Kaskádové styly

4IZ228 – tvorba webových stránek a aplikací

Jirka Kosek

Poslední modifikace: $Date: 2014/10/02 11:38:43 $

Obsah

Úvod 3
Důvody vzniku CSS 4
Problémy s rádoby graficky dokonalými stránkami 5
Řešení problému = CSS 6
Jednoduchý styl v CSS 7
Historie 8
Podpora v prohlížečích 9
Připojení stylu ke stránce 10
Připojení stylu z externího souboru 11
Styl pro celou stránku zapsaný přímo v HTML dokumentu 12
Ukázka stylu vloženého do dokumentu 13
Specifikace stylu přímo u elementu 14
Alternativní styly 15
Selektory 16
Strom dokumentu 17
Univerzální selektor 19
Selektor typu 20
Selektor potomků 21
Selektor dětí 22
Selektor sousedících sourozenců 23
Selektor třídy 24
Selektor ID 25
Atributový selektor 26
Pseudotřídy 27
Pseudoelementy 28
Další selektory podporované v CSS3 29
Další využití selektorů 30
Vlastnosti 31
Dědění vlastností 32
Kaskáda – skládání vlastností 33
Písmo 34
Obecné rodiny písma 35
Uživatelem definovaná písma 36
Barvy, pozadí 37
Formátování textu 38
Formátovací model 39
Podopora několika výstupních médií 40
Přizpůsobení stylu na základě parametrů koncového zařízení 41
Stránkovaný výstup 42
Generování obsahu, automatické číslování 43
CSS pozicování (aneb pryč s rámy a tabulkovým layoutem) 44
Transformace 45
Animace/přechody 46
CSS preprocesory 47
Další zdroje informací 48
Další zdroje informací 49

Úvod

Důvody vzniku CSS 4
Problémy s rádoby graficky dokonalými stránkami 5
Řešení problému = CSS 6
Jednoduchý styl v CSS 7
Historie 8
Podpora v prohlížečích 9

Důvody vzniku CSS

- zákazníci chtějí, aby webové stránky vypadly stejně jako tištěný katalog
  - přesné druhy písma
  - přesné barvy
  - přesné zarovnání a formátování jednotlivých částí stránky
- weboví designeři zákazníkům vyhoví
  - stránky plné tagů `<font color="..." face="..." size="...">`
  - složité, mnohdy vzájemně vnořené tabulky pro dosažení požadovaného efektu
  - mnoho částí stránky tvoří obrázky – umožňují věrně reprodukovat fonty a formátování
  - v horším případě je celá stránka jeden obrázek, případně jeden rozřezaný obrázek

Problémy s rádoby graficky dokonalými stránkami

- zbytečně dlouhý HTML kód, mnoho obrázku → dlouho se přenáší
- schopnosti prohlížečů se využívají až nadoraz – v jednom prohlížeči stránky vypadají dobře a v tom druhém dost špatně
- při použití jiného rozlišení, než ve kterém jsou stránky navrženy, to také nedopadne moc dobře
- změny v designu stránek jsou velice obtížné, protože je potřeba nahradit mnoho výskytů tagů a atributů ovlivňujících formátování

Řešení problému = CSS

- CSS (Cascading Style Sheets) – kaskádové styly
- CSS umožňují oddělit vzhled a obsah stránky
- vzhled jednotlivých elementů je úsporně definován odděleně od HTML kódu
- jeden styl může být sdílen více stránkami
  - jednotný vzhled
  - rychlé změny designu

Jednoduchý styl v CSS

Nejjednodušší kaskádový styl může vypadat asi takto:

```
h1 { color: blue }
```

Pomocí tohoto stylu definujeme, že všechny nadpisy vytvořené pomocí elementu `h1` mají mít modrou barvu. V našem případě je celý styl tvořen pouze jedním *pravidlem*. Každé pravidlo má dvě části – *selektor* (v našem případě `h1`) a *deklaraci* (`color: blue`).

Selektor určuje elementy, na které bude deklarace aplikována.

Každá deklarace se skládá ze dvou částí – z *vlastnosti* a její *hodnoty*. Deklarace můžeme sdružovat dohromady, pokud je oddělíme pomocí středníku:

```
h1 { color: blue; text-align: center }
```

Historie

začátek 90. let
- první prohlížeče, vzhled HTML si může definovat uživatel pomocí jednoduchého stylu

polovina 90. let
- spor: má si styl určovat autor nebo uživatel?
- dočasně vítězí autor

1996
- specifikace CSS1

1998
- specifikace CSS2

2011
- specifikace CSS2.1
- většina prohlížečů umožňuje uživatelské předefinování vzhledu stránky kvůli přístupnosti

???
- modulární specifikace CSS3
- některé moduly již prohlížeče podporují

Podpora v prohlížečích

- současné prohlížeče umí CSS2.1 a některé věci z CSS3
- informace o aktuální podpoře v prohlížečích
  - http://caniuse.com/#cats=CSS
  - http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(Cascading_Style_Sheets)

Připojení stylu ke stránce

Připojení stylu z externího souboru 11
Styl pro celou stránku zapsaný přímo v HTML dokumentu 12
Ukázka stylu vloženého do dokumentu 13
Specifikace stylu přímo u elementu 14
Alternativní styly 15

Připojení stylu z externího souboru

Tento případ je nejvyužívanější, protože umožňuje využití jednoho stylu několika stránkami. Styl se k dokumentu připojuje pomocí elementu `link`, který můžeme použít v záhlaví stránky.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Pokusná stránka se stylem</title>
    <link href="styl.css" type="text/css" rel="StyleSheet">
  </head>
  <body>
  ...
  </body>
</html>
```

Styl pro celou stránku zapsaný přímo v HTML dokumentu

Tento způsob využijeme v případech, kdy chceme definovat vzhled jen jedné stránky a neplánujeme použití stylu na dalších stránkách.

Styl se vkládá do záhlaví dokumentu do elementu `style`. Pomocí atributu `type` můžeme určit typ používaného stylového jazyka, pokud jej neuvedeme, předpokládá se CSS.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Pokusná stránka se stylem</title>
    <style type="text/css">
        ... definice stylu ...
    </style>
  </head>
  <body>
  ...
  </body>
</html>
```

