# Resetování stylů

Resetování stylů je způsob, kterým můžeme vymazat **základní formátování od prohlížeče**, nastavení uživatel a podobného, když chceme, aby stránka co nejlépe odpovídala na všech zařízeních dle vytvoření vývojářem. Základní formátování často bývá velmi obyčejné a v designu stránky překáží. Resetování je tedy souhrn různých pravidel, které např. nastaví správný typ zobrazování, resetuje písmo a podobně. Resetování stylů často říkáme reset. Nejčastěji se používá odnož známého reset.css https://meyerweb.com/eric/tools/css/reset/.

Důležité je, si uvědomit, že po resetování stylů si musíme vše nastavovat sami a nic nebude tak, jak očekáváme. Často se setkávám s podivem toho, že např. ani nadpisy nemají správné velikosti či tloušťku - to je žádané!

Pro účely našeho předmětu máme své vlastní reset.css, které naleznete v Studijních materiálech a nebo ZDE. Tento reset je nutné používat na všech stránkách určený pro tento předmět a to bez výjimky.

Projděte si náš reset.css níže a prozkoumejte, co dělá:

```
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
    display: block;
}
```

```css
body {
    line-height: 1;
}
ol, ul {
    list-style: none;
}
blockquote, q {
    quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
    content: '';
    content: none;
}
table {
    border-collapse: collapse;
    border-spacing: 0;
}
```

# Obrázky

Obrázky jsou dobrý způsob jak oživit webové stránky. Známý fakt je ten, že obrázky umí lidé číst velmi rychlé a často řekne více, než rozsáhly text. Na webových stránkách nejčastěji používáme vektorové a rastrové obrázky. Sem tam se objeví i animované obrázky (např. GIF). Rastrové obrázky obsahují informace o barvách na jednotlivých pixelech a často se využívá komprese pro jejich zmenšení. Vektorové obrázky obsahují pouze matematické rovnice, které se vypočítávají na zařízení pro zobrazení jednotlivých pixelů. Vektorové obrázky můžeme zmenšovat a zvětšovat bez ztráty kvality.

Vektorové typy obrázků:

- SVG

Rastrové typy obrázků:

- PNG, dovoluje průhledné obrázky
- JPG
- GIF, animované

V HTML vytváříme obrázky pomocí nepárového tagu `img`. Má jeden povinný parametr a to `src`, který určuje odkud se obrázek vezme. Může to být jakákoliv URL a můžeme, jako u směrování, použít i lokální soubory.

Jako nepovinný parametr máme alt, který znační alternativní text, který se zobrazí:

1. když se obrázek nenačte

2. když se stránka zobrazí např. v čtečce či podobném zařízení, kde zobrazení "není možné".

Alternativní text píšeme stručně a jednoznačně.

```
<img src="profiles/cajthaml.matej.png">
```

---

Obrázky se načítají na stránkách ihned po získání HTML stránky (resp. jejich načtení). Obrázek se postupně zobrazuje uživatelovi. Obrázky jsou často velmi velké a je nutné šetřit místo. Proto často při publikaci stránky obrázky:

- zmenšujeme tak, aby byly ve velikosti, která se reálně na stránkách používá
- kompresujeme různými způsoby
- a nebo jich používáme méně (softlimit na stránku by měl být cca 15 středně velkých obrázků)
- (nebo použijeme speciální atribut, který obrázky načte tehdy, kdy je uživatel uvidí - viz. ZDE)

---

Obrázky se chovají jako jakýkoliv jiný obsah na stránce a můžeme je např. obalit odkazem.

```
<a href="https://cajthaml.eu">
    <img src="catttto.png" alt="Kocour s jménem Yoko ležící v tunelu.">
</a>
```

