VIII. Cascading Style Sheets (CSS)

Motivace vzniku CSS

Jazyk HTML byl původně navržen pro popis struktury WWW dokumentu – nebyl zamýšlen jako jazyk pro popis vzhledu WWW stránek. Jenže s jeho rozšířením mimo akademickou oblast se objevily *požadavky na definici grafického vzhledu WWW stránek*. Tak byly do HTML přidány nové „užitečné“ značky (např. `font`) a atributy (např. `align`). Ale kód stránek se stal nepřehledným, navíc dodatečné změny vzhledu byly složité.

Koncem 20. století se proto objevuje snaha oddělit prezentaci (vzhled) od struktury WWW stránky. O změnu se postaralo W3C – do HTML 4.0 byly přidány styly. HTML5 se vrací k původní myšlence – definuje strukturu dokumentu a styly používá pro nastavení vzhledu webu.

CSS (Cascading Style Sheets)

CSS je jazyk („šablona stylů“), pomocí něhož autor přiřadí *vzhled a formátování* k dokumentu napsanému ve značkovacím jazyce HTML, resp. XHTML nebo XML. Pomocí CSS tedy může autor WWW stránky definovat, *jak se mají určité části stránky zobrazovat v prohlížečích*.

WWW dokument se potom skládá z vlastního kódu popisujícího jeho *strukturu* (pomocí značkovacího jazyka), a dále z definice stylů popisujících výsledný *vzhled* WWW stránky (přesněji: vzhled libovolného elementu použitého v kódu WWW stránky).

Proč používat CSS

Největší výhodou použití kaskádových stylů je *oddělení prezentace od struktury WWW stránky*, z čehož plyne:

- přehlednější správa obsahu WWW stránky – kód je kratší a jednodušší (neobsahuje zbytečné „grafické“ značky a atributy),
- snadná změna vzhledu WWW stránky (úpravou CSS, nikoli přepisováním částí kódu).

Další výhody přináší definice CSS v externím souboru:

- jednotný vzhled stránek celého webu (sdílení CSS souborů pro všechny stránky),
- možnost snadné změny vzhledu a rozložení všech stránek webu – stačí upravit definici vzhledu na jednom místě.

Vývoj CSS I

Již v roce 1994 se objevuje snaha oddělit strukturu stránek od jejich vzhledu, ale teprve s HTML 4.0 přichází řešení.

CSS1 (CSS level 1) – 1996-2018 (již nepodporováno)

CSS2 (CSS level 2) – specifikace 1998

- CSS 2.1 (CSS level 2 revision 1) – specifikace 2011
- CSS 2.2 (CSS level 2 revision 2) – neoficiální Working Draft z 2016

Vývoj CSS II

CSS3 (CSS level 3)

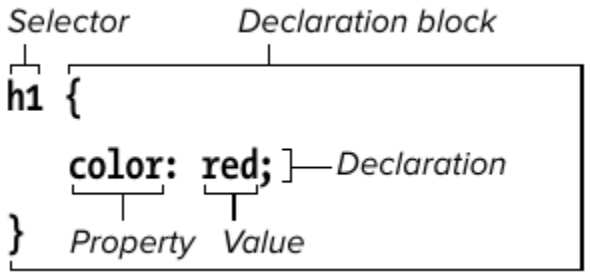
- vývoj od 1999, rozdělen do modulů, které rozšiřují CSS2
- základem jsou CSS 2.1 (resp. CSS 2.2) a CSS style attributes (2013)
- např. hotové moduly (s odkazem na specifikaci):
  - CSS Color Level 3 (2018)
  - Selectors Level 3 (2018)
  - CSS Namespaces Level 3 (2011)
  - Media Queries Level 3 (2012)

CSS aktuálně

- CSS3, CSS4 atd. jsou „lidové“ pojmy, v praxi je vše „CSS bez čísla“
- některé moduly mají čísla, např. CSS Color Level 5
- podporu pro novější prvky CSS v prohlížečích je nutné ověřovat samostatně, např. pomocí https://caniuse.com/
  (případně referenční příručky W3Schools a MDN Web Docs obsahují sekce „Browser support“ resp. „Browser compatibility“ pro každou vlastnost CSS)
- CSS lze použít pro různé značkovací jazyky: HTML, XML, XHTML, SVG, MathML3

Základní syntaxe

- základní dvě části:
  - selektor
    - určí, kterou část webu ovlivníme
  - deklarační blok
    - definice stylu, dvojice *vlastnost: hodnota;*



- dobré přidat komentáře `/* komentář */`
- na umístění bílých znaků (mezera, konec řádku, atd.) nezáleží
  (**pozor**: mezera se používá pro kombinaci selektorů a hodnot pro určité vlastnosti)

Příklady:

Pravidlo s jednou deklarací (pro značku `p` – odstavec):

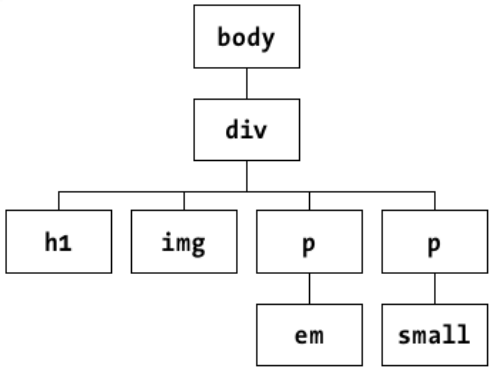
```
p {
  text-align: center;
}
```

Pravidlo se dvěma deklaracemi (pro značku `h1` – nadpis 1. úrovně):

```
h1 {
  font-size: 120%;
  color: blue;
}
```