Ukázka stylu vloženého do dokumentu

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Pokusná stránka se stylem</title>
    <style>
        h1 { color: blue; text-align: center }
        h2 { color: red }
        p  { text-align: justify }
    </style>
  </head>
  <body>
  ...
  </body>
</html>
```

Specifikace stylu přímo u elementu

Poslední možnost definice stylu již trošku odporuje samotné filosofii stylů, která se snaží oddělit definici vzhledu od samotného obsahu dokumentu. U každého elementu můžeme použít atribut `style` a v něm přímo uvést deklaraci stylu. Příklad:

```
...
<p style="color: yellow; text-align: right">Tento jediný
odstavec bude žlutý a zarovnaný vpravo.</p>
...
```

Alternativní styly

- ke stránce může být připojeno více stylů najednou
- uživatel mezi nimi může přepínat

```
<link href="always.css" type="text/css" rel="stylesheet">
<link title="Blue design" href="bluetitle.css" type="text/css" ▶
rel="stylesheet">
<link title="Blue design" href="bluepara.css" type="text/css" ▶
rel="stylesheet">
<link title="Aligned design" href="alignedtitle.css" type="text/css" ▶
rel="alternate stylesheet">
<link title="Aligned design" href="alignedpara.css" type="text/css" ▶
rel="alternate stylesheet">
<link title="Null design" href="null.css" type="text/css" rel="alternate ▶
stylesheet">
```

- většina prohlížečů nenabízí možnost přepnutí stylů a ani si změnu stylu nepamatuje při přechodu mezi stránkami, musí se tak řešit přes JavaScript

Selektory

Strom dokumentu 17
Univerzální selektor 19
Selektor typu 20
Selektor potomků 21
Selektor dětí 22
Selektor sousedících sourozenců 23
Selektor třídy 24
Selektor ID 25
Atributový selektor 26
Pseudotřídy 27
Pseudoelementy 28
Další selektory podporované v CSS3 29
Další využití selektorů 30

Strom dokumentu

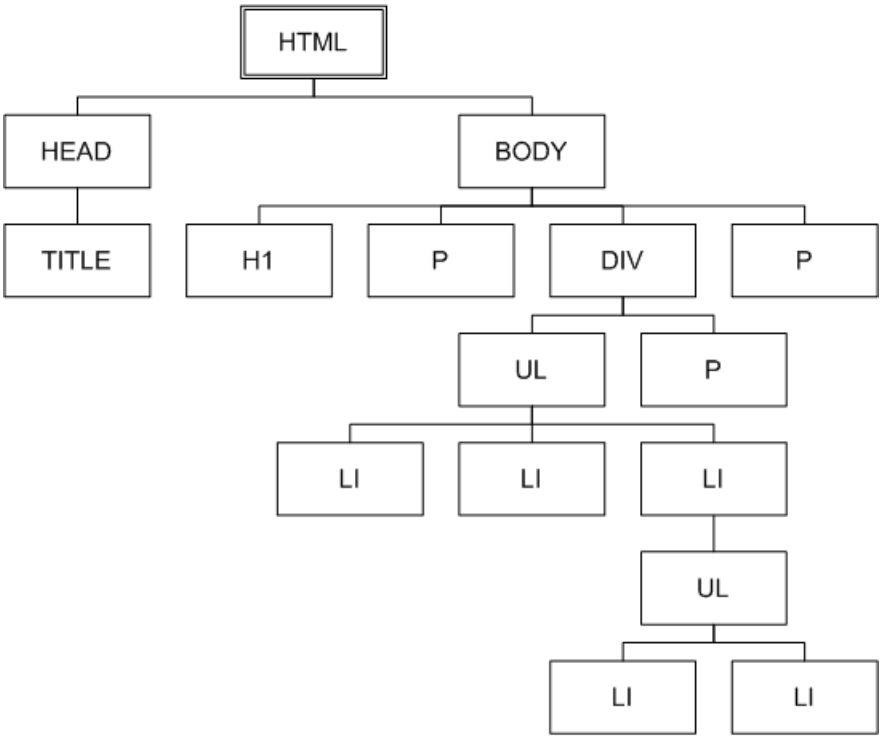
- selektory vybírají části stromu dokumentu
- každému elementu odpovídá uzel ve stromu dokumentu

Příklad 1. Ukázkový dokument HTML

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Sample title</title>
  </head>
  <body>
    <h1 id="chapter1">Title</h1>
    <p>Text1</p>
    <div class="content">
      <ul>
        <li>Item1</li>
        <li>Item2</li>
        <li>
          Item3
          <ul type="circle">
            <li>Nested item1</li>
            <li>Nested item2</li>
          </ul>
        </li>
      </ul>
      <p>Text2</p>
    </div>
    <p>Text3</p>
  </body>
</html>
```

Strom dokumentu (Pokračování)

