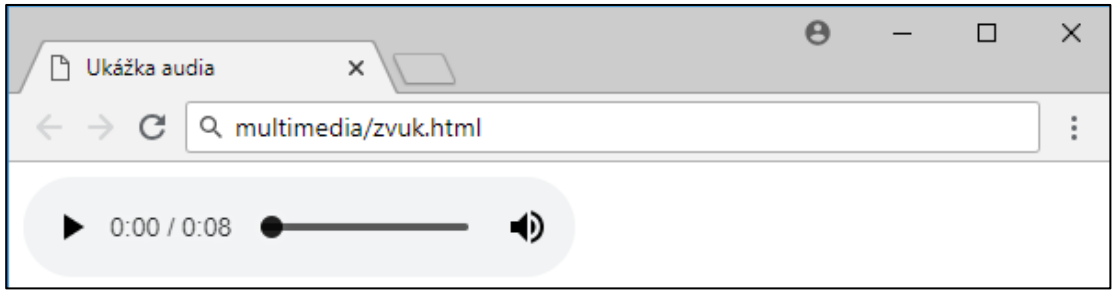
```
<audio src="... "></ audio>
```

PRÍKLAD 22.1

22/01_audio.html

Vytvorme webovú stránku pomocou značkovacieho jazyka HTML5, ktorá bude obsahovať zvuk – výrobu hamburgeru. Hudobný prehliadač musí obsahovať ovládacie prvky pre prehratie/pozastavenie audio súboru, vypnutie/zapnutie zvuku.

V prvom kroku musíme vytvoriť základnú kostru HTML stránky, ktorá bude obsahovať iba značky `<html>`, `<head>` a `<body>`. V ďalšom kroku vložíme značku `<audio>` do kódu našej vytvorenej stránky. Do značky `<audio>` vložíme atribút `src` s názvom nášho súboru `audio.mp3`, ktorý je v audio formáte MP3. Pomocou druhého parametra `type` určíme typ súboru, ktorý sa bude prehrávať – v prípade formátu MP3 je to `audio/mp3`. Pomocou globálneho atribútu `title` môžeme uviesť dodatočné informácie o elemente. Posledný atribút, ktorý vložíme do značky <audio> je `controls`, ktorý nám zobrazí ovládacie prvky na prehrávanie hudby.



Obrázok 22.2 Zvuk na stránke.

```
<!DOCTYPE html>
<html>
<head>
      <title>Ukážka audia</title>
</head>
<body>

 <audio src="audio.mp3" type="audio/mp3" title="zvuk výroby
hamburgeru" controls >
 </audio>

</body>
</html>
```

Atribúty elementu audio

Tabuľka 22.2 Tabuľka atribútov elementu audio.

Atribút	Popis
src	*definuje url adresu k audio súboru*
autoplay	*automatické spustenie audio súboru*
controls	*doplní sa sada ovládacích prvkov*
muted	*vypnutý zvuk audio súboru*
loop	*audio sa bude prehrávať donekonečna*
preload	*„none“ - audio bude načítané až po reakcii používateľa*

22/02_audio2.html

PRÍKLAD 22.2

Upravme webovú stránku s hudobným prehrávačom z príkladu 22.1 tak, aby prehrávač prehral najskôr hudobný formát OGG. Ak webový prehliadač nepodporuje tento formát, tak ako alternatívu prehrá hudobný formát MP3. V prípade, že nepodporuje ani formát MP3, tak vypíše text „Webový prehliadač nepodporuje element audio“.

Pri vkladaní audio súborov na webové stránky, môžeme použiť rôzne audio formáty. V tabuľke 22.1 sme uviedli podporu audio formátov v jednotlivých prehliadačoch. Aby sme zabezpečili kompatibilitu prehrávania na rôznych prehliadačoch, môžeme špecifikovať alternatívu súboru, ktorý sa spustí v prípade, že prehliadač nepodporuje daný audio formát.

```
<!DOCTYPE html>
<html>
<head>
        <title>Ukážka audia</title>
</head>
<body>

 <audio controls>
  <source src="audio.ogg" type="audio/ogg">
  <source src="audio.mp3" type="audio/mp3">
  <p> Webový prehliadač nepodporuje element audio.</p>
 </audio>

</body>
</html>
```

Do kódu stránky z predchádzajúceho príkladu doplníme dve nepárové značky `<source>`. Do prvej značky vložíme atribút `src` s názvom súboru `audio.ogg` a atribút `type` s hodnotou `audio/ogg`. V druhej značke `<source>` nastavíme atribút `src` na `audio.mp3` a typ na `audio/mp3`. Nakoniec do párovej značky `<audio>` vložíme odsek s textom, ktorý sa zobrazí

v prípade, že webový prehliadač nepodporuje značku `<audio>`. Z kódu musíme ešte odstrániť zo značky `<audio>` atribúty `src` a `type`, nakoľko tieto informácie sme uviedli v značkách `<source>`.

ÚLOHA 22.3

Vytvorte vlastnú webovú stránku, ktorá bude prehrávať ľubovoľný zvuk (alebo hudbu). V kóde stránky použite značky `<audio>` a `<source>`. Hudobnému prehrávaču nastavte automatické spustenie zvuku po načítaní stránky.

Video na webe

Správy, televízne programy, náučné videotutoriály, názorné návody a triky, filmy rôzneho druhu, videoklipy a iné zábavné videá, to je dnes samozrejmou súčasťou webu. Tieto dynamické formy médií nám dokážu sprostredkovať hlavnú myšlienku oveľa rýchlejšie a pútavejšie v porovnaní so statickými textovými informáciami. Video môžeme zakomponovať na stránke rôznymi spôsobmi. Najčastejšie sa môžeme stretnúť s jednoduchým nahratím videa na stránku po prihlásení sa do niektorej zo služieb (napr. YouTube, Vimeo, Facebook, Instagram, TedX,...) a následným potvrdením publikovania a zdieľania. Ďalšou možnosťou je skopírovanie URL adresy do samotnej stránky, alebo vloženie videa do stránky pomocou vnoreného kódu, tzv. „embed kódu“**.** Video môžeme zo stránky aj stiahnuť a uložiť ho na svoj počítač, alebo iné mobilné zariadenie, alebo ho vysielať naživo. Niektoré zo spomenutých možností si ukážeme v rámci tejto kapitoly. Najskôr sa ale zoznámme s najčastejšie používanými video formátmi.

Video formáty

Pri vytváraní video súborov máme k dispozícii viacero video formátov. Každý formát má svoje výhody a nevýhody. Medzi najpoužívanejšie video formáty v súčasnosti zaraďujeme nasledujúce.

MPEG (.mpg, *.mpeg)*

MPEG je jeden z prvých video formátov umožňujúcich spúšťať video na internete, ktorý využíva tzv. stratovú kompresiu (názov je odvodený z angličtiny Moving Picture Experts Group, podobne ako pri spomínaných formátoch MP3 a MP4). Tento formát bol podporovaný všetkými prehliadačmi. V súčasnosti ho však nepodporuje značkovací jazyk HTML5.

POZNÁMKA

Videosúbory môžeme transformovať do iných formátov pomocou viacerých nástrojov, napr. Micro Video Converter, HandBrake, atď.

AVI (.avi)*

Názov formátu je odvodený z angličtiny Audio Video Interleave. V minulosti patril medzi najpoužívanejšie formáty. Webové prehliadače v súčasnosti nepodporujú tento video formát.

WMV (.wmv)*

Windows Media Video je komprimovaný súborový video formát pôvodne vyvinutý spoločnosťou Microsoft, ako súčasť prehrávača Windows Media Player. Tento formát nie je možné prehrávať vo webových prehliadačoch, z tohto dôvodu sa v súčasnosti už skoro nepoužíva.

Flash (.swf, *.flv)*

Flash je formát, pomocou ktorého môžeme vytvárať vektorové a bitmapové operácie, obohatené o zvukové efekty. Hlavnou nevýhodou tohto formátu je nutnosť mať nainštalované rozšírenia pre webové prehliadače.

Ogg (.ogg)*

Ogg je už spomínaný formát pre zvukové, resp. multimediálne dáta, ktorý umožňuje ukladať alebo streamovať mulitmediálny obsah vo vysokej kvalite. Formát môže spájať množstvo nezávislých tokov zvuku, videa, textu (napríklad titulkov) a ďalších informácií.

WebM (.webm)*

WebM je formát video súboru, ktorý je určený pre použitie so značkovacím jazykom HTML5. Vytvorili ho spoločnosti Mozilla, Opera, Adobe a Google.

MP4 (.mp4)*

MP4 je formát digitálneho súboru, ktorý vzniká kompresiou dát na menšiu veľkosť. Do formátu MP4 je možné ukladať skoro ľubovoľný obsah (video, zvuk, textové titulky, prípadne obrázky) a je možné ho vysielať naživo aj na webe. Tento formát sa v súčasnosti používa vo väčšine novších kamier a televízorov.

Vkladanie videa do webovej stránky

Oproti zásuvným modulom umožňuje HTML5 vkladať multimediálne prvky priamo do kódu, pomocou novo-vzniknutých značiek. Vývojári pridali túto podporu, aby eliminovali nasledujúce nevýhody zásuvných modulov:

- rýchlosť – vloženie multimediálnych prvkov do HTML5 je podstatne rýchlejšie,
- dostupnosť – netreba sa spoliehať na dostupnosť zásuvného modulu (či má používateľ nainštalovaný daný modul alebo nie).

Jazyk HTML5 podporuje video formáty MP4, WebM a OGG. Tieto formáty patria medzi HTML5 štandardy a v prípade ich použitia na stránkach nie je nutné inštalovať ďalšie kodeky. Nie všetky webové prehliadače však podporujú všetky tri video formáty:

Tabuľka 22. 3 Podpora videoformátov vo webových prehliadačoch.

	MP4	WebM	OGG
Internet Explorer	áno	*nie*	*nie*
Chrome	*áno*	*áno*	*áno*
Firefox	*áno*	*áno*	*áno*
Safari	*áno*	*nie*	*nie*
Opera	*áno*	*áno*	*áno*

Keď chceme vložiť videosúbor priamo na webovú stránku, musíme použiť nasledujúce HTML5 elementy:

- <video>
- <audio>
- <source>

POZNÁMKA

V jazyku HTML existuje ešte značka `<object>`, pomocou ktorej môžeme vkladať video vo formáte Flash.