---

Chceme-li, aby obrázek zabíral celou šířku či výšku rodiče, je potřeba jedné z dimenzí nastavit 100% šířku a druhé nastavit hodnotu auto. Tímto způsobem neztratíme tzv. aspect-ratio, tj. poměr (podíl) stran obrázku a on se zbytečně nebude deformovat. Tímto taktéž vytvoříme responzivní obrázek, o tom si však povíme později.

```
img.full-img {
    width: 100%;
    height: auto;
}
```

# Obrázkové pozadí

Na prvky můžeme v CSS zapnout obrázkové pozadí, které za obsah uvnitř elementu vytvoří obrázek. Prozatím se jedná o nejrozsáhlejší vlastnost, kterou jsme potkali - obrázek na

stránce můžeme různé opakovat, posouvat a zvětšovat. Na pozadí můžeme dávat i průhledné obrázky (resp. je kombinovat s `background-color`).

Zdroj obrázků v CSS musíme vždy zaobalit do příkazu URL (`url("./img/test.jpg")`). Při použití relativních cest se bere základ z toho, odkud se dané CSS čte. Jsme-li tedy např. ve složce `/css/administration/styles.css`, tak `./img/test.jpg` v CSS ukazuje na `/css/administration/img/test.jpg`!

```
.block {
    width: 300px;
    height: 300px;
    color: rgb(247, 247, 247);

    /* Vytvoří opakující pozadí z obrázku na ./block.png */
    background-image: url("block.png");
}
```

Opakování můžeme ovládat pomocí vlastnosti `background-repeat`, hodnoty jsou `no-repeat`, `repeat`, `repeat-x`, `repeat-y`, které jsou velmi jménem-říkající.

Pomocí vlastnosti `background-position` můžeme ovládat zobrazení hlavního obrázku (při opakování toho "základního") v oblastni rodiče. Můžeme používat předpřipravené hodnoty, jako je např. `center`, `left` a další ale i jakékoliv jiné jednotky (nejčastěji však `px` a `%`).

Vlastnost `background-size` slouží k určení velikosti hlavního obrázku (při opakování i těch dalších). Můžeme používat různé jednotky a nebo připravené hodnoty např. `cover` a `contain`. `cover` překreje celé pozadí daného prvku a prvek různě deformuje (části obrázku nemusí být vidět), `contain` se pokusí obrázek zobrazit celý.

Nastavení nějakého pozadí tedy může např. vypadat následovně:

```
.block {
    background-image: url("block.png");
    background-repeat: no-repeat;
    background-position: center top;
    background-size: contain;
}
```

Existují i další vlastnosti pro pozadí, těmi se však budeme zaobírat později.

# Nové vlastnosti

# Průhlednost

Vlastnost `opacity` slouží k určení jak moc je prvek průhledný. Dovoluje hodnoty 0 (neviditelné) až 1 (zcela viditelné).

---

```
strike, del {
   opacity: 0.3; /* Viditelné pouze z 30 %. */
}
```

# Kurzor

Vlastnost cursor dovolí změnit kurzor, který se zobrazí při najetí. Používá se zejména s klikatelnými prvky a pseudotřídami. Seznam všech typů naleznete ZDE.

```
.grab-that {
   cursor: grab;
}
```

# Obrys

Vlastnost `outline` slouží k vytvoření obrysu prvku (resp. jeho obdélníku). Je velmi podobné `border` a používá stejné hodnoty, avšak se nepočítá do velikosti prvku, vykresluje se tedy okolo `border`.

```
.important {
    outline: dashed 4px #936dcf;
}
```

# Obrázkové mapy

**Obrázkové mapy** jsou způsob, jak na webové stránce vytvořit část, na které budou různé odkazy. Obrázkové mapy se používají zejména na složité obrázky, kde chceme aby uživatel mohl přímo klikat na mapu, aby provedl nějakou akci (resp. někam se přesměroval). Obrázkové mapy se skládají z obrazců (obdélníků, kruhů a polygonů).

Obrázkové mapy jsou velmi neoblíbeným tématem studentů, a to zejména kvůli tomu, že jsou velmi neintuitivní a jejich použití je vidět pouze změněním kurzoru a klikem. Jednotlivé obrazce v mapě totiž nelze na webové stránce zviditelnit pomocí žádných pravidel.

