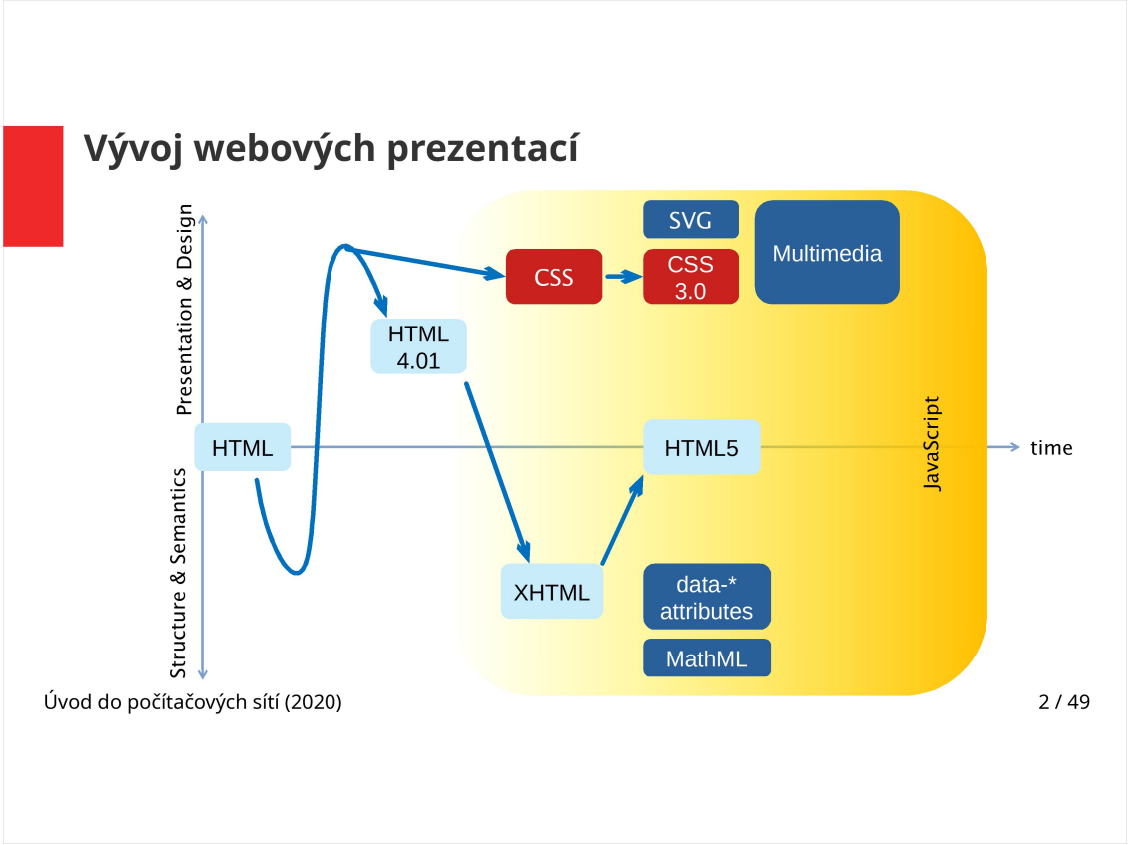
# Cascading Style Sheet – kaskádové styly

Úvod od počítačových sítí

Mgr. Klára Pešková, Ph.D., Klara.Peskova@mff.cuni.cz
Katedra softwaru a výuky informatiky




V době svého vzniku HTML definovalo hlavně strukturu dokumentu, podle této struktury se dokument také zobrazoval.

Postupně v HTML přibývaly možnosti, jak ovlivnit vzhled stránky – atributy pro barvu či zarovnání textu, atd.

Definování vzhledu stránek se postupně zcela přesunulo směrem k CSS stylům.

V XHTML opět zbyla pouze definice struktury dokumentu.

Později přibyla podpora formátu SVG (vektorová grafika, animace) a velká část prezentovaného obsahu se také přesunula do multimédií.

