

15. CSS styly (funkce, vývoj, využití). Práce s CSS styly (vkládání do kódu, pravidlo, selektor, dědičnost, kaskády, id a vlastní třídy)

Formátování HTML

Každý text má obsah a formu. Když mluvím o formátu (formě) webových stránek, myslím tím třeba barvu a velikost písma, pozadí, zarovnání atd., prostě všechno, co nepatří do obsahu. To je formátování.

Protože se jazyk HTML vyvíjel, vznikaly časem **různé** způsoby, jak formátovat text. Proto dnes existují dva odlišné způsoby, jak v HTML třeba obarvit písmo nebo ztučnit text.

1. Starší způsob používá přímo **HTML tagy**. (Například kurzíva se dělá pomocí tagů `<i>` a `</i>`: `<i>kurzíva</i>`). Pomocí tagů se některé věci nedají udělat.
2. Novější způsob -- **CSS styly** -- používá tag `<style>` a obecný atribut "style", kterému se přiřazuje nějaká definice, případně atribut id, class. Je to složitější, ale užitečnější a všeobecné. Dají se s tím dělat různé figle, které budu popisovat níže.

Mimochodem, co je CSS?

CSS vzniklo někdy kolem roku 1997. Je to kolekce metod pro grafickou úpravu webových stránek. Ta zkratka znamená Cascading Style Sheets, česky "kaskádové styly". Kaskádové, protože se na sebe mohou vrstvit definice stylu, ale platí jenom ta poslední.

Nástin možností CSS

Opravdu jenom nástin. (Kdyžtak vizte [kompletní přehled](#).) Tak co třeba CSS dovedou:

- Nastavit libovolnou a přesnou velikost písma, p r o k l á d á n í , KAPITÁLKY
- Udělat odsazení prvního řádku odstavce, zvětšit řádkování
- Zrušit nebo zvětšit prázdný prostor po odstavci
- **Automaticky** formátovat nadpisy (například je všechny udělat zelené)
- Zvýrazňovat odkazy po přejetí myší
- Udělat automaticky grafické odrážky
- Určité části textu zneviditelnit, zprůhlednit nebo nezobrazit
- Předefinovat grafický význam běžných tagů (například všechno, co je kurzívou, udělat i tučně)
- Nastavit pozadí čehokoliv, stránky, tabulky ale třeba i odstavce; pozadí se nemusí opakovat a může mít přesnou pozici!
- Umístit nějaký objekt (třeba kus textu) kamkoliv do stránky, může se to i [překrývat](#)
- Přidat k čemukoli rolovací lišty, oříznout to, orámovat, nastavit okraje
- V kombinaci se skripty je dnes CSS nejmocnější zbraň pro "[rozhybání](#)" stránek.
- Hlavní význam CSS spočívá v tom, že fungují hodně automaticky, přechemž se vzhled celého webu deklaruje jedním souborem.

Trojí použití CSS

Styl se může nadeklarovat třemi způsoby, níže uvádím příklady. Stačí, když se pro začátek naučíte naučíte jeden ze tří způsobů:

1. Přímo v textu zdroje u formátovaného elementu pomocí atributu `style="..."`. Tomu říkám **přímý styl**. Je to nešikovné, ale občas se to používá.
2. Pomocí "**stylopisu**" (angl. "stylesheet") v hlavičce stránky. Stylopis je jakýsi seznam stylů. Je v něm obecně napsáno, co má být jak zformátováno, například že nadpisy mají být zelené. Do stránky se stylopis píše mezi tagy `<style>` a `</style>`.
3. Použitím externího stylopisu -- to je **soubor *.css**, na který se stránka odkazuje tagem `<link>`. V souboru je umístěný stylopis. Hlavní výhoda je v tom, že na jeden takový soubor se dá nalinkovat mnoho stránek, takže pak všechny vypadají podobně.

Samozřejmě stačí ovládnout jenom jeden způsob. Já nejčastěji používám externí css soubor.

