Co je HTML a CSS

HTML a CSS jsou základní stavební kameny webu. To, co vidíš na každé webové stránce nebo uvnitř webové aplikace je tvořeno právě pomocí nich.

**HTML (Hypertext Markup Language) je značkovací jazyk**, kterým vytváříme obsah a strukturu stránky. Říkáme, který kus textu je nadpis, co je obyčejný odstavec, že tento kousek textu je odkaz a kam vede a že zde bude na stránce obrázek.

**CSS (Cascading Style Sheet) je stylovací jazyk**, kterým HTML stránce dodáváme vzhled. Pomocí CSS nastavujeme, jak naše nadpisy vypadají, jakou mají barvu, jakým jsou písmem, jakou velikostí. Můžeme nastavit kolem bloku textu rámeček, přidat barvu pozadí a například i vytvářet efekty při najetí kurzorem myši. Přes CSS nastavíme třeba i to, že web má text ve dvou sloupcích nebo že obrázková galerie má fotky ve čtyřech řadách po pěti. CSS umí z obyčejné textové stránky udělat pastvu pro oči.

‍

Jak vypadá HTML značka

<h1>Nadpis první úrovně</h1>  
<h2>Nadpis druhé úrovně</h2>  
<h3>Nadpis třetí úrovně</h3>  
<p>Odstavec textu.</p>‍

Většina HTML značek je takzvaně párových, tj. mají otevírací polovinu (např. **<p>**) a uzavírací polovinu (např. **</p>**). Mezi otevírací a uzavírací polovinou značky je text, který touto značkou chceme označit.

Existují i značky nepárové, které mají pouze první, otevírací polovinu, ale o těch si povíme až ve druhé lekci.

HTML značky jsou do sebe i vnořovat. Příkladem vnořených značek je například odrážkový seznam:

<ul>  
   <li>První odrážka</li>  
  <li>Druhá odrážka</li>  
   <li>Třetí odrážka</li>  
</ul>  
‍

V kódu vidíme značku **<ul>** pro samotný seznam. Uvnitř jsou pak vnořené značky **<li>** pro jednotlivé položky seznamu.

Pokud jsou do sebe prvky takto vnořené, říkáme tomu vnějšímu **rodič**(parent) a těm vnořeným **potomci**(children). Značky <li> jsou k sobě navzájem **sourozenci**(sibling).

‍

Editor HTML

HTML stránka je obyčejný prázdný soubor, do kterého píšeme prostý text, tedy text bez formátování. Pro psaní HTML stránek nepoužíváme Word a podobné programy, ale stáhneme si programátorský editor jako je např. [VisualStudio Code](https://code.visualstudio.com/).

‍

Struktura HTML dokumentu

Editor spustíme, založíme nový HTML soubor a můžeme začít psát naše značky. Ale správný HTML dokument (říkáme, že je **validní**) má nějakou strukturu. Nelze jen začít psát značky tak, jak jsme si je ukázali nahoře.

Základní struktura dokumentu vypadá takto:

<!DOCTYPE html>  
<html lang="cs">  
<head>  
   <meta charset="UTF-8">  
   <title>Titulek stránky</title>  
</head>  
‍  
<body>  
   <h1>Obsah stránky</h1>  
   <p>Teprve uvnitř značek body píšeme náš kód.</p>  
  
</body>  
</html>  
‍

Není potřeba se děsit a učit se tuto strukturu nazpaměť. Časem si ji zapamatujete a všechny programátorské editory s podporou HTML vám jsou ji schopné vytvořit automaticky.

Na první řádku je **<!DOCTYPE html>**. tato značka nastavuje, o jaký typ dokumentu se jedná. V tomto případě to znamená, že dokument používá nejmodernější HTML verze 5.

Důležité je, že celý dokument je zabalený do značky **<html>**. Uvnitř této značky jsou pak dvě sekce. Hlavička stránky **<head>** a tělo stránky **<body>**.

Do hlavičky stránky píšeme metadata stránky, nastavení pro prohlížeč, jak má stránku zobrazovat.

Do těla stránky pak píšeme vlastní obsah stránky. Veškerý náš kód tedy budeme psát vždy mezi značky **<body>** a **</body>**.

‍

Kontrolní otázky

Jak se jmenuje technologie pro tvorbu obsahu a struktury webové stránky?

* Pomocí čeho nastavujeme vzhled stránky?
* Jak vypadá základní značka pro nadpis a odstavec?
* Umíš vytvořit odrážkový seznam?
* Do jaké části HTML dokumentu píšeme náš kód, který bude tvořit obsah stránky.

### Projekt a podklady ke stažení

V dnešní lekci začneme pracovat na projektu, který budeme vylepšovat po celý zbytek kurzu. Postupně se naučíme vše potřebné a na konci budeme mít hezkou webovou stránku, kterou si budete moci přizpůsobit pro svoje potřeby.

Mám pro vás připravené obrázky a text, který budeme v projektu používat. Prosím stáhněte si podklady pro dnešní lekci z adresy:  
<https://czchts.cz/htmlcss02>

Objeví se rozhraní Google Disku, kde bude buď velké modré tlačítko pro stažení nebo vpravo nahoře ikonka Stáhnout (ikona se šipkou dolů). Podklady si stáhni, rozbal do složky u sebe na počítači a dál postupuj podle videa.

### Další HTML značky

V první lekci jsme si ukázali pouze úplně základní značky pro nadpis, odstavec a seznam. Dnes k nim přidáme další značky.

#### Komentáře

Nejde přímo o značku jako takovou, ale komentář může být velmi užitečný. Používáme v případě, že si chceme do kódu udělat vlastní poznámku. Komentář můžeme také použít pro dočasné "vypnutí" části kódu. Vše, co je uvnitř komentáře totiž prohlížeč ignoruje.

###### <!-- toto je HTML komentář -->

###### <!--

###### Komentář může být i přes několik řádků a lze s ním deaktivovat části kódu

###### <p>Tento odstavec se na výsledné stránce neukáže.</p>

###### -->

#### Titulek stránky

Značka <title> je se na rozdíl od ostatních značek nepíše do sekce <body>, ale do sekce <head> stránky. Pomocí značky <title> nastavíme titulek stránky, který se ukáže v oušku záložky prohlížeče, na titulkové liště ve Windows, použije se při vytváření oblíbeného odkazu v prohlížeči, ale hlavně také jako název stránky ve výsledcích vyhledávání např. na Googlu.

###### <head>

###### <title>Titulek stránky</title>

###### </head>

#### Odkaz

Slouží pro vytváření odkazů vedoucích na další stránky. Může se jednat o stránku v rámci našeho webu nebo na úplně cizí stránku kdekoliv v internetu.

###### <a href="http://www.google.com">Odkaz na Google</a>

U odkazu (značka **<a>**) poprvé vidíme něco, čemu se říká **atribut**. Jde o jakýsi "parametr" značky, který pro značku poskytuje doplňkové informace. V případě odkazu to dává perfektní smysl - nechceme z textu udělat jenom odkaz, ale potřebujeme také říci, kam ten odkaz má vést. U odkazu tedy do atributu **href**uvedeme cílovou adresu odkazu. Po kliknutí na odkaz prohlížeč přesměruje zákazníka právě tam.

Atributy se vždy uvádí do otevírací poloviny značky a vždy jsou ve formátu **atribut="hodnota"**. Hodnota atributu je vždy v uvozovkách.

Konkrétně odkaz může mít i další atributy. Například když do atributu **target**dáme hodnotu **"\_blank"**, bude to znamenat, že se cílová stránka v prohlížeči otevře v nové záložce (nebo v novém okně, pokud prohlížeč záložky nepodporuje).

###### <a href="https://www.czechitas.cz" target="\_blank">Odkaz do nové záložky nebo okna</a>

Můžeme vytvářet i odkazy na e-mailové adresy. Po kliknutí na takový odkaz se otevře poštovní program, vytvoří se v něm nová zpráva a do pole komu se předepíše e-mailová adresa uvedená v našem odkazu. Formát odkazu je stejný, jen do atributu href musíme uvést e-mailovou adresu uvozenou slovíčkem **mailto:**.