Element video

Aby sme mohli pridať video do kódu našej webovej stránky v jazyku HTML5, musíme použiť párovú značku `<video>`. Video súbor, ktorý sa má zobraziť, uvedieme pomocou atribútu `src`.

```
<video src="... "></video>
```

PRÍKLAD 22.4

22/03_video.html

Pomocou značkovacieho jazyka HTML5 vytvorme stránku, ktorá bude prehrávať video – animáciu vytvárania hamburgerov. Video prehliadač musí obsahovať ovládacie prvky pre prehratie/pozastavenie video súboru, vypnutie/zapnutie zvuku a bude mať veľkosť 600x400px.

V prvom kroku musíme vytvoriť základnú kostru HTML dokumentu, ktorá bude obsahovať iba párové značky `<html>, <head>, <body>`. V ďalšom kroku vložíme značku `<video>` do kódu našej vytvorenej stránky. Do atribútu `src` značky `<video>` vložíme názov nášho súboru `mcdonald.mp4`, ktorý je vo video formáte MP4. Okrem atribútu `src` ešte uvedieme atribúty `width` a `height`, pomocou ktorých nastavíme šírku a výšku obrazu s videom. Nakoniec v značke `<video>` napíšeme posledný atribút `controls`, ktorý doplní do videa ovládaciu sadu prvkov.

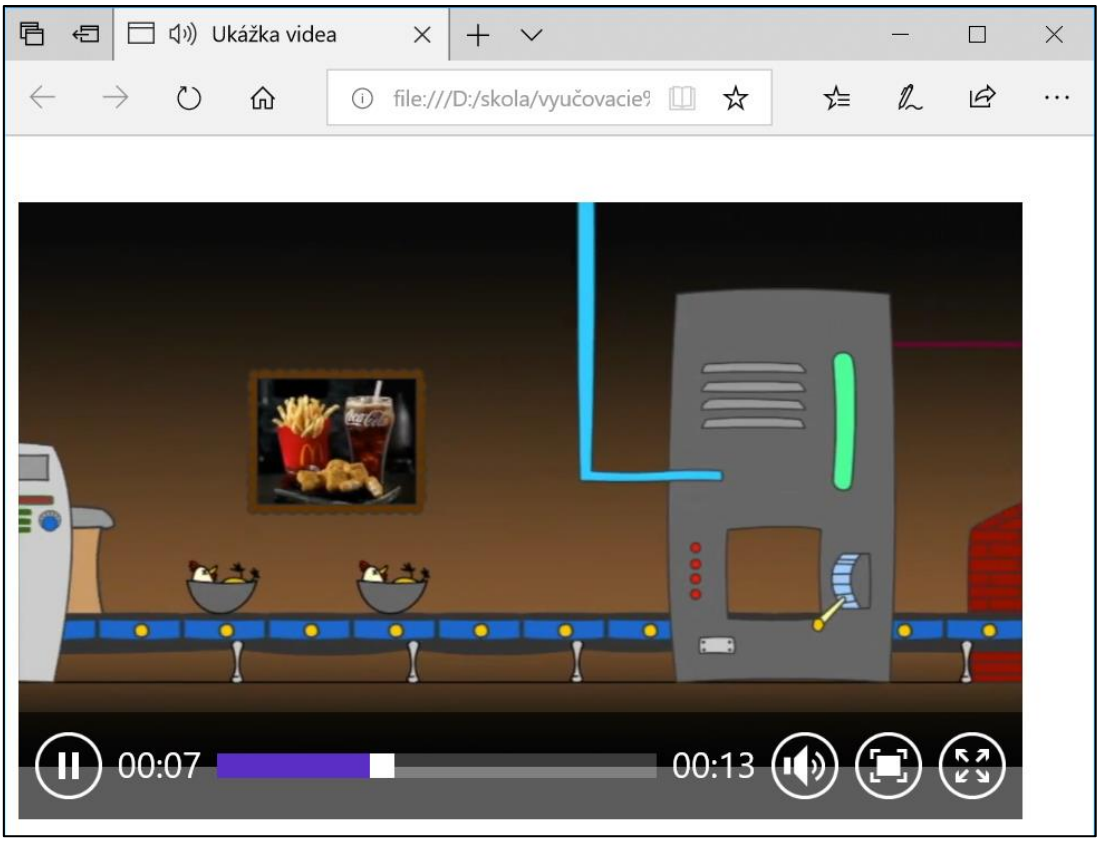
POZNÁMKA

Atribút `controls` patrí medzi atribúty s pravdivostnou hodnotou, ktorý môže mať iba hodnotu `controls="controls"`, a teda nemusíme uvádzať túto hodnotu.

```
<!DOCTYPE html>
<html>
<head>
        <title>Ukážka videa</title>
</head>
<body>

<video src="mcdonald.mp4" width="600" height="400" controls>
</video>

</body>
</html>
```



Obrázok 22.3 Video na webovej stránke.

Atribúty elementu video

V tabuľke 22.4 sú uvedené všetky atribúty, ktoré môže element `<video>` obsahovať.

Tabuľka 22.4 Atribúty elementu video.

Atribút	Popis
src	definuje URL adresu k videosúboru
autoplay	*automatické spustenie videa*
controls	*doplní sa sada ovládacích prvkov*
muted	*vypnutý zvuk videa*
loop	*video sa bude prehrávať donekonečna*
poster	*názov súboru, ktorý sa zobrazí kým sa načíta video*
width	*šírka video v px*
height	*výška videa v px*
preload	*„none“ - Video bude načítané až po reakcii používateľa*

Element source

V úvode kapitoly 22.2.1 sme uviedli video formáty, ktoré jazyk HTML5 štandardne podporuje. V tabuľke 22.3 sme uviedli, ktoré prehliadače podporujú jednotlivé formáty. Čo v prípade, ak chceme použiť formát, ktorý nepodporujú všetky prehliadače? Z tohto dôvodu vznikla v HTML5 značka `<source>`, ktorá umožňuje uviesť rôzne alternatívy použitia video formátov. Tento element môžeme vložiť do elementu `<video>` v ľubovoľnom počte.

POZNÁMKA

Kvôli podpore starších prehliadačov sa odporúča za posledným elementom `<source>` vždy vložiť aj text, ktorý sa zobrazí v prípade, že sa nepodarilo načítať súbory.

```
<video>
  <source src="film.ogg" type="video/ogg">
  <source src="film.webm" type="video/webm">
  <source src="film.mp4" type="video/mp4">
  <p>Webový prehliadač nepodporuje element video</p>
</video>
```

Prehliadač sa pri čítaní HTML kódu „pozrie“, či je definovaný atribút `src` v značke `<video>` a ak nie je, tak sa „pozrie“ do značky `<source>`. V prípade, že vie špecifikovaný formát prehrať, tak ho prehrá a zvyšok kódu v elemente `<video>` ignoruje. Ak nevie prehrať daný formát a sú

definované ďalšie alternatívne formáty, tak webový prehliadač skúsi prehrať ďalší formát. V prípade, že nie sú definované alternatívne formáty, je vhodné uviesť text, ktorý sa zobrazí. Takýto text sa môže vypísať napr. v značkách pre odsek.

Atribúty elementu source

Podobne ako značka `<video>`, môže aj značka `<source>` obsahovať viacero atribútov.

Tabuľka 22.5 Atribúty elementu source.

Atribút	**Popis**
src	*url adresa*
type	*špecifikujeme typ videa*
media	*viacero zdrojov videa s rôznym rozlíšením*

ÚLOHA 22.5

Vložte do vami vytvorenej webovej stránky (z úlohy 22.3) ľubovoľné video. Na stránke použite značky `<video>` a `<source>`. Video prehrávaču nastavte automatické spustenie videa po načítaní stránky.

Sťahovanie audio a video súborov

Sťahovanie audio a video súborov z webových stránok, ako je napríklad YouTube, je možné viacerými spôsobmi. Prvou možnosťou je inštalácia špeciálneho programu zameraného na sťahovanie audio a video súborov (napr. YDT Video Downloader[5], aTube Catcher[6]). Tieto programy väčšinou umožňujú aj ďalšiu prácu s videami (napr. prevod na iný video formát, atď.). Ďalšou možnosťou je nainštalovať si doplnky do webového prehliadača (napr. Video DownloadHelper[7], Easy YouTube Video Downloader[8]., atď.).

[5] http://www.ytddownloader.com/macosx/
[6] https://www.atube.me
[7] https://video-downloadhelper.en.softonic.com
[8] https://addons.mozilla.org/sk/firefox/addon/easy-youtube-video-download/

PRÍKLAD 22.6

Zo stránky YouTube stiahnime video McDonald. Na stiahnutie videa nechceme inštalovať žiadny špeciálny program určený na sťahovanie takýchto videí. URL adresa videa pre sťahovanie je:

https://www.youtube.com/watch?v=2uWlwc-liYY&feature=youtu.be

Na stiahnutie videa dostupného na adrese

https://www.youtube.com/watch?v=2uWlwc-liYY&feature=youtu.be

musíme upraviť URL adresu tak, že vložíme písmená „ss“ za www. a vznikne nám nasledujúca adresa

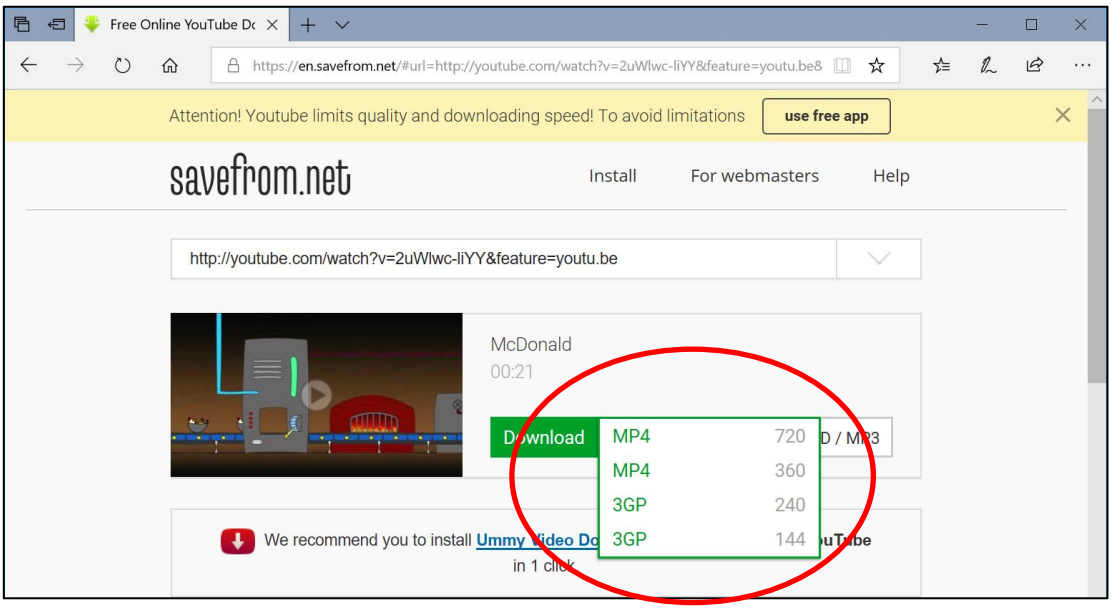
https://www.ssyoutube.com/watch?v=2uWlwc-liYY&feature=youtu.be

Obrázok 22.4 YouTube s url adresou na stiahnutie videa.