Obrázkové mapy se definují pomocí tagu `map`, ve kterém se nacházejí jednotlivé obrazce pomocí tagu `area`. Mapě vždy musíme uvést jméno pomocí atributu `name`. Jednotlivým obrazcům nastavujeme atributy `shape` (který reprezentuje typ obrazce, tedy `rect`, `poly` a `circle`), `coords` (jednotlivé pozice), `href` a často i `alt`.

Mapy se definují pouze jednou a je možné je použít pro více obrázků. Mapy a ani obrazce se na prohlížeči nevykreslují. Pro aplikování mapy používáme na obrázku atribut `usemap`, do kterého vložíme s hashtagem jméno, které jsme uvedli u mapy.

Ukázka, v (nesprávně vytvořených) komentářích, máte definice jednotlivých `coords` u typů:

```
<map name="info">
    <area shape="rect"
          coords="184,6,253,27" <!--- Určujeme horní levý a dolní pravý roh v px! --->
          href="#obdelnik" />
    <area shape="circle"
          coords="130,136,60" <!--- Určujeme střed kruhu a jeho poloměr v px! --->
          href="#kruh" />
    <area shape="poly"
          coords="130,6,253,96,223,106,130,39" <!--- Určujeme jednotlivé body v px, posledn
          href="#polygon"
          target="_blank"
          alt="Polygon" />
</map>

<img src="cat.jpg" usemap="#info">
```

# Celostránková stránka

Při vytváření webů často potřebujeme vytvořit stránku, která bude zabírat místo, které je stejně velké, jako velikost monitoru. Pro tyto účely máme v CSS dvě jednotky a to `vh` a `vw`, neboli viewport height a width. Tato jednotka reprezentuje procento z dané dimenze zařízení. 1vh tedy znamená 1% velikosti zařízení na výšku. Pro celostránkové stránky používáme často 100vh a 100vw na šířku na `body`. Dovnitř dáváme již jakýkoliv obsah. Je nutné využívat reset, abychom zde neměli zbytečné odsazení.

V dětech takové stránky již používáme většinou procenta a absolutní jednotky vynecháváme (zejména při responzivitě).

Ukázka celostránkové stránky:

```
<!-- ... -->
<style>
body {
    width: 100vw;
    height: 100vh;
}
.part1 {
    background-color: #facc00;
    height: 30%;
    width: 100%;
}
```

```
.part2 {
    background-color: #00a0fa;
    width: 100%;
    height: 70%;
}
</style>
<!-- ... -->
<body>
    <div class="part1"></div>
    <div class="part2"></div>
</body>
```

# Seznamy

Seznam, často pojmenovávané listy, jsou způsob vypsání bodů na webové stránce. Vypsání může být neseřazené (unordered) a seřazené (ordered). Na webových stránkách jsou taktéž tzv. definiční listy, které známe např. ze slovníku, kdy jeden pojem je definovaný a vysvětlený.

Seznamem si tedy můžeme představit např. postup práce, nákupní seznam, TODO listy a podobné.

V HTML definujeme listy pomocí tagu `ul` (unordered), `ol` (ordered) a `dl` (description). Pro jednotlivé odrážky v `ul` a `ol` používáme tag `li` (list item). Dovnitř položek lze vkládat jakýkoliv další obsah stránek, včetně obrázků a různého formátování.

---

# Seřazené seznamy

```
<ol>
    <li>naučit se na test</li>
    <li>mít hodně bodů</li>
    <li>mít výborný prospěch</li>
    <li>???</li>
    <li>profit</li>
</ol>
```

Seřazené seznamy se využívají tam, kde víme jak jsou položky seznamu seřazeny a je tato informace důležitá. Při tvorbě webových stránek je dobré si rozmyslet, zda informace jak jsou položky očíslované (označeny) není zbytečná a nezaplní webovou stránku vizuálním smogem.