###### <a href="mailto:muj@email.cz">Odkaz na email</a>

#### Zalomení řádku

Text na webové stránce se standardně zalamuje na konci řádku. Někdy ale potřebujeme text zalomit předčasně nebo na přesně daném místě. Představme si například, že chceme na stránce napsat poštovní adresu a každý údaj chceme na samostatném řádku, stejně jako bychom to napsali na obálku dopisu. Pro zalomení řádku použijeme nepárovou značku **<br>**.

###### <p>

###### Adéla Maříková<br>      Pouliční 1<br>      Městečko nad Řekou<br>      123 45

###### </p>

Značky **<br>** používáme pouze pro zalomení řádku uvnitř odstavce. **Nepoužíváme je jako nástroj pro odsazení dvou bloků textu od sebe.**

#### Zdůraznění textu tučným písmem a kurzívou

Pro zdůraznění části textu (např. slova ve větě) můžeme použít značky **<strong>** a **<em>**. Obě značky znamenají zdůrazněný text, ale je mezi nimi drobný rozdíl.

Značka **<strong>** dělá označený text **tučný**a používá se pro silný důraz. Používáme např. slova, která jsou ve větě důležitá.

Značka **<em>** změní označený kus textu na kurzívu (skloněné písmo). Většinou se používá ve chvíli, kdybychom např. při čtení věty nahlas chtěli dané slovo zdůraznit jinou intonací hlasu.

###### Toto je <strong>tučný</strong> text.      A toto je <em>kurzíva</em>, neboli skloněný text.

### Kontrolní otázky

* K čemu je komentář?
* Jak nastavím titulek stránky? Do jaké části stránky musím značku napsat?
* Co je to atribut značky a do jaké části značky se píše?
* Jaká je značka pro odkaz a jaký má povinný atribut?
* Jaká je značka pro zalomení řádku a na co ji zásadně nepoužíváme?
* Jak zdůrazníme text pomocí tučného textu nebo kurzívy?

### Přidáváme obrázky

Obrázky do stránky přidáváme pomocí značky **<img>**. Jedná se o nepárovou značku, takže má pouze první polovinu značky. Žádné </img> neexistuje a nikde ho neuvádíme.

Samotná značka však nestačí. Prohlížeči musíme dát vědět, jaký obrázek má do stránky vložit. To uděláme pomocí atributu **src**, do kterého jako hodnotu napíšeme **cestu k obrázku**.

Leží-li obrázek na stejné úrovni (ve stejné složce) jako náš index.html, stačí do atributu napsat pouze název obrázku, třeba src="foto.jpg". Máme-li obrázek umístěný v jiné složce, musíme do atributu src uvést i cestu k obrázku. Pokud mám soubor ve složce obrazky, uvedu src="obrazky/foto.jpg".

Obrázek má ještě jeden povinný atribut a to je **alt**. Do atributu alt uvádíme stručný popis obrázku. Tento popis slouží pro asistenční technologie (např. čtečky obsahu pro slepce, apod.), prohlížeč text zobrazí i v případě, že nemůže obrázek načíst. A hlavně tento text použijí i vyhledávače jako je např. Google. Atribut alt proto vždy vyplňujeme (i když se nám nechce!).

Kód pro správné vložení obrázku tedy vypadá třeba takto:

###### <img src="obrazky/foto.jpg" alt="Adéla Maříková">

### Tabulky

Tabulky v HTML jsou tvořeny sérií několika vnořených značek. Každou tabulku tvoří hlavní značka **<table>**. Do ní jsou vnořeny značky **<tr>** pro řádky tabulky (table row). Uvnitř každého řádku pak máme jednotlivé buňky tabulky **<td>** (table data), neboli sloupce.

Chceme-li v tabulce vytvořit záhlaví, použijeme pro buňky záhlaví místo **<td>** značku **<th>** (table header).

Celá tabulka pak bude v kódu vypadat třeba takto:

###### <table>          <tr>                <th>Jméno</th>                <th>Datum narození</th>          </tr>          <tr>                <td>Jana Novotná</td>                <td>6.4.1996</td>          </tr>          <tr>                <td>Alena Hanzlíková</td>                <td>27.11.2001</td>          </tr>     </table>

### Kontrolní otázky

* Jakou značkou vkládáme do stránky obrázky?
* Má značka pro obrázek nějaké povinné atributy? Jaké?
* Jaká je skupina značek pro vkládání tabulky?
* Jak vytvořím tabulku se záhlavím?

## ‍

### Co je CSS?

CSS (Cascading Style Sheets) je stylovací jazyk, kterým nastavujeme vzhled stránky. Při tvorbě webu se snažíme co nejvíce oddělit obsah a vzhled. Obsah a strukturu stránky píšeme v HTML. Vzhled stránky píšeme v CSS.

#### Připojení CSS do HTML

CSS lze psát přímo ke konkrétním značkám nebo do hlavičky HTML souboru, ale nejčastější a nejpoužívanější způsob, který budeme používat i my, je zápis CSS do samostatného souboru.

Aby prohlížeč věděl, že daná stránka používá styl z našeho CSS souboru, musíme styl do stránky tzv. připojit.

Pro připojení externích zdrojů do stránky slouží značka **<link>**. Jedná se o obecnou značku, kterou lze do stránky napojit mnoho různých zdrojů - např. ikony, písma, apod. Aby prohlížeč vědět, o jaký zdroj se jedná, má značka povinný atribut rel, do kterého se jako hodnota uvádí typ zdroje. V případě CSS souboru je to **rel="stylesheet"**.

Druhým povinným atributem je href, do kterého zapíšeme cestu k připojovanému zdroji. může to být kompletní internetová adresa (url) nebo i cesta souboru relativně ke stránce, do které zdroj vkládáme, např. **href="style.css"**.

Značka pro připojení CSS souboru, který leží na stejné úrovni jako HTMl stránka, pak může vypadat takto:

     <link rel="stylesheet" href="style.css">

#### Syntaxe CSS

###### Syntaxe jazyka CSS je velmi jednoduchá. V základní podobě vypadá takto: selektor {    vlastnost: hodnota; }

Chceme-li stylovat nějaký prvek na stránce, musíme CSS říci, o jaký prvek se jedná. K tomu slouží **selektor**. Selektorů je mnoho typů. Nejjednodušší je **selektor typu element**, kdy jako selektor uvedeme název HTML značky, např. h1 (viz dále).

Do složených závorek pak píšeme páry vlastnost: hodnota. Vlastnost je CSS vlastnost, kterou chceme nastavit, např. barvu textu nastavíme pomocí vlastnosti color. Hodnota je hodnota, na kterou chceme vlastnost nastavit. V případě naší barvy to může být např. red (červená).

Celý CSS zápis, který nastaví hlavní nadpis H1 na červenou barvu pak vypadá takto:

###### h1 {      color: red; }

Každý řádek uvnitř složených závorek {} vždy **končí středníkem**. Chceme-li prvku nastavit **více vlastností** najednou, učiníme tak **uvnitř jedné deklarace**. Počet vlastností v deklaraci není ničím omezený, **na pořadí zápisu jednotlivých vlastností nezáleží**.

###### h1 {     color: red;     background-color: yellow; }

‍

#### Selektor typu element

Jedná se o nejjednodušší CSS selektor - prostě uvedeme název HTML značky, kterou chceme stylovat. Selektor na stránce vybere **všechny značky** tohoto typu, tj. pokud napíšeme selektor h2, obarví se všechny nadpisy druhé úrovně na stránce. Jak nastavit barvu pouze jednomu nadpisu si ukážeme v příštích lekcích.

###### /\* ukázka selektorů typu element \*/ h1 {     color: red;     background: yellow; } h2 {     color: purple; } p {     background-color: lightblue; } a {     color: orangered; }

#### Vícenásobný selektor

Někdy potřebujeme stejnou vlastnost nastavit několika různým prvkům. Je zbytečné uvádět samostatné deklarace pro každý typ prvku. Jednotlivé **selektory lze oddělit čárkou**, vlastnosti ve složených závorkách se pak aplikují na všechny prvky v seznamu.

