Kaskádové styly

4IZ228 – tvorba webových stránek a aplikací

Jirka Kosek

Poslední modifikace: $Date: 2007/02/22 16:41:28 $

Obsah

Úvod

Proč jsou dnešní stránky plné zbytečného kódu

- zákazníci chtějí, aby webové stránky vypadly stejně jako tištěný katalog
  - přesné druhy písma
  - přesné barvy
  - přesné zarovnání a formátování jednotlivých částí stránky
- webový 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

nyní
- práce na verzi 2.1 (revize verze 2) a 3
- většina prohlížečů umožňuje uživatelské předefinování vzhledu stránky

Podpora v prohlížečích

- nejlepší podpora (většina z CSS2) – Opera, Mozilla, Safari, IE7
- dobrá podpora (většina z CSS2 až na pár podstatných výjimek) – IE6
- použitelná podpora (většina z CSS1 a velice málo z CSS2) – Internet Explorer od verze 4.0
- nic moc – Internet Explorer 3.0 a Netscape Navigator 4.0

Připojení stylu ke stránce

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 PUBLIC '-//W3C//DTD HTML 4.0 Transitional//EN'>
<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` musíme určit typ používaného stylového jazyka. Obvykle se celá definice stylu uzavírá do HTML komentáře, aby nebyla chybně interpretována staršími prohlížeči bez podpory kaskádových stylů.

```
<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0 Transitional//EN'>
<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 PUBLIC '-//W3C//DTD HTML 4.0 Transitional//EN'>
<html>
  <head>
    <title>Pokusná stránka se stylem</title>
    <style type="text/css">
      <!--
        h1 { color: blue; text-align: center }
        h2 { color: red }
        p  { text-align: justify }
      -->
    </style>
  </head>
  <body>
  ...
  </body>
</html>
```

Načtení externího stylu z interního

Tuto možnost nedoporučuji moc používat, protože ji správně nepodporují všechny prohlížeče. Je tu jen pro úplnost.

```
<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0 Transitional//EN'>
<html>
  <head>
    <title>Pokusná stránka se stylem</title>
    <style type="text/css">
      <!--
        @import url(URL adresa importovaného stylu);
        ... definice stylu ...
      -->
    </style>
  </head>
  <body>
  ...
  </body>
</html>
```

Někdy se používá pro odříznutí starších prohlížečů s chybami v implementaci CSS od stylu

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">
```

- IE nenabízí možnost přepnutí stylů, musí se řešit přes JavaScript

Selektory

Strom dokumentu

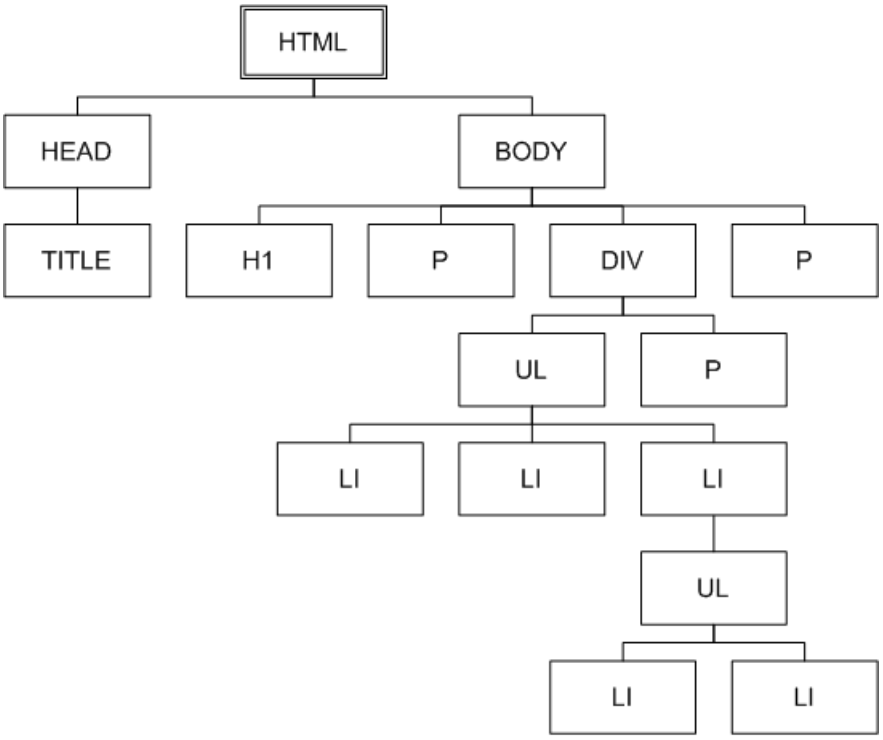
- selektory vybírají části stromu dokumentu
- každému elementu odpovídá uzel ve stromu dokumentu

Příklad 1. Ukázkový dokument HTML