Po načítaní upravenej stránky máme možnosť si vybrať v akom formáte chceme stiahnuť zvolené video. V našom prípade si vyberieme formát MP4 s kvalitou 720p.

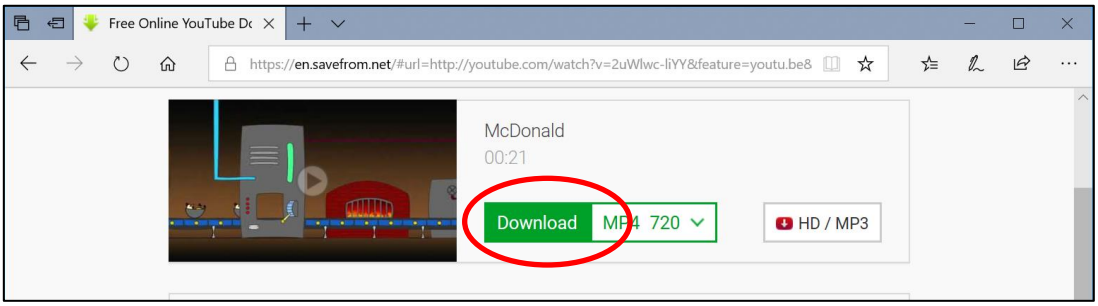
POZNÁMKA

Dostupná kvalita videa závisí od toho, v akej kvalite ho nahral majiteľ videa. Kvalita 720p predstavuje rozlíšenie 1280x720px, často sa označuje aj HD Ready.



Obrázok 22.5 YouTube – výber kvality videa.

Po zvolení konkrétneho formátu stlačíme tlačidlo `Download` a počkáme, kým sa video stiahne do nášho počítača.



Obrázok 22.6 YouTube – stiahnutie videa.

ÚLOHA 22.7

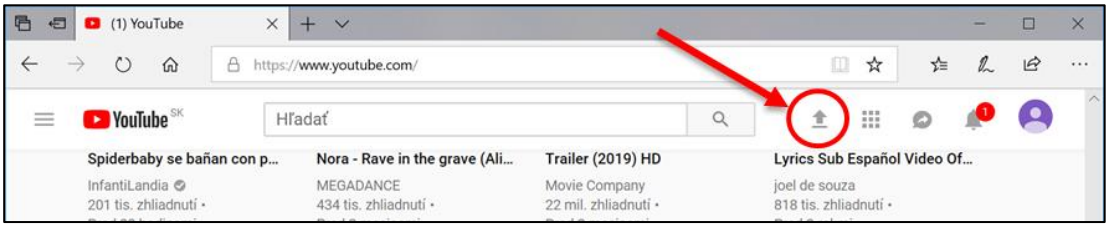
Stiahnite si do počítača ľubovoľný audio/video súbor zo stránky YouTube.

Nahrávanie audio a video súborov

PRÍKLAD 22.8

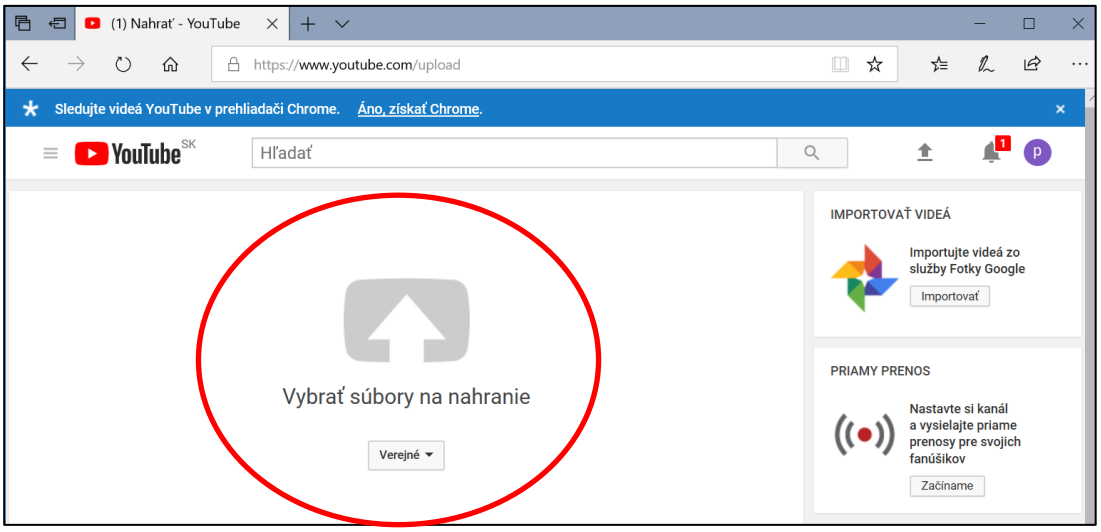
Nahrajme nejaké video, ktoré sme vytvorili v rámci predchádzajúcich hodín, na stránku YouTube.

Pri nahrávaní audio a video súborov na stránku YouTube musíme byť prihlásení. Na prihlásenie môžeme použiť napr. Google účet (rovnaký účet, ako pri emailovom klientovi Gmail). Po prihlásení sa nám zobrazí napravo od vyhľadávacieho tlačidla ikonka na nahrávanie súborov.



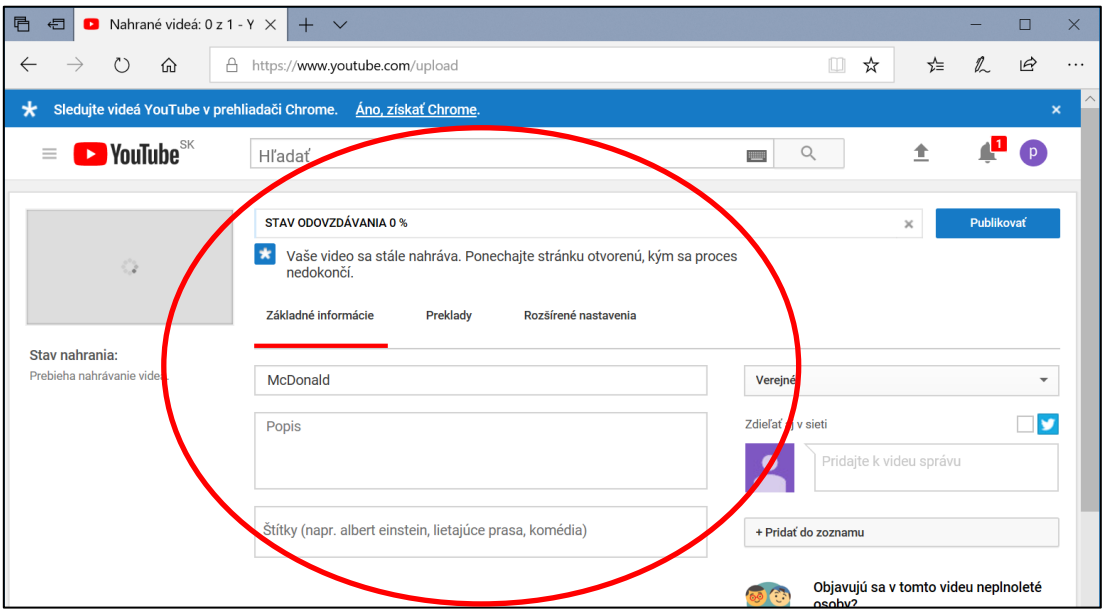
Obrázok 22.7 YouTube – nahrávanie videa.

Po stlačení tohto tlačidla-ikonky sa zobrazí stránka pre nahratie súborov. Súbory nahráme kliknutím na obrázok `Vybrať súbory na nahranie`, alebo presunieme súbory, ktoré chceme nahrať myškou do tohto okna - operácia `potiahni a pusť` (z angl. `drag-and-drop`).



Obrázok 22.8 YouTube – výber súboru na nahratie.

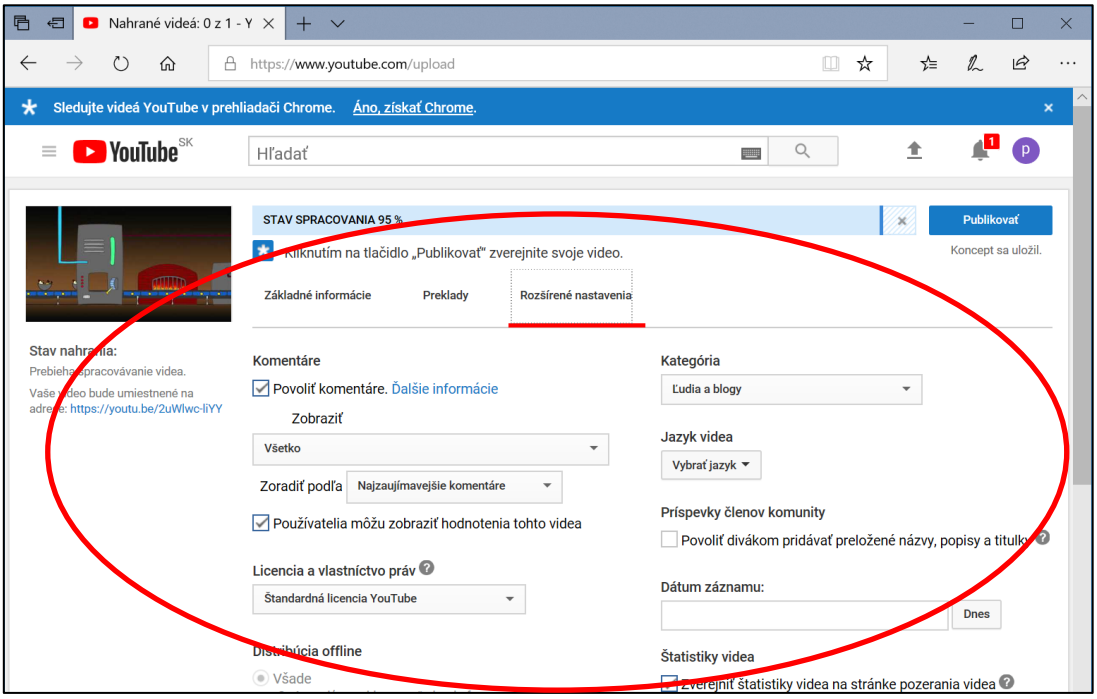
Po nahraní súborov musíme ešte zadať základné informácie o nahranom súbore – povinná položka je minimálne názov. Môžeme a nemusíme špecifikovať ďalšie položky – opis videa, spôsob zobrazenia (súkromné/verejné video). Okrem základných informácií máme možnosť kliknúť na záložku `Preklady` a `Rozšírené nastavenia`. Záložka `Preklady` umožňuje nastaviť iný názov a popis nahraných súborov v iných jazykoch.