U tagu `ol` lze nastavit speciální atribut `type`, který určuje, jak se položky budou číslovat. Základním číslováním je samozřejmě obyčejné číselné číslování v naší, velmi oblíbené, desítkové soustavě. Hodnoty, které můžeme do `type` zadat jsou:

- `1` - číslicový seznam (1., 2., 3., ...)
- `a` - abecední seznam malými písmeny (a., b., c., ...)
- `A` - abecední seznam velkými písmeny (A., B., C., ...)
- `i` - číslicový seznam malými římskými číslicemi (i., ii., iii., ...)
- `I` - číslicový seznam velkými písmeny (I., II., III., ...)

Lze taktéž nastavit atribut `start`, který určuje od jaké hodnoty se bude seznam číslovat. Při nastavení `type` se stále musí nastavit číselná hodnota, tedy i při `type="A"` musí být start číselný, např. `start="4"`, což způsobí začátek na hodnotě `D.`. Při nastavení atributu type na i a start na x, bude se celý seznam číslovat o hodnoty x.

Atribut bez hodnoty `reversed` bude číslovat hodnoty zespoda, tj. např. od 5. do 1., neboli že budou čísla začínat od poslední položky v seznamu.

Každé položce v seznamu (tagu `li`) se dá nastavit číselný atribut `value`, který určuje jak se bude daná a všechny následující položky číslovat.

```
<ol type="a" reversed start="41">
    <li>naučit se na test</li>
    <li>mít hodně bodů</li>
    <li value="31203">mít výborný prospěch</li>
    <li>???</li>
    <li>profit</li>
</ol>
```

Výše uvedená ukázka vytvoří seznam:

ao. naučit se na test
an. mít hodně bodů
atdc. mít výborný prospěch
atdb. ???
atda. profit

---

# Neseřazené seznamy

```
<ul>
    <li>Rostlinné mléko</li>
    <li>Čaj</li>
    <li>Pečivo</li>
    <li>Banán</li>
</ul>
```

Neseřazené listy na rozdíl od seřazených nemají žádné speciální atributy a jejich stylování se koná pomocí CSS. Neseřazené jsou kvůli své jednoduchosti (seřazení je náročnější pro uživatele) používají častěji.

---

# Vnořené seznamy

Seznamy, jak seřazené, tak i neseřazené, lze do sebe vnořovat. Jejich hloubka není omezená a lze je i střídat. Tedy, lze do položky přidat další seznam. V čistém HTML se netvoří u seřazeného klasické číslování kapitol (tj. 2.3.1.1.), ale začíná se od začátku, který určíme, nebo od první hodnoty.

```
<ul>
    <li>listy jsou hezké</li>
    <li>listy jsou zábavné</li>
    <li>listy mají:</li>
    <ul>
        <li>odrážky</li>
        <li>čísla</li>
        <li>znaky</li>
    </ul>
    <li>seřazené listy mají atribut reversed</li>
    <li>seřazené listy mají atribut start</li>
    <ol>
        <li>předměty mohou mít atribut value</li>
        <li>předměty jsou tagu li</li>
    </ol>
</ul>

<ol>
    <li>Nadpis 1</li>
    <ol>
        <li>Sekce 1</li>
```

```
        <ol>
            <li>Subsekce 1.1</li>
        </ol>
    </ol>
</ol>
```

---

# Definiční seznamy

Definiční seznamy jsou nejméně používaným typem a to zejména z důvodu, že jejich použití nedává při spoustě věcí smysl. Taktéž používají více tagů a formátováním si mnoho developerů šáhne po jiných tagech. Máme k dispozici:

- tag `dl` — **d**escription **l**ist
- tag `dt` — **d**efine **t**erm
- tag `dd` — **d**efine **d**escription