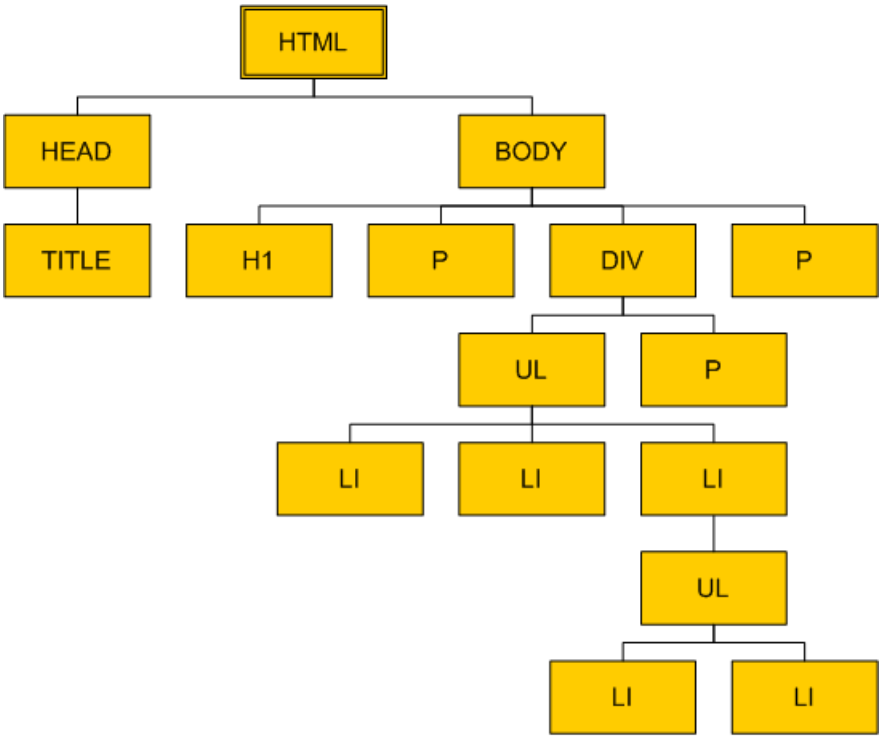
Obrázek 1. Strom dokumentu



Univerzální selektor

- *
- vybere všechny elementy v dokumentu

Obrázek 2. Elementy vybrané pomocí *

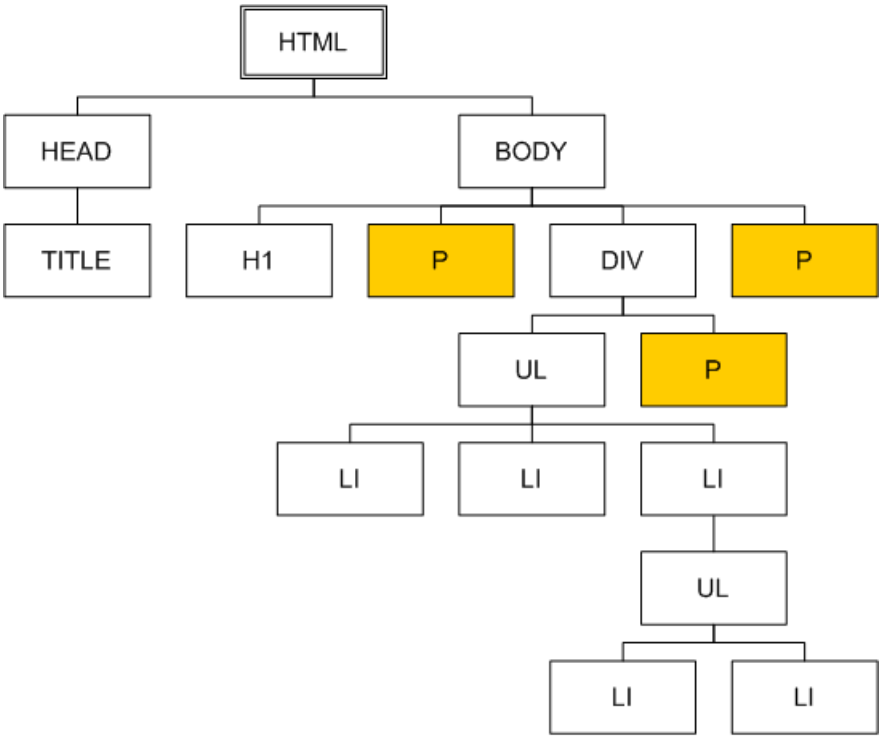


Selektor typu

- vybere element na základě jeho typu (jména)
- příklady:

```
p { … }
h1 { … }
```

Obrázek 3. Elementy vybrané pomocí `p`

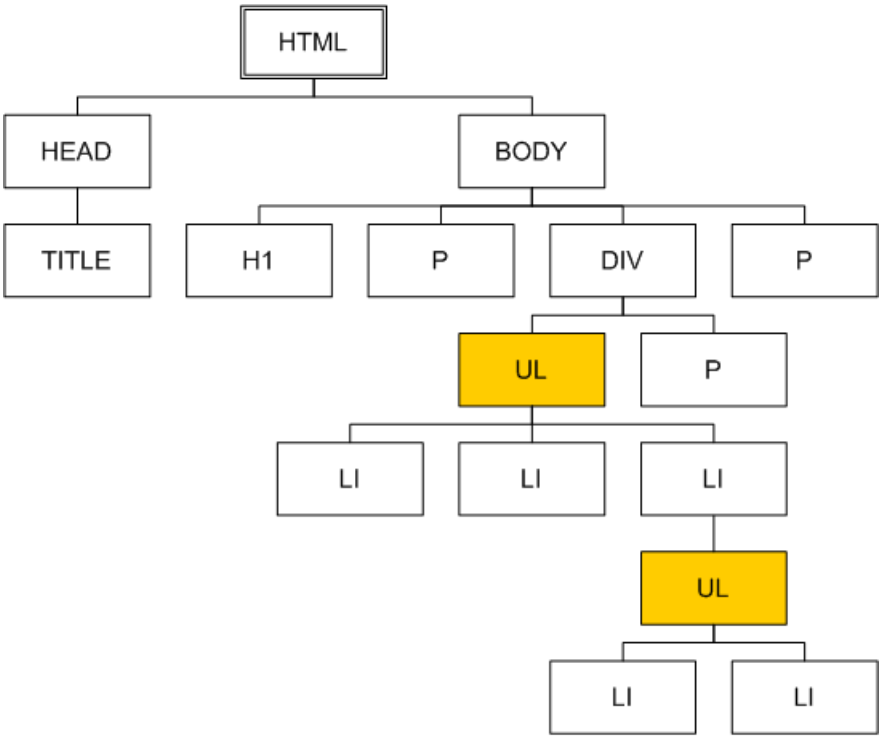


Selektor potomků

- vybere element jen když je potomkem určitého elementu
- příklady:

```
ul li { … }
h1 em { … }
div ol li { … }
```