Obrázok 22.9 YouTube – informácie o nahrávanom videu.

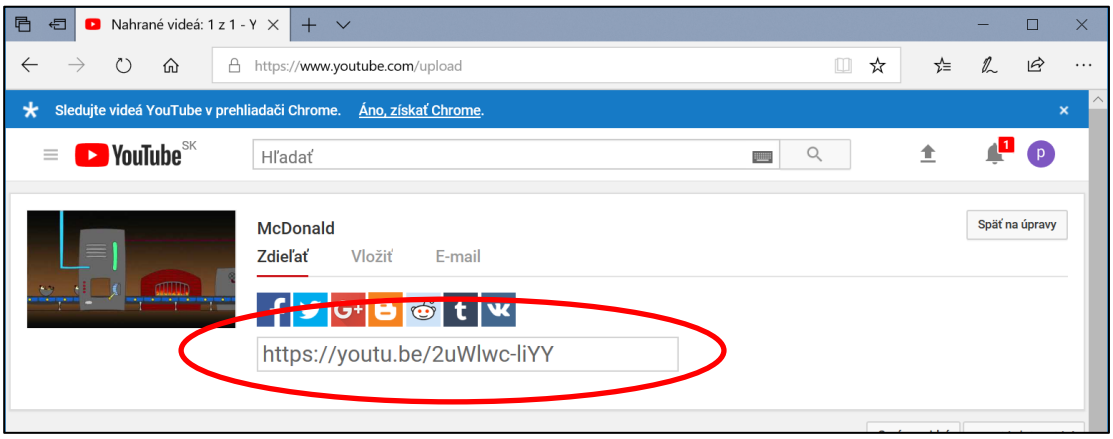
V záložke `Rozšírené nastavenia` máme možnosť napr. povoliť/zakázať komentáre, nastaviť licenciu súborov.

Po nastavení informácií o video súbore musíme ešte stlačiť modré tlačidlo `Publikovať`, aby sa naše video zobrazilo na stránke YouTube.

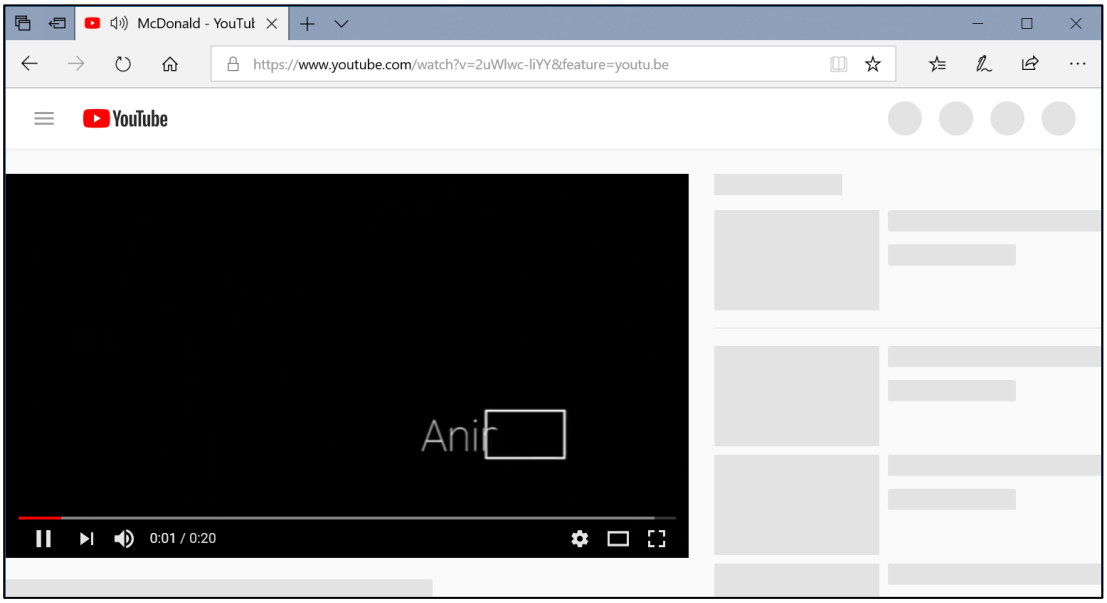


Obrázok 22.10 YouTube – informácie o nahrávanom videu – rozšírené nastavenia.

Po publikovaní videa sa zobrazí stránka s URL adresou nášho nahraného videa, ktorú môžeme ďalej zdieľať.



Obrázok 22.11 YouTube – odkaz na stiahnutie nahraného videa.



Obrázok 22.12 YouTube – prehratie nahraného videa.

ÚLOHA 22.9

Nahrajte na stránku YouTube ľubovoľný audio/video súbor. Pri nahrávaní súboru zadajte názov a opis súboru, zdieľanie súboru nastavte na typ verejné a povoľte možnosť vkladania komentárov.

Vysielanie videa naživo

Sídlo YouTube alebo podobné služby (dailymotion, atď.) umožňujú nahrávanie videosúborov na ich server. Tieto videá môžu byť prehrávané priamo na ich stránke, alebo ich môžeme vložiť na naše stránky pomocou HTML elementu `<iframe>`.

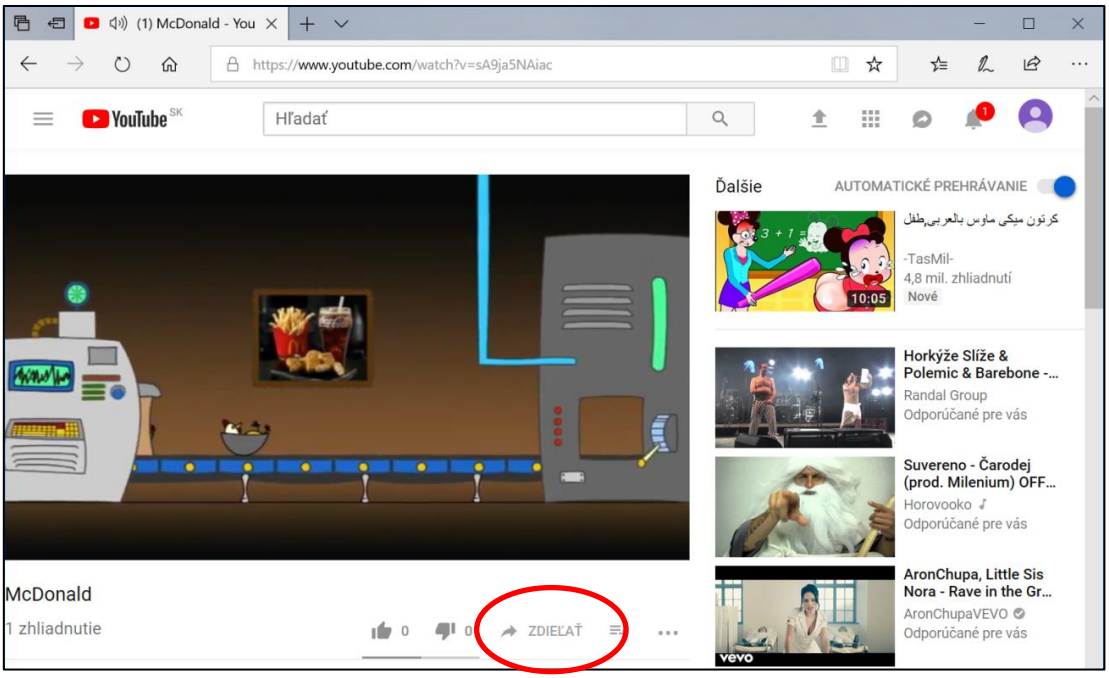
22/06_vysielanie_nazivo.html

PRÍKLAD 22.10

Pomocou jazyka HTML5 vytvorme webovú stránku, na ktorej použijeme video z nasledujúcej URL adresy (video vysielajme naživo zo stránky YouTube).

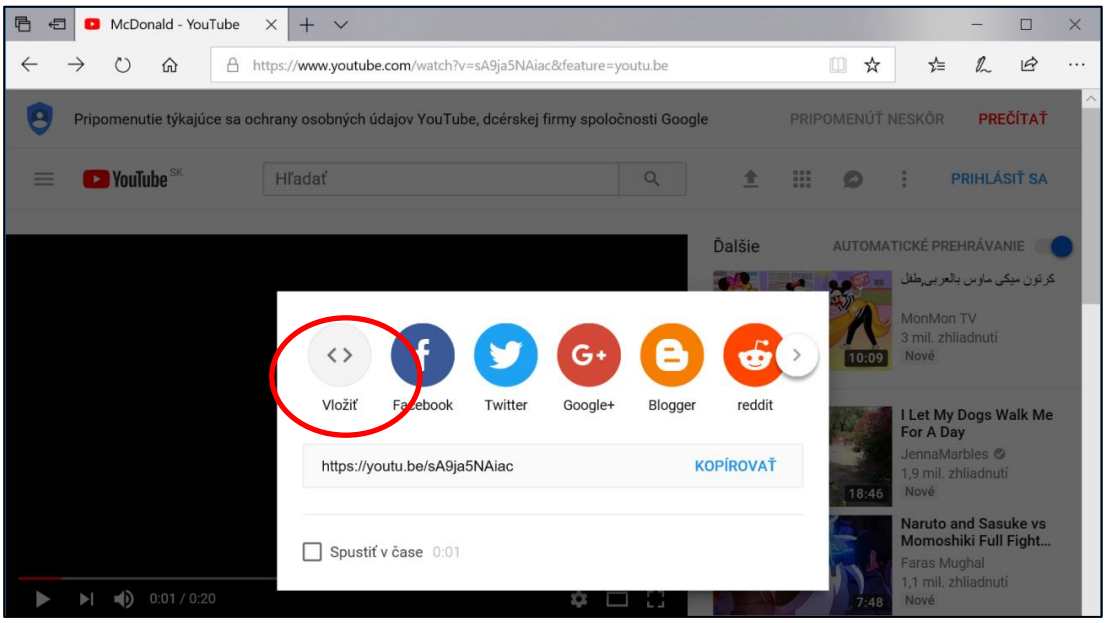
https://www.youtube.com/watch?v=2uWlwc-liYY&feature=youtu.be

Najprv musíme vo webovom prehliadači zobraziť stránku YouTube s videom, ktoré chceme na našej stránke vysielať. Následne stlačíme tlačidlo `Zdieľať`, ktoré sa nachádza pod zobrazeným videom.