Naopak, co se týče struktury dokumentu a dat, v HTML přibyly atributy `data-*`, pomocí kterých je možné předávat data JavaScriptu, nebo MathML pro zápis matematických vzorců (ale např. Google Chrome MathML přestal podporovat).

S tím, jak jsou stále více rozšířené webové aplikace, je také čím dál tím větší část webových prezentací tvořena JavaScriptem.

## Vývoj CSS

- CSS 1 (1996)
  - Základní styly pro text – fonty, zarovnání, …
  - Okraje (padding, margin), rámečky
  - Barva textu a pozadí
- CSS 2 (1998)
  - Nové typy pozicování
  - Koncept @media
- CSS 2.1 (2004–2011)
  - Řešení různých problémů CSS 2

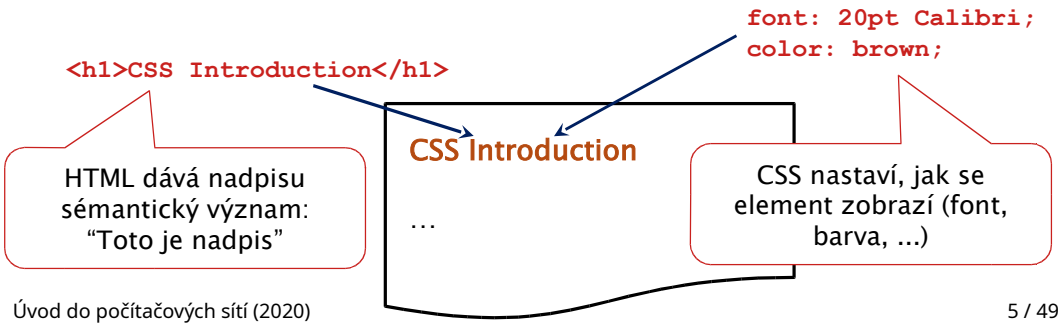
## Vývoj CSS

- CSS 3 (1999–současnost)
  - Vylepšení stávajících vlastností – složitější pozadí, vylepšené rámečky, ...
  - Další moderní efekty - kulaté rohy, stíny
  - Povoluje vlastní fonty
  - Přidány tranzice a animace

## Co je CSS?

- CSS popisuje, jak budou HTML prvky vypadat na obrazovce, telefonu, na papíře...

+ jedno CSS pro několik stránek






Vzhled webové stránky je tvořen propojením HTML elementů s CSS vlastnostmi. HTML elementy mohou mít přiřazeny CSS styly, které ovlivňují způsob, jak prohlížeč HTML prvky vykreslí.

Jeden CSS stylesheet – soubor se styly, je možné použít pro více stránek. Tím je možné snáze udržet jednotný vzhled celého webu.

## CSS syntax

- Obecný tvar pravidel:

```
selektor {
    vlastnost: hodnota;
    vlastnost2: hodnota2
}
```

- Příklad:

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



.css soubor se skládá z jednotlivých CSS pravidel zapsaných za sebou. Každé pravidlo se skládá ze selektoru a bloku deklarací.

Selektor určuje HTML element nebo elementy, pro které následující pravidla platí.

Blok deklarací obsahuje seznam deklarací ve tvaru `vlastnost: hodnota`, jednotlivé vlastnosti jsou odděleny středníky.

CSS syntax je jiná než syntax HTML, oba jazyky ale mohou být zapsané v tom samém souboru (viz další slide).

## Tři způsoby vkládání do HTML

- Inline:

```
<p style="text-align: center;">Text</p>
```

- V hlavičce HTML (uvnitř elementu <head>), jako obsah tagu <style>

```
<head>
   <style>
      p {text-align: center;}
   </style>
</head>
```

- V samostatném CSS souboru:

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

styles.css

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

body { color: blue; }
```



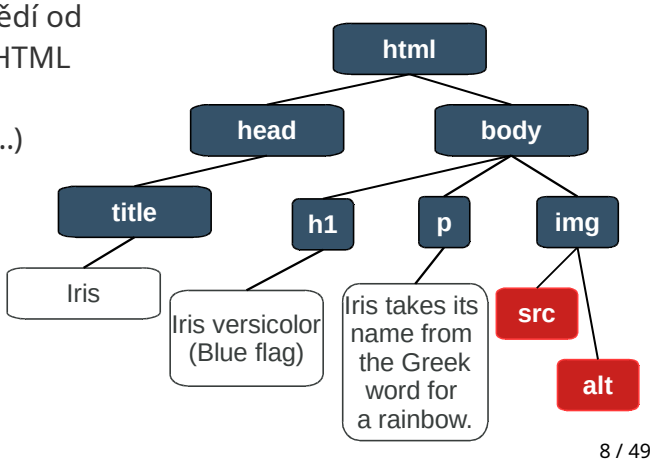
- Inline
  - Je možné použít pro jakýkoliv HTML element
  - Styly jsou hodnota atributu `style`
  - používá se pouze ve speciálních případech (typicky tento způsob využívají skripty)

- V HTML elementu `<style>`
  - Styly se definují v hlavičce HTML (uvnitř elementu `<head>`)
  - Používá se CSS syntax, stejně jako v externím souboru
  - Vhodné pro stránky, které jsou tvořeny pouze jedním hlavním souborem (rychlejší stažení stránky)

- V externím .css souboru
  - Je možné definovat různé .css soubory pro různá použití
  - Jeden .css soubor je možné použít pro více HTML souborů

## Dědičnost

- Některé vlastnosti, např. font, nebo barva textu se dědí od nadřazených prvků v HTML stromě (`<body>` → `<h1>`, `<p>`...)
- Vlastnosti, které se dědí mají nastavenou defaultní hodnotu `inherit`






Dědičnost některých vlastností souvisí s hierarchickou stromovou strukturou HTML dokumentu. Např. styl písma, který je nastaven pro element `<body>` se použije i pro potomky `<body>`, případně i pro jejich potomky.

## Druhy CSS selektorů

- `p` styly pro druh HTML elementu (pro všechny odstavce)
- `#mys` styly pro HTML prvek, s `id="mys"`

  ```
  <h1 id="mys">Myš domácí</h1>
  ```

- `.upozorneni` styly pro prvky s přiřazenou CSS třídou
  - Jednomu HTML tagu může být přiřazeno více tříd

    ```
    <p class="upozorneni cervena">Pozor, pes!</p>
    ```

- `*` univerzální selektor (všechno)



Existuje více způsobů, jak můžeme přiřadit styly vybraným HTML elementům – je možné použít různé druhy selektorů.

- Styly je možné přiřadit všem elementům vybraného typy, např. všem odstavcům,
- nebo konkrétnímu HTML elementu, který má přiřazené `id` (selektor v CSS stylech začíná znakem #)
- Je také možné vytvořit vlastní třídu, a tu přiřadit pomocí atributu `class` vybraným HTML elementům (mohou být různého typu, např. `p` a `h2`); selektor začíná znakem tečka
- Styly deklarované pro selektor * se aplikují na všechny HTML elementy

## Používání selektorů

- Agregace
  - `s1, s2 {…styly…}` stejné styly pro více selektorů
- Kombinování selektorů
  - `p.aktualne` vybere všechny odstavce se třídou `aktualne`
  - `h1#hlavni` vybere `<h1 id="hlavni">`
  - Využití relativní pozice prvku v rámci stromové struktury HTML
    - `E F` vybere prvky `F`, které mají předka `E`
    - `E > F` vybere prvky `F`, které mají přímého rodiče `E`
    - `E + F` vybere prvky `F`, které jsou hned za `E`
    - `E ~ F` vybere prvky `F`, které jsou za `E`



## Agregace

Jeden blok deklarací je možné použít pro více různých selektorů, jejich názvy jsou v tomto případě odděleny čárkami.

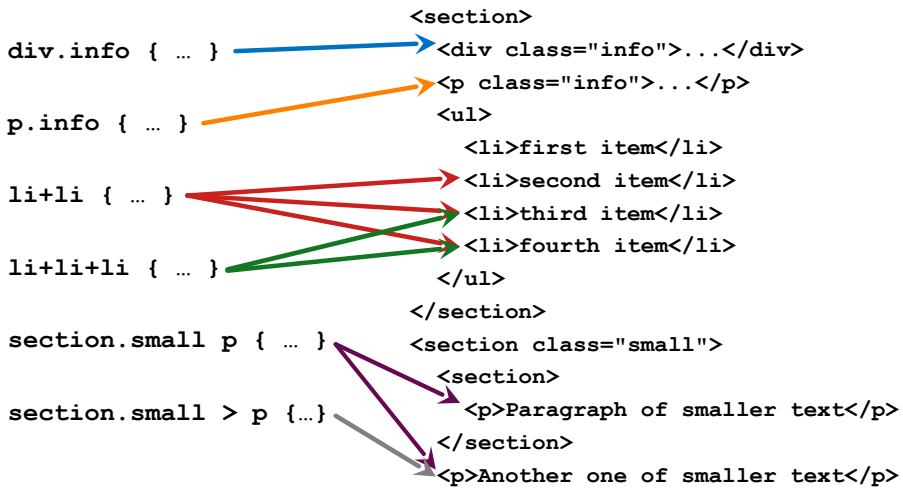
## Kombinování selektorů

Selektor nemusí být pouze jednoduchý název HTML tagu nebo třída, je možné selektory různě kombinovat.

Vysvětlení pojmů:

- Předek – ve stromové struktuře kdekoliv “nad”
- Přímý rodič – ve stromové struktuře hned “nad”
- Hned za – ve stromové struktuře na stejné úrovni, v kódu hned za prvkem
- Za – ve stromové struktuře na stejné úrovni

## Kombinování selektorů – příklad






- div se třídou info
- p se třídou info
- li, jemuž předchází li
- li, jemuž předchází li, kterému opět předchází li
- p, které je kdekoliv uvnitř sekce se třídou small
- p, které má rodiče (nejbližšího předka) section se třídou small

## Kombinování selektorů

- `ul li`
  - `li` kdekoliv uvnitř `ul`
- `p.info` vs. `p .info`
  - Mezera má význam!
- `main ul, ol`
  - main patří jen k ul

