

ROZLOŽENÍ

Úvod do jednoduchého rozložení (tabulky, listy, ...) a pokročilého rozložení pomocí moderních technologií (flexbox, grid, ...).

SEZNAMY

Seznam, často pojmenovávané listy, jsou způsob vypsání bodů na webové stránce. Vypsání může být neseřazené (unordered) a seřazené (ordered). Na webových stránkách jsou také tzv. definiční listy, které známe např. ze slovníku, kdy jeden pojem je definován 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` - číslíkový seznam (1., 2., 3., ...)
- `a` - abecední seznam malými písmeny (a., b., c., ...)
- `A` - abecední seznam velkými písmeny (A., B., C., ...)
- `i` - číslíkový seznam malými římskými číslicemi (i., ii., iii., ...)
- `I` - číslíkový seznam velkými písmeny (I., II., III., ...)

Lze také 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>

```

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

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` zaručí, 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.

Žádné poznámky k této lekci jste ještě nezapsali.

URČENÍ ZOBRAZENÍ PRVKU

Z minulých lekcí víme, že máme v HTML a CSS různé typy zobrazování. Dříve jsme se seznámili se řádkovým (`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;  
}
```

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

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 používány špatným způsobem a to tak, že byly používány 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 rádcích:

- 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>
```

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

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 ukázkou 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

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