Pokud například chceme všechny nadpisy první, druhé a třetí úrovně modrou barvou:

###### h1, h2, h3 {    color: blue; }

#### Komentáře v CSS

Stejně jako v HTML, i v CSS můžeme použít komentáře - ať už na vlastní poznámky v kódu nebo na zakomentování kusu kódu, když ho chceme přechodně "vypnout", ale přitom si ho v souboru zachovat pro pozdější použití.

Komentáře v CSS mají syntaxi:

###### /\* toto je CSS komentář \*/

###### /\* Komentář může být i na více řádků, vše mezi hvězdičkami se v kódu ignoruje. \*/

### Dědičnost v CSS

Pro většinu vlastností v CSS platí princip **dědičnosti**. Pokud jsou HTML prvky do sebe zanořené, můžeme CSS vlastnosti nastavit na nadřazený vnější prvek (rodič). Všechny podřízené vnořené prvky (potomci) tuto vlastnost zdědí. Nebo obráceně: podřízené prvky přejímají CSS vlastnosti ze svých nadřízených prvků.

Skvěle se to hodí třeba v případě, že potřebuji nastavit barvu nebo styl písma pro celý dokument. Nemá smysl vyjmenovávat postupně všechny prvky a nastavovat jim danou barvu. Všechny prvky na stránce jsou přeci vnořené ve značce**<body>** (tělo stránky). Nastavíme tedy barvu na a písmo na tělo stránky a všechny značky na stránce toto nastavení zdědí:

###### body {    color: orange;    font-family: Arial; }

CSS vlastností se dědí na všechny potomky (i potomky potomků) až do chvíle, kdy nějakému prvku danou CSS vlastnost nastavíme konkrétně:

###### /\* veškerý text na stránce bude oranžovou barvou a fontem Arial Pouze nadpisy H2 budou růžové \*/ body {      color: orange;      font-family: Arial; }

###### h2 {     color: pink; }

### Barvy

Barvy v CSS lze zapsat mnoha různými způsoby. Nejjednodušší způsob už jsme si ukázali.

#### Pojmenované barvy

V CSS existuje 141 pojmenovaných barev. Tyto vybrané barvy mají přidělené anglické názvy a v CSS stačí tento název zapsat jako hodnotu barvy. Seznamy pojmenovaných barev lze najít na internetu, zde je několik odkazy na některé z nich:

* [HTML Color Codes](https://htmlcolorcodes.com/color-names/)
* [W3Schools - CSS Colors](https://www.w3schools.com/cssref/css_colors.asp)
* [MDN - Color Values](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value)

#### RGB barvy

Barvy pro zobrazování na monitoru se většinou definují pomocí **RGB** modelu. RGB je zkratka pro Red, Gree, Blue (červená, zelená, modrá) a dokazuje se na způsob, jakým se na moderních zobrazovačích tvoří barvy. Každý pixel (obrazový bod) na obrazovce je většinou tvořen třemi světelnými diodami - červenou, zelenou a modrou. Podle toho, jak hodně která dioda svítí, tak vnímáme výslednou barvu pixelu.

RGB model tedy barvu definuje jako tři hodnoty - množství červeného, zeleného a modrého světla. Hodnota může být 0 (=nesvítí) až 255 (=svítí na maximum). Smícháním světel dostáváme všechny další barvy. Takže např. žlutá je kombinací červené a zelené. Fialová kombinací červené a modré, apod.

Barvy v RGB lze v CSS zapisovat takto:

###### /\* oranžová barva \*/ h1 {    color: rgb(255, 140, 0); } /\* fialová barva \*/ h2 {    color: rgb(190, 20, 240); }

#### Hexadecimální barvy

**Tento zápis se při tvorbě webu používá nejčastěji**, protože se jedná o nejkratší zápis, který ale umožňuje zapsat jakoukoliv barvu. Jedná se o stejné RGB barvy jako výše, pouze zapsané jiným způsobem.

Jednotlivé složky RGB jsou zapsané v [šestnáctkové soustavě](https://cs.wikipedia.org/wiki/%C5%A0estn%C3%A1ctkov%C3%A1_soustava) a celý kód barvy se tak vejde do šesti znaků. Před kód se uvede znak **#**, aby bylo jasné, že se jedná o zápis v šestnáctkové soustavě. Stejnou oranžovou a fialovou barvu z příkladu výše bychom tak mohli zapsat takto:

###### /\* oranžová barva \*/ h1 {    color: #ff8c00; } /\* fialová barva \*/ h2 {    color: #be14f0; }

V CSS se často můžeme setkat i se zkráceným zápisem barvy s pouze třemi číslicemi. V tomto případě se barva interpretuje tak, jako by se každá číslice dvakrát opakovala. Takže např. **#fff** znamená **#ffffff** neboli bílou barvu. Barva **#f83** znamená **#ff8833** neboli oranžová.

Většina editorů kódu nabízí buď vestavěné nástroje pro práci s barvami, nebo je lze stáhnout do editoru jako doplněk. Kódy barev (nebo číselné hodnoty RGB) lze vidět také v grafických editorech jako je Photoshop a další. Existují i online nástroje na tvorbu barevných palet a míchání barev, například:

* [Adobe Color CC](https://color.adobe.com/)
* [Paletton](http://paletton.com/)
* [HTML Color Picker](https://htmlcolorcodes.com/color-picker/)
* … a spousta dalších

### Kontrolní otázky

* K čemu je CSS?
* Kam zapisujeme CSS kód?
* Jak připojíme CSS kód do stránky?
* Napiš jednoduchý selektor pro nadpisy druhé úrovně.
* Napiš vícenásobný selektor, který vybere zároveň všechny odstavce i položky seznamu (<li>).
* Jaké jsou v CSS vlastnosti pro nastavení barvy textu a barvy pozadí?
* Co je nejčastější způsob zápisu barev v CSS?

### Vlastnosti písma a textu

Text je základním stavebním kamenem každé stránky. CSS nabízí spoustu vlastností, díky kterým můžeme vzhled textu na stránce změnit podle svých potřeb.

#### Velikost písma (font-size)

Velikost písma nastavujeme pomocí vlastnosti **font-size**. Jako hodnotu uvádíme číslo včetně jednotky. Nejjednodušší na pochopení jsou asi pixely - jednotka **px**.

###### h1 {    font-size: 80px; }

Můžeme použít i další jednotky - např. **%**, **em**, **rem**, **pt**, **mm**, apod. Jejich podrobný popis lze dohledat na internetu (např. [MDN](https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Values_and_units), [CSS Tricks](https://css-tricks.com/the-lengths-of-css/)).

#### Výška řádku (line-height)

Výška řádku, nebo také řádkování nebo proklad řádků, nastavuje rozestup mezi jednotlivými řádky textu. CSS vlastnost pro nastavení výšky řádku je **line-height**. Řádkování lze nastavit na absolutní hodnotu s jednotkou (např. 40px) nebo lze hodnotu uvést bez jednotky, v tom případě se výška řádku spočítá jako násobek tohoto čísla a velikosti písma.

###### /\* řádkování nastaveno na absolutní hodnotu rozestup mezi řádku bude přesně 50px \*/ body {    line-height: 50px; }

###### /\* řádkování nastaveno relativně rozestup mezi řádku bude vždy jedenapůlkrát velikosti písma \*/ body {    line-height: 1.5; }

Pro běžný text se doporučuje, aby řádkování bylo alespoň 1.5 až 1.6. Zlepší se tak čitelnost velkých bloků textu. Pro nadpisy opravdu velkým písmem je někdy vhodné volit řádkování menší, např. 1.2 až 1.4. Konkrétní "nejlepší" hodnota neexistuje, vše samozřejmě záleží také na vybraném písmu a povaze textu.

#### Tloušťka písma (font-weight)

Tloušťku písma (tučnost) nastavujeme vlastnosti **font-weight**. Ta může mít hodnotu **normal** (výchozí, netučné písmo) nebo **bold** (tučné písmo).

