



**CSS**

kurz Základy  
webových technológií

Eduard Kuric



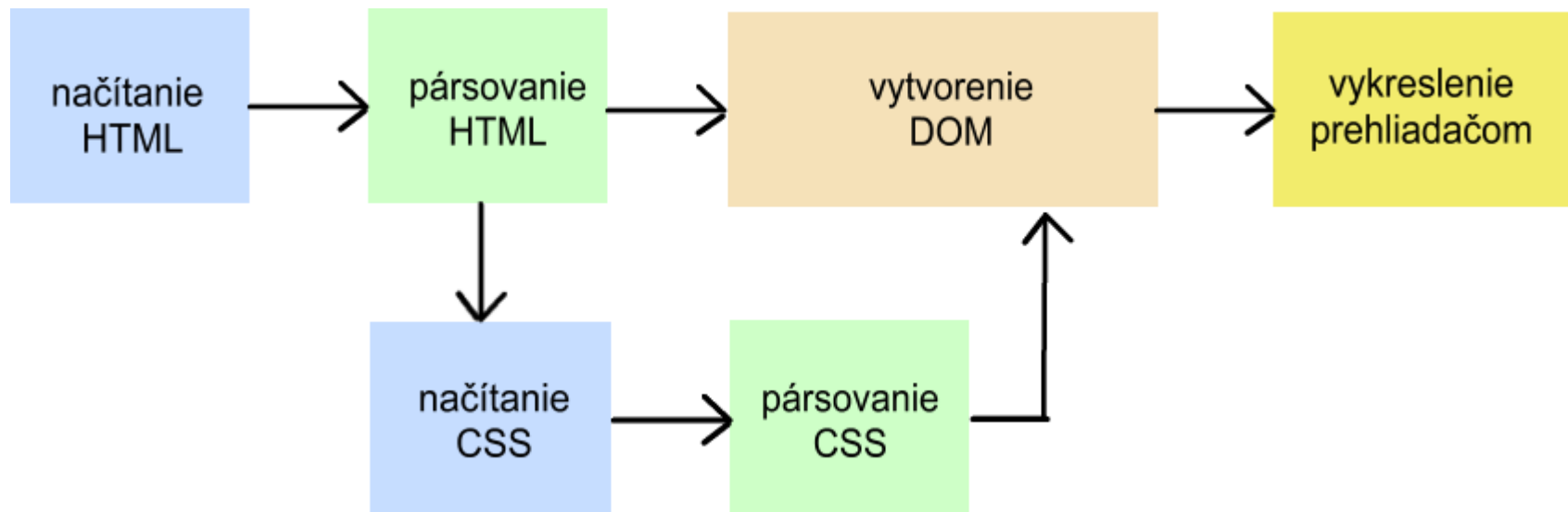
# Čo je CSS?

- **Cascading Style Sheets**
- deklaratívny jazyk, ktorý špecifikuje **výzor stránky/dokumentu (prezentáciu, formátovanie)**
  - sú to štýly, to, čomu zjednodušene hovoríme dizajn
  - písmo, farby, orámovanie, umiestnenie, pozadie,...
- šetrí čas
  - napr. v jednom CSS súbore máme definované rozloženie (layout) viacerých stránok
- CSS preprocesory
  - Sass – premenné, vnorené pravidlá, mixins, ...

# Ako CSS ovplyvňuje HTML?

- prehliadače aplikujú na dokument (elementy) **CSS pravidlá**
  - definujú/určujú, ako sa dokument zobrazí
- poskytuje aparát na zotavenie sa z omylov (syntax)
  - napr. deklarácie, ktorým nerozumie ignoruje
  - nevýhoda - môže byť ťažšie zistiť príčinu chyby

# Vykreslenie dokumentu



[https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/#The\\_rendering\\_engine](https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/#The_rendering_engine)

# 3 spôsoby vloženia CSS v HTML

- 3 spôsoby vloženia (pripojenia) štýlov v HTML
  - externé štýly
  - interné štýly
  - inline štýly

# CSS v HTML - externé štýly

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My experiment</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1>Ahoj svet!</h1>
    <p>Odsek</p>
  </body>
</html>
```

# CSS v HTML - interné štýly

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="utf-8">
```

```
    <title>My experiment</title>
```

```
    <style>
```

```
      p {
```

```
        color: red;
```

```
      }
```

```
    </style>
```

```
  </head>
```

```
  <body>
```

```
    <p>Odsek</p>
```

```
  </body>
```

```
</html>
```

- **môže sa v niektorých prípadoch hodiť**
  - napr. keď pracujete s CMS, kde nemáte priamy prístup k hlavným CSS súborom

# CSS inline štýly

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My experiment</title>
  </head>
  <body>
    <h1>Ahoj svet!</h1>
    <p style="color:red">Odsek</p>
  </body>
</html>
```

**TOTO NEPOUŽÍVAJTE!**  
**iba ak naozaj, naozaj, naozaj musíte**



# Ako vyzerá CSS?

- obyčajný textový súbor s pravidlami
- má pomerne jednoduchú syntax,  
priamočiarú na pochopenie človekom

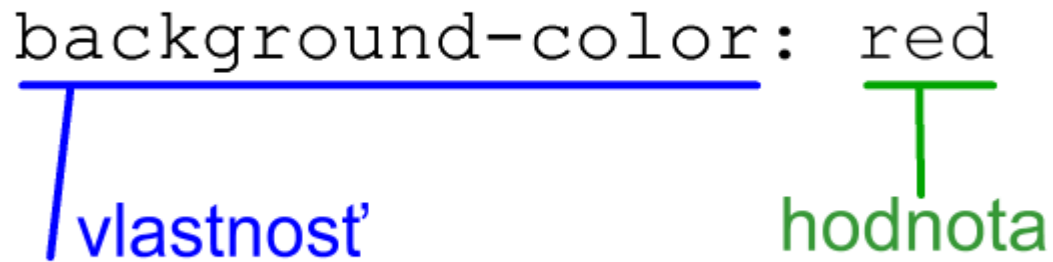
# CSS vlastnost'

`background-color`  
|  
vlastnost'

- **vlastnost'** (angl. property) - **indikuje štylistickú funkciu**
  - **identifikátor** - **človeku zrozumiteľný**,  
napr. `background-color`, `font`, `width`

# CSS hodnota

`background-color: red`

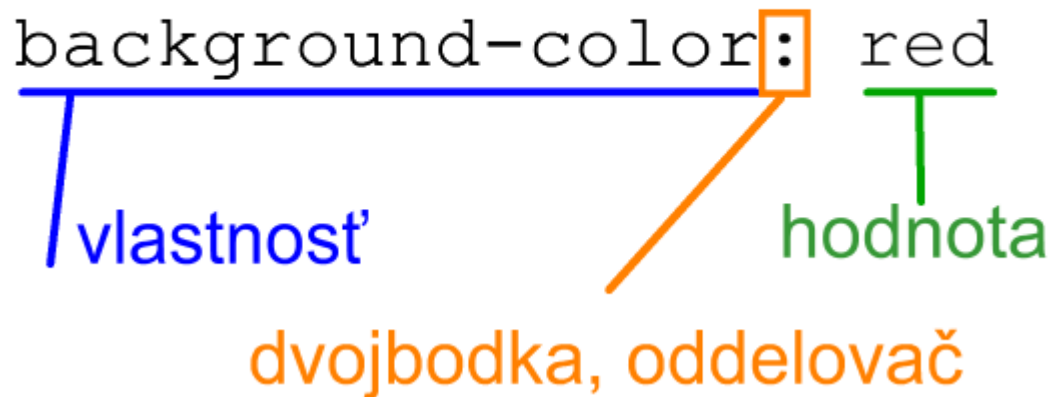


`vlastnost` `hodnota`

- vlastnost má **hodnotu** (angl. value)

# CSS oddeľovač vlastnosť:hodnota

background-color: red



vlastnosť

dvojbodka, oddeľovač

hodnota

- oddeľovačom je **dvojbodka**, teda *vlastnosť: hodnota*

# CSS deklarácia

deklarácia

