CSS

Začínáme designovat webové stránky

Tomáš Vondráček
linkedin.com/in/vondracektomas
tomasvondrac@gmail.com

17. října 2024

Zpracováno pro účely výuky programování v jazyce JavaScript

Obsah

Cíle

- Co to je CSS a jak ho propojit s HTML
- Jak se píše CSS kód a jak se pomocí něho tvoří webová stránka
- Schopnost upravit webovou stránku pomocí CSS

Úvod do CSS

Co to je CSS a k čemu slouží

CSS (**C**ascading **S**tyle **S**heets) je jazyk pro popis zobrazení HTML kódu

- **C**ascading (kaskádové) charakterizuje nejdůležitější vlastnosti stylů, a to dědění stylů (od rodičů k potomkům). Pokud nastavíme rodiči (nějaké značce), že má modré pozadí, tak pak i všichni potomci (značky uvnitř rodiče) budou mít modré pozadí.

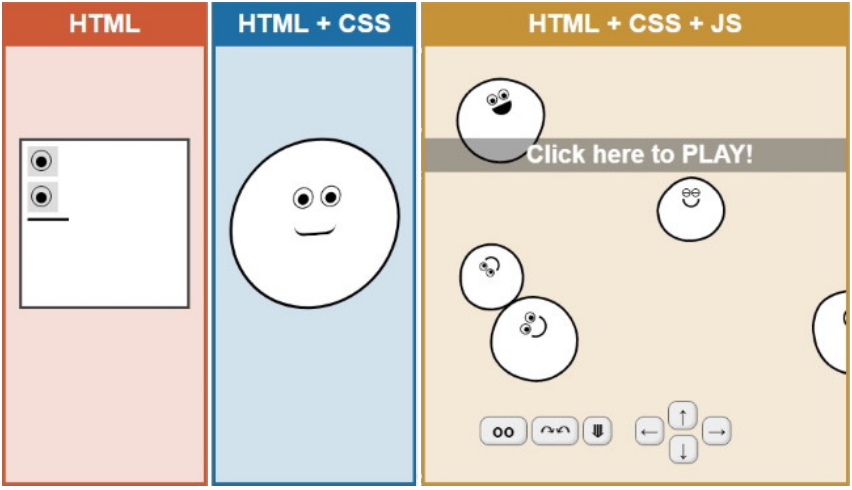
Příklad

- My už jsme se ale s děděním „stylů“ setkali i v HTML, jak to fungovalo?

CSS kód je tedy tvořen pravidly, která:

- jsou vždy přiřazena k jedné, nebo více značkám
- dohromady definují, jak celá stránka vypadá

Jak spolu souvisí HTML, CSS a JS



Obrázek 1: Obrázek převzat z https://html-css-js.com.

Vývoj verzí CSS

Minulost

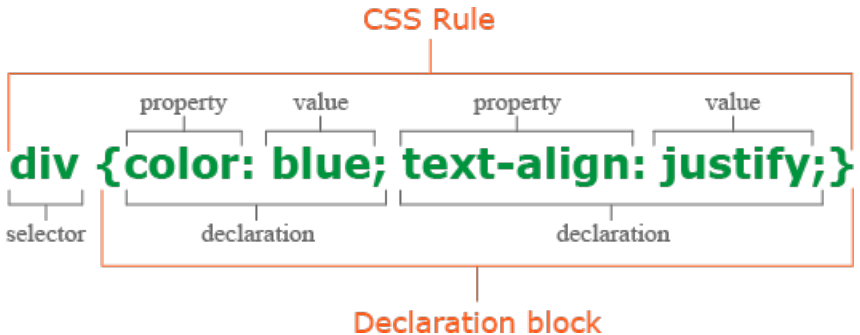
- První verze CSS 1 byla publikována na konci roku 1996
- Druhá verze CSS 2 byla publikována v roce 1998 a přidala možnosti pozicování

Aktuálně

- Aktuálně se používá třetí verze CSS 3. Nové vlastnosti přidané v této verzi se však neustále vyvíjí a postupně jsou adaptovány v prohlížečích.

JAK VYPADÁ CSS KÓD

CSS (kaskádové styly) jsou skupinou určitých pravidel, které se skládají z vlastností a jejich hodnot (podobně jako atributy v HTML).