Obrázek 4. Elementy vybrané pomocí `div ul`



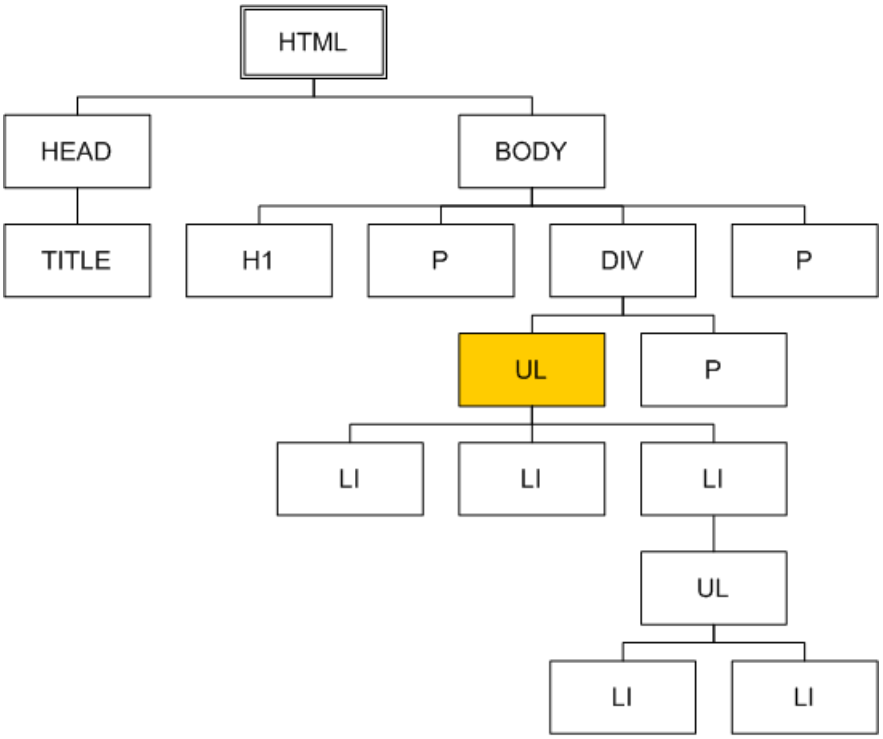
Selektor dětí

- vybere element jen když je dítětem určitého elementu
- příklady:

```
ul > li { … }
h1 > em { … }
```

- dítě je ve stromu právě o jednu úroveň níž
- potomek může být libovolně hluboko

Obrázek 5. Elementy vybrané pomocí `div > ul`

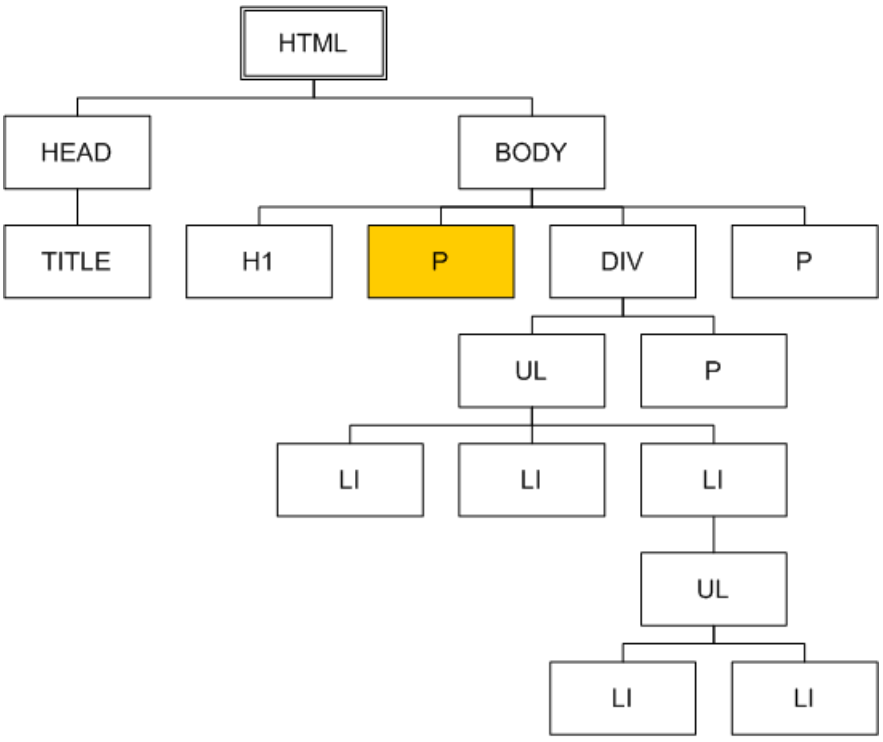


Selektor sousedících sourozenců

- vybere element pouze pokud se vyskytuje bezprostředně za jiným elementem
- oba elementy musí mít stejného rodiče
- příklady:

```
table + p { … }
h1 + p { … }
```