Máme-li k dispozici písmo, které má více řezů (tlouštěk), můžeme jaho hodnotu použít číslo, které odpovídá danému řezů. Přesná označení tlouštěk písma nejsou daná, ale časté značení bývá zhruba takové:

* 100 - thin
* 300 - light
* 400 - regular (normální řez)
* 600 - semibold
* 700 - bold (tučný řez)
* 900 - heavy nebo black

###### /\* odstavec nyní bude tučným písmem \*/ p {    font-style: bold; }

###### /\* má-li stránka tento řez písma k dispozici, odstavec bude tenkým písmem \*/ p {    font-style: 100; }

###### /\* nadpis h1, který ve výchozím stylu bývá tučně, bude nyní normálním (netučným) písmem \*/ h1 {    font-weight: normal; }

#### Sklon písma (font-style)

Sklon písma nastavujeme vlastností **font-style**. Ta může nabývat hodnot:

* **normal** - běžné, neskloněné písmo
* **italic** - tzv. kurzíva, tj. skloněné písmo, tak jak ho vytvořil autor písma
* **oblique** - počítačem uměle nakloněné písmo

Rozdíl mezi variantou **italic** a **oblique** může být značný. Opravdová kurzíva (italic) bývá obvykle krásné skloněné písmo, hezké na pohled. Falešná kurzíva (oblique) je počítačem skloněné normální písmo, které je obvykle poznat na první pohled, protože "vypadá divně". Vždy se snažíme používat opravdovou kurzívu.

###### /\* nadpis bude kurzívou \*/ h1 {    font-style: italic; }

###### /\* značka em, který ve výchozím stylu bývá kurzívou, bude nyní zobrazena normálním (neskloněným) písmem \*/

###### em {    font-style: normal; }

#### Výběr písma (font-family)

Pro výběr písma (fontu) slouží vlastnost **font-family**. Jako hodnotu uvádíme přesný název písma. Má-li písmo v názvu mezeru, musíme název uzavřít do uvozovek.

###### /\* nadpis bude písmem Georgia \*/ h1 {    font-family: Georgia; } /\* název písma s mezerami - musíme použít uvozovky \*/ h2 {    font-family: "Helvetica Neue"; }

Pro případ, že vybraný font není na počítači návštěvníka stránek k dispozici, můžeme uvést více názvů oddělených čárkou. Prohlížeč se pokusí najít a použít písma v pořadí, v jakém jsou uvedena.

Seznam je obvykle vhodné zakončit tzv. generickým fontem. Generický font je název určité skupiny písem, které mají nějakou společnou charakteristiku. Není tak zaručeno použití konkrétního písma, ale na každém počítači (dle operačního systému) se použije výchozí písmo dané kategorie.

* **sans-serif** - bezpatkové písmo
* **serif** - patkové písmo
* **monospace** - neproporcionální písmo se stejně širokými znaky
* **cursive** - vypadá jako rukou psané
* **fantasy** - dekorativní písmo

###### /\* prohlížeč se pokusí použít písma v daném pořadí, nebude-li žádné k dispozici, použije alespoň nějaký bezpatkový (sans-serif) font \*/ body {    font-family: Raleway, Arial, Helvetica, sans-serif; }

###### /\* generický font lze použít i samostatně nejde nám o konkrétní písmo, chceme prostě nějaký patkový (serif) font \*/ h1 {    font-family: serif; }

#### Zarovnání textu

Pro zarovnání textu slouží vlastnost **text-align**. Můžeme nastavit hodnoty:

* **left** - zarovnání doleva, jde o výchozí hodnotu a obvykle ji není třeba nastavovat
* **right** - zarovnání doprava
* **center** - zarovnání na střed
* **justify** - zarovnání do bloku

Máme-li text napsaný velkým písmem nebo obsahuje-li dlouhá slova, může zarovnání do bloku způsobovat v textu nehezké mezery. Prohlížeče totiž neumí na koncích řádků dělit slova a počítač musí zarovnání zároveň k pravému i levému okraji řešit tak, že mezi slova umístí velké mezery, které pak zhoršují čitelnost. V takovém případě je lepší se zarovnání do bloku vyhnout.

###### /\* nadpis bude vycentrován na stránce \*/ h1 {    text-align: center; }

###### /\* v buňce tabulky je text zarovnaný doprava \*/ td {    text-align: right; }

###### /\* všechny odstavece jsou zarovnané do bloku \*/ p {    text-align: justify; }

### Dědění CSS vlastností

Některé vlastnosti se v CSS **dědí z rodiče na potomka**. Týká se to i většiny vlastností týkajících se textu. Pokud například chceme nastavit stejný typ písma pro celou stránku, stačí nám nastavit příslušné vlastnosti značce **<body>**, která je rodičem všech ostatních prvků na stránce.

###### body {    color: red;    font-family: 'Raleway', sans-serif;    font-size: 20px;    line-height: 1.5;    text-align: center; }

Použijeme-li tento CSS kód (viz. výše), bude veškerý text na stránce červenou barvou, písmem Raleway, výchozí velikost písma bude 20px, řádkování 1.5 a vše na stránce bude zarovnané na střed.

**Dědění se přeruší** ve chvíli, kdy nějakému vnořenému prvku (potomek značky **<body>**) nastavíme některou z uvedených vlastností specificky. V tom případě dostane specificky nastavená hodnota **přednost před hodnotou zděděnou**.

Ve výchozím stylu prohlížeče jsou velikosti písma např. pro nadpisy nastaveny jako násobky základní velikosti textu, takže změníme-li velikost písma pro značku **<body>**, přepočítají se i velikosti nadpisů (nezdědí tedy konkrétní hodnotu). Např. velikost nadpisu **<h1>** je standardně nadefinována jako dvojnásobek základní velikosti textu.

### Kontrolní otázky

* Jak nastavujeme velikost písma?
* Jaké jsou dva způsoby nastavení řádkování?
* Jak se pomocí CSS dělá tučné písmo a kurzíva?
* Jak se nastavuje v CSS font?
* Co je to generický font a proč ho chceme vždy použít?
* Jak nastavíme vlastnosti písma pro celou stránku najednou?

### ‍Členění obsahu na oddíly

Obsah stránky často potřebujeme členit na oddíly, ať už za účelem stylování nebo z důvodů logického členění dokumentu. Dlouho se používala a stále používá značka **<div>**, pro jiné než stylovací účely se však dnes používají hlavně sémantické značky **<header>**, **<footer>**, **<section>**, apod.

#### Obecný oddíl - div

Značka **<div>** má mezi ostatními HTML značkami zvláštní postavení. Je to totiž jedna ze dvou značek, která nemá **žádný sémantický význam**. Lze ji použít jako obecný kontejner, do kterého uzavřeme další HTML prvky, které chceme mít z nějakého důvodu pohromadě. Značka <div> jako taková nijak nevypadá. Když obalíme část kódu značkou **<div>**, "nic" se nestane (většinou).

Značka **<div>** se často používá právě a pouze za účelem stylování - např. chceme kolem něčeho udělat rámeček. Na moderních (složitějších) webech můžeme nalézt spoustu značek **<div>** vnořených do sebe - opět pouze a jen kvůli tomu, aby šel obsah v CSS nastylovat určitým způsobem a dosáhlo se požadovaného vzhledu stránky.

###### <div> <h2>Kontakt</h2>    <img src="obrazky/pohoda.jpg" alt="Adéla relaxuje">    <p>Ráda si povídám. O horách, hudbě, prostě o životě.</p> </div>

#### Sémantické značky pro členění obsahu - header, footer, section

HTML5 zavedlo nové sémantické značky pro členění obsahu. Jedná se hlavně o značky:

* **header** - hlavička stránky / dokumentu
* **footer** - patička stránky / dokumentu
* **section** - sekce stránky / dokumentu
* **main** - hlavní obsahová část stránky
* **article** - článek
* **nav** - navigace, menu
* **aside** - doplňkový text k hlavnímu obsahu, někdy také boční pruh (sidebar)
* atd.

Jednoduše popsaný seznam těchto elementů můžeš najít třeba na stránce [W3Schools](https://www.w3schools.com/html/html5_semantic_elements.asp).