```
background-color: red
```

- ak je **vlastnosť neznáma**, alebo **hodnota nie je platná**, deklarácia je **ignorovaná**
- používa sa **US angličtina**, teda color, nie colour

# CSS deklaračný blok

- ohraničený { ... }, obsahuje deklarácie
- bodkočiarka, oddeľuje deklarácie
- posledná bodkočiarka nie je nutná, ale dobrá prax

{

background-color: red;  
color: white

}

# CSS selektor

- selektor určuje **element(y)**, na ktoré sa **aplikuje deklaračný blok**

```
a {  
    background-color: red;  
    color: white  
}
```

# CSS pravidlo

- **pravidlo - selektor spolu s deklaracním blokem**

```
a {  
    background-color: red;  
    color: white  
}
```



# CSS pravidlá *at-rule*

- začína @, poskytnutie **metadát, podmienené pravidlá**, napr. :
  - `@import` - vloženie externého css súboru
  - `@charset` - znaková sada CSS súboru
  - `@font-face` - pripojenie externého súboru s písmom
  - `@media` - media queries

```
@font-face {  
    font-family: "Open Sans";  
    src: url("/fonts/OpenSans-Regular-webfont.woff2")  
        format("woff2"),  
        url("/fonts/OpenSans-Regular-webfont.woff")  
        format("woff");  
}
```

# CSS pravidlá - nested statements

- **špecifická podmnožina *at-rule* pravidiel**

- napr. @media, obsahuje vnorené (podmienené) pravidlá
- pozn.: CSS neumožňuje vnárať pravidlá, toto sú výnimky

```
@media (min-width: 801px) {  
    body {  
        margin: 0 auto;  
        width: 800px;  
    }  
}
```

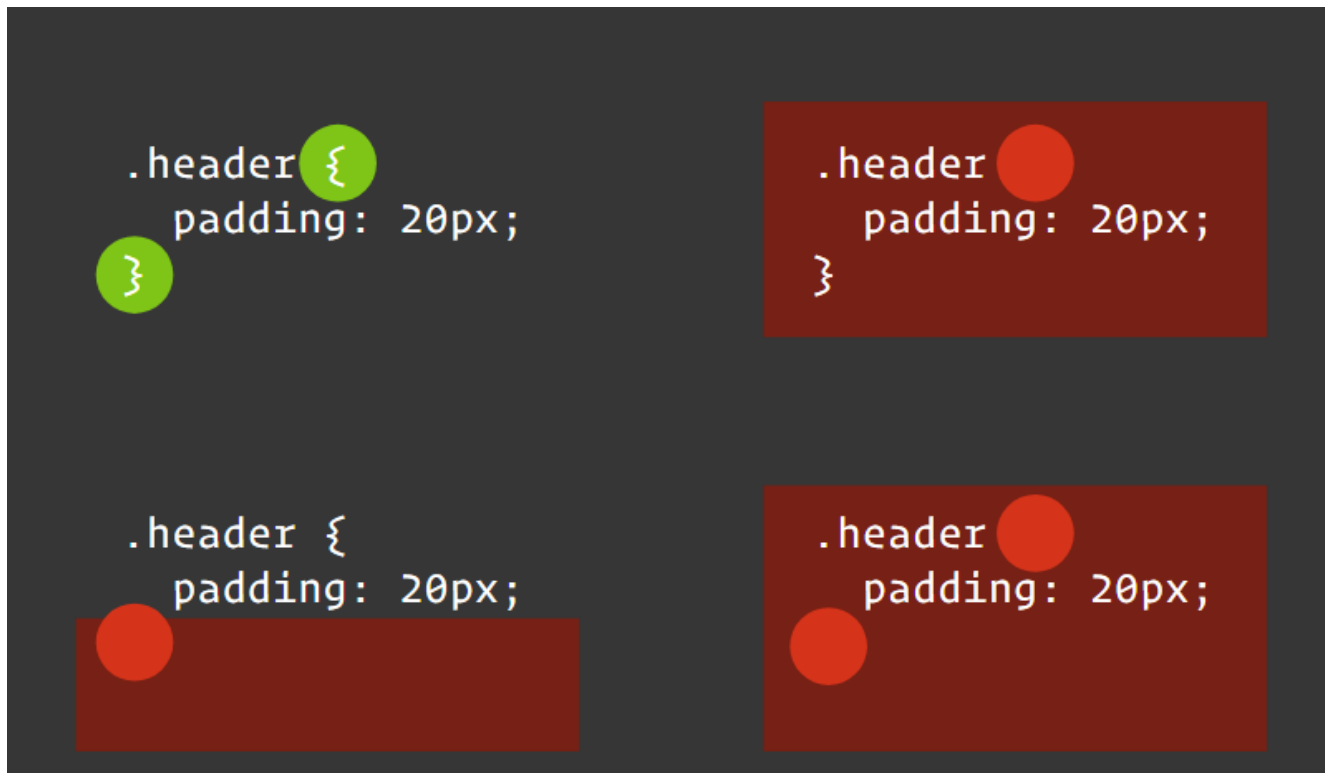
# CSS syntax

- formátujte CSS, aby bolo čitateľné
- prehliadač v princípe ignoruje biele znaky navyše
- dobrá prax:
  - každá deklarácia na novom riadku
  - „štruktúrovať“ dokument pomocou komentárov `/* */`  
štýly pre hlavičku, menu ...

```
/* hlavička dokumentu */  
/* menu */  
/* hlavný obsah */
```

# CSS syntax

- zátvorky sú dôležité



# CSS Selektory

- jednoduché selektory
- selektory podľa atribútov
- pseudo-triedy
- pseudo-elementy
- kombinácie
- viacnásobné selektory (skupiny)

# Jednoduché selektory - element

- keď selektor je **názov elementu**

```
<p>Ktorá farba sa ti páči?</p>  
<div>Páči sa mi modrá.</div>  
<p>Preferujem červenú!</p>
```

```
/* všetky p elementy majú červené písmo */
```

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

```
/* všetky div elementy majú modré písmo */
```

```
div {  
    color: blue;  
}
```

# Jednoduché selektory - class

- keď selektor je **názov triedy**  
(do rovnakej triedy môže patriť viacero elementov)

```
<p>Ktorá farba sa ti páči?</p>
```

```
<div class="blue">Páči sa mi modrá.</div>
```

```
<p>Preferujem červenú!</p>
```

```
/* všetky p elementy majú červené písmo */
```

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

```
/* všetky div elementy s triedou blue majú modré písmo */
```

```
div.blue {  
    color: blue;  
}
```

# Jednoduché selektory - class /2

- keď selektor je **názov triedy**  
do rovnakej triedy môže patriť viacero elementov

```
<p>Ktorá farba sa ti páči?</p>
```

```
<div class="blue">Páči sa mi modrá.</div>
```

```
<p class="blue">Preferujem červenú!</p>
```

```
<p>Naozaj?!?</p>
```

```
/* všetky p elementy majú červené písmo */
```

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

```
/* všetky elementy s triedou blue majú modré písmo */
```

```
.blue {  
  color: blue;  
}
```



# Jednoduché selektory – id

- **keď selektor je identifikátor**  
(v dokumente musí byť identifikátor unikátny)
- ak má viacero elementov rovnaké `id`, prehliadač spravidla uvažuje prvý element, na ktorý „natrafí“, ostatné ignoruje – je to ale nevalidný dokument

```
<p id="question">Ktorá farba sa ti páči?</p>
```

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

```
p#question {  
    color: red;  
}
```

# CSS názvy identifikátorov

- trieda (atribút `class`), identifikátor (atribút `id`)
- môže obsahovať znaky [**a–zA–Z0–9**]
- ISO 10646 znaky **U+00A0** (no break space) a vyššie
- pomlčku (–) a podčiarkovník (\_)
- nemôže začínať číslom, dvomi pomlčkami, alebo pomlčkou nasledovanou číslom

# CSS selektory – univerzálny selektor

- **univerzálny selektor \***
- zahrňa všetky elementy
- často používaný v kombinácii s inými selektormi