```
<ul>
 <li>
  <ol>
   <li> styly se aplikují
        i na toto li
   </li>
  </ol>
 </li>
</ul>
```



Pozor, mezery v selektorech mají význam!
(Význam mezery: Aby se selektor aplikoval, část selektoru za mezerou musí mít jako předka část před mezerou.)

`p.info` vs. `p .info`

- První příklad vybere `p` se třídou `info`
- Druhý – elementy se třídou `info`, které jsou uvnitř `p`

`main ul, ol`

- Styly platí pro:
  - `ul`, které jsou uvnitř HTML elementu `main`
  - A pro všechny `ol` v celém dokumentu

## Pseudotřídy

- Odkazy

| | |
|---|---|
| `a:link` | normální odkaz |
| `a:visited` | navštívený |
| `a:hover` | při přejetí myší |
| `a:active` | právě kliknutý |

- `:first-child` (`:first-of-type`) – styl se aplikuje, pokud je daný prvek první potomek / první potomek daného typu
- `:last-child` (`:last-of-type`), `:only-child` (`:only-of-type`)
- `:nth-child(even)` / `:nth-of-type(3n+1)`
  - př. první `<i>` ve všech `<p>` — `p i:first-child`
  - př. všechna `<i>` v prvním `<p>` — `p:first-child i`



Pomocí pseudotříd můžeme vybrat daný element pouze v nějaké konkrétní situaci.

Např.

- odkaz, po něm uživatel právě přejíždí myší,
- navštívený odkaz,
- řádek tabulky, pokud se jedná o sudý řádek

# Pseudotřídy – příklad

```
tr:nth-child(even) {background-color: #f2f2f2;}
```

| Jméno | Příjmení | Body |
|---|---|---|
| Petr | Smutný | 100 |
| Ludvík | Veselý | 150 |
| Jiří | Černý | 300 |
| Klement | Nový | 250 |

## Pseudo elementy

- Styly pro části HTML prvků