Příklady

Chci udělat odstavec červeným písmem pomocí CSS. Jak už jsem popsal, jde to třemi způsoby:

A) Přímý zápis

Do zdroje se napíše tato deklarace odstavce:

```
<p style="color: red">Tento odstavec bude červený.</p>
```

Vysvětlení: `<p>` je značka vymezující odstavec; z anglického paragraph. Atribut "style" je obecný atribut použitelný u každého prvku. Color znamená barva a red je červená.

B) Stylopisem

Do hlavičky dokumentu se napíše stylopis uzavřený mezi tagy `<style></style>`:

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

a do těla stránky se mohou psát odstavce:

```
<p>Tento odstavec bude červený. </p>
<p>Tento mimochodem také, protože červené budou všechny.</p>
```

To, jak zařídit, aby nebyly červené všechny, ale jenom některé odstavce, se dá pomocí "[tříd](#)" a "[identifikátorů](#)", o tom později.

C) Externím CSS souborem (my jsme se k tomu letos v praxi radši nedostali)

Vytvoří se soubor, který se pojmenuje třeba `style.css`. V něm bude pouze tento text:

```
p {color: red}
```

Do hlavičky html dokumentu, který chci stylem ovlivnit, musím napsat odkaz na tento soubor:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

V těle dokumentu pak budou opět všechny odstavce červené.

Jde tedy v podstatě o obdobu klasického stylopisu v hlavičce, jak jej vytváříme my, s tím rozdílem, že externí soubor můžete připojit (přilinkovat) k množství html souborů, které sdílí design css souboru a v případě jeho změny máte elegantně a v mžiku inovovaný celý web...

Základní selektory

Nejjednoduššími selektory jsou *selektory typu* a *univerzální selektor*.

Selektor typu: `x { ... }`

`x` je libovolná HTML značka. Tento selektor se páruje s každou instancí daného prvku na stránce. **Příklad:** pravidlo `H1 { color: red }` znamená, že uvedený styl se bude aplikovat na všechny prvky typu `<H1>` na stránce; všechny titulky úrovně 1 budou červené.

Selektory tříd a ID

Selektor třídy: `x.jméno_třídy { ... }`

Pravidlu vyhoví ty instance prvků, jejichž třída (`class`) je právě `jméno_třídy`. Tento typ selektoru byl zaveden pouze pro zjednodušení, neboť přesně odpovídá zápisu `X[class=jméno_třídy] { ... }`. V případě selektoru `*.třída` není nutné univerzální selektor `*` uvádět, jako selektor postačí `.třída`. **Příklad:** pravidlo `H1.nadpis { ... }` páruje se všemi prvky `<H1>`, které mají uveden atribut `class="nadpis"`; pravidlu `.nadpis { ... }` - což je totéž jako `*.nadpis { ... }` - pak vyhoví všechny prvky na stránce, které mají `class="nadpis"`.

Selektor ID: `x#ident { ... }`

Styl se použije pouze u prvku, jehož ID je `ident`. Opět, pokud není specifikována konkrétní HTML značka, není nutné uvádět `#ident`, ale postačí `#ident`. **Příklad:** pravidlu `H1#nazev { ... }` vyhoví pouze ten titulek `<H1>`, jehož `ID="nazev"`. Pravidlu `#nazev { ... }` pak vyhoví všechny prvky na stránce, které mají definováno `ID="nazev"`.

Pozn.: Selektory ID mají vyšší prioritu než selektory s atributem - např. #navez je z hlediska kaskádování specifitější než [ID=navez].

Definované hodnoty

Klient tedy nejprve přiřazuje hodnotu každé myslitelné vlastnosti každého prvku v dokumentu. Při tomto procesu používá tato pravidla:

- Pokud proces **kaskády** (viz dále) vrátí nějakou hodnotu - tj. vlastnost je specifikována nějakou definicí stylu, uživatelem atd. - použije se tato hodnota.
- Není-li vlastnost explicitně určena kaskádou, hodnota se **dědí**, tj. použije se hodnota, která je definována pro rodičovský prvek (prvek nadřazený ve smyslu stromu dokumentu).
- Není-li možné tuto vlastnost zdědit (rodičovský prvek danou vlastnost nemá definovanou, vlastnost není dědičná atd.), použije se **výchozí hodnota**. Každý prvek má definovány výchozí hodnoty pro každou vlastnost.