Podobně jako **<div>**, ani tyto sémantické značky "nijak nevypadají" a nemají žádný výchozí styl v prohlížeči. Když do nich zabalíme část obsahu, na pohled se nic nestane. Ale mají sémantický význam, což je důležité například pro vyhledávače a asistenční technologie (čtečky obsahu pro nevidomé, apod.).

Když jejich použití dává smysl, snažíme se upřednostnit použití sémantických elementů před nesémantickým **<div>**.

###### <header>    <h1>Adéla na cestách</h1> </header>

###### <section>    <h2>O mně</h2>    <p>Jmenuji se Adéla, ale všichni mi říkají Áďo...</p> </section>

###### <section>    <h2>Turistický průvodce</h2>    <p>Pracuji jako turistický průvodce...</p> </section>

###### <section>    <h2>Kontakt</h2>    <p>Ráda si povídám. O horách, hudbě, prostě o životě...</p> </section>

###### <footer>    <p>© 2018, Adéla Maříková</p> </footer>

### CSS třídy

Doposud jsme při psaní spoléhali pouze na **selektor typu element**, tj. že stylujeme obecnou HTML značku. To se hodí pro vytvoření obecných výchozích stylů pro stránku, ale pro detailnější nastavení vzhledu už často ne. Při použití selektoru typu element se totiž styl aplikuje na **všechny** značky daného typu. My však často potřebujeme styl přidat pouze k jednomu konkrétnímu nadpisu nebo jednomu konkrétnímu stylu, někdy pouze k několika slovům ve větě.

#### Atribut class

CSS třídu přidáváme v HTML jako atribut **class** k jakékoliv HTML značce. Atribut přidáváme do otevírací poloviny značky.

###### <h1 class="nazevknihy">    Honzíkova cesta </h1>

Jako hodnota atributu **class** uvedeme název třídy (jméno, které jsme si pro danou třídu vymysleli). Název třídy musí splňovat následující pravidla:

* nemůže obsahovat české znaky s diakritikou (ěščřž...)
* nemůže obsahovat speciální znaky (@#$%?!...)
* nemůže začínat číslem (ale uprostřed nebo na konci názvu čísla být mohou)
* nemůže obsahovat mezery
* pokud chceme víceslovný název, musíme pro oddělení slov použít buď pomlčku (-) nebo podtržítko (\_), což jsou povolené speciální znaky

Stejnou CSS třídu můžeme v HTML přidat k **libovolnému počtu různých elementů**. Všechny elementy s touto CSS třídou získají vlastnosti pro tuto třídu nastavené v CSS.

#### Selektor typu třída

Chceme-li stylovat CSS třídu, musíme použít selektor typu třída. Selektor **začíná tečkou**, za kterou následuje název třídy.

###### .nazevknihy {    font-size: 60px;    color: gold; }

**Selektor typu třída má vyšší prioritu než selektor typu element**, a to i v případě, kdyby byl selektor typu element zapsaný v CSS jako poslední. Má-li tedy nějaký prvek nastavenou stejnou vlastnost zároveň jako element i pomocí CSS třídy, zvítězí hodnota nastavená pro danou vlastnost pomocí třídy.

Máme-li například následující kód:

###### <h1 class="nazevknihy">    Honzíkova cesta </h1>

###### .nazevknihy {    color: gold; }

###### h1 {    color: red; }

Náš nadpis je element **h1** a zároveň má na sobě nastavenou třídu nazevknihy. Třída má v CSS nastavenou zlatou barvu, element h1 barvu červenou. Navíc je zápis s červenou barvou v CSS později. Mohlo by se tedy zdát, že minimálně podle pravidla "poslední zápis v CSS vyhrává" by mělo platit, že nadpis bude červený. Selektor typu třída má ale vyšší prioritu (přesný odborný termín v CSS je, že má vyšší **specificitu**).

Při moderním způsobu psaní CSS se snažíme obecné elementy stylovat co nejméně a na psaní většinu stylů aplikovat na HTML právě přes CSS třídy.

### Kontrolní otázky

* K čemu slouží značka <div>?
* Jaké hlavní sémantické elementy pro členění obsahu znáš?
* Kdy se hodí použít <div> a kdy sémantické elementy?
* Jak se jmenuje atribut, kterým se v HTML k elementu přidává CSS třída?
* Jak se zapisuje selektor typu třída v CSS?
* Lze danou CSS třídu použít pouze pro jeden element, nebo ji na stránce můžu použít, kolikrát chci?
* Má vyšší prioritu CSS selektor typu element nebo typu třída?

### Kontextový selektor

V předchozích lekcích jsme používali jednoduché selektory typu element a třída. Jsou ale situace, kdy potřebujeme selektorem vybrat prvek v kontextu jiného prvku. Například chceme nadpis uvnitř konkrétní sekce, nebo odkaz uvnitř menu, apod.

Přesně k tomu slouží kontextový selektor:

###### /\* selektor pro nadpis <h2> uvnitř <section> section h2 {  color: red; }

###### /\* selektor pro výběr obrázku uvnitř prvku s třídou fotogalerie \*/ .fotogalerie img {  width: 300px; }

Kontextový selektor se interpretuje zprava doleva. Nastavujeme vlastnosti prvku uvedeného vpravo za mezerou, který je vnořený (je potomkem) prvku uvedeného vlevo.

Kontextovému selektoru se také někdy říká **selektor typu potomek**.

### Box model

Každý prvek v HTML je v prohlížeči zobrazen jako obdélník neboli box. I když prvek nemá rámeček nebo barvu pozadí a box není ve skutečnosti vidět, je do něj prvek uzavřen.

Přesvědčit se o tom můžeme například otevřením vývojářských nástrojů v prohlížeči (stiskni klávesu F12) a kliknutím na libovolný prvek v zobrazeném HTML zdrojovém kódu. Prohlížeč nám v okně se stránkou zobrazí box pro daný prvek.

**Box model je souhrnný název pro skupinu CSS vlastností, které ovlivňují vzhled a chování tohoto boxu (obdélníku).**

Do box modelu patří vlastnosti pro nastavení velikosti prvku, rámečku, odsazení obsahu prvku, odstup od okolních prvků, typ zobrazení prvku, apod.

### Velikost prvku

Velikost prvku nastavujeme pomocí CSS vlastností **width** a **height**. Můžeme použít libovolné délkové jednotky: px, em, rem, vw, vh, % a mnoho dalších,

###### /\* nastaví velikost obrázku \*/ img {  width: 400px;  height: 300px; }

Nastavujeme-li velikost obrázku, můžeme použitím nesprávného poměru mezi výškou a šířkou obrázek zdeformovat. Obrázek má implicitní poměr stran - je v grafickém editoru uložený v nějaké konkrétní velikosti.

Pokud tedy nastavujeme velikost obrázku, je obvykle snazší nastavit pouze jeden rozměr (šířku nebo výšku) a prohlížeč druhý rozměr sám dopočítá tak, aby zůstal obrázek ve stejném poměru stran jako originál.

#### Nastavování velikosti v procentech

Velikost můžeme nastavovat i v procentech. V tom případě se velikost prvku spočítá jako procenta z prostoru, který prvku dává k dispozici jeho rodičovský prvek.

Pokud máme obrázek vložený jen mezi značky <body>…</body>, bude šířka vypočítaná jako daný počet procent ze šířky stránky. Je-li obrázek (nebo jakýkoliv jiný prvek) vložený například dovnitř sekce, bude šířka uvedená v procentech spočítána z šířky, kterou vnořenému prvku sekce poskytuje.

Příklad:

###### <!-- máme následující HTML kód --> <div class="prvni">  První obdélník </div> <div class="druhy">  Druhý obdélník   <div class="treti">    Třetí obdélník  </div> </div>

###### /\* a následující CSS kód \*/ .prvni {  width: 50%; } .druhy {  width: 30%; } .treti {  width: 50%; }

V příkladu uvedeném výše bude první obdélník zabírat 50% šířky stránky, druhý obdélník 30% šířky stránky. Třetí obdélník je vnořený do druhého (je jeho potomkem), takže bude zabírat 50% šířky druhého obdélníku, tj. 50% z 30%, tedy 15% šířky stránky.