```
::first-letter
::first-line
::selection
::after
::before

h1::after {content: url(smiley.gif)}
p::first-letter {color: #f00; font-size: xx-large;}
```



`::after` vloží obsah hned za daný element, v příkladu uvedeném na slidu se zobrazí za všemi nadpisy úrovně jedna smajlík

`::selection` – styly se aplikují na část textu, kterou uživatel vybere myší

## Kaskády

- Pro jeden HTML prvek může platit více stylů
- Poměrně složitá pravidla, v jakém pořadí se styly se aplikují
- Váha je určena (podrobněji na dalším slidu):
  1. Podle toho, kde jsou styly zapsané – záleží na "blízkosti" definice stylu HTML elementu (např. inline styly mají větší prioritu než styly v externím souboru)
  2. Podle specificity selektorů
  3. Podle pořadí, ve kterém jsou styly zapsány (předchozí přepíše následující)
- Absolutní přednost `color: blue !important;`



Jednomu selektoru můžeme přiřadit i více bloků deklarací. Např. společné vlastnosti pro více selektorů můžeme deklarovat v jednom bloku a potom pro každý selektor ještě zvlášť další styly přidat v dalších blocích.

`!important` je potřeba používat s maximální opatrností, jde o porušení struktury návrhu stylů. Nemělo by se používat pokud možno vůbec.

## Specificita selektorů

- Definuje pořadí – prioritu selektorů
- Výpočet:
  - Připočtěte 1000 za deklaraci v atributu style
  - 100 za ID
  - 10 za třídu nebo pseudo-třídu
  - 1 za název HTML elementu nebo pseudo-elementu
- Zjednodušeně: atribut style > ID > třídy > elementy
- Tedy např. `a:hover` má vyšší prioritu než samotné `a`
- Pokud je specificita stejná, vítězí naposledy definovaný (přepíše předchozí)



Obecně, čím konkrétněji specifikujeme daný selektor, tím se předpokládá, že nám více záleží na deklarovaných stylech.

Např. písmo všech odstavců je modré, pouze u jednoho vybraného, který má přiřazené ID, chceme písmo červené.

Příklad výpočtu specificity:

```
A: h1 {color: black;}
B: #mys h1 {color: red;}
C: <div id="mys">
     <h1 style="color: #ffffff">Myš domácí</h1>
   </div>
```

Specificita A = 1 (1x element)
Specificita B = 101 (1x ID + 1x element)
Specificita C = 1000 (inline styly)
=> použije se pravidlo C

## Stylování textu – fonty

- Typ fontu: `font-family: Arial CE, sans-serif`
- Velikost: `font-size: 12px, 12pt, 1.2em` (relativní k velikosti textu aktuálního prvku)

  `font-style: normal|italic`

  `font-weight: normal|bold`

- Zkrácený zápis:

  `font: italic bold 20px Arial`



Jelikož nevíme, jaké fonty prohlížeč podporuje, dáváme mu zpravidla na výběr z více možností. Jednotlivé možnosti jsou oddělené čárkou. Fonty tedy specifikujeme od toho, který si nejvíce přejeme, až po obecné určení, zda se má použít patkové či bezpatkové písmo.

## Stylování textu - pokračování

- Zarovnání textu

  `text-align: left|right|center|justify`

- Podtržení atd.

  `text-decoration: none|underline|line-through`

- Text a prostor (mezi, kolem)

  `text-indent, line-height, letter-spacing, word-spacing`

## Barvy

- barvu je možné nastavit různými způsoby:

```
Red (Tomato, MediumSeaGreen)
#161616
rgb(0,0,100)
```

- Průhlednost `opacity` nebo `rgba`