Dědičnost v CSS

Některé vlastnosti se mohou v rámci stromu dokumentu dědit. To znamená, že pokud není řečeno jinak, vlastnost definovanou pro jeden prvek přebírají i všechny objekty uvnitř tohoto prvku. Nedědí se ale všechny vlastnosti - každá vlastnost má definováno, zda její hodnota je či není dědičná.

Příklad:

Vlastnost `color` (barva textu) je dědičná. Pokud v tomto příkladu:

```
<H1>Toto je <EM>důležitý</EM> titulek</H1>
```

nebude mít prvek `EM` nijak specifikovanou vlastnost `color`, převezme ji od rodičovského prvku `H1` (z hlediska stromu dokumentu je zde `EM` potomkem `H1`, neboť prvek `EM` je obsažen uvnitř prvku `H1`) - celý text titulku bude tedy zobrazen stejnou barvou textu.

Pozn.: ve většině případů se nedědí definované hodnoty, ale až hodnoty *přepočítané* - tzn. má-li např. rodičovský prvek definovanou vlastnost v procentech, dědí se až hodnota přepočítaná na absolutní jednotky.

Kaskáda

Styly, ovlivňující jeden dokument, mohou pocházet ze tří různých zdrojů: od autora, od uživatele a z klientu. **Autor** (tvůrce dokumentu) specifikuje styly přímo v dokumentu (či v externích souborech k němu připojených - viz dříve). **Uživatel** si v některých prohlížečích může zvolit své vlastní styly (např. připojit svůj vlastní soubor se styly nebo nastavit některé vlastnosti přímo v prohlížeči). A v neposlední řadě každý **klient** (prohlížeč) má za povinnost mít definovanou *výchozí tabulku stylů*, která předchází všem dříve jmenovaným a definuje tak výchozí hodnoty podle možností daného klientu. Ve specifikaci CSS je [ukázka typické výchozí tabulky stylů pro HTML4](#). Všechny tyto styly se navzájem ovlivňují a tvoří výsledné hodnoty podle pravidel kaskádování.

Kaskáda ohodnotí každý styl - přidělí mu tzv. *váhu*. Pokud je pro danou vlastnost více pravidel, přednost dostane to, které má větší váhu. Standardně mají styly autora větší váhu než styly uživatele; pouze při použití direktivy `!important` (viz dále) je tomu naopak. Styly a autora i uživatele ale mají vždy větší váhu než výchozí styly klientu. Váha také záleží na pořadí a způsobu načtení: styly z dané tabulky stylů mají větší váhu a přepíšou styly importované z jiné tabulky.

Aby se mohla určit hodnota dané vlastnosti daného prvku, klient musí všechna pravidla nejprve setřídit podle následujícího **třídění kaskády**:

1. Nejprve **najde všechna pravidla**, která se vztahují k danému prvku, dané vlastnosti a použitému médiu (výstupní zařízení). Pravidla se použijí, pokud jejich selektor páruje s daným prvkem (viz Selektory dříve).
2. Poté **setřídí pravidla podle jejich váhy a zdroje** - styly autora mají přednost před styly uživatele, výchozí styly klientu mají nejmenší váhu atd.
3. Následně **setřídí pravidla podle jejich specifičnosti** (viz dále) - specifičtější pravidla mají přednost před obecnějšími. Se pseudo-třídami a pseudo-prvky se zde nakládá, jako by to byly běžné třídy a prvky.
4. V posledním kroku jsou pravidla **setříděna podle pořadí** v dokumentu. Z pravidel, která mají stejnou váhu, zdroj i specifičnost, se použije to, které je definováno nejpozději. Importované styly se považují za definované dříve než styly popsané přímo v dané tabulce.