#### Rámeček

Rámeček (obrys) obdélníku se nastavuje pomocí vlastností, které začínají slovem **border**. Rámečku můžeme nastavit tloušťku čáry **(border-width**), styl čáry (**border-style**) a barvu čáry (**border-color**).

###### /\* přidáme rámeček široký 3px, plnou čarou, červenou barvou \*/ img {  border-width: 3px;  border-style: solid;  border-color: red; }

Pro nastavení rámečku existuje i vlastnost **border**, která funguje jako zkratka, kterou jdou nastavit všechny tři hodnoty najednou. Pro nastavení rámečku se používá nejčastěji. Hodnoty se oddělují mezerou a na jejich pořadí nezáleží:

###### /\* stejný rámeček jako v předchozím příkladu \*/ img {  border: 3px solid red; }

Jako styl čáry můžeme uvést hodnoty **solid**(plná čára), **dashed**(čárkovaná), **dotted**(tečkovaná), **double**(dvojitá) a další, viz. např. článek na [CSS Tricks](https://css-tricks.com/almanac/properties/b/border/).

Rámeček nemusíme vždy nastavovat kolem dokola celého prvku, ale lze ho nastavit na libovolné straně boxu (nebo kombinaci více stran) pomocí vlastností **border-left**, **border-right**, **border-top** a **border-bottom**. Zápis je stejný jako při použití zkratky **border**.

###### img {  border-left: 3px solid red;     /\* rámeček vlevo \*/  border-right: 10px dashed blue; /\* rámeček vpravo\*/  border-top: 1px dotted green;   /\* rámeček nahoře \*/  border-bottom: 7px doublepink;  /\* rámeček dole\*/ }

#### Zaoblené rohy

Další vlastnost, která souvisí s nastavením rámečku, je vlastnost **border-radius**. Pomocí této vlastnosti můžeme nastavit zaoblené (zakulacené) rohy boxu.

###### /\* zakulacené rohy \*/ img {  border-radius: 20px; }

Hodnota za **border-radius** představuje poloměr zakulacení. Lze ji nastavit v libovolných délkových jednotkách, ale nejčastěji se nastavuje v pixelech.

Zajímavé je i použití s procenty. Pokud nastavíme zaoblení rohu na 50% a máme čtvercový prvek, dostaneme jako výsledek kolečko. To se na webu často používá pro různé profilové obrázky apod.

###### img {  border-radius: 50%; }

Uvedeme-li za**border-radius** pouze jedno hodnotu, použije se stejné zaoblení na všechny čtyři rohy. Za **border-radius** ale můžeme napsat **až 4 hodnoty oddělené mezerou** a ty se pak použijí tak, že první hodnota se použije jako poloměr pro levý horní roh boxu, další hodnoty v pořadí pak **po směru hodinových ručiček**. Pomocí této techniky můžete v CSS vyrábět zajímavé tvary.

###### /\* hodnoty se použijí postupně pro levý horní roh pravý horní, pravý dolní a nakonec levý dolní \*/ img {  border-radius: 30px 0 80px 50px; }

### Kontrolní otázky

* **K čemu slouží** kontextový selektor?
* Jakými vlastnostmi nastavujeme šířku a výšku boxu v CSS?
* Z čeho se počítá šířka objektu nastavená v procentech?
* Jaká vlastnost se používá nejčastěji pro nastavení rámečku?
* Vyjmenuj alespoň tři typy čáry, které můžeme použít pro rámeček?
* Jak uděláme v CSS z libovolného prvku kolečko?

### Box model - pokračování

Budeme pokračovat dalšími vlastnostmi **box modelu**, s jehož popisem jsme začali v minulé lekci. Tentokrát se budeme věnovat nastavování rozestupů a vytváření volného prostoru uvnitř a vně boxu.

#### Odsazení obsahu od okraje boxu (padding)

Nechceme-li, aby byl obsah boxu namáčknutý přímo na okraj (nebo rámeček) boxu, musíme přidat odsazení. Odsazení obsahu boxu od jeho okraje se provádí vlastností **padding**. Padding tedy nastavuje odsazení obsahu od okraje boxu **směrem dovnitř boxu**.

###### .box {  padding: 30px; }

Hodnotu vlastnosti **padding** můžeme uvést v libovolných délkových jednotkách, takže kromě **px** samozřejmě i **%**, **em**, **rem**, a další.

Jako hodnotu vlastnosti **padding** můžeme uvést:

* 1 číslo - stejné odsazení se použije na všech stranách boxu
* 2 čísla oddělená mezerou - první hodnota se použije pro horní a dolní odsazení, druhá hodnota pro odsazení vpravo a vlevo
* 4 čísla oddělená mezerou - první hodnota se použije pro horní odsazení, ostatní se pak aplikují postupně na strany boxu po směru hodinových ručiček (tj. vpravo, dole a poslední vlevo)

###### /\* odsazení obsahu od okraje stejné kolem dokola \*/ .box1 {  padding: 30px; }

###### /\* nahoře a dole odsazení 30px, po stranách 50px \*/ .box2 {  padding: 30px 50px; }

###### /\* osazení nahoře 30px, vpravo 50px, dole 70px, vlevo 10px \*/ .box3 {  padding: 30px 50px 70px 10px; }

Potřebujeme-li nastavit odsazení pouze na jednu stranu, můžeme použít vlastnosti **padding-left**, **padding-right**, **padding-top** a **padding-bottom**. Například:

###### .box {  padding-right: 20px; }

#### Rozestup mezi prvky (margin)

Pomocí vlastnosti **margin** můžeme nastavit odsazení prvku od okolních prvků. Pomocí **margin** nastavujeme **mezeru vně boxu**. Co se zápisu týče, funguje vlastnost **margin** úplně stejně, jako výše zmíněný **padding**:

* Hodnoty uvádíme v libovolných délkových jednotkách
* Můžeme uvést 1, 2 nebo 4 hodnoty oddělené mezerou
* Pro nastavení odstupu pouze na jedné straně můžeme použít **margin-left**, **margin-right**, **margin-top**, **margin-bottom**.

###### /\* stejný odstup od prvku na všech stranách \*/ .box1 {  margin: 30px; }

###### /\* nahoře a dole odstup 30px, po stranách 50px \*/ .box2 {  margin: 30px 50px; }

###### /\* odstup nahoře 30px, vpravo 50px, dole 70px, vlevo 10px \*/ .box3 {  margin: 30px 50px 70px 10px; }

###### /\* odstup lze nastavit i pro jednu stranu samostatně \*/ .box4 {  margin-left: 30px;  margin-top: 20px; }

#### Prolínání rozestupů

Pro vertikální rozestupy mezi prvky nastavené pomocí **margin** platí pravidlo, že když jsou nad sebou prvky a obojí mají nastavený **margin**, použije se jako rozestup mezi nimi pouze ten větší z nich. Vertikální rozestupy (**margin**) se tedy nečítají, ale vytváří se něco jako jejich "průnik" (řečeno zjednodušeně a nepřesně).

Má-li horní prvek **margin** směrem dolů nastavený na 50px a spodní prvek **margin** směrem nahoru 30px. nebude výsledný rozestup mezi prvky 80px, ale pouze 50px (ten větší z nich).

Toto pravidlo se anglicky označuje jako "**collapsing margins**".

#### Řídíme rozestup mezi prvky

Pro zvětšení rozestupu mezi dvěma prvky nikdy nepoužíváme značku **<br>**. Neděláme v obsahu prázdné řádky, abychom roztáhli obsah od sebe. Potřebujeme-li mezi prvky větší rozestup, upravíme vždy jejich **margin**.

Spousta prvků má ve výchozím nastavení prohlížeče nastavenou nějakou hodnotu buď **paddingu** nebo **marginu**. Někdy se nám tento výchozí rozestup nehodí a potřebujeme se ho zbavit. Stačí nastavit **margin** nebo **padding** na 0 (nulu). Při nulování nemusíme uvádět jednotky, nula v jakýchkoliv jednotkách je vždy nula.