Definice vypadá pak následovně:

```
<dl>
    <dt>SSPŠag</dt>
    <dd>Smíchovská střední průmyslová škola a gymnázium</dd>

    <dt>WBA</dt>
    <dd>webové aplikace &mdash; předmět na SSPŠag</dd>
</dl>
```

# Seznamy v CSS

# Typ odrážek

Vlastnost `list-style-type` slouží k určení jaké odrážky budou u seřazeného nebo neseřazeného listu zobrazovat. Seznam všech hodnot lze nalézt ZDE, pro neseřazený seznam nejčastěji používáme:

- `circle`
- `disc`
- `square`

Pro seřazený nejčastěji používáme:

- `decimal`
- `lower-alpha`
- `upper-alpha`

Hodnota může být i `none`, a ta způsobí to, že odrážky nebudou viditelné.

```
ol {
    list-style-type: georgian;
}
```

# Obrázkové odrážky

Vlastnost `list-style-image` slouží k nastavení odrážek jako obrázku, který se bude za ně zobrazovat. Obrázkem se odkazuje stejně, jako např. na obrázkovém pozadí. Při použití vlastnosti bude hodnota z `list-style-type` ignorována.

```
ol, li {
    list-style-image: url("/img/list/cat.jpg");
}
```

# Pozice odrážek

Vlastnost `list-style-position` slouží k úpravě pozice odrážek. V základním designu jsou odrážky vykreslovány mimo prvek (hodnota `outside`) a tedy např. nastavení barevného pozadí neovlivní odrážku. Hodnota `inside` zaručení, že odrážka bude uvnitř daného prvku.

```
ul.inside > li {
    list-style-position: inside;
    background-color: rgb(51, 138, 153);
}
```

---

**POZOR:** náš reset resetuje právě i odrážky. Je nutné si je tedy nastavit ručně pomocí zmíněných vlastností.

# Určení zobrazení prvku

Z minulých lekcí víme, že máme v HTML a CSS různé typy zobrazování. Dříve jsme se seznámili se řádkovém (`span`, `b` a další) a blokovém (`p`, `div` a další) zobrazení. Toto zobrazování je ale normálně určené pouze prohlížečem a tedy si jej můžeme v CSS přenastavit a to i pomocí různých selektorů. Ukážeme si taktéž typ zobrazení `inline-block` a `none`.

K nastavení zobrazení používáme vlastnost `display`. A prozatím máme tyto hodnoty:

- `inline` - prvek zabere místo, které potřebuje na řádku
- `block` - prvek začne na novém řádku a na něm i skončí
- `inline-block` - prvek si bude moci změnit velikost a bude na řádku

- `none` - prvek se vůbec nevykreslí a na stránce nezabere místo

```
b.block-inline {
    display: inline-block;
    width: 100px;
    height: 100px;

    background-color: brown;
    color: white;
}
```

---

Vlastnost s hodnotou display: none; často ale tvoří něco, co nechceme. Často na stránce chceme, aby prvek nebyl viditelný ale zabral místo, které mu bylo určené. K tomu slouží vlastnost `visibility` s hodnotami `visible` (viditelné) a `invisible` (neviditelné). Hodnota `invisible` se chová velmi podobně jako vlastnost s hodnotou `opacity: 0`.

```
.hidden {
    visibility: hidden;
}
```

# Tabulky

Tabulky jsou způsob vytvoření dvojrozměrného pole v HTML. Uživatel v HTML tabulce vidí řádky a sloupce, v kódu však přímo definujeme jen řádky a poté jednotlivé buňky v rádcích. Můžeme mít tedy v každém řádku jiný počet buněk. Buňka je jeden takový obdélník a zrovna v HTML může uvnitř něj být cokoliv, například i další tabulka.

V historii byly tabulky používané špatným způsobem a to tak, že byly používané na rozložení stránky. Rozložení je vlastně systém umístění různých prvků stránek do určitých míst. Například jsme tedy pomocí tabulky vytvořili "hlavičku" a tu dále dělili další tabulkou a tak podobně. Problém je v tom, že tabulky nejsou responzivní, resp. nemůžeme styly upravovat jejich velikost a přeorientování.