CSS a dědičnost

- většina vlastností (především u písma) se dědí od nadřazeného elementu
- pokud chcete definovat nějakou vlastnost společnou všem elementům, definujte ji pro element body



Hodnoty initial a inherit

- `initial`
  - počáteční hodnota dané vlastnosti, lze ji také explicitně vynutit („resetování“ vlastnosti)
- `inherit`
  - deklaruje, že element danou vlastnost dědí od rodiče, lze i pro automaticky nedědičné vlastnosti

Příklad: `h1 {font-size: inherit;}`

Kaskádovost

- každý prohlížeč má svůj výchozí styl
- styly lze:
  - nahrát z externího souboru
  - vložit je na začátek html dokumentu
  - vložit je přímo do kódu
- co se stane, když pro jeden selektor použiju více definic?
  - různé definice => sjednocení
  - definice se stejnou vlastností => přepíšou se podle váhy selektoru
    - čím více specifikovaný selektor, tím vyšší váha (nejvíc specifický `id`)
    - pokud máme selektory se stejnou váhou, později uvedený přepíše původního
  - definice `!important;`
    - styl s touto definicí přepíše vše

Hodnoty vlastností

- každá hodnota má jistý typ hodnot, které mohou být vyjádřená číselnou hodnotou, procentem, barvou, url, ...
- hodnota `inherit`, `initial`
- předdefinované hodnoty (vlastnost `float` a hodnota `left`, `right`, `none`)
- hodnota délky: musí být uvedené ve formátu `čísloJednotka` (`3em`, `10px`, výjimka `0`)
  - lepší `em` (relativní) než `px` (absolutní)
- samotná čísla, např. pro `line-height`, `opacity`
- hodnota `url`, např. pro `backgroud-image:url(file.ext)`

Hodnoty vlastností - barvy

- buď můžeme použít předdefinované hodnoty (https://www.w3.org/TR/css-color-3/#svg-color) a nebo použít RGB, HSL, RGBA, nebo HSLA kód

zápis	hodnoty r, g, b	příklad (yellow)
`#rgb`	0,1,...,9,A,B,...,F	`#FF0`
`#rrggbb`	šestnáctkové (00-FF)	`#FFFF00`
`rgb(r,g,b)`	desítkové (0-255)	`rgb(255,255,0)`
`rgb(r%,g%,b%)`	procenta (0-100)	`rgb(100%,100%,0%)`

Výtvor CSS

Externí soubor

- vytvoříme jednotný styl pro všechny stránky daného webu
- soubor s příponou **style.css**, kódování UTF-8

```
img{
    border: 4px solid red;
}

img{
    border-width: 4px;
    border-style: solid;
    border-color: red;
}
```

Nalinkování souboru do HTML

- do head sekce vložíme relativní adresu vzhledem k serveru, ne html souboru!

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

- při změně souboru style.css je vzhled webu automaticky modifikován všude, kde je tento soubor nalinkován

Vnoření stylu do html

- kompletní deklaraci stylu vložíme do head sekce html souboru
- použijeme element `<style> ... </style>`

```
<head>
<meta charset="UTF-8" />
<title>El Palau de la Música</title>
<style>
img {
          border: 4px solid red;
}
</style>
</head>
<body>
```

- vnořené styly přepíší styly z externích souborů

Inline styly

- málo používané, protože narušují stylizaci html dokumentu
- nejvíc se používají při testech, aneb jak to bude vypadat
- ovlivňují pouze element, u kterého vystupují

```
<img src="img/palau.jpg" width="250"
   height="163" alt="El Palau de la Música"
   style="border: 4px solid red" />
```

- lépe pomocí class:

```
<img src="img/palau.jpg" width="250"
    height="163" alt="El Palau de la Música"
    class="frame" />
```

```
.frame {
  border: 4px solid red;
}
```

Medium-specifický styl

- definujeme pro které zobrazovací médium má být styl použit
- přidáme `media="output"` do elementu `<link>` nebo `<style>`
- hodnota `"output"` nabývá hodnot `"all"`, `"screen"`, `"aural"` , `"braille"` `"handheld"`, `"print"`, `"projection"`, `"tty"` a `"tv"`
- alternativně

```
@media print {
 body {
      /* make text larger */
      font-size: 25pt;
 }
 p {
      /* hex for black */
      color: #000;
 }
}
```

Definice selektorů

- selektor určuje, kterou část má daný styl ovlivnit
- selektor může definovat až pět různých kritérií pro výběr prvků, které mají být formátovány:

Typ nebo název elementu

Name of desired element

```
h1 {
    color: red;
}
```

Kontext plus element

Context

Name of desired element

```
h1 em {
    color: red;
}
```

Třída nebo ID elementu

Class

```
.error {
    color: red;
}
```

ID

```
#gaudi {
    color: red;
}
```

Name of desired element

Class

```
strong.error {
    color: red;
}
```

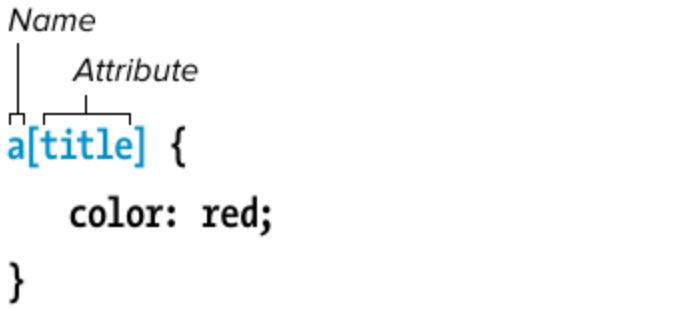
Pseudo-element, pseudo-třída

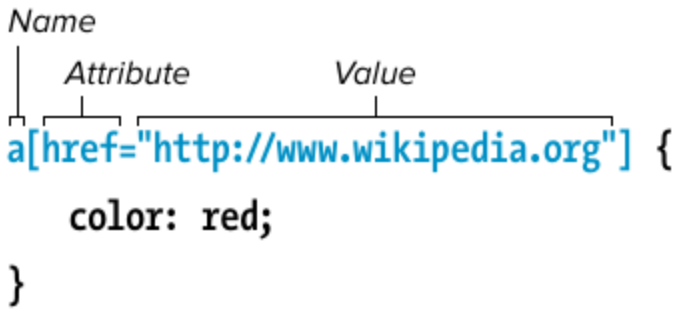
Name

Pseudo-class

```
a:link {
    color: red;
}
```

Element s atributami s bebo bez hodnot





Specifikace selektorů a dědičnost

Sestupná dědičnost

```
.architect p {
 color: red;
}
```

- mezera mezi `.architect` a `p` znamená, že tento selektor najde jakýkoli element `p` který je potomkem jakéhokoli elementu s výrazem architect, bez ohledu na jeho generaci

```
/* Any p that is a descendant of any article. The least specific of the three. */
article p {
color: red;
}
/* Any p that is a descendant of article elements with the architect class. The most specific of the three. */
article.architect p {
color: red;
}
```

Dědičnost na jednu generaci

- tento selektor vybere pouze ty elementy `p` které jsou dětmi (ne vnuky, ne pra- vnoučata a tak dále) elementů s atributou třídy architect

```
.architect > p {
color: red;
}
```

Dědičnost v rámci sousedních sourozenců

- `h1` a `p`, resp. `p` a `h2` jsou **sousední sourozenci**, ale `h1` a `h2` ne, ale `h1`, `p` a `h2` jsou **sourozenci**

```
<body>
<h1>...</h1>
<p>...</p>
<h2>...</h2>
</body>
</html>
```

- všechny sousední sourozenci za `p` typu `p` budou mít červené písmo (všechny odstavce vyjma prvního budou červení)

```
.architect p+p {
color: red;
}
```

- všichni sourozenci (ne nutně sousední) za `h1` typu `h2` budou červení

```
.architect h1~h2 {
color: red;
}
```

Dědičnost v rámci prvního/posledního potomka

- tento selektor vybere pouze element `li`, který je prvním potomkem svého rodiče

```
li:first-child {
color: red;
}
```

- alternativně `li:last-child`

Výběr prvního písmene, řádku

- pseudoelement `:first-letter` a `:first-line`
- ne všechny vlastnosti lze takto specifikovat

```
p:first-letter {
 color: red;
 font-size: 1.4em; /* make letter larger */
 font-weight: bold;
}
```

- v CSS3 `::first-letter` a `::first-line` (starší prohlížeče :: nepodporují)

Výběr odkazů na základě na základě jejich stavu

- `link` pro změnu vzhledu odkazů, které ještě nebyly aktivovány nebo na které se právě neukazuje.

```
a:link {
color: red;
}
```

- `visited` pro změnu vzhledu aktivovaných odkazů

```
a:visited {
color: orange;
}
```

- `focus` pro změnu vzhledu pokud je odkaz vybrán pomocí klávesnice a je připraven k akci (též po aktivaci)

```
a:focus {
color: purple;
}
```

- `hover` pro změnu vzhledu při ukázání na něj myší

```
a:hover {
color: green;
}
```

- `active` pro změnu vzhledu aktivovaných odkazů

```
a:active {
color: blue;
}
```

Výběr elementů na základě atribut

- př: odstavec `p` s atributou `class` bude mít červené písmo

```
p[class] {
color: red;
}
```

- případně můžeme specifikovat i hodnotu

```
pp[class="intro"] {
color: red;
}
```

- hodnotu atributy lze dále specifikovat takto:

1. `="value"`, přesná shoda hodnot

```
a[rel="external"]{
color: red;
}
```

2. `~="value"`, přesná hodnota libovolného celého slova (hodnoty) v seznamu slov(hodnot) oddělených mezerami

```
a[href][title~="howdy"]{
color: red;
}
```

3. `|="value"`, přesná hodnota nebo začíná danou hodnotou, za kterou bezprostředně následuje - (`en` a `en-US`)

```
a[lang|="es"]{
color: red;
}
```

4. `^="value"`, hodnota začíná danou hodnotou, nebo je částí delšího slova

```
a[href^="http://"]{
color: red;
}
```

5. `$="value"`, hodnota končí danou hodnotou nebo je koncovou částí slova

```
img[src$=".png"]{
border: 1px solid green;
}
```

6. `*="value"`, hodnota je částí jiné hodnoty

```
a[href][title*="how"]{
color: red;
}
```

- další atributy dáváme do `[]` zvlášť

Specifikace skupin elementů

- jednotlivé elementy oddělujeme čárkou

```
h1,
h2 {
  color:red;
}
```

Kombinace více selektorů

- styl platí jen pro ty `em`, které jsou v elementu `p`, který je přímým sousedem (sourozencem) `h2[lang|="es"]`, který je uvnitř jekéhokoliv elementu s třídou `.project`

```
.project h2[lang|="es"] + p em {
color: red;
}
```

Formátování textu pomocí CSS

Motivace: naformátovat web tak aby byl přívětivější

Výběr fontu

```
body{
  font-family:Geneva;
}
h1, h2 {
font-family: "Gill Sans";
}
```

- **POZOR**, uživatel na webu uvidí jen fonty, které má u sebe nainstalované, proto je dobré definovat alternativní fonty (list fontu nazýván *font stack*)

```
body{
  font-family: Geneva, Tahoma, sans-serif;
}
h1, h2 {
  font-family: "Gill Sans", "Gill Sans MT", Calibri, sans-serif;
}
```