```
opacity: 0-1
rgba(255,0,0,0.5)
```

- `color` – barva popředí (např. písmo)



Barvy je možné zadávat několika způsoby:

- Předdefinované “pojmenované” barvy (ne každý prohlížeč podporuje všechny barvy, tyto jsou bezpečné) https://www.w3schools.com/colors/colors_names.asp
- Zápisem v hexadecimálním tvaru – první dvě číslice reprezentují červenou složku, druhé dvě zelenou a poslední dvě modrou
- Zápisem pomocí rgb (Red, Green, Blue)

Poznámka:

- Pokud všechny barevné složky – červená, zelená, modrá – mají stejnou hodnotu, výsledkem je odstín šedé.

## Nastavení pozadí

- `background-color` – barvená výplň pozadí
- Obrázky na pozadí

  `background-image: url("obrazek.gif");`
  `background-position: right top;` – umístění v rámci elementu
  `background-repeat: repeat-x | no-repeat` – dá se použít na různé textury
  `background-attachment: fixed;` – nastaví, zda je pozice pozadí svázaná s dokumentem nebo oknem prohlížeče
- Nastavení pozadí jednou vlastností

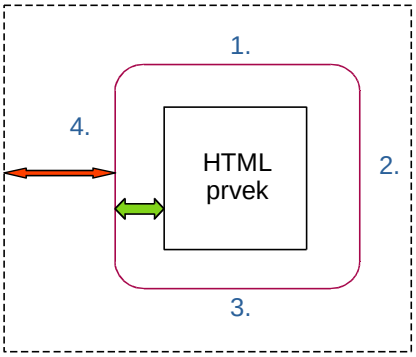
  `background: #ffffff url("tree.png") no-repeat right top`
- Barevný přechod

  `background: linear-gradient(to bottom right, red, blue)`

Na slidu je nastavené pozadí pomocí opakujícího se obrázku. Jak je vidět, ne vždy je to dobrý nápad.

## Box Model (Okraje a rámeček)

- Padding
  - 10px (jedna hodnota) – všechny strany
  - 10px 0px (dvě hodnoty) – nahoře-dole, vlevo-vpravo
  - 0px 5px 10px 5px (čtyři hodnoty) – nahoře, vpravo, dole, vlevo
- Margin
- Border `border-radius:10px`






Kolem samotného prvku je vnitřní okraj (`padding`), rámeček (`border`) a vnější okraj (`margin`).

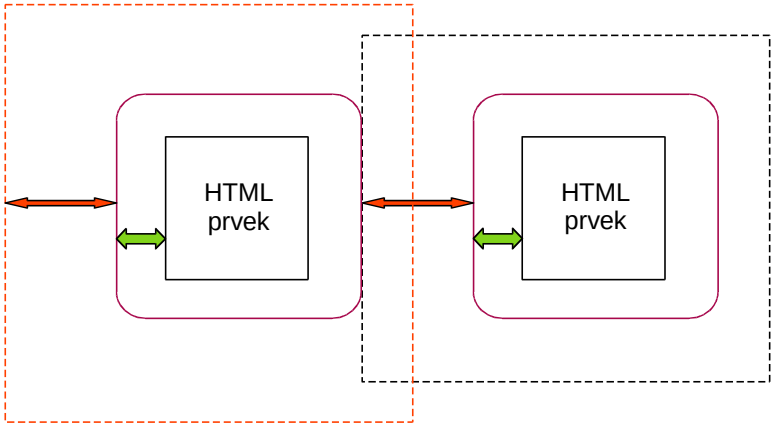
Vnitřní okraje je možné nastavit buď pomocí vlastnosti `padding`, a to jednou, dvěmi nebo čtyřmi hodnotami, nebo konkrétně pomocí

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

To samé platí pro vlastnosti `margin` a `border`.

## Box Model (Okraje a rámeček)

- Margin dvou prvků se může překrývat



## Rámeček (border)

`border: 2px solid blue;` (nastavení všech vlastností najedou)

- `border-style:`

  `dotted` – tečkovaně
  `dashed` – čárkovaně
  `solid` – normální čára
  `double` – dvojitě
  `groove` – 3D vyrytě
  `ridge` – 3D vystouple
  `inset` – zamáčknutě
  `outset` – vystouple
  `none` – bez rámečku
  `hidden` – skrytý rámeček

- Smíšený rámeček (nahore vpravo dole vlevo)

  ```
  border-style: dotted dashed solid double;
  ```

- Tabulka bez rámečku `border-collapse: collapse;`

## Stíny

- Stín textu (text-shadow)