Tabulky se skládají ze čtyř základních tagů:

- `table` - obaluje celou tabulku obsahuje definice řádek
- `tr` - table row, označuje řádek, obsahuje definice buněk
- `td` - table data, označuje generickou buňku
- `th` - table heading, označuje nadpisovou buňku, většinou nahoře/po straně

Definice tabulky tedy můžeme vypadat např. následovně:

```
<table>
    <tr>
        <th>Hra</th>
        <th>Hodnocení</th>
        <th>Nahráno hodin</th>
    </tr>

    <tr>
        <td>Slime rancher</td>
        <td>9.9 / 10</td>
        <td>103 h</td>
    </tr>
</table>
```

Výše máme definovanou jednu tabulku, dva řádky, a postupně v řádcích:

- 3x nadpisovou buňku
- 3x datovou buňku

---

Náš reset styly tabulek odstraňuje, ale když si představíme tabulku, tak má většinou nějaké ohraničení. My tedy ohraničení přidáme:

```
table, td, tr, th {
    border: 4px solid black;
}
```

Toto pravidlo ale způsobí, že mezi jednotlivými buňkami (a případně tabulkou a buňkou) budou dvě čáry. Abychom to zamezili, musíme si představit novou vlastnost a to `border-collapse` s hodnotou `collapse`, která spojení sousedních ohraničení umožní.

```
table {
    border-collapse: collapse;
}
td, tr, th {
    border: 4px solid black;
}
```

Jednotlivé buňky můžeme stylovat již jak chceme.

---

Jako první tag (jinde nemůže být!) tabulky můžeme vložit nový párový tag `caption`, který slouží k popisu tabulky. Takový popis bude viditelný na začátku tabulky:

```
<table>
    <caption>Tyto statistiky jsou smyšlené.</caption>

    <tr>
        <th>Hra</th>
        <th>Hodnocení</th>
        <th>Nahráno hodin</th>
    </tr>

    <!-- ... -->
</table>
```

---

Po tagu `caption` může následovat přímo definice `tr` a nebo můžeme (v pořadí!) použít následující prvky na obalení jednotlivých částí:

- `thead` - označuje hlavu tabulky (lze vynechat)
- `tbody` - označuje tělo tabulky (lze vynechat, pokud použijeme tr)
- `tfoot` - označujeme patku tabulky (lze vynechat)

Tyto tagy mají pouze sémantický účel a slouží k rozdělení tabulky do čitelnější podoby robotem. Můžeme tím však i jednotlivé části stylovat.

---

Každou buňku můžeme v tabulce roztáhnout na více sloupců a řádek. Na tagu `td` a `th` můžeme použít atribut `rowspan` a `colspan`. Jako hodnotu dáváme počet sloupců / řádek. Ukázka:

```
<tr>
    <td rowspan="4">Kočka</td>
    <td>černá</td>
    <td>Bambi</td>
</tr>
```

# Pseudotřídy výběru

Nyní si ukážeme další pseudotřídy. Jedná se o pseudotřídy výběru, které vybírají různé prvky v závislosti na rodiči. Ukážeme si:

- `first-child` - vybere prvek, pokud je umístěn jako první prvek v rodiči

- `last-child` - vybere prvek, pokud je umístěn jako poslední prvek v rodiči
- `nth-child(k)` - např. `nth-child(3)`, vybere prvek, pokud je umístěn na třetí pozici v rodiči (pozice se počítají od 1)
- `nth-child(kn)` - např. `nth-child(3n)`, vybere prvek, který je právě na pozici znásobené daným číslem, pro ukázku např. na pozici 3, 6, 9, a další
  - můžeme používat pokročilejší věci, jako např. `nth-child(3n + 1)`, které vyberou stejné jako 3n ale posunuté o jedna dopředu, tento příklad tedy 1, 4, 7, 10 a další