-defultně uvést **serif** , **sans-serif** , **cursive** ,**fantasy** , nebo **monospace**

Kurzíva

- defultně nastaveno pro `cite`, `em`, a `i`
- extra kurzívu můžeme přidat přes `font-style` s hodnotou `italic` nebo `oblique` (šikmý), pro navrácení zpět `font-style: normal;`

```
body {
  font-family: Geneva, Tahoma, Verdana, sans-serif;
}
h1,
h2 {
  font-family: "Gill Sans", "Gill Sans MT", Calibri, sans-serif;
}

p
{
  font-style: italic;
}
```

- **POZOR:** fonty jako Geneva nemají kurzívu (italic)

Tučný text

- defaultně jsou například nadpisy `h1` - `h6` vysázené tučně
- definice pomocí `font-weight` s hodnotou `bold` pro tučné, `normal` pro normální, `bolder`, `lighter`, nebo násobek 100 až do 900 (normál 400 a ne všechny fonty mají tolik hodnot)

```
h1,
h2 {
  font-family: "Gill Sans", "Gill Sans MT", Calibri, sans-serif;
  font-weight: normal;
}
em,
a:link,
.intro .subhead {
  font-weight: bold;
}
```

Velikost písma

- dvě možné definice velikosti písma
    i. v pixelech - absolutní
    ii. relativní v procentech, em (rem) - vůči svému předkovi!
- vlastnost `font-size`

```
body {
  font-family: Geneva, Tahoma, Verdana, sans-serif;
  font-size: 100%; /* 16px */
}
h1 {
  font-size: 2.1875em; /* 35px/16px */
}
.intro p {
  font-size: 2em; /* 32px/16px */
}
.intro a {
  font-size:1.0625em; } /*pozor na relativnost vůči rodiči*/
```

- pokud je výše uvedený odkaz v paragrafu s třídou `intro`, pak se relativní jednotka `em` nepočítá vůči 16px ale vůči 32px!
- `rem` škáluje vše vůči kořenu, odpadá předchozí problém (ale problém s kompatibilitou vůči starším prohlížečům)
- absolutní velikost lze zadat buďto přímov pixelech, nebo pomocí `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, nebo `xx-large`.

Řádkování

```
.project p {
  font-size: .9375em; /* 15px/16px */
  line-height: 1.65; /* 15px*1.65 = 24.75px */
}
```

- alternativně můžeme hodnotu udat v `em`, `px`, `pt`, `%`
- při definici v `em` a `%` se dědí výsledná hodnota a ne poměr

Nastavení všech vlastností písma najednou

- použijeme vlastnost `font`
- nemusíme specifikovat vše, minimum je `font-size` a `font-family`
- vlastnosti definujeme takto:
  - (`normal` / `italic` / `oblique`), (`normal` / `bold` / `bolder` / `400`), (`normla` , `small-caps`), **`font-size`** , `line-height` , **`font-family`**
  - `small-caps` varianta fontu, vše převede na velká písmena
  - na pořadí záleží až od `font-size`

```
.example-3 {
  font: italic small-caps bold .875em/1.3 "Palatino Linotype", Palatino, serif;
}
```

Nastavení barvy

- definujeme vlastnost `color` s danou hodnotou barvy, viz dříve

```
body {
  color: blue;
  font: 100% Geneva, Tahoma, Verdana, sans-serif;
}
```

Nastavení pozadí

- jednotlivé vlastnosti

1. barva pozadí `background-color` (možnost `transparent` plus barva)

```
body {
  background-color: #88b2d2;
  font: 100% Geneva, Tahoma, Verdana, sans-serif;
}
```

2. obrázek na pozadí `background-image` (relativní adresa vůči souboru s definicí css stylu)

```
body{
  background-color: #ccc;
  background-image: url(bg-pattern.png);
  ...
}
```

3. opakování vzoru na pozadí `background-repeat`

- udáme směr, ve kterém se má daný obrázek opakovat (`repeat-x` pro horizontální směr, `repeat-y` pro vertikální směr, `repeat` pro oba směry)

```
body {
  background-image: url(sky.png);
  background-repeat: repeat-x;
...
}
```

4. fixování obrázku na pozadí `background-attachment`

- `fixed` (pozice je pevná) vs `scroll` (pohybuje se při scrollování)

```
body {
  background-color: #000;
  background-image: url(sky.png);
  background-repeat: repeat-x;
  background-attachment: fixed;
  }
```

5. pozice obrázku na pozadí `background-position`

- hodnota `x y`, vyjádřená v `px` (absolutní) nebo v `%` (relativní)
- alternativně pro `x` hodnoty `left`, `center` , `right` a pro `y` `top`, `center`, `bottom`

```
body {
  background-color: #000;
  background-image: url(sky.png);
  background-repeat: repeat-x;
  background-attachment: fixed;
  background-position: left bottom;
}
```

- pomocí `background` lze definovat vše najednou

```
body {
  background: #004 url(../img/ufo.png) no-repeat 170px 20px ;
  color: greenyellow;
...
}
```

- CCS3 umožňuje specifikovat pozadí dalšími vlastnostmi
  - `background-clip`, `background-origin` (http://css-tricks.com/transparent-borders-with-background-clip)
  - `background-size` kontrola velikosti obrázku na pozadí, hodnoty:
  - `contain` co nejvíce velký a současně celý viditelný, nemusí pokrýt celé pozadí
  - `cover` co nejmenší, vyplní celé pozadí, část nemusí být vidět
  - hodnota `x y` vyjádřená v `px` nebo `%`