Obrázek 6. Elementy vybrané pomocí `h1 + p`



Selektor třídy

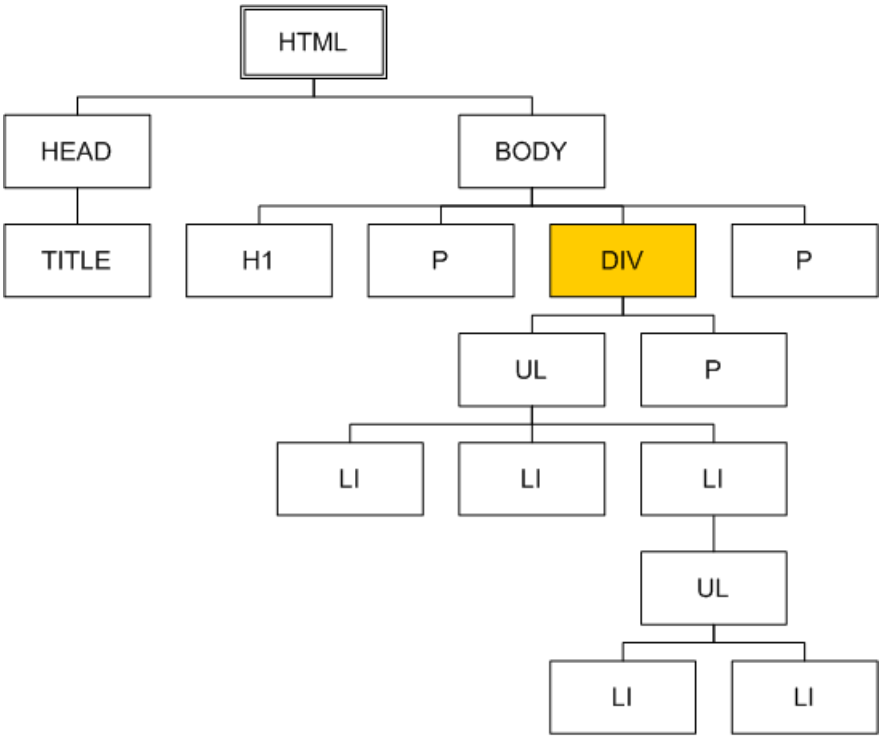
- vybere elementy, které mají nastavenu určitou třídu pomocí atributu `class`

```
<div class="content">…</div>
```

- atribut `class` může obsahovat mezerami oddělený seznam tříd
- příklady:

```
.content { … }
div.content { … }
```

Obrázek 7. Elementy vybrané pomocí `.content`

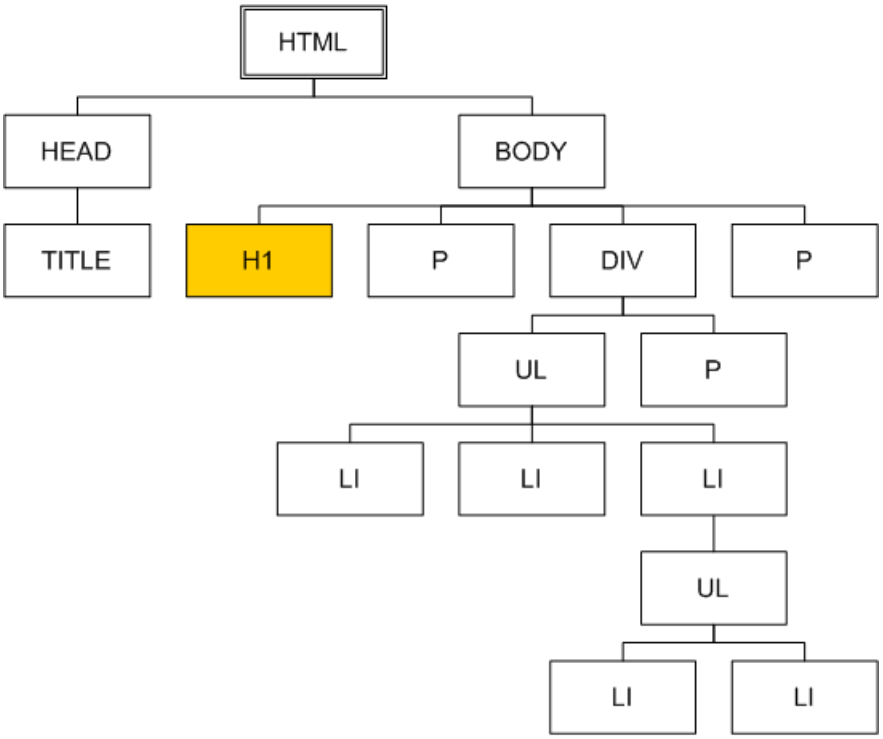


Selektor ID

- vybere elementy, které mají v atributu typu ID určitou hodnotu
- v HTML je atribut `id` dostupný na skoro všech elementech
- hodnota ID musí být v celém dokumentu jedinečná
- selektor se používá pro ošetření výjimek v dokumentu
- příklady:

```
#chapter1 { … }
div#chapter1 { … }
```

Obrázek 8. Elementy vybrané pomocí `#chapter1`



Atributový selektor

- vybere elementy na základě přítomnosti atributu nebo určité hodnoty uvnitř atributu
- `[foo]`
  vybere elementy, které mají atribut `foo`