```
<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'>
<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)

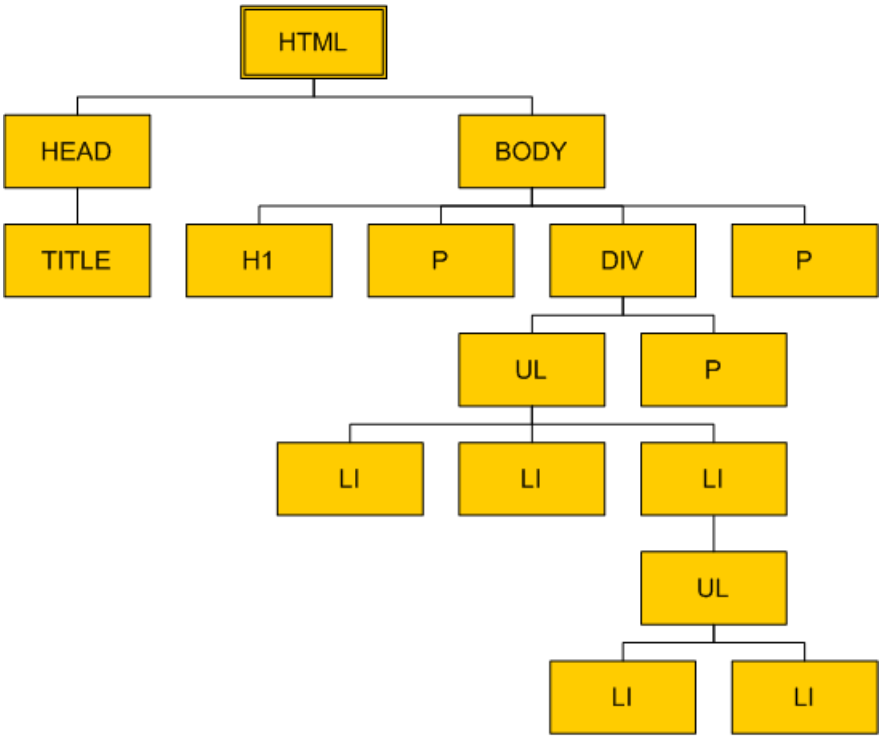
Obrázek 1. Strom dokumentu



Univerzální selektor

- *
- vybere všechny elementy v dokumentu

Obrázek 2. Elementy vybrané pomocí *

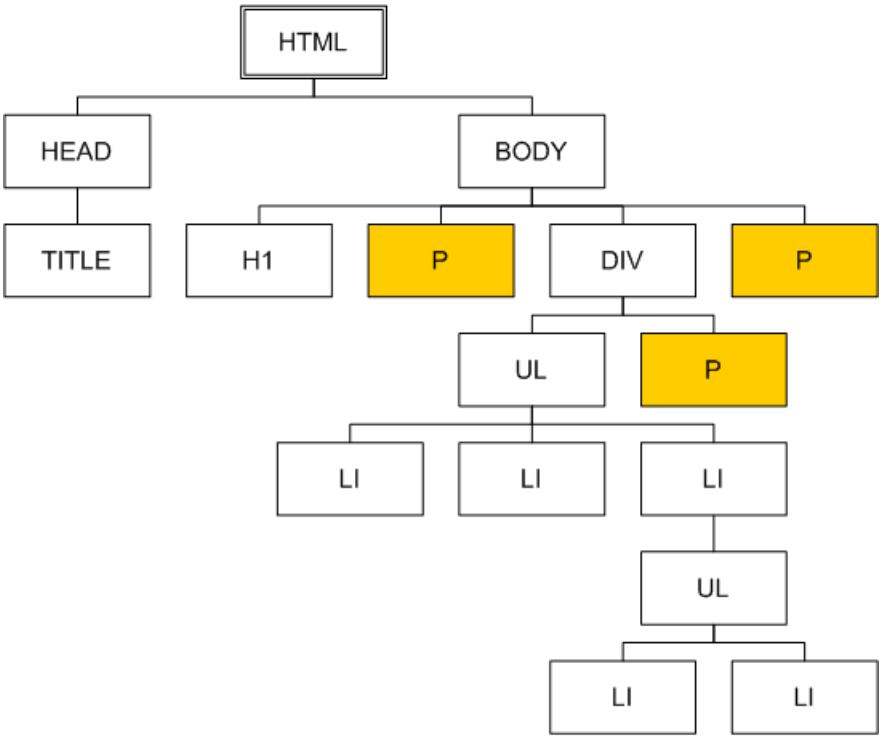


Selektor typu