Obrázek 2: Převzato z https://puzzleweb.ru/en/css/1_css_syntax.php

JAK PŘIPOJIT CSS KÓD DO HTML KÓDU

Způsoby připojení CSS do HTML se liší podle toho, kde se CSS kód nachází:

1. **externí CSS** – CSS je zapsáno v externím souboru, např. `styles.css` a ten je pak připojen do HTML
2. **interní CSS** – CSS je zapsáno uvnitř HTML souboru ve značce `style`
3. inline CSS – CSS je zapsáno jako hodnota atributu `style` u dané značky

Inline přípojení CSS

Inline CSS jsou kaskádové styly zapsané přímo v atributu `style` u dané značky.

```
<p style="color: red">červený text</p>
```

Tento přístup má několik nevýhod:

- Pokud bych chtěl na červenou obarvit každý odstavec, musel bych daný styl neustále kopírovat – a co kdybych se později rozhodl, že text v odstavci nebude červený, ale modrý?
- Pokud bych potřeboval vložit více vlastností, tak by se kód stal velmi rychle nepřehledný
- Když prohlížeč načítá webovou stránku, musí vždy načíst i dané styly, protože jsou uvnitř souboru HTML

INTERNÍ PŘÍPOJENÍ CSS DO HTML

Interní CSS jsou kaskádové styly zapsány uvnitř značky `style`, která se musí nacházet uvnitř hlavičky (značka `head`). Stylovat tak lze celou stránku z jednoho místa.

```
<head>
    <style>
        p {
            color: red;
        }
    </style>
</head>
<body>
    <p>červený text</p>
</body>
```

EXTERNÍ PŘÍPOJENÍ CSS DO HTML

Externí připojení CSS souboru do HTML souboru je nejlepší možnost, jak připojit CSS, protože:

- Stejně jako u interního CSS lze pravidla použít na libovolné množství značek
- Nyní je však CSS kompletně odděleno od HTML a tím se oba soubory stávají přehlednější
- Pokud naše webová stránka má více stránek a na každé stránce načítá CSS soubor, tak tento soubor načte pouze 1×. Tomuto mechanismu zapamatování souboru se říká **cache paměť** prohlížeče.

```
<head>
    <link href="styles.css" rel="stylesheet" />
</head>
<body> <p>červený text</p> </body>
```

Shrnutí vlastností 3 způsobů připojení CSS do HTML

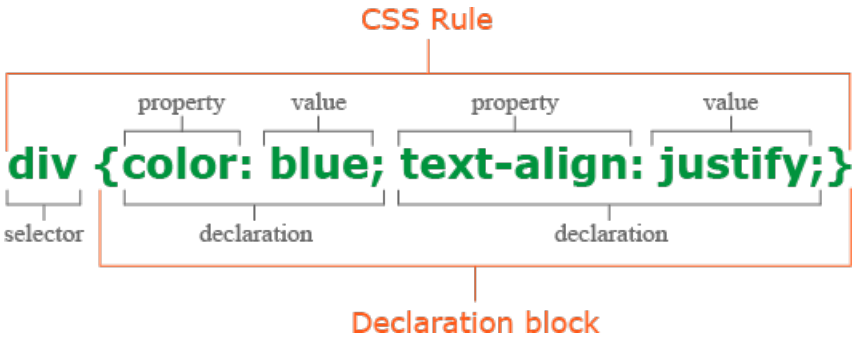
	Znovupoužitelnost	Rozdělení zodpovědnosti	Rychlost načítání
Externí CSS	✓	✓	✓
Interní CSS	✓	∼	✗
Inline CSS	✗	✗	✗

Tabulka 1: Vždy chceme mít možnost znovupoužívat již vytvořena pravidla, chceme, aby tyto pravidla byla oddělena od HTML a chceme, aby se stránka rychle načítala.

Selektory

SELEKTORY

Selektory určují, na jaké značky se deklarované styly aplikují.



Obrázek 3: V tomto případě se styly aplikují na všechny značky `div`.