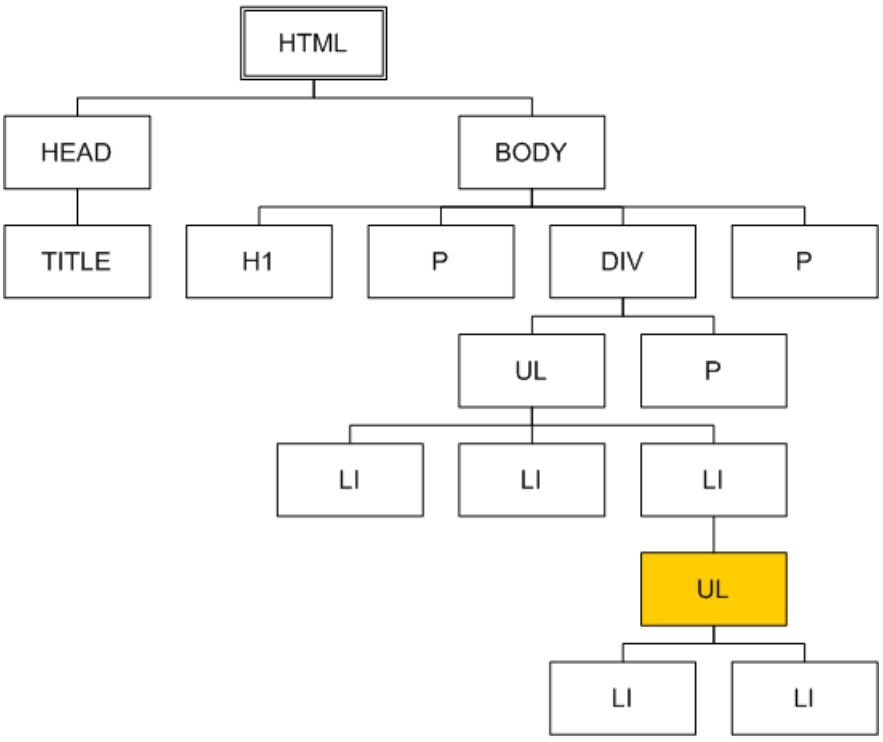
  `[foo="bar"]`
  vybere elementy, které mají v atributu `foo` hodnotu `bar`

  `[foo~="bar"]`
  vybere elementy, které mají atribut `foo`, v němž je seznam hodnot oddělených mezerou a jednou z těchto hodnot je i `bar`

  `[foo|="bar"]`
  vybere elementy, které mají atribut `foo`, v němž je seznam hodnot oddělených pomlčkou a první z těchto hodnot je `bar`
- příklady:

```
ul[type="circle"] { … }
[type="circle"] { … }
*[type="circle"] { … }
h1[align] { … }
```

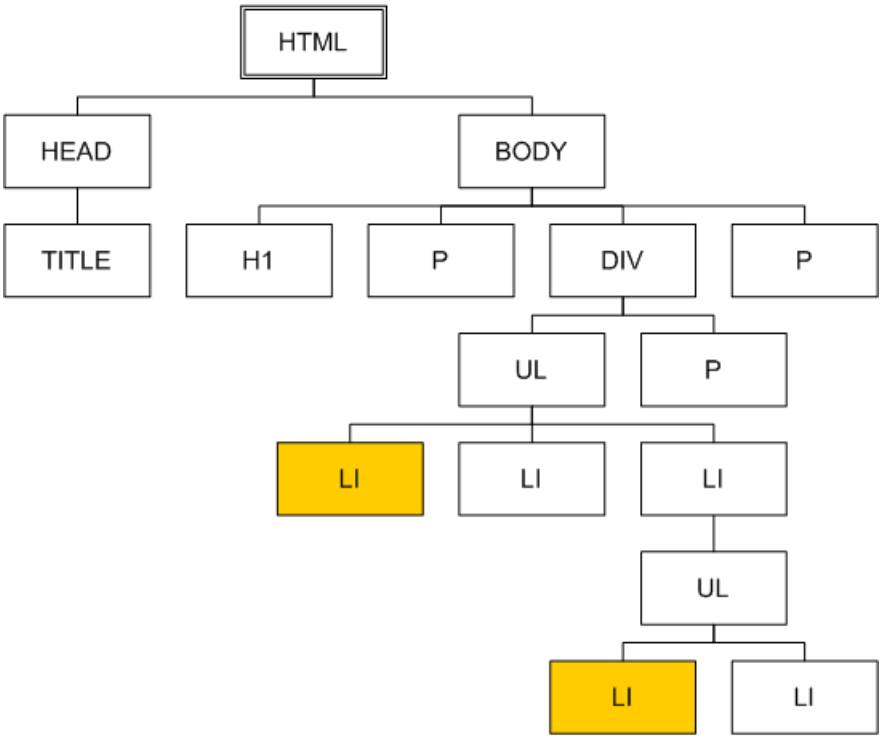
Obrázek 9. Elementy vybrané pomocí `ul[type="circle"]`



Pseudotřídy

- výběr nezáleží na pozici elementu ve stromu dokumentu, ale na stavu prohlížeče a interakci s uživatelem
- `:first-child`
  vybere element, pokud je to první díte svého rodiče

Obrázek 10. Elementy vybrané pomocí `li:first-child`



`:link`
vybere odkazy, které ještě nebyly navštívené

`:visited`
vybere již navštívené odkazy

`:hover`
vybere element, který je pod kurzorem myši

`:active`
vybere element, který je aktivovaný uživatelem

`:focus`
vybere element, který má fokus (přijímá vstup z klávesnice)

`:lang(cs)`
vybere elementy v určitém jazyce (v tomto případě v češtině)

Pseudoelementy

- vybírají „virtuální“ elementy, které nejsou součástí stromu dokumentu
- `:first-line`
  Vybere první řádku elementu, který je formátován jako odstavec

  `:first-letter`
  Vybere první písmeno elementu

Další selektory podporované v CSS3

`:root`
: kořenový element

`:nth-child(`*n*`)`, `:nth-last-child(`*n*`)`
: *n*-tý podlement

`:nth-of-type(`*n*`)`, `:nth-last-of-type(`*n*`)`
: *n*-tý podlement daného typu

`:first-of-type`, `:last-of-type`, `:only-child`, `:only-of-type`, `:empty`, `:target`, `:enabled`, `:disabled`, `:checked`
: podrobnější vysvětlení ve specifikaci Selectors Level 3[1]

[1] http://www.w3.org/TR/selectors/

Další využití selektorů

- dotazovací jazyk v javascriptových frameworcích (např. jQuery)

```
$("ol > li").addClass("blue");
```

- kvůli oblíbenosti bylo nakonec vytvořeno standardní javascriptové rozhraní Selectors API[2]

```
var tableRows = document.querySelectorAll("tr");
```