Mezery

- `word-spacing` pro mezery mezi slovy
- `letter-spacing` pro mezery mezi písmeny
- hodnoty uvedeny v `px` nebo v `em`, hodnota `normal` nastaví mezery na výchozí nastavení

```
h2 {
  font-family: "Gill Sans", "Gill Sans MT", Calibri, sans-serif;
  letter-spacing: normal;
}
```

Odrážky

- vlastnost `text-indent` s hodnotou vyjádřenouv `em` nebo `px`
- typicky nastavujeme pro odstavce `p`

```
.project p {
  font-size: .9375em; /* 15px/16px */
  line-height: 1.65;
  text-indent: 2em; /* 30px */
}
```

Zarovnání textu

- vlastnost `text-align`, hodnoty `left`, `center`, `right`, `justify` (na obě strany)
- nefunguje pro `strong`, `em`, `a`, `cite`

```
p {
  text-align: justify;
}
```

Transformace textu

- vlastnost `text-transform` s hodnotami
  - `capitalize` všechny slova (i spojky, předložky, ...) mají první písmeno velké
  - `uppercase` vše na velké
  - `lowercase` vše na malé
  - `none` nic nemění

```
h1{
  font-size: 2.1875em;
  text-transform: uppercase;
}
```

Small Caps

- převod všeho na velké písmena s velikostí malých
- vlastnost `font-variant` s hodnotou `small-caps` nebo `none`

```
h2{
  background-color: #eaebef;
  color: #7d717c;
  font-size: 1.75em;
  font-variant: small-caps;
}
```

- ne všechny fonty mají *small caps* variantu

Dekorace textu

- vlastnost `text-decoration`
- převážně používáno pro odkazy
- hodnoty `underline`, `overline`, `line-through`, `none`

```
a:link {
  color: #e10000;
  text-decoration: none;
}
```

Bílá místa (Whitespace)

- většina mezer a prázdných míst v HTML dokumentu jsou zobrazena jako jedno prázdné místo nebo jsou ingnorovány
- vlastnost `white-space` s hodnotami
  - `pre` - nic se neignoruje, stejné mezery jako v HTML souboru
  - `nowrap` - žádná mezera není zalamující, vše na jedné řádce
  - `normal` - standardní chování

```
.intro .subhead {
  font-size: 1.125em;
  color: lime;
  text-shadow: 3px 2px 2px black;
  white-space: nowrap;
}
```

Cvičení

- vytvořte CSS styl, který dané stránky převede, styl pro:

1. `<body>`:
   barva pozadí *#88b2d2*
   font: velikost *100%*, fonty *Geneva, Tahoma, Verdana, sans-serif*

2. Pro:

   `<h1>, <h2>`

   mezery mezi písmeny 1px

   zarovnání textu doprava

   fonty "Gill Sans", "Gill Sans MT", Calibri, sans-serif

   tučnost normální (ne tučný)

   `<h1>`

   transformace na velká písmena

   velikost písma *2.1875em*

   výška řádku *1.19318*

3. Pro:
   `<h2>`
   barva pozadí *#eaebef*
   barva textu *#7d717c*
   velikost písma *1.75em*
   nastavit převod na malá
   `<p>`
   zarovnání na obě strany

4. Pro:

   `em, a:link, .intro .subhead`

   tučný text

   `.architect`

   barva pozadí `#fff`

5. Pro:

   `.intro`

   barva pozadí *#686a63*

   barva textu *#fff*

   výška řádku *1.45*

   `.intro .subhead`

   velikost písma *1.125em*

   zarovnání na střed

   `intro p`

   velikost písma *1.0625em*

6. Pro:

   `.project p`

   odrážka velikosti *2em*

   velikost písma *.9375em*

   výška řádku *1.65*

   `.photos`

   zarovnání textu na střed

   `a:link`

   barva *#e10000*

   dekorace žádná

   `a:visited`

   barva *#b44f4f*

7. Pro"

   `a:hover`

   barva textu *#f00*

   dekorace podtržením

   `.intro a`

   barva textu *#fdb09d*

   `.intro a:hover`

   barva textu *#fec4b6*

Rozvržení stránky pomocí CSS

- stránku rozvrhneme tak abychom dobře vizuálně oddělili od sebe různý typ obsahu
- základní typy rozvržení

1. S pevným rozvržením (pevnou šířkou)

- stránka má pevnou šířku danou v `px`
- rozvržení se nepřizpůsobuje při škálování obrazu na jiných zařízení

2. S responzivním rozvržením

- šířka stránky je daná v procentech (relativně)
- "tekuté" prostředí se přizpůsobuje různým zařízením (tablet, smartphone, ...)

- v základě budeme definovat strukturu základních kontainerů HTML
- pro strukturizaci stránky použijeme `article`, `aside`, `main`, `nav`, `section`, `header`, `footer`, a `div`
  - `div` jeden na celou stránku a další, který obsahuje dvě hlavní části
  - `header` pro záhlaví a navigační panel
  - `main` rozdělíme na `section` s vlastním `footer`
  - postranní `div` piužívající `article` a `aside` pro informace o stránce, autorovi, ...
  - `footer` pro zápatí

- přidej na stránku obsah do částí kam se nejlépe hodí
- přidávejte konzistentně nadpisy
- označujte sémanticky obsah pomocí `<p>`, `<figures>`, ...
- přidávejte komentáře do HTML pro lepší orientaci

Reset nebo normování

- prohlížeč más svůj defaultní styl, každý trochu jiný
- abychom mohli začít nezávisle na prohlížeči, máme dvě možnosti:

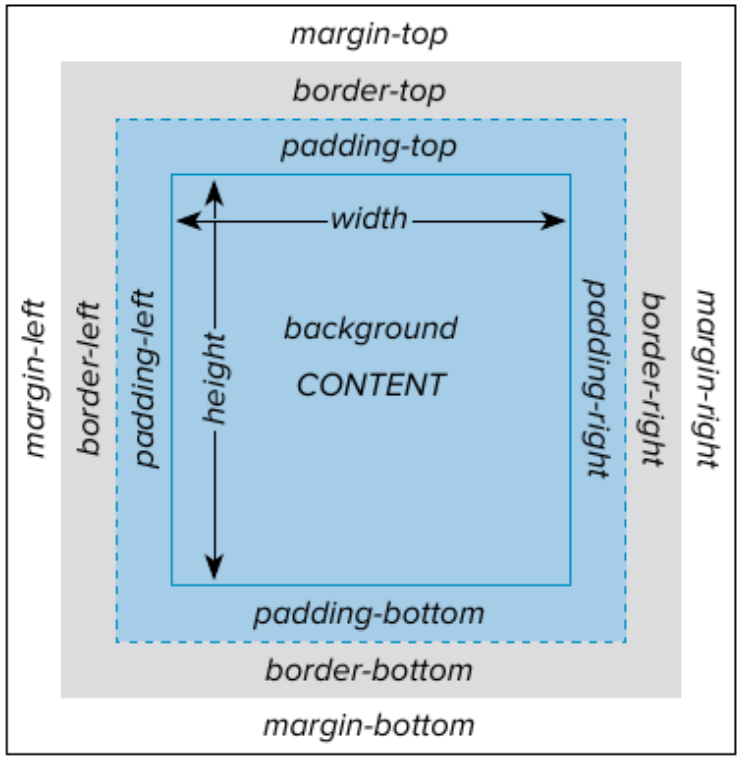
1. reset - vše nastavíme na nulu
   https://meyerweb.com/eric/tools/css/reset/
2. Normování - defaultní styly se upraví aby byly "shodné" napříč prohlížeči
   http://necolas.github.io/normalize.css/

Cvičení:

- porovnejme vliv reset vs normalize na danou stránku

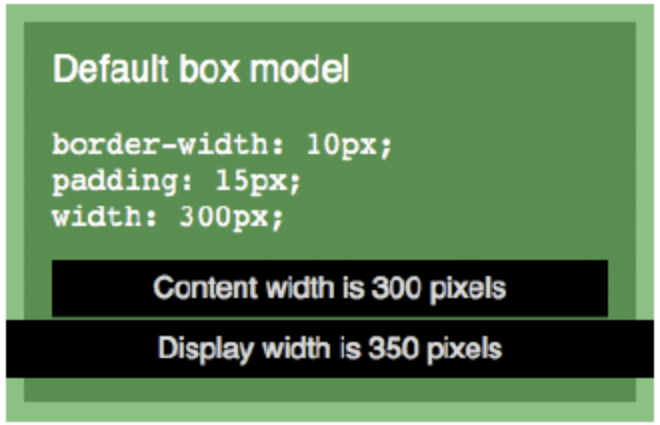
Uspořádání stránky, box model

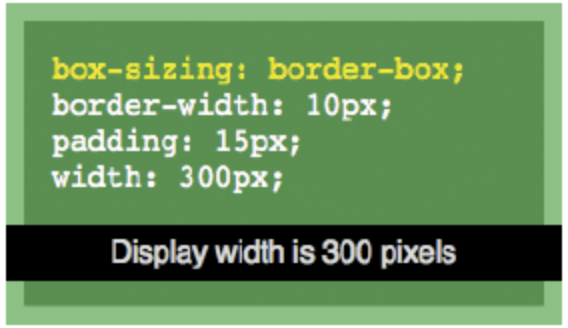
- CSS pracuje s modelem boxu, tj. každý element je uvnitř jednoho boxu
- *padding* = oblast obklopující hlavní obsah
- *border* = okraje okolo padding
- *margin* = poslední část, za border