```
text-shadow: v_posun h_posun r_rozmazání barva
color: white; text-shadow: 2px 2px 4px #000000;
```

- Stín rámečku (box-shadow)

```
box-shadow: v_posun h_posun r_rozmazání
            r_rozšíření barva
box-shadow: 3px 10px 10px 5px #555
```

Text se stínem



Na jeden element je možné použít více stínů, jejich nastavení v CSS vlastnosti `text-shadow` je odděleno čárkou.

Např. následující stín je složený z červeného a modrého:

Text-shadow with red and blue neon glow

```
text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
```

## Transformace

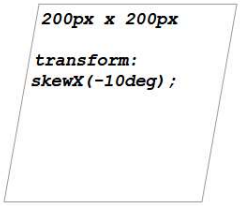
- 2D transformace:

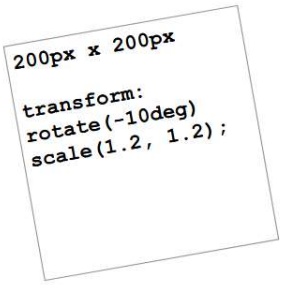
```
translate(x px, y px)
rotate(20deg)
scale(2,3) – 2x širší, 3x delší
skewX(20deg), skewY(20deg) – zmáčknutí podle osy x, y
skew(X,Y)
matrix(scaleX(), skewY(), skewX(), scaleY(),
       translateX(), translateY())
```

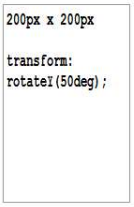
- 3D transformace:

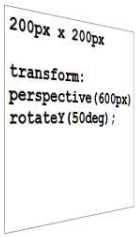
```
rotateX(90deg)
rotateY(90deg)
rotateZ(90deg)
```

# Příklady transformací



## Tranzice

`transition: width 2s` – jaká vlastnost se mění, trvání

`transition: width 2s, height 3s`

`transition-timing-function:`

- `ease` – default
- `linear`
- `ease-in`
- `ease-out`

`transition-delay: 1s`

`transition: width 2s linear 1s`



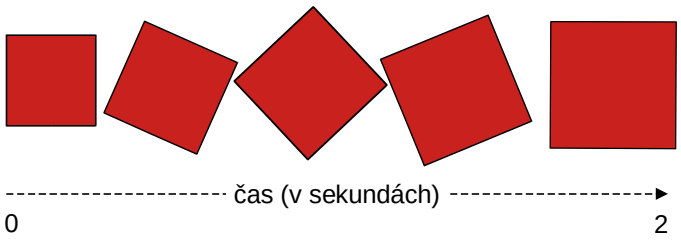
Díky tranzici můžeme pomocí CSS vytvořit animaci, během které se prvku mění např. barva, velikost, otočení, průhlednost, atd.

Vždy se nastavuje počáteční a cílová hodnota, můžeme nastavit trvání tohoto přechodu, a jeho průběh. Např. na začátku může změna probíhat pomaleji, potom se zrychlí (hodnota `ease-in`).

## Tranzice – příklad

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s, height 2s, transform 2s;
}

div:hover {
   width: 300px;
   height: 300px;
   transform:
   rotate(90deg);
}
```






Animace se spustí po přejetí myší přes div.

## Obecné – co všechno může být hodnota

- Numerické hodnoty (velikost, úhel, trvání)
  - `font-size: 12pt;`
- Barva
  - `background-color: #00ff00;`
- Odkaz na externí zdroj (např. obrázek)
  - `background-image: url("paper-texture.png")`
- Znakový řetězec
  - `font-family: "Courier New";`
- Hodnota z výčtu
  - `border-style: solid;`

# Obecné - jednotky

- Všechny číselné hodnoty kromě 0 musí mít zadanou jednotku

| | |
|---|---|
| `cm, mm, in` | Centimeters, Millimeters, Inches (1in = 2.54cm) |
| `px` | Pixels (1px = 1/96 in) |
| `pt` | Typographical points (1pt =1/72 in) |
| `pc` | Picas (1pc = 12pt) |
| `em` | Relative to the font-size of current element |
| `ex` | Relative to the height of 'x' in current font size |
| `%` | Special – relative to some existing/inherited value |
| `vh, vw` | Relative to 1% of width/height of the viewport |
| `deg` | Degrees (rotation) |
| `s` | Seconds |