[2] http://www.w3.org/TR/selectors-api/

Vlastnosti

Dědění vlastností 32
Kaskáda – skládání vlastností 33
Písmo 34
Obecné rodiny písma 35
Uživatelem definovaná písma 36
Barvy, pozadí 37
Formátování textu 38
Formátovací model 39
Podopora několika výstupních médií 40
Přizpůsobení stylu na základě parametrů koncového zařízení 41
Stránkovaný výstup 42
Generování obsahu, automatické číslování 43
CSS pozicování (aneb pryč s rámy a tabulkovým layoutem) 44
Transformace 45
Animace/přechody 46
CSS preprocesory 47

Dědění vlastností

Snadnost a intuitivnost použití kaskádových stylů je do velké míry dána děděním vlastností. Pokud nějakou vlastnost u elementu nastavíme a tento element obsahuje další elementy, automaticky tyto elementy dědí vlastnosti rodičovského elementu. Tímto způsobem je ve většině případů dosaženo nejlepšího výsledku bez nutnosti tvorby složitých selektorů.

Příklad 2. Dědění vlastnosti

```
h1 { color: blue }

<h1>Tohle je modré <em>a tohle kupodivu taky</em> -
barva se dědí</h1>
```

U některých vlastností není dědění vhodné a proto se tyto vlastnosti nedědí.

Kaskáda – skládání vlastností

- pro některé vlastnosti může být ve stylu/stylech určeno několik konfliktních hodnot
- kaskáda určuje, která deklarace má nejvyšší váhu
- primárně se priorita určuje podle:
  1. uživatelova stylu s příznakem important
  2. autorova stylu s příznakem important
  3. autorova normálního stylu
  4. uživatelova normálního stylu
  5. výchozího stylu prohlížeče
- pokud to nestačí, nejvyšší prioritu mají
  - inline deklarace stylu (atribut `style`)
  - pak deklarace se selektorem ID
  - pak deklarace s atributovým selektorem a pseudotřídami
  - a nakonec deklarace se selektorem typu nebo pseudoelementy
- pokud to pořád nestačí, použije se pozdější deklarace

Písmo

- rodina písma
- styl písma – normální, kurzíva, skloněné
- varianta písma – normální a kapitálky
- duktus písma (síla tahu)
- velikost

Příklad 3. Nastavení písma

```
blockquote { font-weigth: bold;
             font-style: italic;
             font-size: 12pt;
             line-height: 14pt;
             font-family: "Times Roman", serif }
```

Obecné rodiny písma

Jednotlivé OS používají různá písma → CSS pak nemusí být přenositelné

Řešení obecné rodiny písma, které se použijí, když se nenajde konkrétní font:

- `serif` – patkové písmo
- `sans-serif` – bezpatkové písmo
- `cursive` – ozdobná kurzíva
- `fantasy` – ozdobné písmo
- `monospace` – neproporcionální písmo

Uživatelem definovaná písma

- CSS nabízejí prostředky pro definici a stažení písma do prohlížeče
- prohlížeče podporují různé formáty písem
- v poslední době konečně došlo ke shodě na jednom formátu WOFF (Web Open Font Format)
- existují různé služby pro snadné zařazení písma v mnoha formátech – http://www.fontsquirrel.com/, http://www.google.com/webfonts, http://html.adobe.com/edge/webfonts/
- licence mnoha písem toto použití neumožňuje
- pozor na podporu českých znaků (http://misc.maly.cz/webfonts.php)

```
@font-face
{
  font-family: Baskerville;
  src: url(http://example.com/fonts/baskerville.woff);
}

h1 { font-family: Baskerville }
```

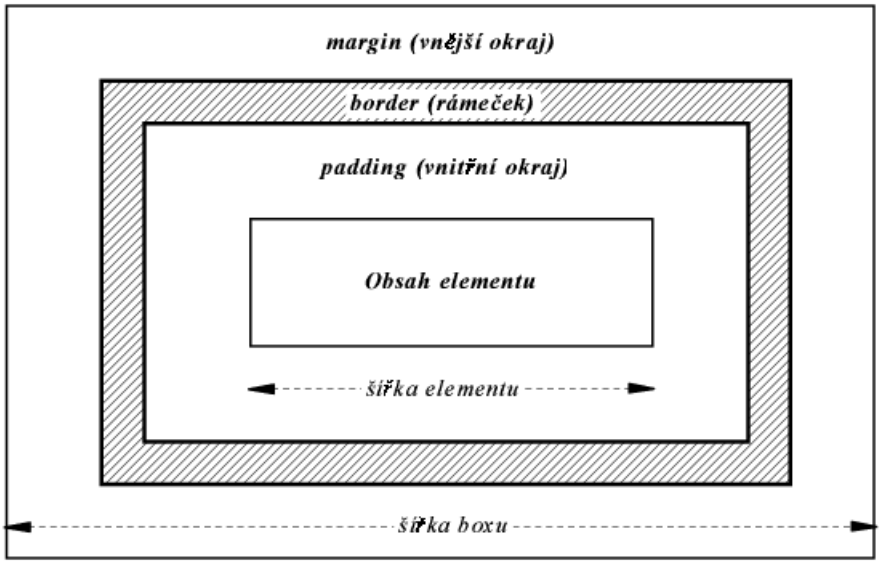
Barvy, pozadí

- barva textu
- barva pozadí
- obrázek na pozadí
  - opakování
  - souřadnice umístění
  - způsob rolování při posunu v okně

Formátování textu

- způsob zarovnání – doleva, doprava, centrování, do bloku
- řádkování
- vertikální zarovnání objektů na řádce
- velikost odstavcové zarážky – odsazení první řádky odstavce

Formátovací model



- u všech vnějších a vnitřních okrajů a u rámečku lze nastavit přesné rozměry
- u rámečku navíc barvu a tvar
- způsob zobrazení elementu – block, inline, list-item, none
- další možnosti – plovoucí elementy, výška, šířka, zachování mezer a konců řádek, vzhled seznamů apod.