###### /\* Nadpis <h1> má pod sebou a nad sebou odstup od okolních prvků 30px. Vlevo a vpravo je nulová mezera. Tj. obsah je až do kraje. \*/ h1 {  margin: 30px 0; }

###### /\* vynulujeme výchozí padding u těla stránky \*/ body {  padding: 0; }

### Kontrolní otázky

* Co nastavujeme vlastností padding?
* Co nastavujeme vlastností margin?
* Když uvedu za padding 2 hodnoty, jaký budstup obsahu od okraje boxu?
* Když za margin uvedu 4 hodnoty oddělené mezerou, tak jaký bude rozestup od okolních prvků po všech stranách boxu?
* Jak se chovají okraje (margin), když máme nad sebou dva prvky?

‍

### Obtékání obrázků textem - float

Obtékání obrázků se v CSS nastavuje vlastností **float**. Vlastnost **float**může nabývat hodnot **left**nebo **right**, podle toho, na jaké straně chceme, aby byl obrázek umístěný.

###### img {  float: left; }

Hodnota **left**umístí obrázek vlevo a text ho obtéká zprava. Vlastnost **right**umístí obrázek vpravo a text ho obtéká zleva. Obrázek je obtékaný veškerým textem, který za obrázkem následuje, dokud není obtékání zastaveno.

Vlastnost **float** můžeme nastavit jakémukoliv prvku, nemusíme se omezovat jen na obrázky. Obtékán může být jakýkoliv prvek, který nezabírá celou šířku stránky (nemůže být obtékán textem, když vedle něj není na text místo).

#### Zastavení obtékání - clear

Obtékání lze zastavit pomocí CSS vlastnosti **clear**. Ta může mít hodnotu **left**, **right**nebo **both**, podle toho, jaké obtékání chceme zastavit. Hodnota **both** zastaví obtékání objektu ať už je obtékán zleva nebo zprava.

Vlastnost **clear**se nastavuje **prvnímu prvku**, u kterého už **nechceme**, aby obtékal objekt/obrázek s nastaveným **float**.

Představme si následující příklad:

###### <img src="foto.jpg" alt="Foto">

###### <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Modi libero eaque id maiores temporibus voluptatum porro ipsum voluptatem sequi, enim cupiditate minus in, culpa quas magnam, esse ducimus sint reprehenderit.</p>

###### <h2>Nadpis</h2>

###### <p>Itaque, id dignissimos! Natus recusandae quia facilis magni, error illo, optio fugit sint iste fugiat dolorum, architecto dignissimos. Libero modi placeat quae, alias voluptatum dolorum. Quaerat ipsa voluptates quo eius!</p>

###### img {  float: left; }

###### h2 {  clear: left; }

V příkladu výše je obrázek umístěn vlevo a text ho obtéká zprava. Protože ale nadpis <h2> má zastavené obtékání pomocí vlastnosti **clear**, tak nadpis už nebude obrázek obtékat. Vedle obrázku tedy bude pouze první odstavec. Nadpis a druhý odstavec budou pod obrázkem.

#### Přetečení obtékaného prvku mimo box

Obrázek (nebo jiný prvek), který má nastavené obtékání vlastností **float**, je tzv. vyjmut z toku dokumentu. Pro rodičovský element (prvek, uvnitř kterého je obtékaný obrázek uzavřen) jakoby obrázek přestal existovat. Protože obrázek najednou "neexistuje", rodičovský prvek na něj nebere ohled a upraví svoji výšku tak, aby obklopoval pouze text uvnitř (a další neobtékané prvky).

Kvůli tomuto jevu může někdy dojít k situaci, kdy obtékaný obrázek **přeteče ven mimo hranice**svého rodičovského boxu, když je vedle něho pouze malé množství textu.

###### <div class="box">  <img src="foto.jpg" alt="Foto">  <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Modi libero eaque.</p> </div>

###### img {  float: left; }

Někdy je toto chování žádoucí, ale obvykle mu chceme zabránit a chceme, aby i obtékaný obrázek byl celý uvnitř rodičovského boxu. Nechceme, aby trčel mimo box.

Existuje několik způsobů, jak tomuto přetečení zabránit. Nejjednodušší moderní způsob (nefunguje vůbec nebo jen částečně ve starých prohlížečích), je nastavit boxu vlastnost **overflow**na hodnotu **auto**.

Box se roztáhne tak, aby automaticky obklopoval i prvky, které by z něho normálně přetekly ven, tedy i náš obrázek.

###### .box {  overflow: auto; }

### Obrázkové pozadí

Stejně jako jsme barvu pozadí nastavovali pomocí **background-color**, můžeme prvku nastavit i obrázkové pozadí pomocí vlastnosti **background-image**.

Cesta k obrázku, který má být na pozadí, se uvádí do závorek jako parametr CSS funkce **url()**.

###### .box {  background-image: url(pozadi.jpg); }

#### Opakování pozadí

Obrázkové pozadí se standardně opakuje, tj. když obrázek není dost velký, aby pokryl celý prvek, opakuje se jako dlaždice vedle sebe ve vodorovném i svislém směru.

Opakování pozadí můžeme řídit pomocí vlastnosti **background-repeat**, která může mít hodnotu:

* **repeat-x** - opakuje se pouze ve vodorovném směru
* **repeat-y** - opakuje se pouze ve svislém směru
* **repeat** - opakuje se v obou směrech (výchozí nastavení)
* **no-repeat** - opakování je vypnuté, zobrazí se pouze jedna "dlaždice"

###### .box {  background-repeat: no-repeat; }

#### Pozice pozadí

Obrázek se ve výchozím nastavení umístí na pozadí objektu tak, že je v levém horním rohu. Polohu pozadí uvnitř objektu můžeme řídit pomocí vlastnosti **background-position**, za kterou můžeme uvést několik různých hodnot nebo jejich kombinaci.

* **left top** - levý horní roh (výchozí nastavení)
* **right bottom** - pravý dolní roh
* **center center** - uprostřed, lze zkrátit pouze na jedno **center**
* jakákoliv další kombinace slov **left**, **right**, **top**, **bottom**, **center**
* **30% 70%** - číselné nastavení polohy levého horního rohu obrázku na souřadnice uvnitř boxu. První je horizontální souřadnice, druhá vertikální. **0% 0%** je levý horní roh, **100% 0%** je pravý horní roh, **50% 100%** je uprostřed dole, atd.
* **10px 60px** - viz. výše, v jakýchkoliv CSS délkových jednotkách

###### .box {  background-position: center; }

#### Velikost pozadí

Obrázek na pozadí se ve výchozím nastavení zachovává svou původní velikost (jak je uložen z grafického editoru). Velikost můžeme změnit pomocí vlastnosti **background-size**. Ta může mít hodnoty:

* **100% 100%** (nebo samozřejmě jakoukoliv jinou hodnotu v %) - roztáhne pozadí tak, aby zabíralo na šířku a výšku daná procenta ze šířky boxu. Pokud má nastavený rozměr jiný poměr stran než obrázek, může dojít k deformaci obrázku.
* **50px 30px** - stejné jako viz výše, ale v jakýchkoliv jiných CSS délkových jednotkách.
* **cover**- zvětší pozadí tak, aby nikdy nebylo deformované (zachovalo si poměr stran), ale přitom pokrylo kompletně celý box.

###### .box {  background-size: cover; }

#### Zkrácený zápis

Pro zápis všech vlastností pozadí (**background-color**, **background-image**, **background-position**, **background-size**, **background-repeat**, a dalších**)** můžeme použít zkratku **background**, kterou jdou nastavit všechny tyto vlastnosti najednou.

###### .box {  background: #806030 url(foto.jpg) right 50% / top 30% no-repeat; }

### Kontrolní otázky

* Jakou CSS vlastností nastavujeme objektu obtékání?
* Jako CSS vlastností můžeme obtékání zastavit? Na jaký prvek ji přidáváme?
* Jak zabráníme přetečení obtékaného obrázku mimo hranice boxu?
* Jak v CSS nastavujeme obrázkové pozadí?
* Lze ovlivnit velikost a pozici obrázkového pozadí?
* Jakou CSS vlastnosti se nastavuje opakování obrázkového pozadí?