Jak můžeme selektovat (vybírat) značky

Selektory můžeme řadit od nejvíce obecného po nejvíce specifický:

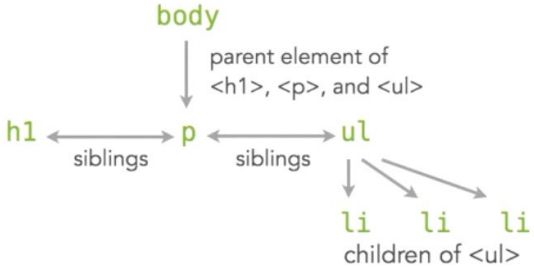
- **Značka** – značky vybíráme na základě jejich názvu (`div`, `p`, ...)
- **Třída** – značky vybíráme na základě přiřazené třídy, což je hodnota v atributu `class`
- **Identifikátor** – značku vybereme na základě přiřazeného identifikátoru, což je hodnota v atributu `id`

Ale nejen to... značky můžeme vybírat i na základě atributů a jejich hodnot. Výše zmíněné spolu lze dále kombinovat, např. vyberu pouze ty značky, které mají určitou třídu.

VZTAHY MEZI ZNAČKAMI (RODIČ × DÍTĚ × POTOMEK)

HTML je reprezentováno jako **DOM** (Document Object Model), což je stromová struktura.

```
<body>
  <h1>Heading</h1>
  <p>Paragraph.</p>
  <ul>
    <li>List item</li>
    <li>List item</li>
    <li>List item</li>
  </ul>
</body>
```



Obrázek 4: Převzato z
https://medium.com/@jubileekim/css-essential-1-selectors-typography-layouts-cf9a83293e96

Přehled základních selektorů

Selektor	Příklad	Popis
element	p	Vybere všechny `<p>`
.class	.auto	Vybere všechny značky s `class="auto"`
#id	#nadpis	Vybere značku s `id="nadpis"`
*	*	Vybere úplně všechny značky
.class1.class2	.name1.name2	Vybere značky s `class="name1 name2"`
.class1 .class2	.name1 .name2	Vybere značky s `class="name2"`, kteří jsou potomci značek `class="name1"`
.class1, .class2	.name1, .name2	Vybere značky s `class="name1"` a `class="name2"`
element.class	p.name	Vybere značky `<p class="name">`
element > element	div > p	Vybere všechny `<p>`, jejichž rodič je `<div>`
[attribute=value]	[target=_blank]	Vybere značky s atributem `target="_blank"`

Tabulka 2: Celý přehled selektorů: https://www.w3schools.com/cssref/css_selectors.asp

POJMENOVÁNÍ TŘÍD A IDENTIFIKÁTORŮ

- Jméno selektoru **nesmí začínat číslicí**
- Jméno typicky začínají malými písmeny
- V případě **víceslovného názvu** proměnné používáme **(kebab-case)**, tj. jednotlivá slova jsou oddělena pomlčkou (`-`), např. `user-table`, `phone-number`, `description-list`, ...

Kódovací výzva #3

docs.google.com/document/d/15OBVuBMWWPVx3I1A5PH_7wKqSs048PvlRxtpCJI1v

- **Cílem je obarvit text podle popisu v komentářích**

K ZAMYŠLENÍ...

PŘÍKLAD

- Jakou barvu bude mít text v odstavci?
- Jak se změní situace, pokud bych do CSS přidal třídu nebo identifikátor?

```
<style>
    p {
      color: red;
    }
    p {
      color: blue;
    }
 </style>

 <p id="unikatni-barva" class="barva">Jakou budu mít barvu?</p>
```

CSS SPECIFICITA

Každý selektor má určitou specificitu, což je číslo, které určujte jeho prioritu při uplatňování stylů.

Selektory	Specificita
Elementy, pseudoelementy (`p`, `::before`)	`0,0,0,1`
Třídy, pseudotřídy, atributy (`.box`, `:hover`, `[lang=cs]`)	`0,0,1,0`
Identifikátory (`#box`)	`0,1,0,0`
Inline styly (style="...")	`1,0,0,0`

Tabulka 3: Pěkná vizualizace počítání specificity: https://specificity.keegan.st

- Selektor (`*`) a operátory (`>`, `+`, ∼, mezera) a negace (`:not`) nemají na specificitu vliv
- Použití `!important` zajistí, že daný styl bude aplikován bez ohledu na specificitu
- Pokud má více selektorů stejnou specificitu, pak se použije poslední selektor

Záleží na pořadí tříd uvnitř atributu class?