```
<div>  
  <p>Lorem ipsum dolor it samet <strong>border</strong> or  
  <em>border em</em>,tub sith si gnitteg <strong>tuo fo  
  andh</strong>!</p>  
</div>
```

```
* {  
  padding: 5px;  
  border: 1px solid black;  
  background: rgba(255,0,0,0.25)  
}
```

# CSS selektory - atribúty

- výber elementov na základe zhody atribútov a ich hodnôt
  - selektor je ohraničený [ ]
- `p[attr]` všetky elementy s atribútom `attr`, nezáleží na hodnote
- `[attr=val]` všetky elementy s atribútom `attr`, ak jeho hodnota je `val`
- `[attr~val]` všetky elementy s atribútom `attr`, ak sa **v hodnote atribútu vyskytuje slovo** `val` (slovo - postupnosť znakov oddelená medzerou), napr. názov triedy

# CSS selektory – atribúty RegEx

- `[attr|=val]` všetky elementy s atribútom `attr`, kde hodnota je presne `val`, alebo začína `val-`
- `[attr^=val]` všetky elementy s atribútom `attr`, ak hodnota začína `val`
- `[attr$=val]` všetky elementy s atribútom `attr`, ak hodnota končí `val`
- `[attr*=val]` všetky elementy s atribútom `attr`, ak hodnota obsahuje reťazec `val`

# CSS selektory – pseudo triedy

- kľúčové slovo, pred ktorým je : , pridáva sa na koniec selektorov
- štýl je aplikovaný na elementy, ktoré sú v určitom stave
- napr.:
  - `a:hover`
  - `a:active`
  - `a:focus`
  - `a:visited`

# CSS selektory – pseudo triedy

:active	:any	:checked
:default	:dir()	:disabled
:empty	:enabled	:first
:first-child	:first-of-type	:fullscreen
:focus	:focus-within	:hover
:indeterminate	:in-range	:invalid
:lang()	:last-child	:last-of-type
:left	:link	:not()
:nth-child(2)	:nth-last-child()	:nth-last-of-type()
:nth-of-type()	:only-child	:only-of-type
:optional	:out-of-range	:read-only
:read-write	:required	:right
:root	:scope	:target
:valid	:visited	

[https://www.w3schools.com/css/css\\_pseudo\\_classes.asp](https://www.w3schools.com/css/css_pseudo_classes.asp)

# CSS selektory – pseudo elementy

- kľúčové slovo, pred ktorým je :: , pridáva sa na koniec selektorov
- štýlovanie určitej časti elementu
  - ::after
  - ::before
  - ::first-letter
  - ::first-line
  - ::selection
  - ::backdrop

[https://www.w3schools.com/css/css\\_pseudo\\_elements.asp](https://www.w3schools.com/css/css_pseudo_elements.asp)



# CSS selektory – skupiny

- deklaračný blok môže byť aplikovaný na viacero selektorov súčasne
  - selektory sa oddeľujú čiarkou

```
p, li {  
    font-size: 1.6em;  
}
```

# CSS selektory – kombinácie

- výber elementov na základe ich vzájomného vzťahu
- nech A a B reprezentujú ľubovoľný selector (doteraz predstavený)
- $A \ B$  – každý element zodpovedajúci selektoru B, ktorý **je potomkom** elementu, ktorému zodpovedá selektor A (aj potomok potomka, ...)
- $A > B$  – každý element zodpovedajúci selektoru B, ktorý **je priamym potomkom** elementu, ktorému zodpovedá selektor A

# CSS selektory – kombinácie

- $A + B$  – element zodpovedajúci selektoru B, ktorý je **bezprostredným súrodencom** elementu, ktorému zodpovedá selektor A, obaja sú potomkami rovnakého rodiča
- $A \sim B$  – element zodpovedajúci selektoru B, ktorý je súrodencom elementu, ktorému zodpovedá selektor A, pričom element B **nemusí byť bezprostredným súrodencom**, obaja sú potomkami rovnakého rodiča

# CSS syntax

- medzera môže byť dôležitá

$\neq$  `body ul li`  
`bodyulli`

$=$  `.header .title`  
`.header .title`

$\neq$  `.header .title`  
`.header.title`

# CSS syntax

- medzera môže byť dôležitá

```
✓ background-image: url(tiger.jpg);  
background - image: url(tiger.jpg);  
background-image: url (tiger.jpg);
```

# CSS syntax

- bezvýznamná medzera

✓ `.header {  
padding: 20px;  
}`

✓ `.header {  
padding:20px;  
}`

✓ `.header{  
padding: 20px;}`

✓ `.header {  
padding: 20px;  
}`

✓ `.header{padding:20px;}`

✓ `.header{  
padding: 20px; }`

# CSS syntax

- dôležité bodkočiarky

```
.header {  
  padding:      20px;  
  max-width:    600px;  
  background:   black;  
  color:        white;  
}
```

```
.header {  
  padding:      20px;  
  max-width:    600px;  
  background:   black;  
  color:        white  
}
```

# CSS syntax

- pozor na znaky navyše



```
.header {  
  padding: 20px;  
}
```

```
.header {  
  /padding: 20px;  
}
```

```
.header { {  
  padding: 20px;  
}
```

```
.header {  
  padding: ~20px;  
}
```



# CSS syntax

- (niekedy) zbytočné nové riadky

```
✓ .header {  
    box-shadow:  
        0    0    5px    rgba(0, 0, 0, 0.5),  
        20px 2px 5px -5px rgba(0, 0, 0, 0.5);  
}  
  
✓ .bar  
{  
    background-image:  
        url(texture.png),  
        url(tiger.jpg);    }  
  