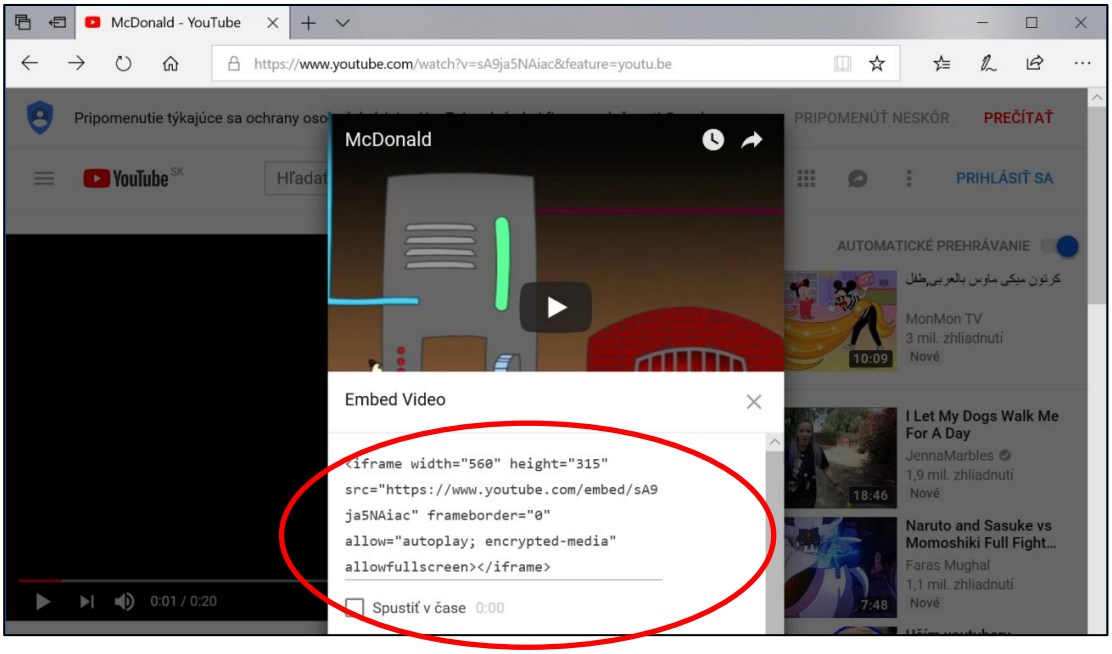
Obrázok 22.13 YouTube – zdieľanie videa.

Po stlačení tlačidla `Zdieľať` sa zobrazí dialógové okno, v ktorom máme možnosť vybrať si, ako chceme zdieľať video.



Obrázok 22.14 YouTube – zdieľanie – videa zobrazenie dialógového okna.

Aby sme ho mohli vložiť na našu stránku, musíme stlačiť tlačidlo `Vložiť`. Po stlačení tohto tlačidla sa zobrazí ďalšie dialógové okno s vygenerovaným kódom, ktorý vložíme do kódu našej stránky.



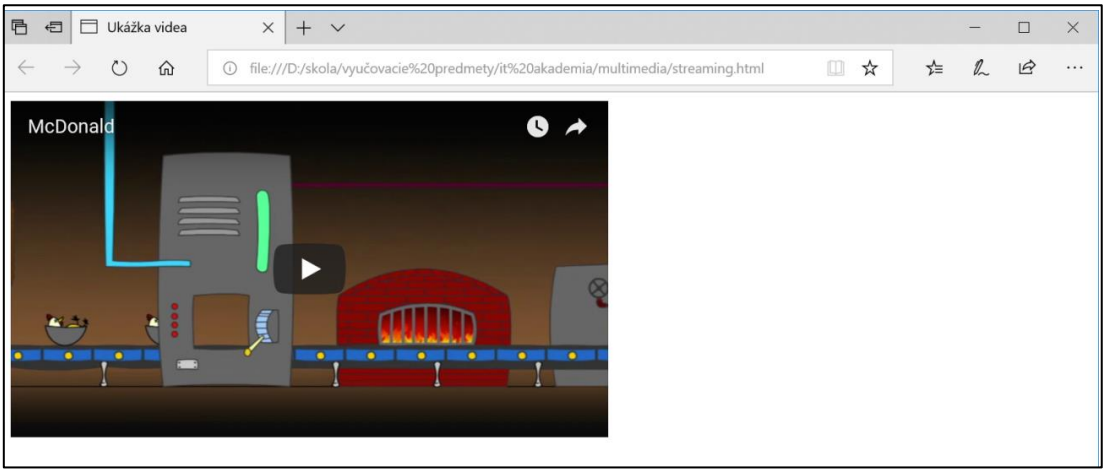
Obrázok 22.15 YouTube – zdrojový HTML kód na vloženie videa na stránku.

Konkrétne, ak by sme chceli použiť na našej stránke video zo stránky YouTube, ako sa vytvára McDonald hamburger, museli by sme vložiť nasledujúci (vygenerovaný) kód do kódu našej stránky.

```
<!DOCTYPE html>
<html>
<head>
	<title>Ukážka videa</title>
</head>
<body>

<iframe width="560" height="315"
src="https://www.youtube.com/embed/2uWlwc-liYY" frameborder="0"
allow="autoplay; encrypted-media" allowfullscreen></iframe>

</body>
</html>
```



Obrázok 22.16 Vložené video z YouTube na našej webovej stránke.

ÚLOHA 22.11

Do Vami vytvorenej webovej stránky, vložte naživo vysielané ľubovoľné video zo sídla YouTube.

Autorské práva

Každý multimediálny súbor, ktorý sa nachádza na internete (a nielen tam), musel niekto vytvoriť. Autor, resp. vydavateľ takéhoto súboru má autorské právo na tento súbor. Môže sa rozhodnúť, či ho niekde zverejní, alebo poskytne právo na jeho používanie ďalším osobám, firmám. Pod autorské právo spadajú:

- audiovizuálne diela (filmy, videá nachádzajúce sa na internete, televízne relácie),
- zvukové nahrávky a hudobné skladby,
- počítačový softvér,
- vizuálne diela (obrazy, reklamy),
- písomné diela (články, knihy),
- dramatické diela (divadelné hry, muzikály).

DISKUTUJTE

- Keby ste chceli použiť dielo chránené autorským právom, čo by ste museli urobiť, aby ste ho mohli použiť (napr. použiť cudziu hudobnú skladbu vo vami vytvorenom video súbore)?

Použitie materiálu chráneného autorskými právami bez povolenia vlastníka autorských práv

Každý štát má na základe Bernskej dohody možnosť obmedziť autorské práva k vytvoreným dielam. Tieto obmedzenia však nesmú narúšať normálne využívanie diela a nesmú spôsobovať neospravedlniteľnú ujmu. Takejto možnosti sa hovorí **čestné používanie** (inak povedané aj rozumné, spravodlivé používanie, v USA sa takejto legislatíve autorského práva hovorí „fair use“, v Spojenom kráľovstve „fair dealing“).

Zákony na Slovensku pojem čestné používanie neobsahujú. Pri použití diela sa musí dodržiavať **autorský zákon** 618/2003 Z.z. v znení neskorších predpisov (§ 25 Citácia diela):

„Bez súhlasu autora možno použiť krátku časť zverejneného diela vo forme citácie v inom diele len na účel recenzie alebo kritiky tohto zverejneného diela alebo na vyučovacie účely, vedeckovýskumné účely alebo umelecké účely. Takéto použitie musí byť v súlade so zvyklosťami a jeho rozsah nesmie presiahnuť rámec odôvodnený účelom citácie. Pri citácii sa musí uviesť meno autora alebo jeho pseudonym, ak nejde o anonymné dielo, alebo meno osoby, pod ktorej menom sa dielo uvádza na verejnosti, ako aj názov diela a prameň. Za takéto použitie nevzniká povinnosť uhradiť autorovi odmenu.“

POZNÁMKA

Podľa americkej legislatívy je možné použiť čestné používanie „fair use„ v prípade, že sa jedná o kritiku, komentáre, spravodajstvo, vzdelávanie a výskum. Vlastník autorských práv má v USA možnosť preveriť, či sa jedná o použitie čestného používania „fair use“ na súde, pričom sudcovia posudzujú nasledujúce faktory:

- účel a charakter použitia (ide o použitie komerčnej povahy alebo použitie na neziskové vzdelávacie účely, či bolo niečo pridané k pôvodnému dielu a následne nové dielo nadobúda nový význam),
- druh diela chráneného autorskými právami,
- rozsah a významnosť použitej časti vzhľadom na celok diela chráneného autorskými právami (či sme použili celé pôvodné dielo, alebo iba jeho časť, ak sme použili iba jeho časť, tak sa berie do úvahy, či to nie je ťažiskom pôvodného diela),
- vplyv použitia diela chráneného autorskými právami na trh alebo na jeho hodnotu (či sa nejedná o náhradu pôvodného diela, pričom pôvodný autor prichádza o potencionálne zisky).