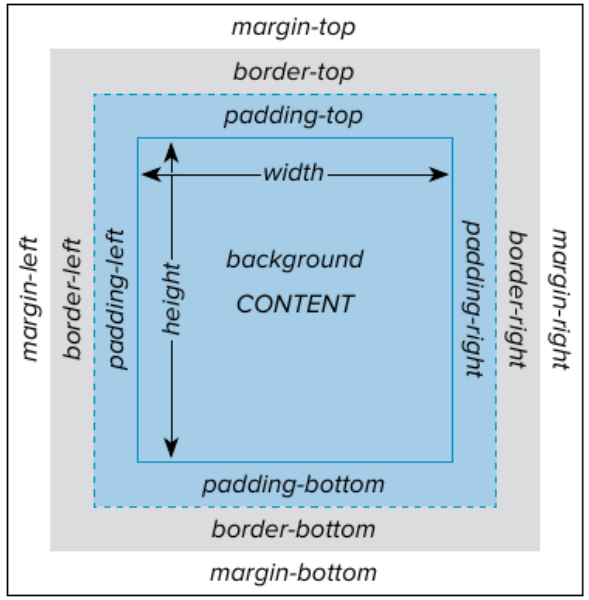


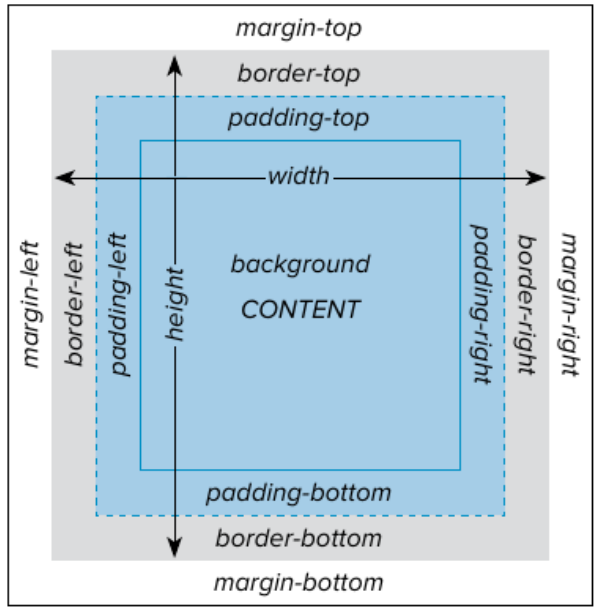
Šířka, výška v box modelu

- klasicky definovaná šířka pomocí `width` udává šířku obsahu (content area)
- přidáním `box-sizing:border-box;` k danému prvku, pak `border-width` definuje šířku okrajů, `padding` šířku padding (výplně) a `widht` celého boxu









Typ zobrazení

- *block-level* elementy (`<p>, <h1>, ...`) a inline *elementy* (`<em>, <cite>`)
- vlastnost `display` s hodnotami `block`, `inline`, `inline-block` a `none`
- typicky u `inline` prvků nemá smysl definovat šířku
- dále můžeme nastavit `margin-left` a `margin-right`

```
em{
  background: lightgreen;
  display: inline-block;
  width: 300px;
}
```

Padding (obálka)

- přidáme extra prostor okolo prvku
- nedefinujeme pro to barbu a texturu, to se přebírá od pozadí
  - `padding-top: 5em;`, `padding-right: 6em;`, `padding-bottom: 5em;` a `padding-left: 6em;` definují šířku v daných směrech
  - `padding: 5px;` - jedna hodnota aplikovaná do všech směrů
  - `padding: 5px 9px;` - dvě hodnoty, první na top a bottom, druhá na right a left
  - `padding: 5px 9px 11px;` - tři hodnoty, první na top, druhá na right a left, třetí na bottom
  - `padding: 5px 9px 11px 0;` - top, right, bottom, left

- levo bez obalky, pravo s obalkou

Okraje (border)

- stejně jako obálku můžeme definovat okraje daného elementu
- pro definici použijeme vlastnost `border-style` s hodnotou `none`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`
- pro nastavení šířky použijeme `border-width` s hodnou v px nebo em
- obdobně definujeme barvu `border-color`
- pro sloučenou definic použijeme `border` nebo `border-left`, `border-top`, … pro definici okraje na dané straně
- případně můžeme použít sloučenou vlastnost, např. `border-right-style`
- okraje se nedědí
- CSS3 zavádí vlastnost `border-image`

```
p {
  border: 10px solid red;
  padding: 15px;
}
.ddd{
  border-width:4px;
  border-style: dotted dashed double;
}

.ridge {
  border-style:border-color:
}
```

Vnější okraj (margin)

- průhledný prostor mezi jednotlivými elementy
- vlastnost `margin: x`, kde `x` je délka v daných jednotkách (absolutní/relativní/ `auto`)
- případně lze specifikovat okraj v daném směru `margin-top: x;` , `margin-right: x;` , …
- pokud máme dva elementy nad sebou, tak se pro margin mezi nimi použije maximum z `margin-bottom` prvku nad a `margin-top` prvku pod

```
.links li {
  margin-bottom: 1.1em;
}
```

Vnoření prvku do textu

- někdy se hodí například obrázek nechat obtékat textem, proto použijeme vlastnost `float` s hodnotami
  - `float` - obtékání zprava
  - `right` - obtékání zleva
  - `none` - žádné obtékání
- při použití `float` je dobré definovat `width` abychom měli kontrolu jak daleko boudou ostatní prvky
- tato vlastnost se nedědí a neovlivňuje výšku svých rodičů

```
.sidebar {
  float: right;
  margin-top: 1.875em; /* 30px/16px */
  width: 300px; /* 31.25% = 300px/960px */
}
```

- vlastnost `clear` - umožňuje zastavit efekt obtékání
- hodnoty `left` (ukončí levý), `right`, `both`, `none`
- např. chtějme aby zápatí bylo vysázeno až za obtéaným obrázkem:

```
.post-footer{
  clear: left;
}
```

- v některých případech je dobré umožnit automatický *clear* (clearfix method)

```
.clearfix:before,
.clearfix:after {
  content: " ";
  display: table;
}
.clearfix:after {
  clear: both;
}
.clearfix {
  *zoom: 1
}
```

- alternativně lze použít vlastnost `overflow` s vlastností `hidden` (zařízne obsah) nebo `auto` (přidá scroll bar) pro danýho předka

Pozicování elementů

- každý prvek má svou danou pozici podle HTML dokumentu
- pro upravení této pozice vytvoříme styl s vlastností `position`
- hodnoty jsou `static` (výchozí hodnota), `relative`, `absolute`, `fixed` (nedoporučeno - bugs pro smartphones)

Relativní pozice `relative`

- uvedeme vlastnost `top`, `right`, `bottom` , nebo `left` s hodnotou udávající offset od původní pozice (v px nebo v em)
- ostatní prvky toto ingnorují, může dojít k překryvu

```
.example {
  position: relative;
  top: 35px;
  left: 100px;
}
```

Absolutní pozice `absolute`

- udáme přesnou hodnotu pozice vůči `body` nebo předkovi s určenou pozicí (stačí mu přidat `position:relative;`)
- uvedeme vlastnost `top`, `right`, `bottom`, nebo `left` s hodnotou udávající offset od původní pozice (v px nebo v em)