✓ .button::after  
{  
    content:  
        ">";  
}
```

# CSS vlastnosti a ich hodnoty

- číselné hodnoty
- farby
- súradnice
- funkcie

# Číselné hodnoty

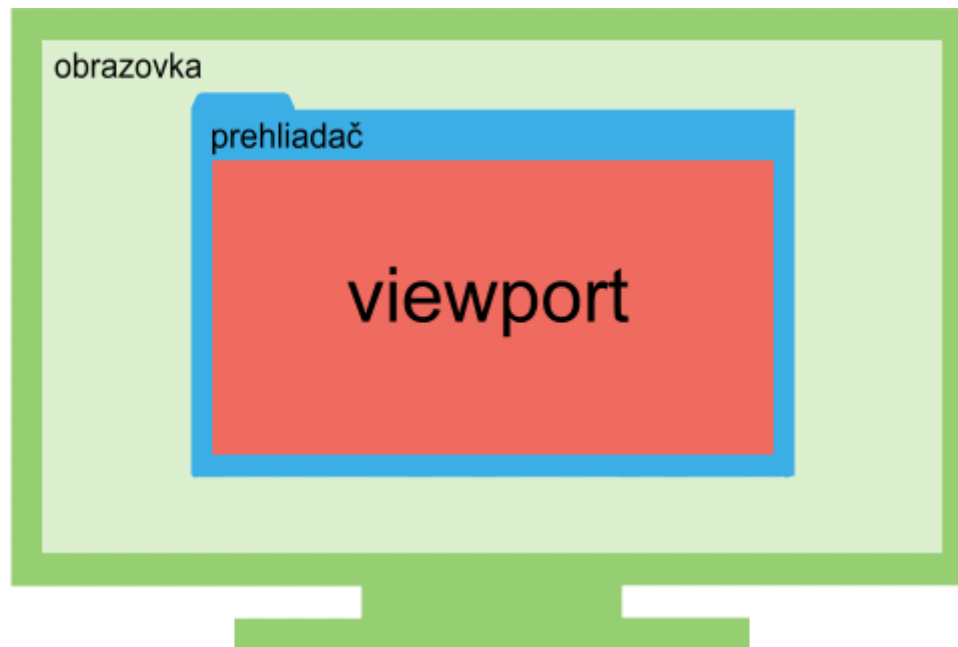
- délka (length, nápr. výška elementu/bloku)
- velikost (size, napr. písma)
- pixel – picture element, px
  - absolutní jednotka
  - má „vždy“ rovnakú veľkosť
- ďalšie absolútne jednotky
  - mm, cm, in
  - pt – point - 1/72 z palca
  - pc – pica (pajka) – 12pt

# Číselné hodnoty /2

- relatívne jednotky:
  - `em` – `1em` – prednastavený štýl prehliadača - 16px
  - `2em` - veľkosť písma bude dvojnásobná **voči aktuálnej veľkosti písma v elemente**
  - pozor na dedenie
- ďalšie relatívne jednotky:
  - `ex` – `1ex` - výška písmena „x“ aktuálneho písma
  - `ch` – `1ch` - šírka čísla „0“ aktuálneho písma
  - `rem` – `1rem` – vždy relatívne **voči veľkosti písma v koreňovom elemente**

# Číselné hodnoty /3

- `vw`, `vh` – 1/100 šířky, resp. výšky viewportu
- `viewport` - viditeľná plocha/oblasť webovej stránky



pr1-ciselene-hodnoty.html

# Číselné hodnoty /4

- nie každá číselná hodnota musí mať uvedenú jednotku
- napr. keď chceme vynulovať `margin: 0;`
  - netreba uvádzať `px`, ani nič iné – je to zbytočné
- podobne, `line-height: 1.5;` (výška riadku)
  - výška riadku bude 1,5 násobok aktuálnej veľkosti písma
  - môžu sa použiť aj jednotky

# Hodnoty - percentá

- najčastejšie šírka/výška elementov
- definovanie v percentách umožňuje, napr.
  - vytvárať bloky/boxy, ktorých šírka/výška sa prispôsobí šírke/výške ich rodičovského elementu
- blok s fixnou šírkou/výškou (napr. v px)
  - angl. fixed-width/height block
- blok s premenlivou veľkosťou (v %)
  - angl. liquid block

# Hodnoty - farby

- možností je niekoľko
  - rovnaká farba môže byť určená rôznymi spôsobmi
- kľúčové slová
  - `color: red`
- hexadecimálna hodnota
  - `color: #ff0000;` (skrátene `#f00`)
- modely - RGB(A), HSL(A)
  - `color: rgb(255, 0, 0);`
  - `color: rgba(255, 0, 0, 0, 5);`  
(polovičná nepriehľadnosť, angl. opacity, alfa-kanál pixelu)
- [prevod farieb z/do rôznych farebných modelov](#)



# Hodnoty - funkcie

- `rgb()` – počíta, aká farba sa má aplikovať
- `transform: rotate(45deg);`
  - otočí element o 45 stupňov
- `transform: translate(50px, 60px);`
  - posunie element o 50px (os x) a 60px (os y)
- `width: calc(90%-15px);`
  - vypočíta šírku ako premenlivých 90% - fixných 15px

# CSS ako sa aplikujú pravidlá?

- CSS - kaskádové štýly – KASKÁDA

**Na poradí pravidiel záleží**

# Poradie aplikovania štýlov

- Deklarácie sa aplikujú v tomto poradí, pričom neskoršie majú prednosť:
  1. Prednastavené štýly prehliadača (keď iné štýly nie sú nastavené)
  2. Externé štýly (`<link>`) // autor
  3. Interné štýly (`<style>`) // používateľ
  4. Inline štýly // používateľ
  5. Autorom deklarovaný `!important`
  6. Používateľom deklarovaný `!important`

pozn.: má logiku, že používateľ môže prekonať cez `!important` autorovu deklaráciu – už spomínané CMS

# Deklarácia !important

- !important platí nad všetko
  - môže ho "prebiť" iba iný !important, ktorý bude deklarovaný neskôr
    - s rovnakou alebo s vyššou mierou konkrétnosti (angl. specificity)
- **nepoužívať!**
  - ak to nie je naozaj nutné (napr. CMS)
  - je to „násilný“ zásah do normálneho fungovania kaskády

# Konfliktné deklarácie

- inline štýly majú prednosť pred internými (`<style>`) a externými štýlmi, ale
  - ak je deklarácia **!important** v internom štýle a v inline štýle nie je deklarácia **!important**, potom interná deklarácia má prednosť pred inline deklaráciou
  - ak je deklarácia **!important** v externom štýle a v inline štýle nie je deklarácia **!important**, potom externá deklarácia má prednosť pred inline deklaráciou
  - ak je deklarácia **!important** v internom/externom štýle a aj v inline štýle, potom deklarácia v inline štýle má prednosť

# Miera konkrétnosti selektora

- **angl. specificity**
- selektor na triedu má nižšiu mieru konkrétnosti ako selektor na identifikátor
- ako určiť mieru konkrétnosti selektora
  - napr. kombinovaného?

```
li > a[href*="en-US"] > .inline-  
warning
```

<https://www.smashingmagazine.com/2007/07/css-specificity-things-you-should-know/>

# Miera konkrétnosti selektora /2

- **počíta sa ako spojenie štyroch zložiek:**
- tisícky - 1000 – ak je deklarácia v atribúte `style`
- stovky – za každý selektor na identifikátor: +100
- desiatky – za každý selektor na triedu, selektor na atribút, alebo selektor na pseudo triedu: +10
- jednotky – za selektor na názov elementu, alebo pseudo element: +1

pozn.: univerzálny selektor `*`, kombinácie `(+, >, ~, '')`, a pseudo trieda `:not` nemajú vplyv na mieru konkrétnosti

# Miera konkrétnosti - príklad

```
li > a[href*="en-US"] > .inline-warning
```

Tisícky: -

Stovky: -

Desiatky: 20

10: [href\*="en-US"]

10: .inline-warning

Jednotky: 2

1: a

1: li

= 2,2



# Miera konkrétnosti – príklad 2

**#important div > div > a:hover**

deklarácia je vo vnútri `style`

# Miera konkrétnosti – príklad 2

**#important div > div > a:hover**  
deklarácia je vo vnútri `style`

Tisícky: **1000** (deklarácia v `style`)

Stovky: **100**

100: `#important`

Desiatky: **10**

10: `:hover`

Jednotky: **3**

1: `div`

1: `div`

1: `a`

= 1113

# Miera konkrétnosti – príklad 3

```
p.red {  
    color: red !important;  
}
```

```
p.blue {  
    color: blue !important;  
}
```

```
<p class="red blue">Odsek1...</p>
```

```
<p class="red blue">Odsek2...</p>
```

**Akú farbu písma bude mať odsek1/2?**

# Miera konkrétnosti – príklad 3

```
p.red {  
    color: red !important;
```

**Ak je rovnaká dôležitosť (importance) a miera konkrétnosti (specificity), potom vyhráva neskoršia deklarácia.**

```
<p class="blue red">Odsek...</p>
```

**Akú farbu písma bude mať odsek? - modrú**

# Všetko je to na úrovni vlastností

**Ako bude naformátované slovo „dôležitý“?**

```
/* specificity: 0002 */  
p strong {  
    background-color: yellow;  
    color: green;  
}
```

```
/* specificity: 0001 */  
strong {  
    text-decoration: underline;  
    color: red;  
}
```

<p>Som <strong>**dôležitý**</strong> odsek.</p>

# Všetko je to na úrovni vlastností

**Ako bude naformátované slovo „dôležitý“?**

The word "important" is displayed in a green serif font, underlined with a thick green line. It is set against a solid yellow rectangular background.