Použitie materiálu chráneného autorskými právami s povolením vlastníka autorských práv

Vlastník autorského práva môže dovoliť šírenie svojho diela viacerým osobám. V prípade, že chce dovoliť šírenie osobám, ktoré nepozná, môže použiť rôzne licencie. Napríklad licencia „Creative Commons“ umožňuje autorom udeliť takéto povolenie na šírenie svojho diela. Stránka YouTube umožňuje pri vkladaní multimediálnych súborov označiť takéto súbory licenciou „Creative Commons CC BY“, pričom si vlastník autorského práva ponechá svoje vlastníctvo, ale umožní ďalším autorom opakovane používať a upravovať jeho multimediálne súbory v súlade s podmienkami licencie.

POZNÁMKA

„Creative Commons“ je americká nezisková organizácia, ktorá sa od svojho založenia v roku 2001 venuje legálnemu využívaniu a zdieľaniu autorských diel. Patrí medzi najznámejšie a najpoužívanejšie licencie.

Voľné dielo

Autorské práva nie sú večné, časom diela strácajú ochranu. Doba, kedy sa dielo považuje za voľné dielo, záleží od miesta a času zverejnenia. Na Slovensku môžeme hovoriť o voľnom diele, podľa Zákona č. 185/2015 Z. Z. § 9 Autorský zákon.

Doba, kedy uplynie trvanie majetkových práv podľa § 32 je na Slovensku stanovená na 70 rokov po smrti autora.

Metodika pre učiteľa

CIEĽ

- Cieľom tejto kapitoly je naučiť študentov používať multimédiá na webe. Zoznámiť ich s audio a video formátmi, ukázať im, ako sa sťahujú, nahrávajú a vysielajú naživo multimediálne súbory na webe. Študenti sa naučia vkladať audio a video súbory na webové stránky pomocou elementov jazyka HTML5.

MOTIVÁCIA

Na predchádzajúcich hodinách sa študenti naučili vytvárať audio a video nahrávky a upravovať ich na počítači. V tejto kapitole sa naučia, ako majú vytvorené audio a video nahrávky zverejniť na webe.

Odporúčame pokračovať s diskusiou o tom, či majú študenti skúsenosti so sťahovaním a nahrávaním súborov na web. V závere hodiny je možné rozvinúť diskusiu so študentami o dodržiavaní autorských práv.

VÝKLAD

Študentov je potrebné oboznámiť:

- s pojmami zásuvný modul, kodek, nahrávanie (z angl. upload), sťahovanie (z angl. download), vysielanie naživo (z angl. streaming),
- s najpoužívanejšími audio a video formátmi,
- s vkladaním HTML elementov pre audio a video,
- so sťahovaním, nahrávaním a vysielaním videa naživo s využitím stránky www.youtube.com

Výklad je potrebné doplniť praktickými ukážkami na uvedených príkladoch a striedať s úlohami, ktoré budú riešiť študenti.

Problematiku autorských práv je vhodné prebrať formou diskusie.

- Odporúčame tiež so študentami diskutovať o etických aspektoch publikovaného audia a videa (nenarúšať dôstojnosť a súkromie nahrávaných osôb, pýtať si od nich súhlas, vyhnúť sa urážlivému a vulgárnemu obsahu a pod.).

Nadväznosť na predchádzajúce kapitoly: V predchádzajúcich kapitolách sme sa naučili vytvárať statické webové stránky a v predchádzajúcej kapitole vytvárať a upravovať audio a video súbory. V tejto kapitole sa naučíme vkladať multimediálne súbory na web.

23 PROGRAMOVANIE MULTIMÉDIÍ V JAZYKU PYTHON

Programovací jazyk Python dnes naberá na popularite hlavne kvôli svojej jednoduchej a prehľadnej syntaxi, kvôli efektívnosti, rozsiahlej základnej knižnici a obrovskému množstvu externých modulov, ktoré vznikli vďaka komunite združenej okolo jazyka Python.

Táto kapitola slúži ako doplňujúci materiál k prezentovaným informáciám o multimédiách a formou webovej aplikácie prináša do učebnice množstvo praktických ukážok a príkladov využitia jazyka Python na programovanie multimédií. Všetky lekcie elektronickej učebnice sú dostupné na adrese http://multimedia.umb.sk/

Programovací jazyk Python

Vďaka zanietenému vedcovi a programátorovi menom Guido van Rossum začal v roku 1982 na univerzite v Amsterdame vznikať nový programovací jazyk, ktorý mal slúžiť primárne pre vedcov a ktorý mal byť jednoducho osvojiteľný, efektívny a ľahko aplikovateľný pre vedecké účely. Jazyk, ktorý autor pomenoval Python[9], si začal získavať mnohých nadšencov z radov jeho kolegov, ale aj širšej programátorskej verejnosti. Na stránke www.python.org[10] sa nachádza kompletná dokumentácia ku všetkým verziám jazyka, združuje sa tu komunita a indexujú sa aj moduly vytvorené nezávislými vývojármi. Okrem toho sa na nej nachádzajú pracovné ponuky pre „pythonistov“ a skutočné príbehy ľudí, ktorým Python zmenil život. Stránka je v správe združenia Python Sofware Foundation, ktoré vzniklo v roku 2001 a dohliada na správny vývoj jazyka, zachovanie jeho dobrého mena a zásad, ako aj jeho neustále zlepšovanie a šírenie.

Vzrastajúci záujem o programovací jazyk Python sa prejavil nielen v zahraničných firmách, ako Google, Dropbox, IBM, ale začali ho používať aj niektoré slovenské firmy, ako Pantheon Technologies, Accenture, Kistler, vnet, Sufio a ďalšie.

Všetky ukážky zdrojových kódov, ako aj teoretické informácie v našej aplikácií sa prikláňajú k prezentácii programovania v jazyku Python vo verzii 2.x.

Python a multimédiá

Za jednu z najväčších výhod **jazyka Python** sa považuje jeho prirodzenosť. Táto vlastnosť výrazne uľahčuje prvotné zoznámenie sa so syntaxou jazyka. Programy napísané v jazyku Python sú kompaktné, prehľadné a rýchlo editovateľné. Jednoduchá syntax vedie k zvýšeniu produktivity programátora, čím znižuje náklady na vývoj. Produktivite napomáha aj fakt, že Python má rozsiahlu štandardnú knižnicu, ktorá presahuje zvyčajne zahrnuté funkcie. Knižnica obsahuje:

- moduly na prácu s dátami, súbormi, regulárnymi výrazmi a databázami,

[9] názov dostal podľa seriálu Monty Python’s Flying Circus
[10] V roku 1997 bola prvý krát spustená stránka www.python.org

- moduly, ktoré spracovávajú dáta z internetu, ako aj moduly na prácu so značkovacími jazykmi ako HTML, či XML,
- moduly užitočné pre multimediálne aplikácie, ako aj modul Tkinter, ktorý umožňuje programátorovi jednoducho vytvoriť základné GUI pre svoj program.

Knižnica je nainštalovaná automaticky spolu s jazykom Python a je navrhnutá tak, aby ešte viac prehĺbila medziplatformovú kompatibilitu jazyka. Programy napísané v jazyku Python fungujú v rôznych platformách rovnako. Nezáleží na tom, či programátor používa Linux, Windows, Mac OS X alebo inú platformu.

Python umožňuje programátorovi písanie zaujímavých a výkonných programov bez toho, aby musel vynaložiť priveľa nadbytočného úsilia, ktoré nie je spojené s riešením pôvodného problému. Pri práci s multimédiami sa netreba sústreďovať na syntaktické pravidlá, zaoberať sa kompiláciou a ani rozdielmi, ktoré by mohli vzniknúť pri zmene platformy. Programátor sa môže sústrediť priamo na spracovanie a prezentáciu multimediálneho obsahu. Môže vyhľadávať a implementovať balíčky/moduly, ktoré sa zameriavajú na prácu s mediálnymi elementami, využívať flexibilitu jazyka Python a vyvíjať unikátne aplikácie v kratšom čase.

Existuje niekoľko desiatok modulov, knižníc a rozšírení, ktoré sú kompatibilné s jazykom Python a slúžia na úpravu mediálnych elementov. Niektoré moduly sa venujú konkrétnemu elementu, iné pracujú s niekoľkými mediálnymi elementami naraz. V našej aplikácií sme sa zamerali na vybrané moduly na prácu s multimédiami. Štandardná knižnica jazyka Python ponúka moduly na spracovanie textu, ďalej sú to externé balíčky Pillow, Pi3d, Pydub a MoviePy. Balíček Pillow je určený na spracovanie 2D grafiky, balíček Pi3d na vytváranie 3D grafiky, balíček Pydub podporuje spracovanie zvuku a balíček MoviePy ponúka niekoľko možností spracovania videa a animácie.

Spracovanie textu prostredníctvom štandardnej knižnice

Obrázok 23.1 Logá balíčkov Moviepy, Pillow, Pi3d, Pydub.

Štandardná knižnica jazyka Python ponúka široké spektrum možností spracovania textu. Prvou možnosťou sú metódy dostupné pre prácu s reťazcami. Sú to napríklad metódy, ktoré slúžia na zmenu veľkosti písmen, alebo umožňujú rôzne spôsoby zarovnania reťazca, alebo vyhodnocujú prítomnosť zadaného podreťazca v reťazci, ďalšie ponúkajú zmenu častí reťazca, alebo zabezpečujú rozdeľovanie a spájanie reťazcov.

Moduly balíčka Pillow na prácu s 2D grafikou

Existuje niekoľko techník spracovania 2D grafiky prostredníctvom **knižnice Pillow**. Python podporuje väčšinu bežne používaných grafických formátov na ukladanie rastrovej grafiky, ktoré sú sprístupnené na čítanie aj zápis. Okrem nich dokáže knižnica identifikovať a sprístupniť na čítanie aj ďalšie formáty, ako napríklad grafický formát PSD, príp. zápis do formátu PDF.

Súčasťou knižnice Pillow je niekoľko modulov. Základné funkcie, ako načítanie, vytvorenie alebo uloženie grafiky, ponúka modul Image, ktorého ďalšie funkcie zabezpečujú aj geometrické, farebné transformácie, či základné grafické operácie, filtrovanie, rozličné úpravy, ale aj možnosti kreslenia grafických objektov.

Vytváranie 3D grafiky pomocou balíčka Pi3d

Balíček pi3d umožňuje vytváranie a spracovanie 3D objektov. Funkcionalita balíčka Pi3d je zameraná na tvorbu jednoduchých skriptov pracujúcich s trojrozmernými objektami, ktoré sú dostupné aj na platformách ako Raspberry Pi alebo Android a je závislá od niekoľkých ďalších balíčkov, ktoré je potrebné nainštalovať. Po spustení príkazov na inštaláciu prerekvizít je možné naplno využívať funkcionalitu balíčka a vytvárať obrazovky, rotovať objekty, definovať vlastnosti 3D objektov a pod.