Podopora několika výstupních médií

- přímo ve stylu:

```
@media print {
  BODY { font-size: 10pt }
}
@media screen {
  BODY { font-size: 12pt }
}
@media screen, print {
  BODY { line-height: 1.2 }
}
```

- připojení několika různých stylů:

```
<link rel="stylesheet" type="text/css" href="zaklad.css" media="all">
<link rel="stylesheet" type="text/css" href="online.css" media="screen">
<link rel="stylesheet" type="text/css" href="tisk.css" media="print">
```

- stránka by měla mít připojení minimálně speciální styl pro tisk, který při tisku skryje navigaci a další „zbytečnosti“

Přizpůsobení stylu na základě parametrů koncového zařízení

- „media queries“ – umožňují jednu a tutéž stránku inteligentně zobrazit na různých koncových zařízeních
- styly se použijí selektivně na základě podmínek testujících např. rozlišení obrazovky, orientaci displeje, barevnou hloubku
- podmíněně lze načíst celý styl:

```
<link rel="stylesheet" href="velkaobrazovka.css" media="screen and ▶
(min-width: 1200px)">
<link rel="stylesheet" href="smartphone.css" media="screen and ▶
(max-device-width: 480px)">
```

- případně můžeme mít jen podmíněné sekce uvnitř jednoho stylu:

```
@media screen and (min-width: 500px) and (max-width: 1200px) {
  …pravidla…
  }

@media screen and (min-width: 1200px) {
  …pravidla…
  }

@media screen and (max-width: 500px) {
  …pravidla…
  }
```

- ukázka[3]

[3] http://www.alistapart.com/d/responsive-web-design/ex/ex-site-FINAL.html

Stránkovaný výstup

- nastavení okrajů na stránce; zvlášť pro liché, sudé a první
- řízení stránkového zlomu
- kontrola vdov a sirotků
- možnosti se postupně rozšiřují a pro jednodušší publikace je již je možné generovat PDF v tiskové kvalitě přímo z HTML+CSS
  - Prince XML[4]
  - Antenna House CSS Formatter[5]
  - Weasy Print[6]

```
h2, h3, h4 { pabe-break-after: avoid; }
h1 { page-break-before: always;
thead, tr { page-break-inside: avoid; }
```

[4] http://www.princexml.com/
[5] http://antennahouse.com/product/ahf60/ahf6top.htm
[6] http://weasyprint.org/

Generování obsahu, automatické číslování

- pseudoselektory `before` a `after`, vlastnost `content`

```
p.note:before { content: "Note: " }
```

- automatické číslování

```
h1:before {
    content: "Chapter " counter(chapter) ". ";
    counter-increment: chapter;  /* Add 1 to chapter */
    counter-reset: section;      /* Set section to 0 */
}
h2:before {
    content: counter(chapter) "." counter(section) " ";
    counter-increment: section;
}
```

CSS pozicování

aneb pryč s rámy a tabulkovým layoutem

- CSS umožňují vytvářet složitý layout jednodušším a kratším kódem než tabulky
- změny takového layoutu jsou pak velmi jednoduché
- vlastnost `position` může nabývat mnoho hodnot (např. static, relative, absolute, fixed)
- pokročilejší možnosti layoutu je možno realizovat pomocí vlastnosti `display` a hodnot table, flex, grid

Transformace

- jakýkoliv objekt je možné libovolně 2D transformovat (posunutí, otočení, skosení, zvětšení/zmenšení)
- lze aplikovat několik transformací najednou
- prohlížeče zatím většinou implementují jen verzi s „vendor prefixem“

```
h1 { color: red;
     -o-transform: rotate(30deg) translateY(200px);
     -webkit-transform: rotate(30deg) translateY(200px);
     -ms-transform: rotate(30deg) translateY(200px);
     -moz-transform: rotate(30deg) translateY(200px);
     transform: rotate(30deg) translateY(200px); }
```

Animace/přechody

- vlastnosti CSS lze jednoduše nechat přejít do jiné hodnoty a tím vytvořit dojem animace
- animovat lze několik vlastností najednou
- lze ovládat rychlost a tempo animace
- podporováno jen na některých vlastnostech
- podpora se teprve rozšiřuje, proto se opět používají „vendor prefixy"

CSS preprocesory

- pro profesionální použití některé věci CSS neumí
  - proměnné pro snadné modifikace hodnot, které se vyskytují na více místech stylu (např. použité barevné schéma)
  - ošetření prefixovaných/neprefixovaných vlastností
  - výpočty
- lze řešit pomocí preprocesorů CSS
  - Sass[7]
  - LESS[8]
  - Stylus[9]

[7] http://sass-lang.com/
[8] http://lesscss.org/
[9] http://learnboost.github.com/stylus/

Další zdroje informací

Další zdroje informací 49

Další zdroje informací

- CSS2.1[10]
- úvod do CSS v češtině[11]
- on-line přehled vlastností CSS v češtině[12]
- historie CSS[13]
- úvod do CSS od W3C[14]
- validátor CSS stylů[15]
- další validátor CSS stylů[16]
- impresivní ukázka CSS[17]
- přehled podpory CSS v jednotlivých prohlížečích[18]

[10] http://www.w3.org/TR/CSS21
[11] http://htmlguru.cz/css.html
[12] http://www.kosek.cz/clanky/dhtml/css-properties.html
[13] http://www.w3.org/Style/LieBos2e/history/
[14] http://www.w3.org/MarkUp/Guide/Style
[15] http://jigsaw.w3.org/css-validator/
[16] http://www.htmlhelp.com/tools/csscheck/
[17] http://www.csszengarden.com/
[18] http://www.quirksmode.org/css/contents.html