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

```
p { … }
h1 { … }
```

Obrázek 3. Elementy vybrané pomocí `p`

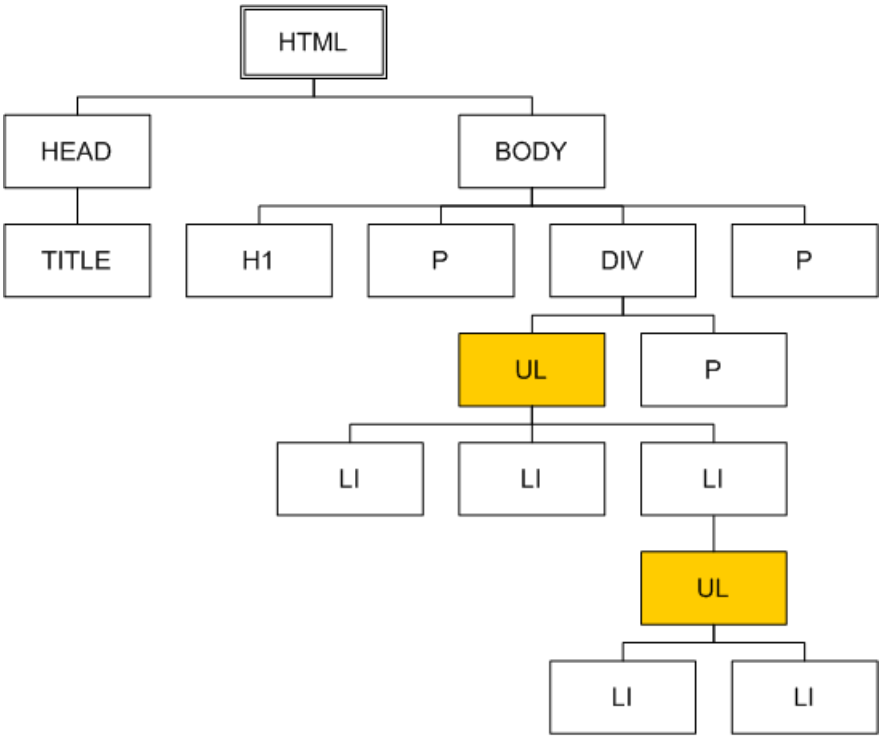


Selektor potomků

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

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

Obrázek 4. Elementy vybrané pomocí `div ul`



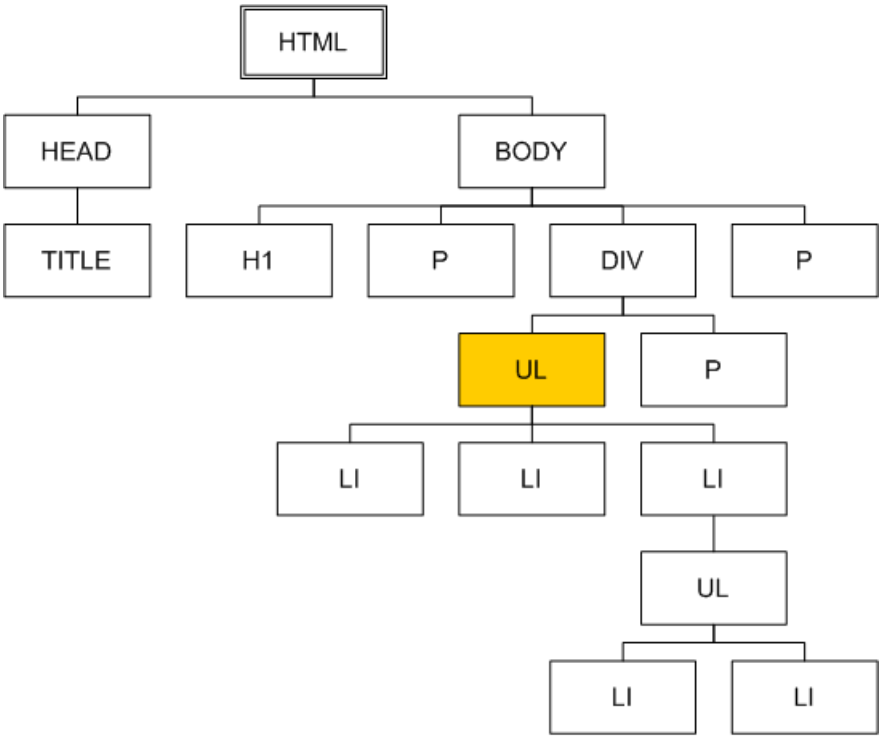