```
/* specificity: 0002 */  
p strong {  
    background-color: yellow;  
    color: green;  
}
```

```
/* specificity: 0001 */  
strong {  
    text-decoration: underline;  
    color: red;  
}
```

`<p>Som <strong>dôležitý</strong> odsek.</p>`

# Dedenie

- posledná „kocka“ skladačky, aby sme pochopili, aký štýl je aplikovaný na element
- niektoré hodnoty vlastností sú dedené potomkami, niektoré nie sú
- prečo nie všetko alebo nič? guláš?
  - napr. `font-family` a `color` (sa dedia)  
predstavte si že by ste mali nastavovať každému elementu typ písma, alebo farbu
  - naopak, `margin`, `padding`, `border`, `background-image` (sa nededia)  
predstavte si že nastavíte elementu `body` `background-image` a museli by ste potom každému potomkovi zmeniť/vypnúť `background-image`

# Dedenie – treba sa postupne naučiť

- ktoré vlastnosti sa dedia (v prednastavenom režime), a ktoré nie sa treba postupne – **používaním** naučiť
- alebo, zamyslieť sa, ma globálny charakter vlastnosti zmysel?

<https://www.w3.org/TR/CSS21/propidx.html>



# Dedenie – pod kontrolou

- CSS poskytuje tri špeciálne hodnoty, ktoré je možné použiť ako hodnoty vlastností, na ovplyvňovanie dedenia
- `inherit`: hodnota vlastnosti sa nastaví na rovnakú, akú má predok
- `initial`: hodnota sa nastaví na prednastavenú hodnotu štýlu prehliadača
  - ak nie je def. a vlastnosť sa implicitne dedí, potom `inherit`
- `unset`: ak sa vlastnosť implicitne dedí, potom `inherit`, inak `initial`

# Dedenie – pod kontrolou /2

- ```
<div class="wrapper">  
  <p class="one">Had<p>  
  <p class="two">Mačka<p>  
  <p class="three">Myš</p>  
  <p>Potkan</p>  
</div>
```
- ```
.wrapper { color: orange; }  
.wrapper p { color: purple; }  
p.one { color: inherit; }  
p.two { color: initial; }  
p.three { color: unset; }
```

# Dedenie – pod kontrolou /2

- ```
<div class="wrapper">  
  <p class="one">Had<p>  
  <p class="two">Mačka<p>  
  <p class="three">Myš</p>  
  <p>Potkan</p>  
</div>
```
- ```
.wrapper { color: orange; }  
  .wrapper p { color: purple; }  
p.one { color: inherit; }  
  p.two { color: initial; }  
  p.three { color: unset; }
```

# Dedenie – pod kontrolou /2

- ```
<div class="wrapper">  
  <p class="one">Had<p>  
  <p class="two">Mačka<p>  
  <p class="three">Myš</p>  
  <p>Potkan</p>  
</div>
```
- ```
.wrapper { color: orange; }  
.wrapper p { color: purple; }  
p.one { color: inherit; }  
p.two { color: initial; } // def. black  
p.three { color: unset; }
```

# Dedenie – pod kontrolou /2

- ```
<div class="wrapper">  
  <p class="one">Had<p>  
  <p class="two">Mačka<p>  
  <p class="three">Myš</p>  
  <p>Potkan</p>  
</div>
```
- ```
.wrapper { color: orange; }  
  .wrapper p { color: purple; }  
  p.one { color: inherit; }  
  p.two { color: initial; }  
p.three { color: unset; } // inh.
```

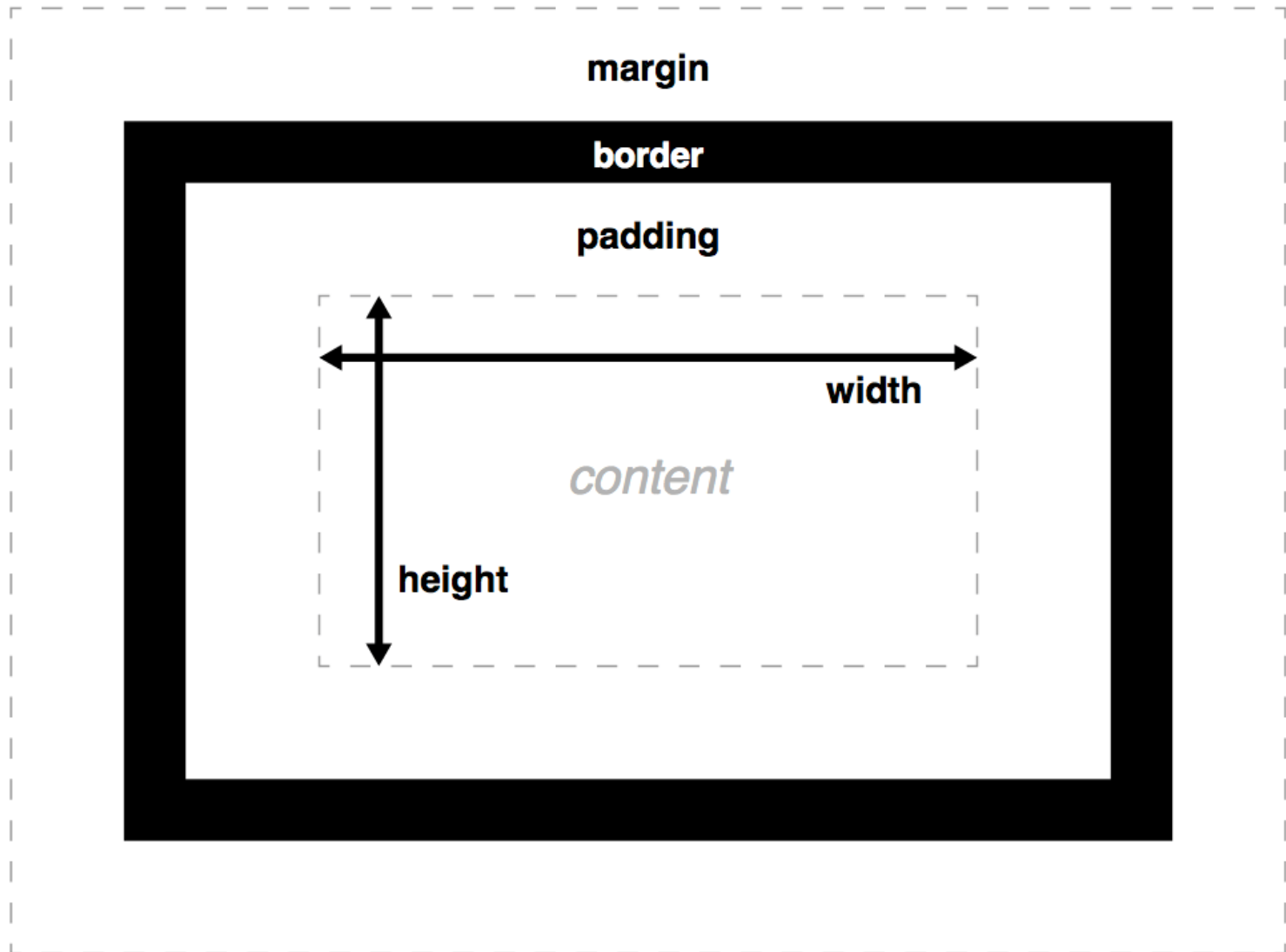
# Dedenie – pod kontrolou /2