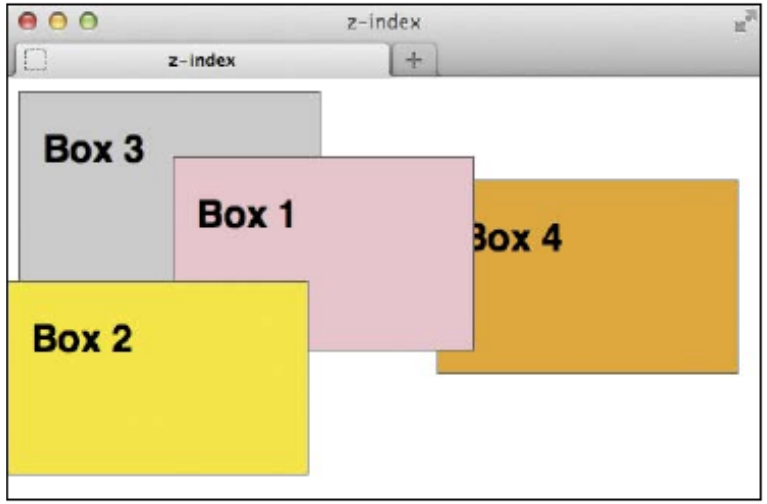
- vlastnost `position` se nedědí

Co s překryvy => z-index

- pokud chceme ovlivnit jak se jednotlivé prvky překrývají, uvedeme vlastnost `z-index` s hodnotou udávající pozici v překrytí (neplatí pro `static`)

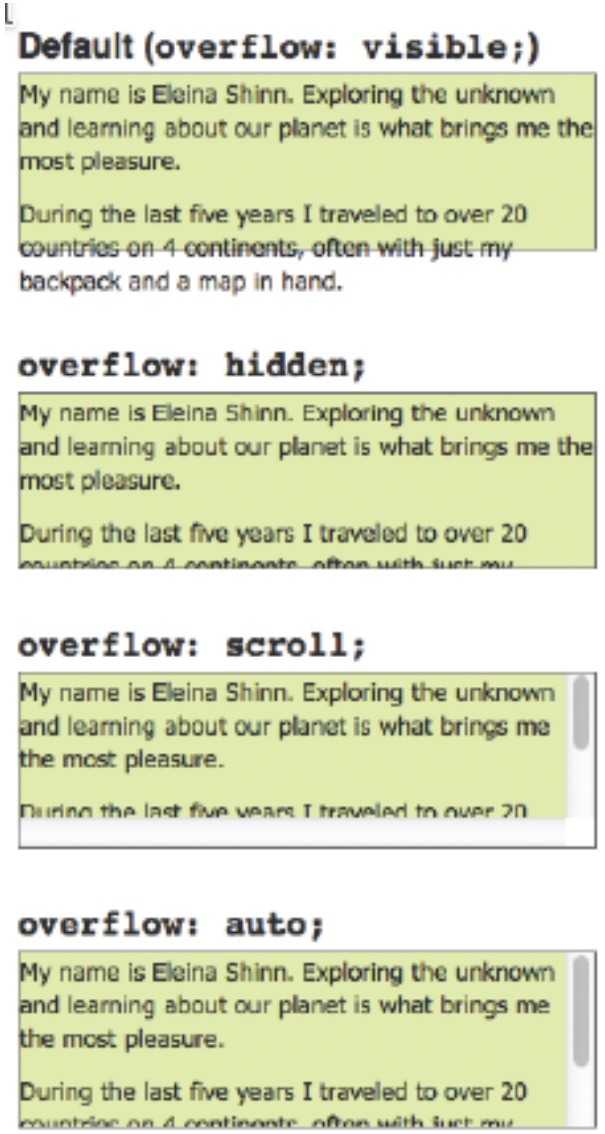
Co s překryvy => z-index

```
.box1 {
  background: pink;
  left: 110px;
  top: 50px;
  z-index: 120;
}
.box2 {
  background: yellow;
  left: 0;
  top: 130px;
  z-index: 530;
}
.box3 {
  background: #ccc;
  position: static;
  z-index: 1000;
}
.box4 {
  background:left: 285px;
  top: 65px;
  z-index: 3;
}
```



Co s přesahy

- některé prvky mohou přelézt ven ze svého boxu, příkladem je třeba příliš velký obrázek
- přesahy lze kontrolovat pomocí vlastnosti `overflow`
- hodnoty:
  - `visible` - přesah je vidět (váchozí hodnota)
  - `hidden` - přesah se nezobrazí
  - `scroll` - přidán scroll bar při překryvu
  - `auto` - scroll bars se přidají jen když jsou potřeba
- tato vlastnost se nedědí



Vertikální zarovnání

- použití vertikálního zarovnání vylepší grafickou podobu daného webu
- vlastnost `vertical-align` s hodnotami
  - `baseline` - element se zarovná na *baseline* svého předka
  - `middle` - střed prvku se zarovná na *baseline* svého předka
  - `sub` - zarovnání na dolní index předka
  - `super` - zarovnání na horní index předka

```css
input[type="image"] {
  vertical-align: bottom;
}
```

- další hodnoty `text-top`, `text-top`, `top`, `bottom`, `x%`, `xpx`, `xem`

Změna kurzoru

- výchozí CSS většiny prohlížečů změnu kurzoru obsluhuje, tj. například klasický kurzor se změní na "prst" při najetí na odkaz,...
- chování lze změnit vlastností `cursor` s hodnotami
  - `pointer`, `default`, `crosshair`, `move`, `wait`, `help`, `text`, `progress`, `auto`
  - `x-resize` oboustranná šipka, kde x určuje kam směřuje (`n`, `nw`, `e`, ...)

Cvičení:

- danou stránku stylizujte podle zadání

Zdroje:

Castro, E. (2006). HTML, XHTML, and CSS: Visual QuickStart Guide. Pearson Education.