Selektor dětí

- 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`

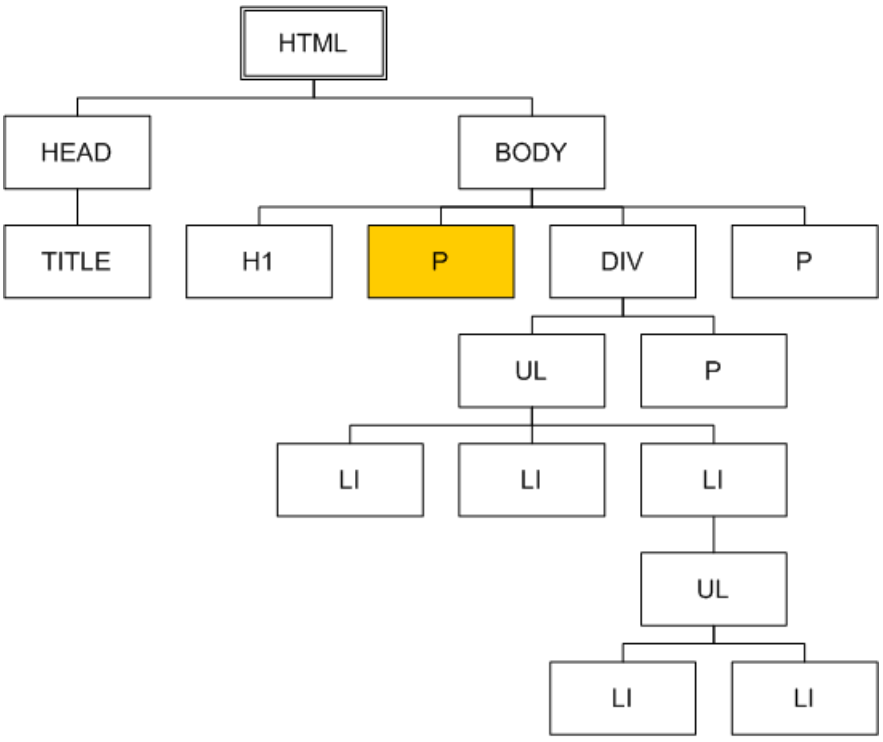


Selektor sousedících sourozenců

- 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

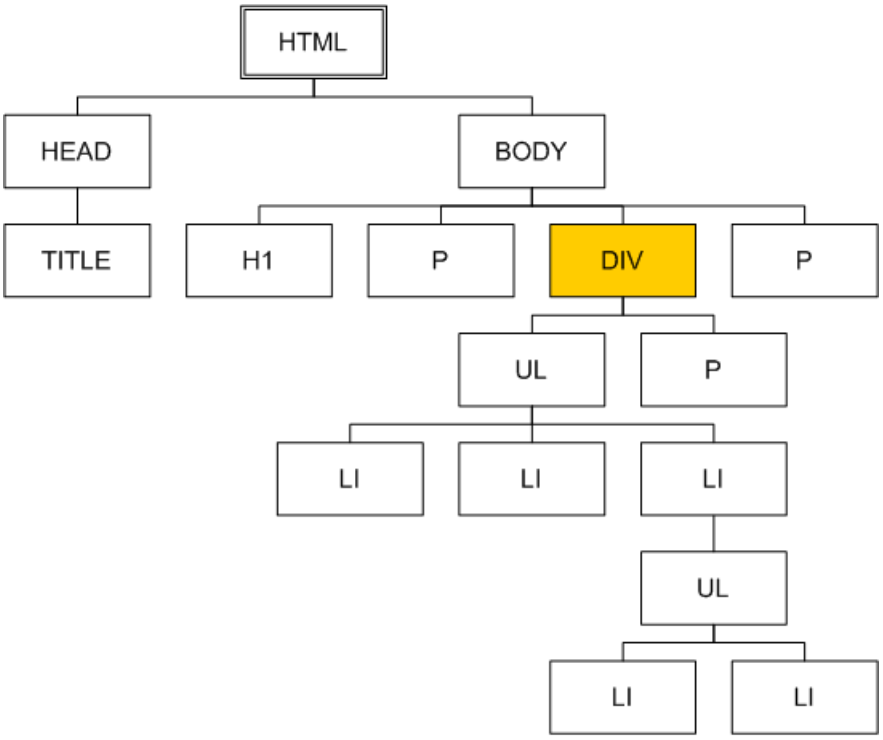
- vybere elementy, které mají nastavenu určitou třídu pomocí atributu `class`

```
<div class="content">…</div>
```