Příklad

- Jakou barvu bude mít text v odstavci?

```
<style>
    .a {
      color: red;
    }
    .b {
      color: blue;
    }
 </style>

 <p class="a b">Jakou budu mít barvu?</p>
 <p class="b a">Jakou budu mít barvu?</p>
```

CSS vlastnosti

Barvy textu nebo pozadí

Barvu textu nastavuje vlastnost `color` a barvu pozadí vlastnost `background-color`. Barvy lze zapsat 3 způsoby:

Jméno barvy	RGB barva	HEX barva
`black`	`rgb(0,0,0)`	`#000000`
`red`	`rgb(255,0,0)`	`#ff0000`
`green`	`rgb(0,255,0)`	`#00ff00`
`blue`	`rgb(0,0,255)`	`#0000ff`
`white`	`rgb(255,255,255)`	`#ffffff`

Tabulka 4: Všeobecně známé hodnoty barev.

Barvě můžeme nastavit **průhlednost** pomocí `rgba(18, 67, 201, 0.5)`. Průhlednost se zadává v rozmezí 0–1 a nulu před desetinnou tečkou je možné vynechat, tj. 0.5 je totéž co .5. Písmeno a v RGB značí tzv. alfa kanál a udává onu průhlednost.

KDE LZE NASTAVIT BARVU

- **Barva textu** se nastaví např. `color: red`
- **Barva pozadí** se nastaví např. `background-color: rgba(123, 13, 55, 0.3)`
- **Barva ohraničení** se nastaví např. `border-color: #abc` ale ještě se musí nastavit i typ ohraničení, např. `border-style: solid`
  - Případně lze použít zkrácený zápis: `border: 1px solid #abc`
- **Barva stínu** se nastaví vlastností `box-shadow`. Tato vlastnost má velké množství konfigurací, takže je často lepší si stín vygenerovat, např. cssmatic.com/box-shadow

CSS JEDNOTKY (ANGL. CSS UNITS)

Stejně jako v životě, tak i v CSS potřebujeme měřit šířku (`width`) a výšku (`height`) prvků nebo velikosti písma (`font-size`). K tomu slouží několik jednotek, které se rozdělují na `absolutní` a `relativní`.

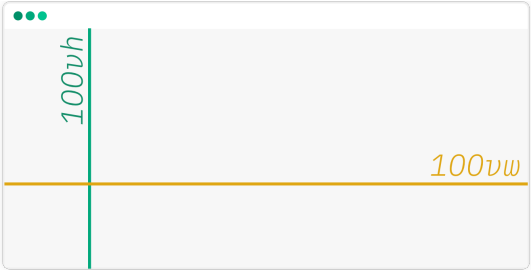
Přehled nejpoužívanějších jednotek:

- **Absolutní jednotky** – Absolutní jednotky jsou fixní a velikosti se neměnní.
  - **`px`** – Pixel je základní CSS jednotka (budeme pracovat hlavně s ní)
- **Relativní jednotky** – Relativní jednotky mění velikost na základě jiné velikosti
  - `%` – Procenta mění velikost (prvku, písma, ...) relativně k velikosti rodičovského prvku
  - **`em`** – Pokud je jednotka `em` použita pro velikost písma, pak je velikost relativní vzhledem k velikosti písma rodičovského prvku. Při použití u jiných vlastností je relativní vůči velikosti písma samotného prvku. Výpočet: `počet_em * velikost_pisma_px = velikost_px`
  - **`rem`** – Jednotka `rem` (`root em`), vychází z velikosti písma kořenového prvku, což je prvek `<html>`. A pokud prvek `<html>` nemá specifikovanou velikost písma, použije se výchozí velikost prohlížeče, tj. `16 pixelů`.

POKRAČOVÁNÍ RELATIVNÍCH CSS JEDNOTEK

Poslední **2 relativní jednotky**, které si ukážeme, jsou podobné procentům (`%`). Liší se tím, že nejsou relativní vůči rodičovskému prvku, ale vůči velikosti výřezu zobrazení webové stránky (angl. viewport).

- **`vh`** (**v**iewport **h**eight) – Např. `50vh` se rovná `50%` výšce výřezu zobrazení.
- **`vw`** (**v**iewport **w**idth) – Např. `100vw` se rovná `100%` šířce výřezu zobrazení.