‍

### Menu a skok na sekce stránky

Menu nebo navigační „rozcestník“ je součástí téměř každé stránky. Náš projekt je tvořený pouze jednou stránkou, takže rozsáhlá navigace nedává velký smysl, ale přesto do projektu přidáme jednoduché menu pod hlavičku. Jednotlivé odkazy v menu budou skákat na sekce naší stránky (O mně, Fotogalerie, Turistický průvodce, Kontakt).

#### Menu

Menu můžeme považovat za jednoduchý **seznam odkazů**. Sémanticky tedy dává smysl, vytvořit menu v HTML pomocí značek pro seznam. Obvykle používáme neseřazený seznam <ul>. Uvnitř každé položky seznamu <li> je odkaz, který povede na sekci stránky. Atribut **href**, kam bychom normálně napsali cílovou adresu odkazu, zatím necháme prázdný:

###### <ul class="menu">  <li><a href="">O mně</a></li>  <li><a href="">Fotogalerie</a></li>  <li><a href="">Turistický průvodce</a></li>  <li><a href="">Kontakt</a></li> </ul>

#### ID prvku

Chceme-li skákat na jednotlivé sekce stránky, musíme je pro prohlížeč nějak identifikovat. K tomu slouží atribut **id**, do jehož hodnoty dáme název identifikátoru.

ID prvku musí být na stránce jedinečné. Je to „číslo občanky“ prvku, prvek jde podle něj jednoznačně identifikovat. Na stránce by nikdy nemělo být více prvků se stejným ID.

ID se může skládat pouze z malých a velkých písmen, číslic, pomlčky, podtržítka, dvojtečky a/nebo tečky. Musí začínat písmenem. Nemůžeme použít znaky s diakritikou. Nejjednodušší je držet se pravidla, kdy vše píšeme jako jedno slovo dohromady, malými písmeny.

Atribut **id** lze přidat k libovolnému prvku, my ho v našem projektu přidáme k jednotlivým sekcím:

###### <section id="omne"> … </section>

###### <section id="fotogalerie"> … </section>

###### <section id="pruvodce"> … </section>

###### <section id="kontakt"> … </section>

#### ID jako cíl odkazu

Když má prvek přidělené ID, můžeme toto ID použít jako cíl odkazu. Do atributu **href**odkazu napíšeme znak mřížky **#** následovaný identifikátorem prvku:

###### <a href="#omne">O mně</a>

Takto napsaný odkaz znamená „na této stránce skoč na prvek, který má identifikátor omne“. Prohlížeč posune obsah stránky v okně tak, aby prvek s daným ID byl co nejvýše na stránce.

V našem projektu to nepotřebujeme, ale skok na konkrétní ID jde použít i v případě, že máme odkaz na jinou stránku:

###### <a href="strana2.html#omne">O mně</a>

Takto napsaný odkaz prohlížeči řekne, že má otevřít stránku strana2.html a na ní se přesunout na prvek s **id="omne"**.

Chceme-li skákat na různá ID v rámci aktuální stránky, neuvádíme adresu stránky (název souboru), ale pouze mřížku následovanou ID prvku.

### Blokové a řádkové prvky

Každý HTML element je určitého typu - blokový, řádkový, řádkově-blokový, a další. Typ prvku určuje, jakým způsobem se prvek zobrazuje uvnitř stránky.

#### Blokové prvky

* zabírají na šířku celý prostor, který mají k dispozici (buď celou šířku stránky nebo celý prostor, který mají k dispozici od svého nadřazeného prvku / rodiče), výška se přizpůsobuje podle obsahu prvku (není-li nastaveno jinak)
* začínají vždy na novém řádku
* blokové prvky se v dokumentu skládají **pod sebe**

Příkladem blokového prvku je například nadpis **<h1>**, **<h2>**, atd., odstavec **<p>**, oddíl **<div>** nebo **<section>**, seznam **<ul>** i položka seznamu **<li>** a mnoho dalších.

#### Řádkové prvky

* v řádku zabírají pouze nezbytně nutnou šířku, dle svého obsahu
* jsou uvnitř řádku, nezačínají v dokumentu nový řádek
* na stránce se v rámci řádku skládají **vedle sebe**

Příkladem řádkového prvku je např. odkaz **<a>**, značky **<em>** a **<strong>** pro zdůraznění textu, a mnoho dalších.

**Řádkovým prvků nejdou nastavit některé vlastnosti box modelu:**

* nejde nastavit šířka a výška
* jde nastavit padding a margin, ale ve vertikálním směru (nahoře a dole) nefunguje „správně“ - prvek má sice nastavený padding nebo margin, ale ten neodsouvá okolní řádky textu od sebe, jak bychom očekávali.

#### Řádkově-blokové prvky

Existují ještě hybridní prvky, neboli prvky řádkově-blokové. Ty kombinují vlastnosti obou předchozích typů. Tj. jdou jim nastavit a správně fungují všechny vlastnosti box-modelu (šířka, výška, padding, margin), ale přitom se skládají v řádku za sebe.

Příkladem takového prvku je například značka **<img>** pro obrázek.

### Změna typu prvku - CSS vlastnost display

Typ prvku můžeme v CSS libovolně měnit. Z prvku, který je ve výchozím stavu prohlížeče blokový, můžeme udělat prvek řádkový nebo řádkově-blokový, a naopak.

Typ prvku se mění CSS vlastností **display**. Vlastnost může mít hodnotu:

* **block**- blokový prvek
* **inline**- řádkový prvek
* **inline-block** - řádkově-blokový prvek
* **none**- skrytý prvek
* a další

Existuje spousta dalších hodnot pro tabulkové prvky, flexbox, css grid a další, které jsou ale mimo rozsah tohoto kurzu.

Zajímavá je hodnota **none**. Když prvku nastavíme **display: none;**, bude prvek skrytý a prohlížeč se bude chovat, jako by v dokumentu neexistoval.

Chceme-li na naší stránce udělat horizontální menu, kde jsou jednotlivé položky vedle sebe, musíme z položek seznamu (blokové prvky) udělat řádkové nebo řádkově-blokové prvky, čímž dostaneme položky na řádek vedle sebe.

###### li {  display: inline-block; }

### Pseudotřídy

V předchozích lekcích jsme si ukazovali CSS třídy, které jsme k vybraným prvkům přiřazovali pomocí atributu **class="trida"**.

Pseudotřídy prvkům nepřidáváme, existují na prvku samy od sebe ve chvíli, kdy je prvek v určitém stavu - například když nad prvek najedeme kurzorem myši, když klikneme na tlačítko, apod.

Pro tlačítka v menu jsou pro nás zajímavé tyto pseudotřídy:

* **hover** - prvek, nad kterým je kurzor myši
* **focus**- vybraný (zaostřený) prvek, ať už pomocí myši nebo klávesnice (stiskem klávesy Tab)
* **active** - prvek, který uživatel právě aktivuje (tj. kliká na něj, ale ještě nepustil tlačítko myši)

Pseudotřídy zapisujeme v CSS za dvojtečku hned za selektorem prvku. Chceme-li například změnit barvu pozadí odkazu, na který se najelo kurzorem myši, zapíšeme do CSS:

###### a {  background-color: yellow; }

###### a:hover {  background-color: red; }

Kód výše způsobí, že odkazy na stránce budou mít žluté pozadí. Ve chvíli, kdy na odkaz najedeme myší, změní se barva na červenou. Stejně se používají i ostatní pseudotřídy. Uvnitř CSS deklarace pak pro daný stav prvku můžeme samozřejmě změnit jakékoliv CSS vlastnosti.

Konkrétně pro odkazy jsou zajímavé ještě pseudotřídy:

* **link** - ještě nenavštívený odkaz
* **visited**- navštívený odkaz, uživatel už na cílové stránce někdy byl

### Kontrolní otázky

* Pomocí jakých značek v HTML vytvoříme menu? (není to nutná podmínka, ale jde o většinovou zavedenou praxi)
* Jaké typy HTML prvků znáš? (podle způsobu, jakým se v dokumentu zobrazují)
* Jaký je rozdíl mezi řádkovým a blokovým prvkem?
* Jako CSS vlastností měníme typ prvku?
* Co je CSS pseudotřída a k čemu se používá?