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

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

Obrázek 7. Elementy vybrané pomocí `.content`

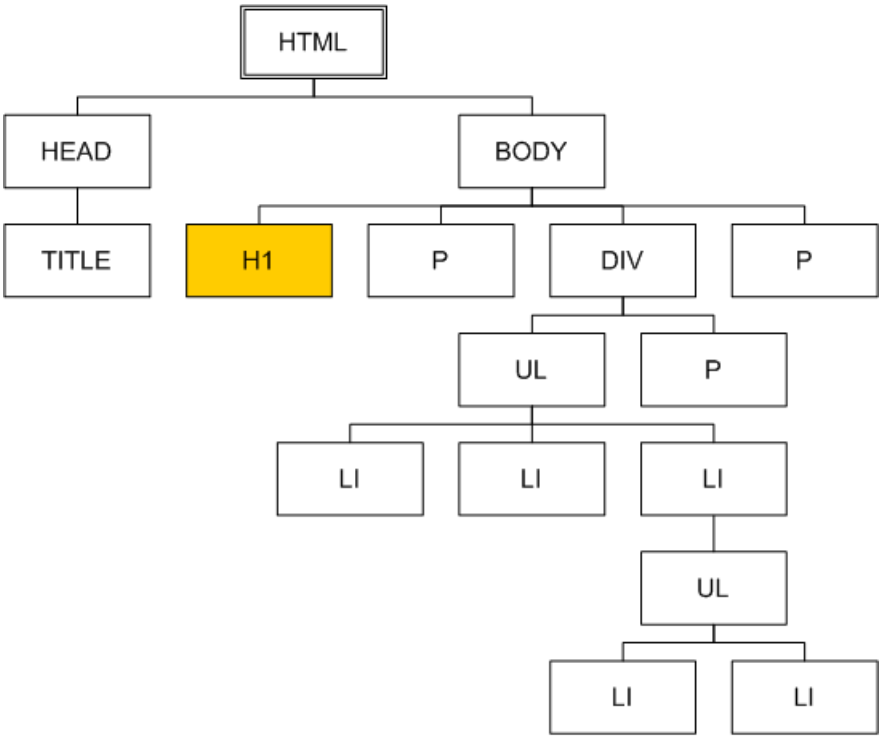


Selektor ID

- 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`

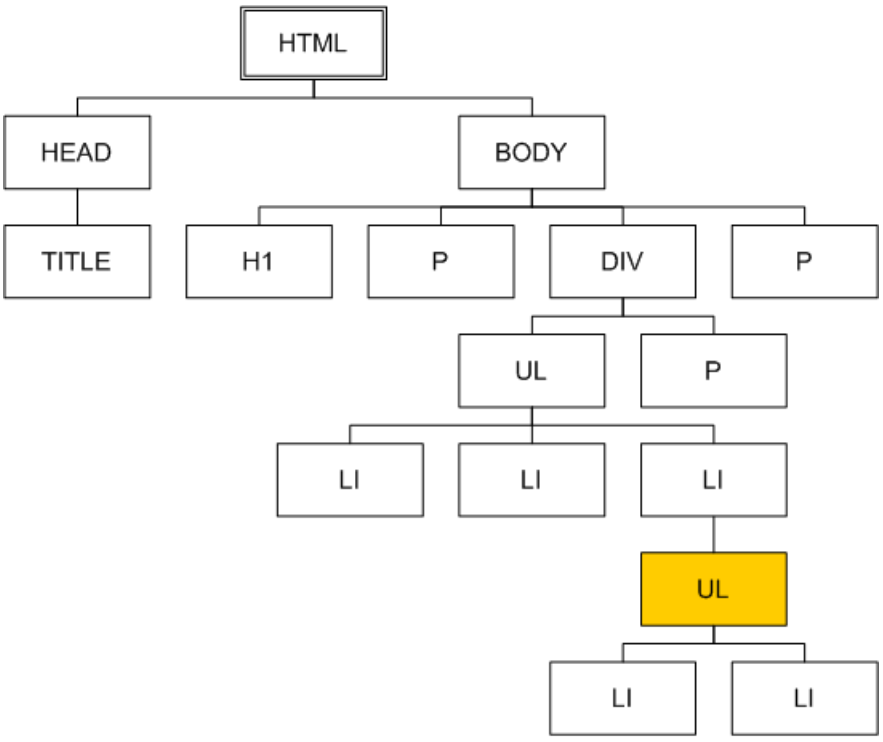
  `[foo="bar"]`
  vybere elementy, které mají v atributu `foo` hodnotu `bar`

  `[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] { … }