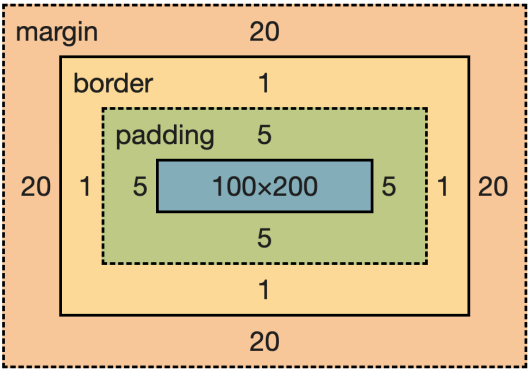


Formátování písma/textu

- **`font-size`** – Změna velikosti písma.
- **`font-style`** – Změna písma na zkosené. Hodnota: `italic`
- **`font-weight`** – Změna tučnosti písma.
  - Hodnota: `bold`, `normal`, nebo čísla mezi `100`–`900`. Zkratky `bold` = `700` a `normal` = `400`
- **`font-family`** – Změna písma. Hodnota může být jedno písmo, nebo i seznam písem a prohlížeč vybere první písmo, které je dostupné.
  - Hodnota: `serif`, `monospace`, `fantasy`, ...
  - Spousta kvalitních bezplatných písem: https://fonts.google.com
- **`text-align`** – Horizontální zarovnání textu
  - Hodnota: `center`, `left`, `right`, `justify`
- **`text-decoration`** – Dekorace textu
  - Hodnota: `underline`, `overline`, `line-through`, `underline wavy`, ...
- **`line-height`** – Nastavení výšky řádku
- **`letter-spacing`** – Nastavení mezer mezi písmeny
- **`word-spacing`** – Nastavení mezer mezi slovy

Box model

Každý prvek na stránce je reprezentován jako obdélníkový box, který má nějakou šířku a výšku (width×height) a který může definovat: odsazení mezi obsahem a ohraničením (`padding`), ohraničení (`border`) a odsazení mezi boxem a okolím (`margin`).



BOX MODEL – ŠÍŘKA, VÝŠKA A VLASTNOST `DISPLAY`

Šířka a výška lze ve výchozím nastavení aplikovat pouze na blokové prvky, tj. prvky, které mají nastavenou vlastnost `display: block`. Všechny řádkové prvky, např. `span`, mají ve výchozím nastavení `display: inline`, a tudíž na ně nelze aplikovat šířka a výška.

Nastavení šířky a výšky na řádkové prvky:

- **`display: inline-block`** – Prvek bude zobrazen jako řádkový, ale lze nastavit šířku a výšku
- **`display: block`** – Prvek bude zobrazen jako blokový

Prvky lze na stránce i zneviditelnit:

- **`display: none`** – Prvek na stránce nezabere žádné místo
- **`visibility: hidden`** – Prvek na stránce zabere místo, ale nebude vidět

BOX MODEL – BORDER

Jednotlivé vlastnosti pro definování ohraničení:

- **`border-width`** – Šířka ohraničení
- **`border-style`** – Typ ohraničení (`dotted`, `solid`, `double`, `dashed`)
  - Typ ohraničení lze nastavit různé na všechny strany obdélníku
- **`border-color`** – Barva ohraničení

Zkrácený zápis ohraničení: **`border: border-width border-style border-color`**

- `border: 1px solid red`
- S ohraničením lze tvořit různé tvary: w3schools.com/howto/howto_css_shapes.asp