Spracovanie zvuku pomocou balíčka Pydub

Balíček Pydub je určený na manipuláciu so zvukovými stopami. Priamo po nainštalovaní umožňuje spracovanie zvukových stôp uložených vo formáte WAV, pretože využíva modul wave, ktorý je súčasťou štandardnej knižnice jazyka Python a slúži na manipuláciu so zvukovými stopami práve vo formáte WAV. Pri práci s inými audio formátmi je balíček Pydub závislý od ďalších knižníc (libav alebo ffmpeg). Pri vykonávaní zmien nad audio obsahom je okrem jeho načítania dôležité aj prehrávanie, rozdeľovanie a spájanie jeho častí, postupné zvyšovanie a znižovanie hlasitosti, uloženie vykonaných zmien a pod.

Spracovanie videa a animácií pomocou modulu MoviePy

Modul MoviePy prináša širokú škálu predpripravených funkcií a metód, ktoré umožňujú úpravu rôznych mediálnych elementov so zameraním na vytvorenie finálnej podoby videa alebo animácie. V kombinácií s ďalšími balíčkami umožňuje vytváranie animácií z vektorovej grafiky. Hlavné objekty balíčka MoviePy sú video a audio klipy, ktoré sú definované ako inštancie tried VideoClip a AudioClip. Od týchto tried dedia vlastnosti všetky ďalšie triedy, ktoré reprezentujú klipy. Najčastejšie používaná trieda je VideoFileClip, ktorá umožňuje načítanie klipu zo súboru. Podporovaná je väčšina najčastejšie používaných formátov, napríklad OGV, MP4, FLV, MPEG, AVI, MOV a iné. Prostredníctvom modulu MoviePy je možný jednoduchý strih, spájanie, vkladanie a vyberanie sekvencií, pridávanie efektov a transformácií. Z videa je veľmi jednoducho možné vytvoriť animáciu, napr. do formátu GIF. Prvým krokom je načítanie videa zo súboru, na ktoré je následne možné aplikovať akékoľvek transformácie a filtre, ktoré sú pre video klipy dostupné.

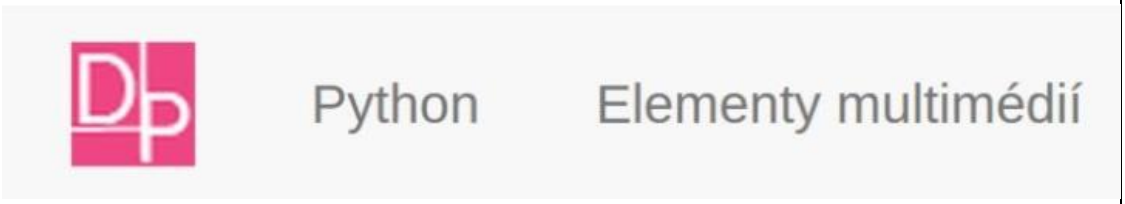
Webová aplikácia na demonštráciu práce s jazykom Python

PRÍKLAD 23.1

Zoznámime sa s prostredím aplikácie „Prezentácia multimédií prostredníctvom programovacieho jazyka Python“ na adrese http://multimedia.umb.sk/. Preskúmame jej možnosti a funkcionalitu. V jednotlivých externých balíčkoch ako sú Pillow, Pi3d, Pydub a MoviePy sa naučíme naprogramovať a spracovať mediálne elementy potrebné aj pri tvorbe webstránok.

Rozdelenie a popis častí prostredia

Základné časti **webovej aplikácie „Prezentácia multimédií prostredníctvom programovacieho jazyka Python“** sú dostupné cez navigáciu. Navigácia obsahuje dve položky – `Python` a `Elementy multimédií`. Nasledujúci obrázok je obrazovou ukážkou navigácie v prostredí na demonštráciu práce s programovacím jazykom Python.



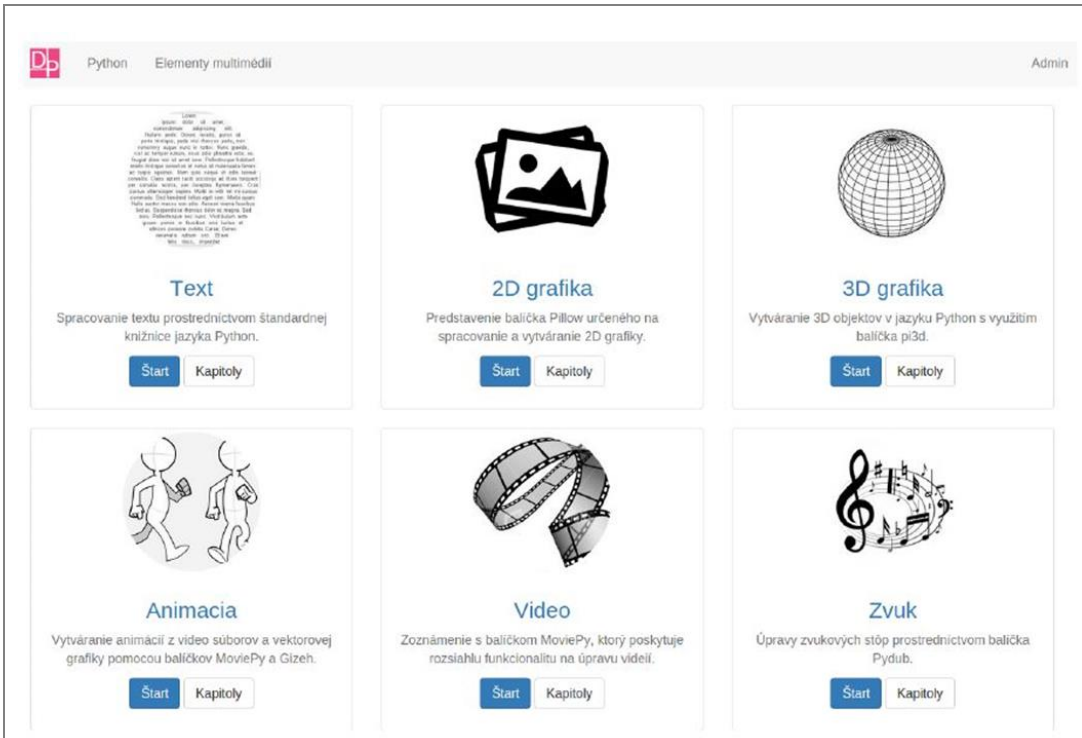
Obrázok 23.2 Navigácia.

Po kliknutí na položke Python sa používateľovi zobrazí časť prostredia, ktorá sprístupňuje obsah demonštrujúci prácu s jazykom Python. Táto časť prostredia umožňuje zoznámenie sa so základnou syntaxou jazyka Python, popisuje číselné dátové typy, ponúka prehľad operátorov, predstavuje riadiace štruktúry jazyka, opisuje tvorbu a používanie funkcií a modulov.

Po kliknutí na položke `Elementy multimédií` sa zobrazí stránka s mediálnymi elementami. Každý mediálny element je spracovaný samostatne a obsahovo tvorí individuálnu časť prostredia, ktorá demonštruje spracovanie vybraného mediálneho elementu prostredníctvom programovacieho jazyka Python.

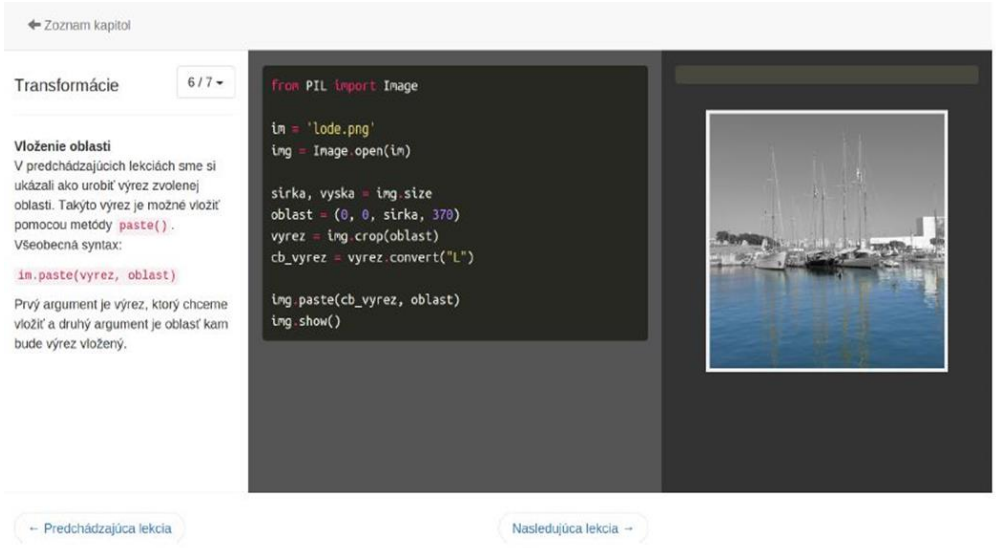
Kapitoly a lekcie

Jednotlivé časti prostredia sú rozdelené do kapitol. Rozdelenie celkov do kapitol umožňuje používateľovi jednoduchšiu orientáciu. Ukážka rozdelenia celku do kapitol je na obrázku 23.3. Kapitoly majú okrem názvu aj krátky popis, ktorý informuje o obsahu kapitoly.