```

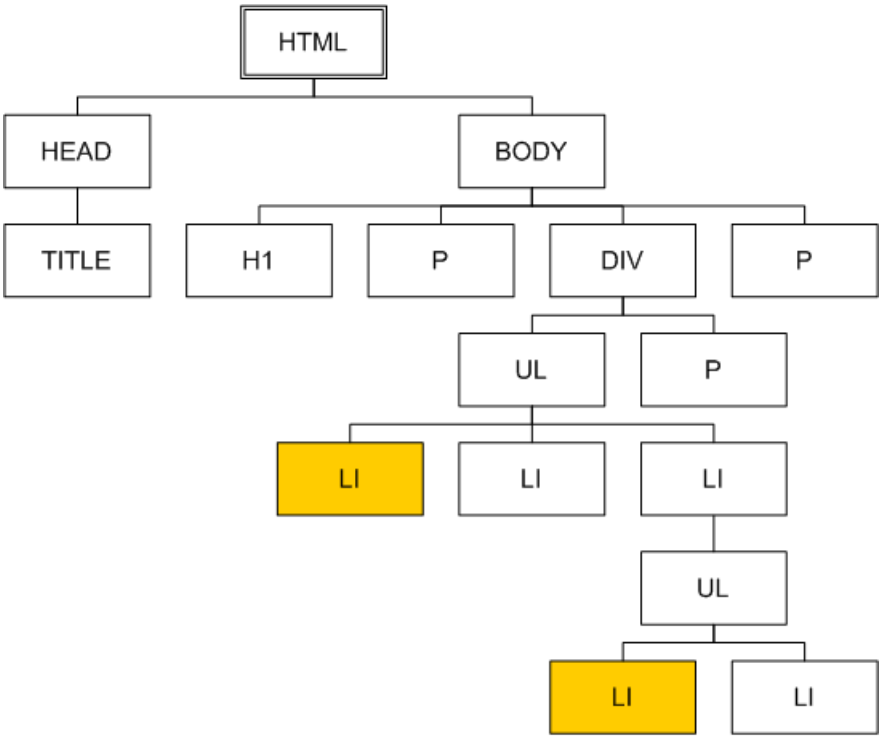
Obrázek 9. Elementy vybrané pomocí `ul[type="circle"]`



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

Vlastnosti

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

CSS nabízejí mechanismus pro stahování fontů:

- prohlížeče podporují různé formáty písem
- licence většiny písem toto použití neumožňuje

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

h1 { font-family: Baskerville }
```