- ```
<div class="wrapper">  
  <p class="one">Had<p>  
  <p class="two">Mačka<p>  
  <p class="three">Myš</p>  
  <p>Potkan</p>  
</div>
```
- ```
.wrapper { color: orange; }  
.wrapper p { color: purple; }  
p.one { color: inherit; }  
p.two { color: initial; }  
p.three { color: unset; }
```

# CSS box model

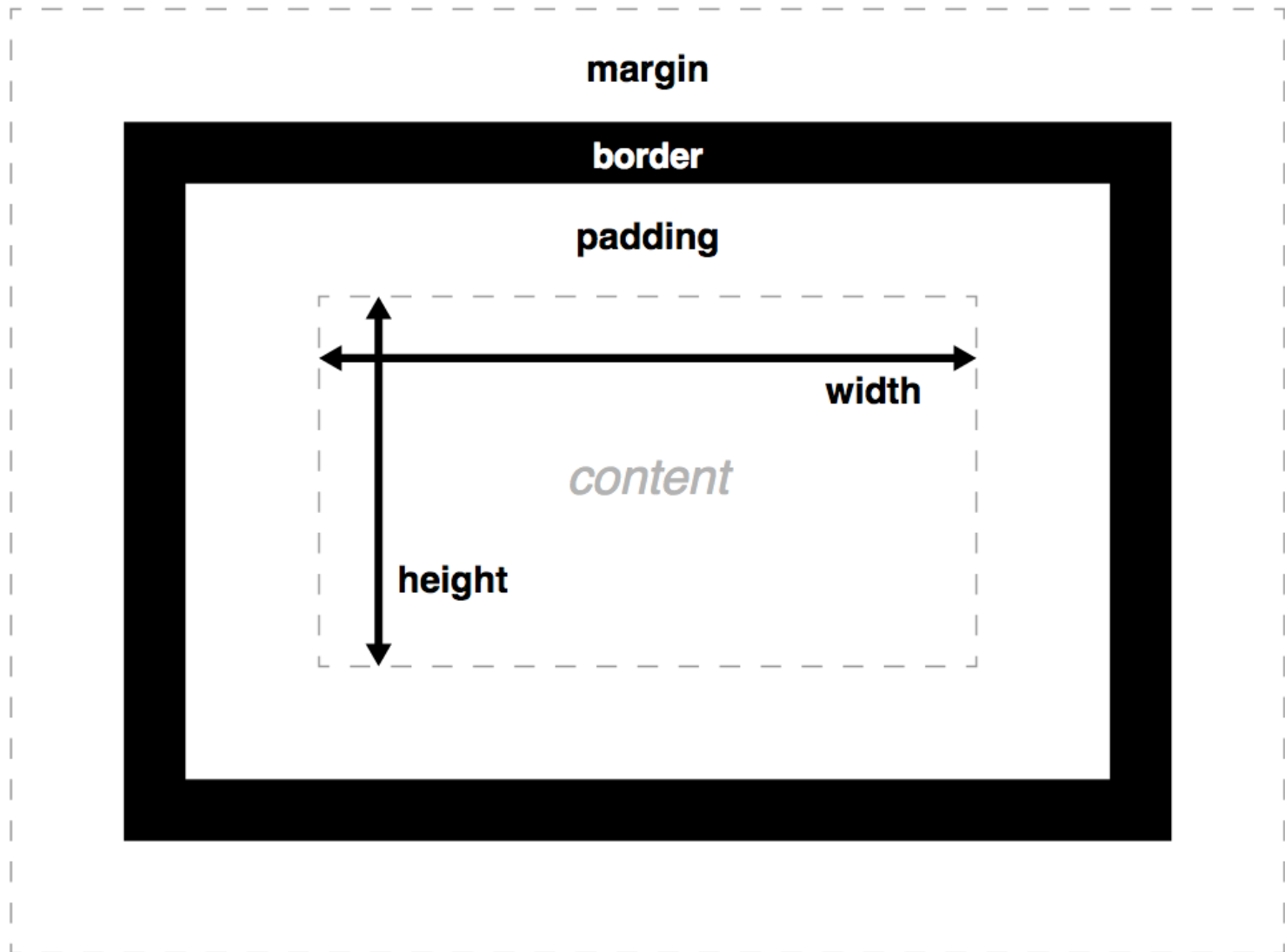
- aby sme mohli začať vytvárať „layout“ stránky, musíme dobre rozumieť tzv. ***box modelu***
- základ na vytváranie rozloženia/usporiadanie stránky (page layout)
- každý element je reprezentovaný obdĺžnikovým blokom, spolu s:
  - margin – vonkajší okraj
  - padding – vnútorný okraj
  - border – rámik
  - + obsah

# CSS box model

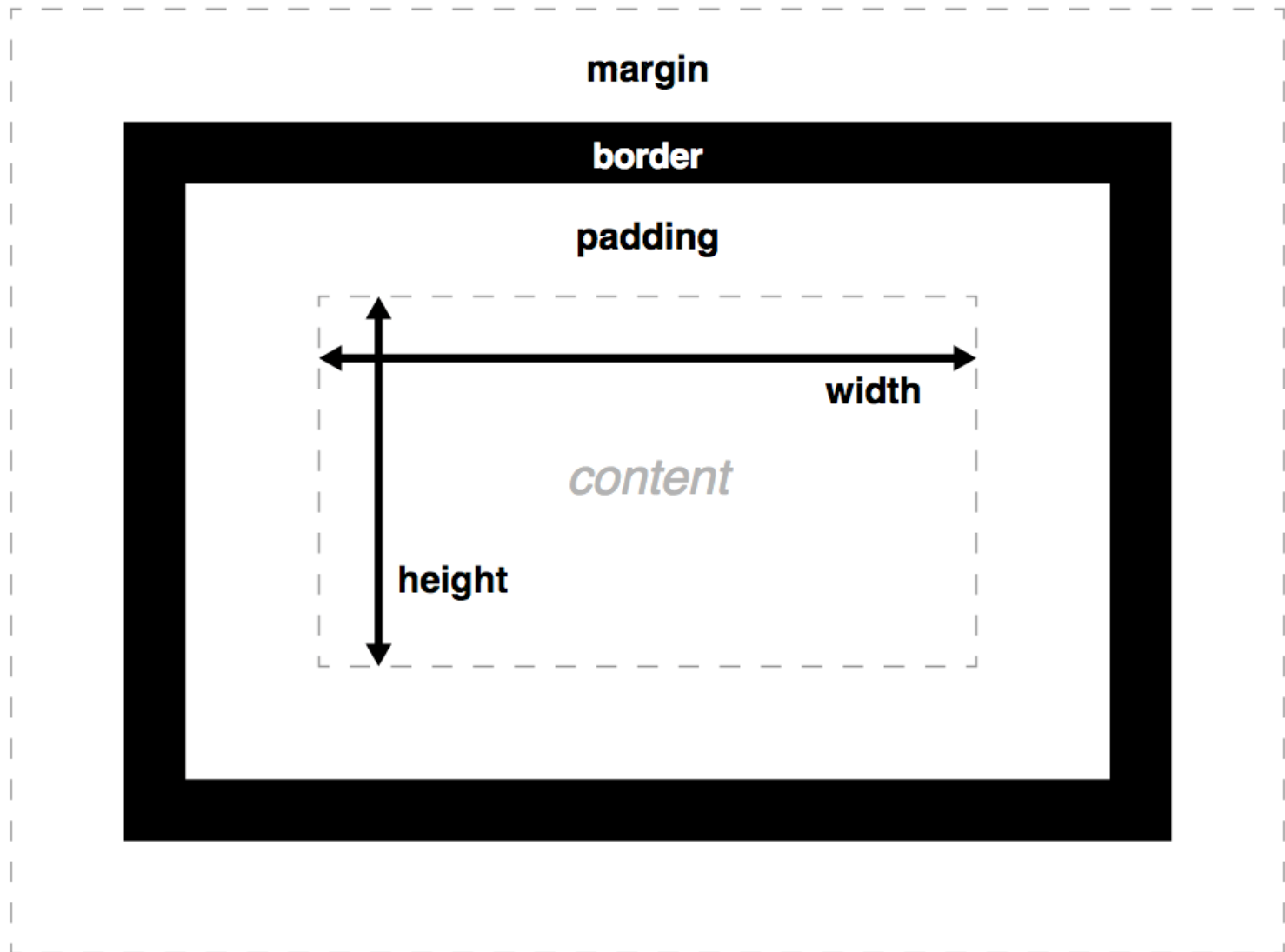




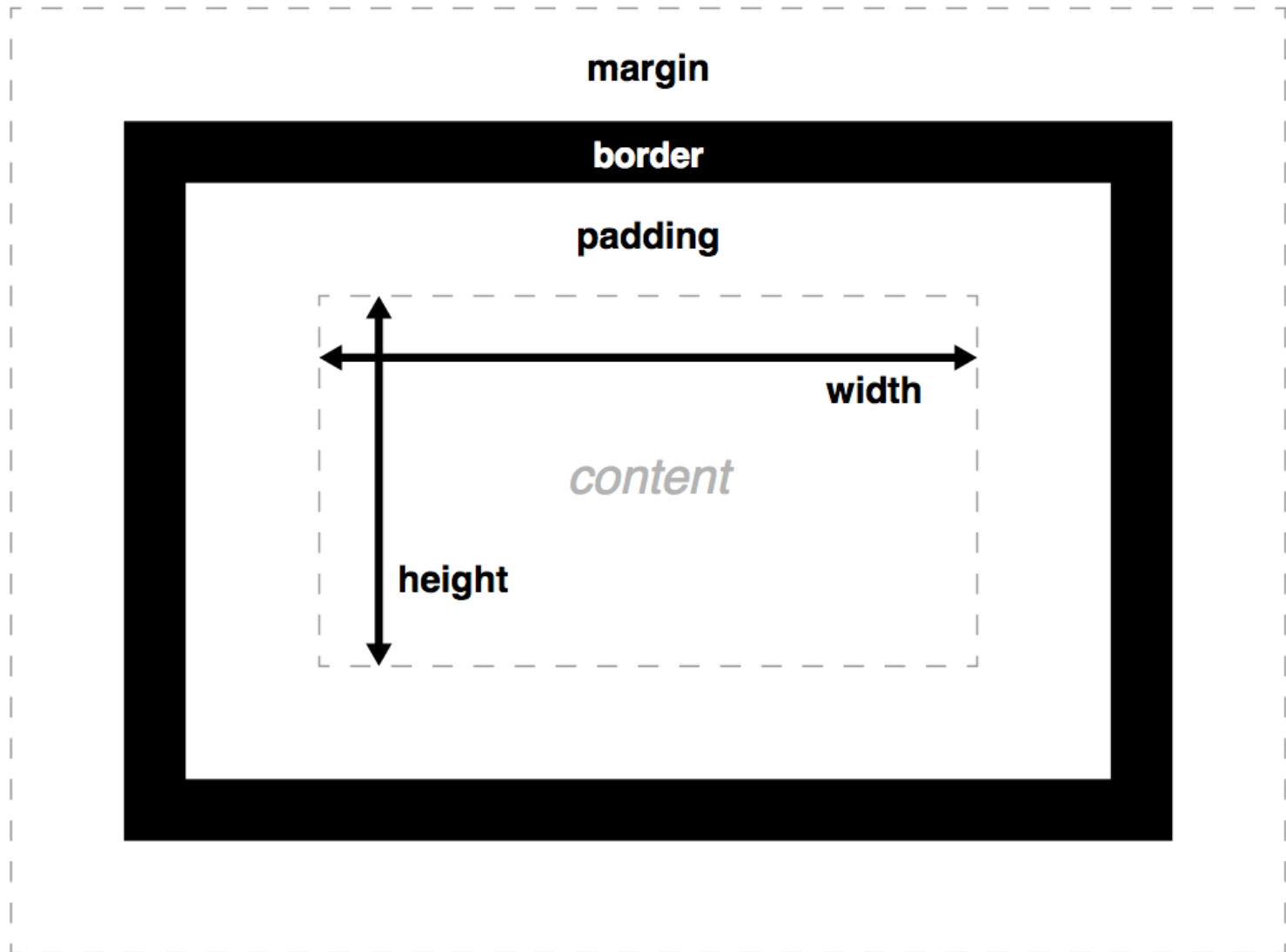
`width/height` – šírka/výška obsahového bloku



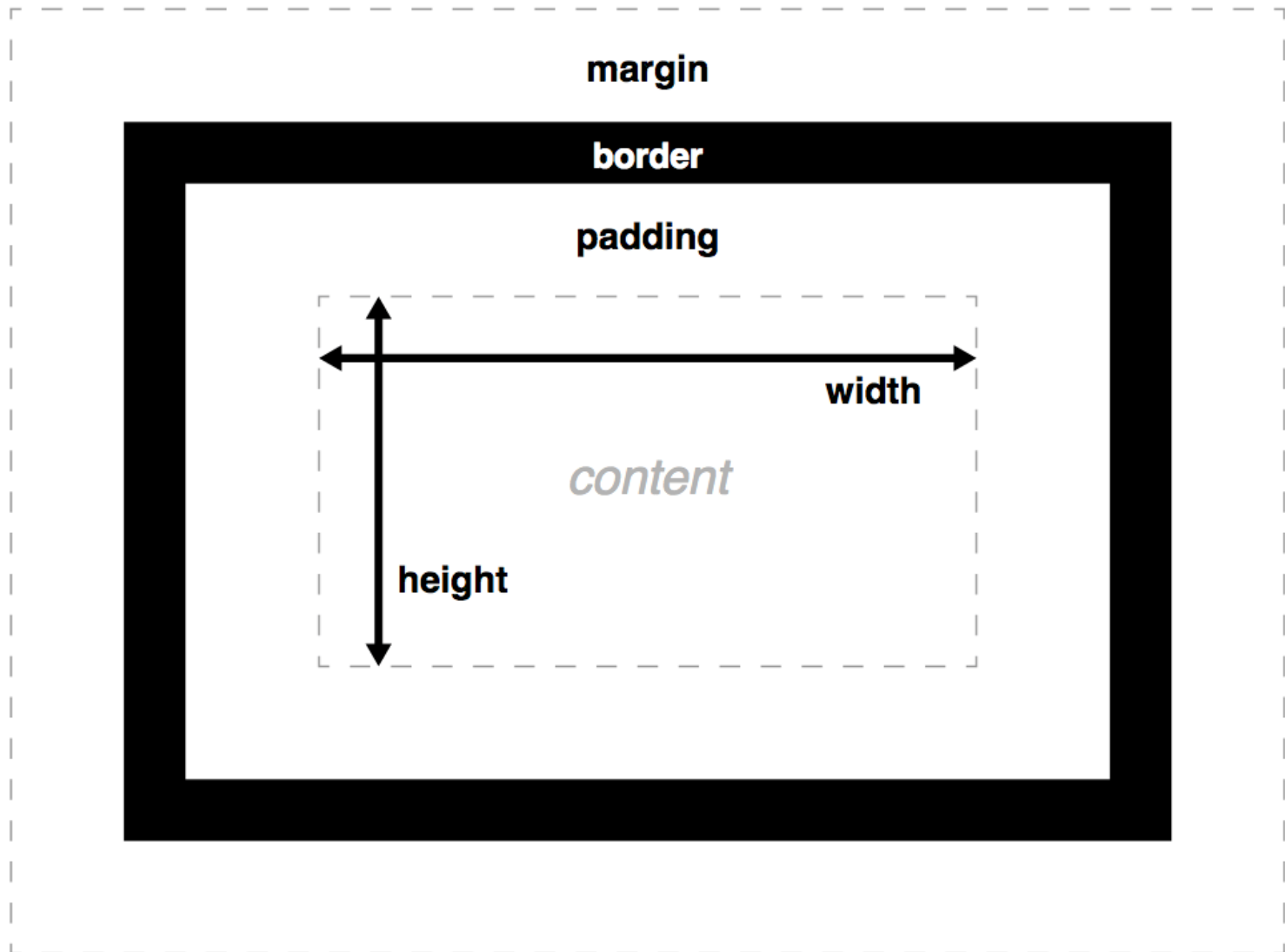
padding – vnútorný okraj bloku



`border` – rámik, medzi vnútorným  
a vonkajším okrajom bloku



`margin` – vonkajší okraj bloku



# Box model – obsah

- `width`, `height` – šírka/výška obsahu bloku
- obsahový blok môže obsahovať jednak text, ale tiež ďalšie bloky (elementy)
- tiež
  - `min-width` – blok nemôže mať menšiu šírku, ako určuje hodnota
  - `max-width`
  - `min-height`
  - `max-height`
- napr.:

```
{  
    width: 50%;  
    min-width: 480px;  
}
```

# Box model –padding

- obsahový blok môže obsahovať jednak text, ale tiež ďalšie bloky (elementy)
- padding - vnútorný okraj
  - možnosť nastaviť každú zo strán
    - padding: 10px 15px 5px 10px;  
(top:10, right:15, bottom:5, left:10)
    - padding: 10px;  
(top=right=bottom=left:10)
    - padding: 10px 15px;  
(top=bottom:10, right=left:15px)
    - padding-top: 10px;
    - padding-(right|bottom|left): 10px;

# Box model – margin

- **margin** - vonkajší okraj
  - možnosť nastaviť každú zo strán
    - `margin: 10px 15px 5px 10px;`  
`(top:10, right:15, bottom:5, left:10)`
    - `margin: 10px;`  
`(top=right=bottom=left:10)`
    - `margin: 10px 15px;`  
`(top=bottom:10, right=left:15px)`
    - `margin-top: 10px;`
    - `margin(right|bottom|left): 10px;`
- **margin collapsing** – keď sa dva bloky dotýkajú, vzdialenosť medzi nimi nie je suma vonkajších okrajov na danej strane, ale veľkosť okraja väčšieho z nich

# Box model – border

- rámik medzi margin a padding
- prednastavene je nulový
- možnosť nastaviť, hrúbku, farbu, a štýl čiary (napr. bodkovaná)
- možnosť nastaviť každú zo strán zvlášť:
  - `border: 1px solid red;`
  - `border: 5px dotted rgb(254,211,10);`
  - `border-(top|right,bottom,left)`
  - `border-(width|style,color)`
  - `border-(top|right,bottom,left) - (width|style,color)`



# Box model – poznámky

- výšku bloku a rámik nie je možné nastaviť v percentách
- celková šírka bloku je
  - width +  
padding-right +  
padding-left +  
border-right +  
border-left
  - + margin-left + margin-right
- background-color, background-image
  - **siaha až po hranu rámika**

# Box model – background-clip

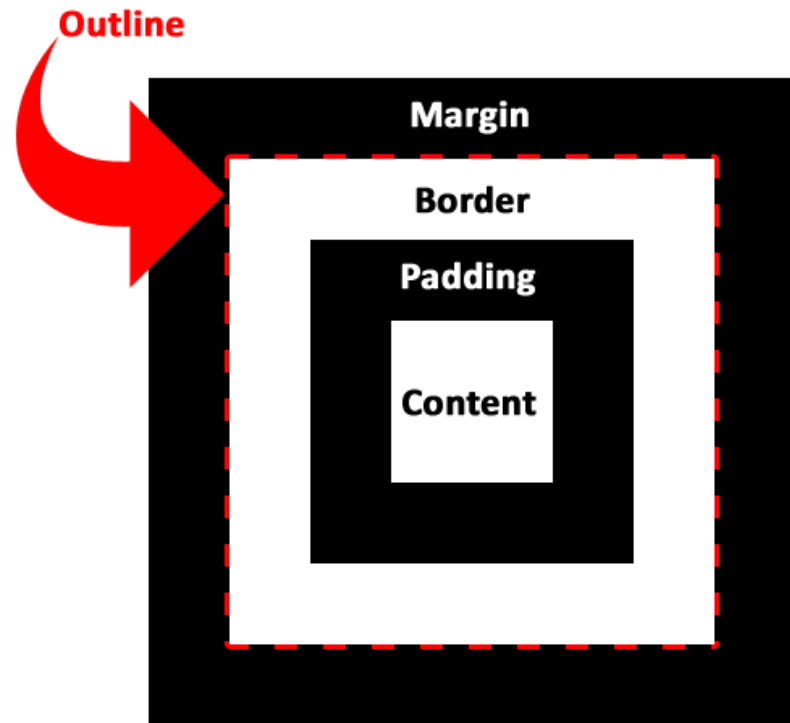
- pokiaľ má v bloku siahať obrázok (ako pozadie), príp. farba pozadia môžeme explicitne určiť:
- `background-clip: border-box;`
- `background-clip: padding-box;`
- `background-clip: content-box;`

# Box model – overflow

- ak nastavíte fixnú veľkosť bloku (napr. v px), môže sa stať, že sa obsah doň nezmestí – obsah bude pretekať
- pretekajúci obsah vieme dostať pod kontrolu
- **vlastnosť** `overflow`:
  - `auto` - pretekajúci obsah sa skryje a zobrazia sa posuvníky (angl. scrollbars)
  - `hidden` - pretekajúci obsah sa skryje
  - `visible` – prednastavená hodnota, pretekajúci obsah je viditeľný, môže spôsobiť vizuálne – nechcené nedostatky

# Box model – outline

- vyzerá ako rámik (border) ale nie je priamou súčasťou box modelu
  - nemení veľkosť bloku,
  - je vykreslený mimo border-boxu
  - vo vnútri marginu



# Box model – typy blokov

- elementy: blokové, inline
- CSS poskytuje aparát na zmenu typu bloku elementu
- cez vlastnosť `display`
- `block` – obsah pred/za blokom je na novom riadku, na elementy je aplikovaný štandardný box model
- `inline` – obsah na rovnakom riadku, ako okolitý text a iné `inline` elementy;
  - obsah sa zalamuje s tokom textu, podobne ako riadky v odseku – „stratí svoj tvar“
  - nastavenie šírky/výšky nemá vplyv;
  - nastavenie `margin`, `padding` a `border` ovplyvní pozíciu okolitého textu, neovplyvní okolité blokové elementy

# Box model – typy blokov

- `inline-block` – niečo medzi `block`/`inline`
  - obsah na rovnakom riadku, ako okolitý text
  - je možné nastaviť šírku/výšku
  - obsah sa nezalamuje s tokom textu, ak nemá miesto v danom riadku, je celý presunutý na ďalší riadok – nestratí svoj tvar
- blokové elementy majú prednastavené `display: block;`
- Inline element majú prednastavené `display: inline`
  - `b`, `i`, `strong`, `span`, `var`, `a`
- **inline-block?** `button`, `textarea`, `input`,...
  - <http://w3c.github.io/html-reference/img.html#img-display>
  - <http://jsfiddle.net/AQ2yp/>

# Štýlovanie textu

- **štýly písma**
  - veľkosť, tučné písmo, kurzivá
- **štýl bloku textu**
  - zarovnanie textu, výška riadku, medzera medzi slovami,

# Štýl písma - color

- farba písma/textu
- veľa vlastností ma prefix `font-`(size) , proste iba `color`
- `color: red | rgb(255,0,0) | #f00;`



# Štýl písma – font-family

- umožňuje určiť typ písma, konkrétne písmo, alebo zoznam písiem
- prehliadač aplikuje písmo, iba ak je dostupné na danom počítači – z ktorého pristupujeme na web
  - ak nie je, aplikuje prednastavené písmo