BOX MODEL – PADDING A MARGIN

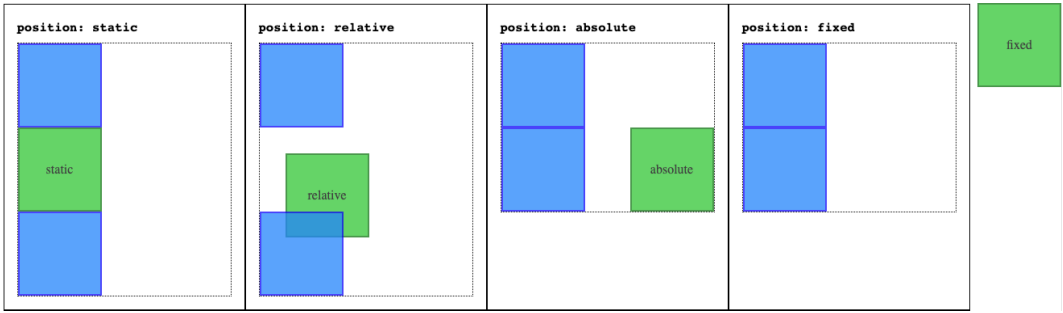
Odsazení uvnitř boxu (`padding`) a odsazení zvenku boxu (`margin`). Odsazení lze nastavit v obou případech nastavit pro všechny strany obdélníku separátně.

- `margin: 10px` – Odsazení je ze všech stran `10px`
- `margin: 10px 20px` – Vertikálně `10px` a horizontálně `20px`
- `margin: 10px 15px 20px` – Z vrchu `10px`, horizontálně `15px` a zespodu `20px`
- `margin: 10px 15px 20px 25px` – Je zkrácený zápis těchto vlastností:
  - `margin-top` – Z vrchu `10px`
  - `margin-right` – Zprava `15px`
  - `margin-bottom` – Zespodu `20px`
  - `margin-left` – Zleva `25px`

POZICOVÁNÍ

Pozice prvků na stránce je ve výchozím nastavení ovlivněna pouze okolními prvky. Pozici lze měnit vlastností `position` s výchozí hodnotou `static`.

- Prvkům lze měnit pozice vlastností `top`, `right`, `bottom` a `left`

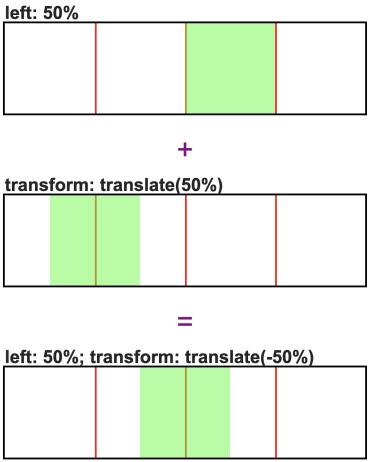


Obrázek 5: Interaktivní ukázka pozicování: https://codepen.io/huijing/pen/PowbeXJ

Pozicování v krocích

1. Prvku nastavíme vlastnost `position` na jinou hodnotu než výchozí, např:
   - **`relative`** – Prvek není vyjmut z toku dokumentu.
     - Pozice je relativní vůči původní pozici
   - **`absolute`** – Prvek je vyjmut z toku dokumentu, ale reaguje na posun na stránce
     - Pozice je relativní vůči nadřazenému prvku, který má jinou pozici než `static`. Pokud žádný takový prvek není, tak je to až prvek `body`.
   - **`fixed`** – Prvek je vyjmut z toku dokumentu a nereaguje na posun na stránce
     - Pozice je relativní vůči výřezu webové stránky a nemění pozici s posunem na stránce
2. Poté lze nastavit vlastnosti `top`, `right`, `bottom` a `left`
   - Hodnoty těchto vlastností jsou typicky v pixelech, nebo procentech

JAK CENTROVAT POZICOVÁNÍ



Zelený čtverec je posunutý o 50 % šířky rodiče doprava ke svému levému okraji.

Zelený čtverec je posunutý o 50 % své šířky doprava.

Nejprve zelený čtverec posuneme o 50 % šířky rodiče doprava, ale pak ho posuneme o 50 % své šířky doleva, a tím pádem bude zarovnány na střed.

Obrázek 6: Kód vytvořeného obrázku: https://codepen.io/vondry/pen/yLPddqY

Kódovací výzva #4

docs.google.com/document/d/1MOvXOxBLq3O0i7DfxS9cL6ZJ2I6D6RC9n9dJlAPzDAA

- **Cílem je rozmístit nad obrázek pomocí pozicování 5 textů**

Psuedoelementy

PSUEDOELEMENTY

Pseudoelementy umožňují ve stránce vybrat „virtuální elementy“, které nejsou přímo součástí stránky, ale může být užitečné s nimi pracovat a přiřadit jim odlišný způsob zobrazení.