Příklad s odkazy :pseudotřídy

Pomocí stylopisů se dají formátovat libovolné HTML tagy, ne pouze nadpisy. Obzvlášť efektní je to u odkazů. Nebudu vypisovat celý zdroj, omezím se na stylopis:

```
<style type="text/css">
a      {text-decoration: none}
a:link {color: green}
a:visited {color: navy}
a:active {color: black}
a:hover {color: red; text-decoration: underline}
</style>
```

Celý soubor s tímto stylopisem si můžete [zobrazit](#). Setkáváme se tu s deklarací formátu odkazů -- vnitřku tagu `<a>` ``. Navíc tento tag má **pseudotřídy** (různé stavy), které umožňují různé zobrazení podle toho, zda už je odkaz navštívený nebo zda po něm jede myš. Takže konkrétně:

- `text-decoration: none` znamená, že odkazy nebudou podtržované
- `a:link` znamená nenavštívený odkaz (bude zelený)
- `a:visited` je už navštívený (tmavě modrý)
- `a:active` je ten, na který se zrovna kliklo (černý), nebo ten, po kterém jede tabulátor
- `a: hover` je ten, přes který se jede myší (červený podtržený)
- `text-decoration: underline` znamená podtržení.

Tag A je jediný, u něhož se vyskytují pseudotřídy. Ještě pozor na syntaxi: mezi *a* a dvojtečkou není mezera!

První řádek stylopisu definuje nepodtrhávání odkazů pro všechny pseudotřídy. Pouze `a:hover` tuto deklaraci přebíjí, protože je uvedena později.

Podtitul (tvorba vlastní třídy)

Příkladem vlastního stylu může být podtitul. (Nepatří do nadpisu a přece by měl být formátován odlišně než normální text.) Dá se formátovat přímo, ale aby byl ve všech dokumentech stejný, je dobré nadefinovat jej jako styl. HTML ale nemá pro podtitul žádný tag `<podtitul>`, a tak si musím pomoci jinak. Vytvořím *třídu* s názvem `podtitul`, ve stylopisu mu nadefinuji vlastnosti (třeba tučnost, zarovnání na střed) a u daného textu jenom řeknu, že *patří do třídy* podtitul.

Jak vypadá stylopis:

```
<style>
.podtitul { text-align: center; font-weight: bold; text-decoration:
overline} /* zarovnání na střed, tučné písmo a nadtržení*/
</style>
```

a potom v těle dokumentu to vypadá takhle:

```
<p class="podtitul">Text podtitulu</p>
```

A v prohlížeči potom takhle:

Text podtitulu

Text uvnitř "zaklasovaného" elementu se bude formátovat podle definice ve stylopisu. Ještě je třeba všimnout si **tečky** na začátku deklarace ve stylopisu. Ta vyjadřuje, že deklarace se nebude týkat html tagu, ale třídy.

Atribut `class` (třída) se může použít u libovolného elementu (tagu). Symbolicky:

```
<tag class="jméno_třídy">
```

Element se stejnou `class` se v dokumentu může vyskytovat mnohokrát (na rozdíl od `ID` -- identifikátoru, o tom později). Potom se tento element zformátuje podle definice.

Identifikátor (tvorba ID třídy)

Pro jednoznačný popis nějakého elementu (zejména pro potřeby [skriptů](#)) existuje univerzální atribut `ID`. I jemu se může ve stylopisu přiřadit nějaká deklarace, ale na rozdíl od třídy

nezačíná tečkou, ale **dvojkřížkem #**. V těle dokumentu by se element s jedním identifikátorem měl vyskytovat jenom jednou.

Kdybych v předchozím příkladu použil identifikátoru namísto třídy, deklarace by vypadala takhle:

```
#podtitul { text-align: center; font-weight: bold; text-decoration: overline }
```

a v těle by se odstavci přiřadila identifikace atributem **id**:

```
<p id="podtitul">Text podtitulu</p>
```

Identifikátor **id** se z hlediska CSS chová stejně jako třída **class**. Rozdíly jsou právě jen ve skriptech a v parsování dokumentu.

Pseudoelementy

Ve specifikaci CSS1 se vyskytují pseudoelementy **:first-line** a **:first-letter**. Znamenají první řádek a první písmeno. Například zápis:

```
p:first-line { color: blue }
```

```
p:first-letter { font-size: 200% }
```

by měl způsobit, že odstavce budou mít první řádek modrý a první písmeno dvakrát větší.