```
p {  
    font-family: arial;  
}
```

# Bezpečné písmo

- angl. web safe fonts
- vývojári chcú mať často písmo pod kontrolou
  - je súčasťou dizajnu, má silnú estetickú hodnotu a je výrazom určitého vkusu
- bezpečné písmo, ktoré sú dostupné naprieč všetkými systémami
  - Arial, Courier New, Georgia, Times New Roman, Trebuchet MS, Verdana
- písmo sa dá k webu aj pripojiť

[zoznam bezpečných písiem: http://www.cssfontstack.com/](http://www.cssfontstack.com/)

# Prednastavené písma

- CSS definuje 5 generických názvov pre písma
  - `serif` – pätkové
  - `sans-serif` – bezpätkové
  - `monospace` – znaky s rovnakou šírkou
  - `cursive` – písmo napodobňujúce rukopis
  - `fantasy` – dekoratívne písmo
- aké konkrétne písmo (font face) prehliadač použije pre daný generický názov – závisí – aj na OS, na ktorom beží
  - prvé tri relatívne ok, ale `cursive` a `fantasy` - uvážene

# Zásobník písíem

- angl. **font stack**
- poskytneme prehliadaču viacero písíem, z ktorých môže vybrať
- prehliadač vyberie prvé dostupné písmo
- dobrým zvykom je poskytnúť na konci vhodný generický názov písma

```
p {  
    font-family: "Trebuchet MS", Verdana, sans-serif;  
}
```

- pozn.: písma, ktoré sú z viacerých slov musia byť v " "

# Veľkosť písma

- `font-size: px | em | rem`
- veľkosť písma sa dedí z rodiča
- uvažujte že nastavíte v `<article>` veľkosť `1.5em` = `24px`
- potom chcete vnorenému odseku nastaviť `20px`, akú veľkosť v `em` nastaviť?
  - $20/24 = 0.8333333em$
- lepšie používať `rem`

# W3C CSS validátor

- <http://jigsaw.w3.org/css-validator>
- vyskúšajte aj CSS LINT:
- <http://csslint.net/>