Pseudoelementy umožňují:

- vybrat první písmeno textu,
- vybrat první řádek textu,
- vložit text/obrázek před nebo za prvek
- a mnoho dalšího. . .

Výběr prvního písmena v textu

CSS kód:

```
p::first-letter {
    color: red;
}
.flb::first-letter {
    color: blue;
}
```

Výsledek v prohlížeči:

Red

Blue

HTML kód:

```
<p>Red</p>
<span class="flb">Blue</span>
```

Výběr prvního řádku v textu

CSS kód:

```
p::first-line {
    color: green;
}
```

HTML kód:

```
<p>Lorem ipsum dolor sit
consectetur adipisicing elit.
Quam in quisquam a incidunt
exercitationem? Qui deserunt
porro error reprehenderit,
perferendis magnam.</p>
```

Výsledek v prohlížeči:

Lorem ipsum dolor sit consectetur adipisicing elit. Quam in quisquam a incidunt exercitationem? Qui deserunt porro error reprehenderit, perferendis magnam.

VLOŽENÍ TEXTU/OBRÁZKU PŘED NEBO ZA PRVEK

CSS kód:

```
p::before {
    content: "Před";
    color: pink;
}
p::after {
    content: url(obrazek.png);
}
```

HTML kód:

```
<p>Odstavec</p>
```

Výsledek v prohlížeči:

PředOdstavec<obrazek.png>

Pokročilejší ukázka vložení textu z atributu prvku (Odznak/Badge)

CSS kód:

```
.badge {
    position: relative;
}

.badge::after {
    content: attr(data-value);
    position: absolute;
    top: -10px;
}
```

Výsledek v prohlížeči:

Odznak[4]

HTML kód:

```
<span class="badge" data-value="4">Odznak</span>
```

Psuedotřídy

PSUEDOTŘÍDY

Psuedotřídy umožňují zachytit prvek v určitém **stavu**, nebo stejně jako pseudoelementy, vybrat nějakou jeho část. Od pseudoelementů se liší tím, že se před jejich názvem místo dvojitých dvojteček (::), píše jenom jedna dvojtečka (:).

Psuedotřídy umožňují:

- zachytit stav, kdy uživatel pohybuje myší nad prvkem,
- rozlišit stavy u odkazů (navštívený $\times$ nenavštívený),
- vybrat první, poslední, n-té dítě prvku,
- a mnoho dalšího...

ZACHYCENÍ POHYBU MYŠI NAD PRVEK

Selektor: `:hover`

CSS kód:

```
p:hover {
    color: white;
    background: black;
}
```

HTML kód:

```
<p>Hover over me</p>
```

Zachycení stavu odkazu

Selektory, které je vhodné aplikovat v následujícím pořadí:

- `:link` – Odkaz, který ještě uživatel nenavštívil
- `:visited` – Odkaz, který uživatel navštívil
- `:hover` – Odkaz, nad který uživatel právě umístil myš
- `:active` – Odkaz, na který uživatel právě klikl

Pokud nebudou selektory aplikovány v tomto pořadí a všechny budou nastavovat stejnou vlastnost, např. barvu, tak nastavení barvy nebude fungovat tak, jak bychom očekávali kvůli pořadí (udávající prioritu) selektorů.

Výběr prvků

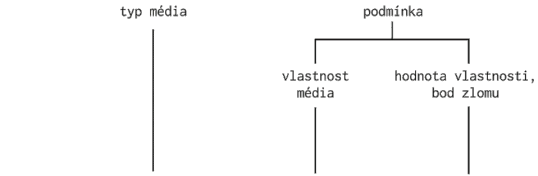
- `:first-of-type` – Výběr prvního prvku mezi skupinou sourozeneckých prvků
- `:last-of-type` – Výběr posledního prvku mezi skupinou sourozeneckých prvků
- `:not(selektory)` – Výběr prvků, které neodpovídají seznamu selektorů.
  - `div:not(:first-of-type)` – Mezi sourozenci `div` vybere všechny vyjma prvního
- `:nth-child(2)` – Vybere druhý prvek mezi sourozenci
  - `li:nth-child(2n)` – Vybere každý druhý prvek `li` mezi sourozenci
  - `li:nth-child(even)` – Vybere každý sudý prvek `li` mezi sourozenci
  - `li:nth-child(odd)` – Vybere každý lichý prvek `li` mezi sourozenci

Media Queries

MEDIA QUERIES

Dotazování na vlastnosti média (angl. Media Queries) představují **způsob, jak získat informace o zobrazovacím zařízení, na kterém je právě zobrazována webová stránka.**

- Pro počítač je typickým zobrazovacím zařízením monitor a pro mobilní zařízení zase jejich displej.