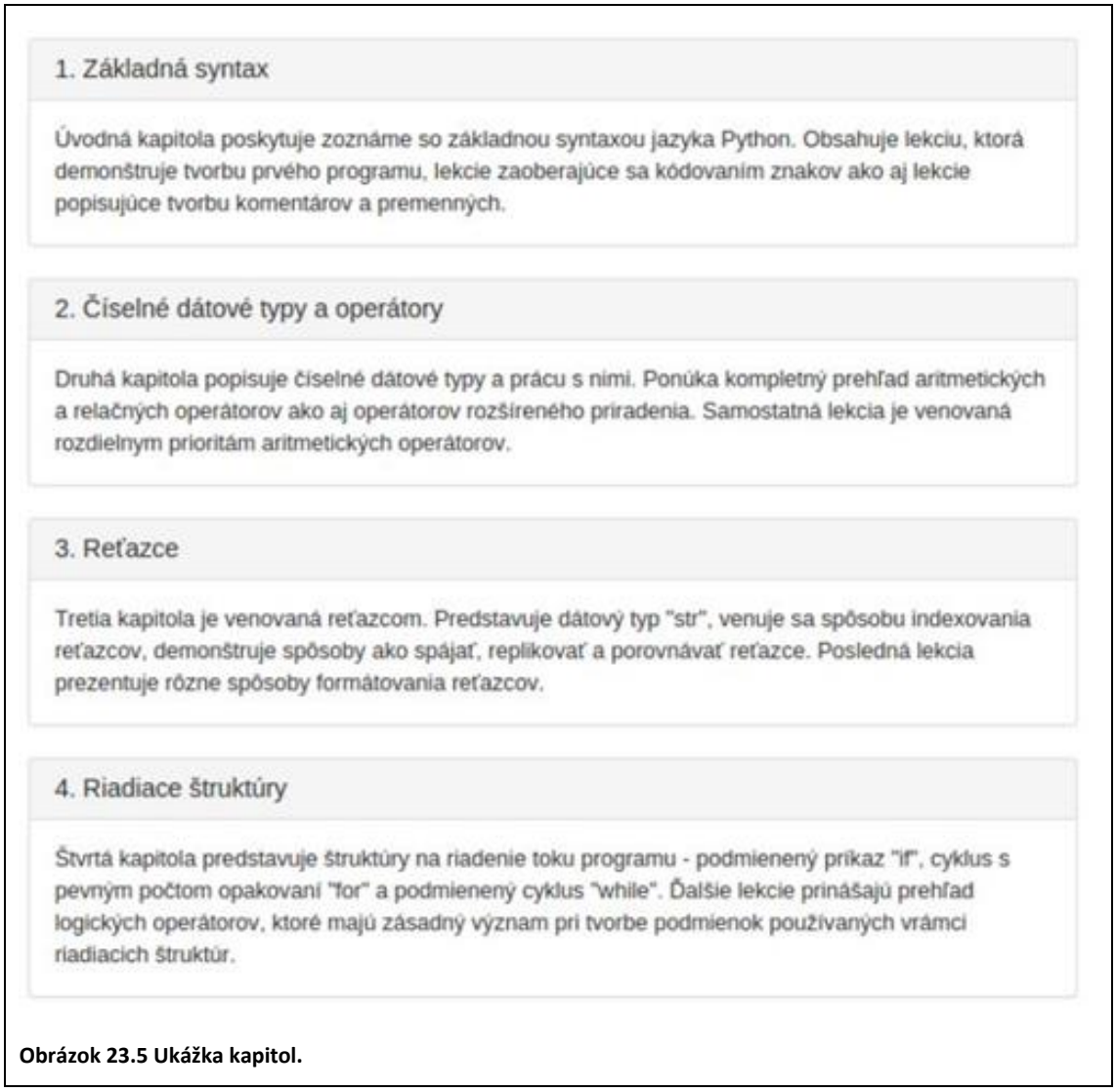
Obrázok 23.3 Zobrazenie webovej aplikácie na displeji osobného počítača.

Kapitoly sa skladajú z lekcií, ktoré demonštrujú prácu s jazykom Python. Lekcie boli navrhnuté tak, aby okrem textovej časti zobrazovali aj ukážky zdrojového kódu a ukážky výstupu na obrazovke.



Obrázok 23.4 Obrazová ukážka lekcie.

Všetky lekcie, ako vidieť napr. na obrázku č. 23.4, obsahujú vľavo textovú časť s popisom, v strede ukážku zdrojového kódu a napravo výstup na obrazovke.



Obrázok 23.5 Ukážka kapitol.

Metodika pre učiteľa

CIEĽ

- Táto kapitola má slúžiť najmä pokročilejším záujemcom o tvorbu a úpravu mediálnych elementov v programovacom jazyku Python. Snažíme sa priblížiť základnú syntax jazyka prostredníctvom vzdelávacej aplikácie, ktorá poukazuje na možnosti vytvárania a úpravy mediálnych elementov multimédií pomocou dostupných externých modulov. Je tu k dispozícii viac ako 100 lekcií s návodom ako programovať a spracovávať mediálne elementy. Vzdelávacia aplikácia je k dispozícii na stránke http://multimedia.umb.sk

MOTIVÁCIA

Odporúčame rozvinúť diskusiu na tému, ako je možné vytvoriť multimédiá na našej webstránke bez použitia HTML elementov a ktorý z programovacích nástrojov (skriptovacích jazykov) je na to najvhodnejší. Bolo by dobré zistiť, aké majú študenti skúsenosti s programovaním multimédií v jazyku Python.

VÝKLAD

V tejto kapitole je okrem stručného zoznámenia sa s jazykom Python prezentovaná webová aplikácia na ukážku praktického využitia jeho modulov na prácu s multimédiami:

- Pillow na prácu s 2D grafikou,
- Pi3d na vytváranie 3D grafiky,
- Pydub na spracovanie zvuku,
- MoviePy na pracovanie videa a animácií.

Nadväznosť na predchádzajúce kapitoly: Táto kapitola nemá priamu nadväznosť na predchádzajúce kapitoly. Pre programovanie multimédií v jazyku Python a lepšie porozumenie jednoltivých lekcií je vhodné, aby mal žiak zvládnuté aspoň základy dátových typov, riadiacich štruktúr (cyklov), zoznamov, funkcií a metód v jazyku Python. Nie je to však nevyhnutnou podmienkou.

ZHRNUTIE

V tejto kapitole sme sa zoznámili s programovacím jazykom Python a jeho externými modulmi na prácu s multimédiami. Prostredníctvom webovej vzdelávacej aplikácie je k dispozícii viac ako 100 lekcií s návodom ako programovať a spracovávať mediálne elementy. Lekcie sú usporiadané do jednotlivých kapitol, ktoré poskytujú úlohy a zadania s postupnými návodmi vo forme ukážok zdrojových kódov. Náročnosť jednotlivých lekcií má stúpajúcu tendenciu a na začiatku nevyžaduje žiadnu znalosť programovacieho jazyka Python.

INDEX OBRÁZKOV, GRAFOV A TABULIEK

BIBLIOGRAFIA

[1] Hrušecký, R. a kol.: Tvorba webových dokumentov, študijný materiál dostupný v LMS pre študentov a pedagógov FMFI UK Bratislava, https://moodle.uniba.sk/fmfi

[2] ŠPINAR, D.: Tvoříme přístupné webové stránky. Praha: Zoner Press, 2004, ISBN: 80-86815-11-0.

[3] Wagner, M., Hrušecký, R.: Internet: princípy a tvorba webu 1, Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika, Bratislava: ŠPÚ, 2010, ISBN 978–80–8118–035–4.

[4] Conversion Uplift, Get more from digital, The Inverted Pyramid Writting Style, https://www.conversion-uplift.co.uk/glossary-of-conversion-marketing/inverted-pyramid/

[5] Digitálne technológie pre žiakov so špeciálnymi vzdelávacími potrebami, http://www.edi.fmph.uniba.sk/~jaskova/IKTH/

[6] Christopher Reeve Stars in „Rear Window", https://thehitchcockreport.wordpress.com/2013/06/16/christopher-reeve-stars-in-rear-window/

[7] Kurz Prístupnosť elektronických dokumentov, http://www.edi.fmph.uniba.sk/~jaskova/ped/

[8] Let's run together, Responzívny dizajn, http://www.run.sk/responzivny-dizajn

[9] Nie ste náhodou farboslepý?, https://magazin.centrum.sk/zdravie/nie-ste-nahodou-farboslepy/676553.html

[10] Použitelnost stránek, https://www.jakpsatweb.cz/pouzitelnost.html

[11] Použiteľnosť, http://pouzitelnost.kincel.sk/uvod-do-pouzitelnosti.htm

[12] Pravidla tvorby přístupného webu, http://pristupnost.nawebu.cz/texty/pravidla-standardy.php

[13] Přístupnost a použitelnost, http://www.pestujemeweb.cz/obsah/ruzne/pristupnost-a-pouzitelnost.php

[14] Responzívny a adaptívny dizajn – štandardy pri tvorbe nových webov, https://www.alejtech.sk/sk/blog-o-webdizajne/responzivny-dizajn-standard-pri-tvorbe-novych-webov.html

[15] Seven things about Stephen Hawking you may have wondered about, https://kormorant.co.za/39993/7-things-stephen-hawking-may-wondered/

[16] Špeciálne pomôcky, https://www.specialnepomocky.sk/ (obrázky)

[17] Testovanie použiteľnosti webovej stránky, http://www.vytvoreniewebstranky.eu/testovanie-pouzitelnosti-stranky.html

[18] Tvorba-webu.cz, Použiteľnosť, https://www.tvorba-webu.cz/tipy/pouzitelnost.php

[19] w3schools.com, https://www.w3schools.com/

[20] Web Accessibility in Mind, https://webaim.org/

[21] Web Support, Aký je to responzívny web?, https://www.websupport.sk/blog/2013/11/aky-je-to-responzivny-web/

[22] 7 Critical WebsiteUsability Principles You Should Learn, https://www.advisorwebsites.com/blog/blog/general/7-website-usability-principles-you-should-know

[23] Salanci, Ľ. Informatika pre stredné školy – Práca s grafikou, Slovenské pedagogické nakladateľstvo, Bratislava 2004, ISBN 80-10-00531-2.

[24] Šnajder, Ľ., Kireš, M. Informatika pre stredné školy – Práca s multimédiami, SPN, Bratislava 2005, ISBN 80-10-00422-7
[25] Pekárová, J., Guniš, J., Jašková, Ľ., Mikolajová, K., Mikuš, Ľ.: Multimédiá, Digitálna gramotnosť učiteľa, Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika, Bratislava, ŠPÚ, UPJŠ, UK, UKF, UMB, ŽUŽa 2009, ISBN 978-80-89225-51-4, http://www.statpedu.sk/files/sk/o-organizacii/projekty/projekt-dvui/publikacie/multimedia.pdf
[26] Vítko, P., Horváthová, D.: Multimédiá a ich využitie vo vzdelávaní. Banská Bystrica : Akadémia umení, 2008, ISBN 978-80-89078-47-9
[27] Horváthová, D.: Tvorba multimediálnych výučbových materiálov pre DV a e-learning. Banská Bystrica: UMB, 2011, ISBN 978-80-557-0182-0,
[28] HTML Media, www.w3schools.com
[29] Autorské práva v službe YouTube, https://www.youtube.com/intl/sk/about/copyright/#support-and-troubleshooting.