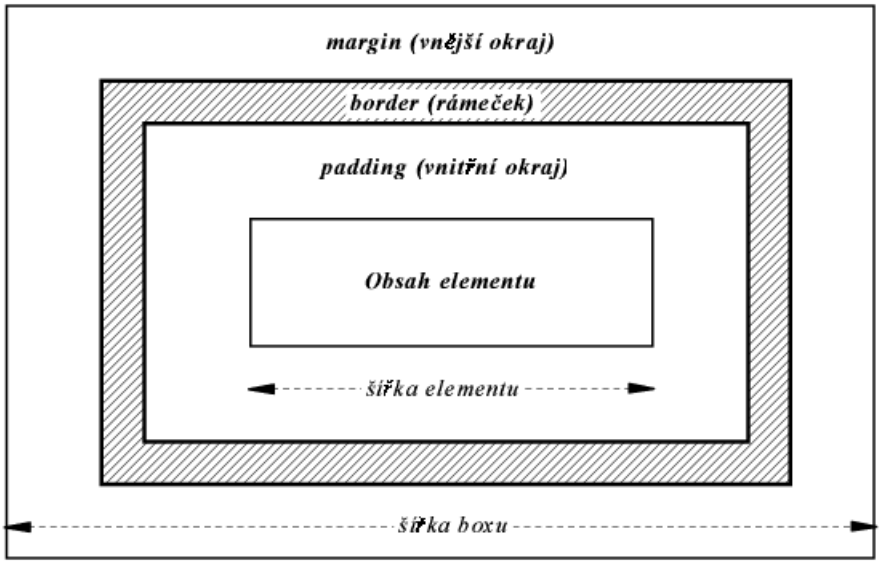
Barvy, pozadí

- 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.

CSS2

Rozšíření oproti CSS1

- zcela zpětně kompatibilní
- nové selektory s lepší podporou XML
- vlastnosti pro přesné umístění elementů na stránce
- podpora několika výstupních médií v jednom stylu
- stránkovaný výstup – lepší podpora tisku
- vlastnosti pro hlasovou syntézu
- lepší práce s fonty, download fontů
- formátovací model pro tabulky
- generování obsahu, automatické číslování

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">
```

Stránkovaný výstup

- nastavení okrajů na stránce; zvlášť pro liché, sudé a první
- řízení stránkového zlomu
- kontrola vdov a sirotků

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
- ukázka – menu pomocí `position: fixed`

Příklad 4. Třída `menu` definuje prostor pro menu

```
.menu { position: fixed;
        left: 10px;
        top: 0px;
        width: 182px;
        border: 2px blue solid;
        padding: 2px;
        height: 100%;
        overflow: auto;

        /* IE nepodporuje position: fixed, ale lze to obejít pomocí JS */
        position: expression("absolute");
        top: ▶
expression(document.body.scrollTop+this.offsetHeight-this.offsetHeight);
      }

.content { margin-left: 200px }
```

Příklad 5. HTML stránka obsahuje dva oddíly, jeden pro menu, druhý pro obsah stránky

```
<div class="menu">
  ... obsah menu ...
</div>
<div class="content">
  ... samotný obsah stránky ...
</div>
```

Další zdroje informací

Další zdroje informací

- Cascading Style Sheets, level 1[1]
- CSS2[2]
- návrh CSS2.1[3]
- on-line přehled vlastností CSS v češtině[4]
- úvod do CSS v češtině[5]
- historie CSS[6]
- CSS tutoriál[7]
- úvod do CSS od W3C[8]
- články o CSS na Interval.cz[9]
- validátor CSS stylů[10]
- další validátor CSS stylů[11]
- impresivní ukázka CSS[12]

[1] http://www.w3.org/TR/REC-CSS1
[2] http://www.w3.org/TR/REC-CSS2
[3] http://www.w3.org/TR/CSS21
[4] http://www.kosek.cz/clanky/dhtml/css-properties.html
[5] http://www.kosek.cz/clanky/html/16.html
[6] http://www.w3.org/Style/LieBos2e/history/
[7] http://www.w3schools.com/css/
[8] http://www.w3.org/MarkUp/Guide/Style
[9] http://interval.cz/?idcategory=14&idsubcategory=168
[10] http://jigsaw.w3.org/css-validator/
[11] http://www.htmlhelp.com/tools/csscheck/
[12] http://www.csszengarden.com/