Obrázek 7: Obrázek převzat z: vzhurudolu.cz/prirucka/css3-media-queries

Bod zlomu (angl. breakpoint)

Bod zlomu představuje bod, ve kterém se něco stane/zlomí v závislosti na dané vlastnosti média. O bodech zlomu tzv. „**breakpointech**“ mluvíme jako o sadě nějakých hodnot (rozměrů), které jsou specifické pro daný web.

Breakpoint	Class infix	Dimensions
X-Small	*None*	<576px
Small	`sm`	≥576px
Medium	`md`	≥768px
Large	`lg`	≥992px
Extra large	`xl`	≥1200px
Extra extra large	`xxl`	≥1400px

Obrázek 8: Knihovna Bootstrap definuje 5 breakpointů.

Implementace breakpointů v knihovně Bootstrap

```
// X-Small devices (portrait phones, less than 576px)
// No media query for `xs` since this is the default in Bootstrap

// Small devices (landscape phones, 576px and up)
@media (min-width: 576px) { ... }

// Medium devices (tablets, 768px and up)
@media (min-width: 768px) { ... }

// Large devices (desktops, 992px and up)
@media (min-width: 992px) { ... }

// X-Large devices (large desktops, 1200px and up)
@media (min-width: 1200px) { ... }

// XX-Large devices (larger desktops, 1400px and up)
@media (min-width: 1400px) { ... }
```

Obrázek 9: Implementace breakpointů v knihovně Bootstrap.

CSS PRO GENEROVÁNÍ PDF A TISK

```
@media print {
    a[href]::after {
        /* Zkopíruj hodnotu atributu href za každý odkaz */
        content: " (" attr(href) ")";
    }

    h2 {
        /* Zalom stránku před každým nadpisem h2 */
        break-before: page;
    }
}
```

Stránku lze *uložit jako PDF*/*vytisknout* po kliknutí pravým tlačítkem na myši a vybráním volby *Tisk...*

Co dalšího se může hodit

KOMENTÁŘE

Může se stát, že si do CSS potřebujeme udělat nějakou poznámku. Řešením tohoto problému jsou **komentáře**.

- Komentář začíná textem `/*` a končí `*/`
- Všechno mezi těmito dvěma texty nebude bráno jako CSS

```
<style>
    p {
        color: red; /* nemenit */
    }
</style>
```

VALIDACE CSS KÓDU

Vždy když vytvoříme webovou stránku je vhodné si otestovat, že jsme neporušili žádné CSS pravidla. Testovat lze `.css` soubory i veřejné webové stránky pomocí URL adresy.

https://jigsaw.w3.org/css-validator/#validate_by_upload

Shrnutí a závěr

SHRNUTÍ

- CSS zapisujeme pomocí pravidel (angl. rules), které se skládají ze selektorů a jednotlivých vlastností
- CSS lze do HTML připojit 3 způsoby, kdy nejefektivnější způsob je pomocí `.css` souboru
- Prvky jsou na stránce reprezentovány jako obdélníkový box (Box model)
- Prvky můžeme na stránce posouvat pomocí vlastnosti `position`
- Kromě klasických prvků existují i pseudoprvky a pseudotřídy
- Na typ zobrazovacího zařízení nebo změny ve velikosti obrazovky lze reagovat skrze dotaz `@media`
- Do CSS lze přidávat komentáře a výsledné CSS bychom měli vždy validovat (ověřovat jeho správnost)

Co dál...

1. Techniky rozložení prvků na stránce (**`Flexbox`**, `Grid`, ~~`Float`~~)
2. CSS preprocesory (**`Sass`**, `Less`)
3. CSS knihovny/frameworky (**`Bootstrap`**, `Tailwind`)
4. Konvence pro psaní CSS (**`BEM`**)
5. Tvořit, tvořit, tvořit, ...

Závěr

Děkuji Vám za pozornost.