## Obecné - Zkrácený zápis vlastností

- Mnoho CSS vlastností je možné zapsat v “úsporné” podobě
- Např. tři vlastnosti rámečku, je možné nastavit buď každou zvlášť:

```
border-width: 2px;
border-style: solid;
border-color: blue;
```

- Nebo pomocí jedné vlastnosti:

```
border: 2px solid blue;
```

## Umístění a zobrazení prvků na stránce

- Základní nástroje:
  - Vlastnost `float`
  - Pozicování
  - Vlastnost `display`



Tyto tři nástroje budou rozebrány na následujících slidech.

## Vlastnost float a obtékání elementů

- Element se “vznáší” – vlevo nebo vpravo, ostatní elementy kolem něj “obtékají”

```
float: left|right|none
```

- Ostatním prvkům je možné nastavit, aby kolem float elementů neobtékaly, pomocí vlastnosti clear:

```
clear: left; clear: right; clear: both;
```

  - Na jedné (nebo obou) stranách se nesmí “vznášet” žádný element (obsah elementu s nastavenou vlastností `clear` je posunut pod “vznášející” se element)

## Vlastnost float – příklad




Kolem druhého nadpisu se nemůže nic vznášet, nadpis se tedy umístí až pod (vznášející se) obrázek.

## Vlastnost float – příklad 2

- Tři (stejné) obrázky mají všechny nastavenou vlastnost `float: left`

  → vznáší se všechny vlevo, vedle sebe
- Je možné využít pro rozložení prvků na stránce (např. pro menu - navigaci)

## Zobrazení prvků

- Pro každý HTML element je dáno, jestli se zobrazuje v toku textu – v řádku (inline) nebo jako samostatný blok (block)
- Toto můžeme změnit pomocí vlastnosti `display`

  `display: block|inline`

  `display: none`

  `display: inline-block` – jako inline, ale je možné nastavit šířku, výšku a okraje

- `visibility: hidden|visible` – element zabírá místo na stránce

## Pozicování

- Prvky se vykreslují v pořadí, ve kterém jsou definované ve zdrojovém kódu stránky
  - Kromě prvků s nastavenou vlastností `float`
- Toto můžeme změnit pomocí pozicování

```
position: static|relative|fixed|absolute|sticky
```

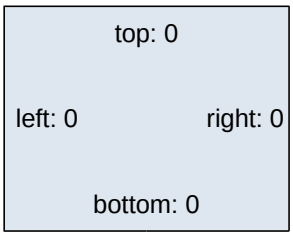
- `z-index` vrstva, ve které se zobrazí prvky, které se překrývají



Jednotlivé možnosti pozicování budou rozebrány na následujících slidech.

## Pozicování

- `static` – defaultní hodnota
- `relative` – relativně vzhledem k pozici, ve které by byl prvek normálně vykreslen
  - Je možné nastavit vlastnosti `top`, `right`, `bottom`, `left`
- `fixed` – relativně vzhledem k viewportu – zařízení, na kterém se stánka zobrazuje; t.j. když uživatel scrolluje, prvek zůstává na stejném místě
  - Nastavuje se pomocí `top`, `right`, `bottom`, `left`




...



Vlastnosti `top`, `right`, `bottom`, `left` určují vzdálenost prvku od vybraného okraje.

## Pozicování

...

- `absolute` – pozice je relativní vůči nejbližšímu napozicovanému předku
- `sticky` – něco mezi `relative` a `fixed`, pozice prvku je relativní; pokud ale uživatel scrolluje a prvek by nebyl vidět, zůstane "viset" v dané pozici (jako při `position: fixed`)
  - Fixní pozice se nastavuje pomocí `top`, `right`, `bottom`, `left`



Napozicovaný prvek je prvek, který má `position` nastavenou na jakoukoliv hodnotu kromě `static`.

## CSS vlastnosti související s pozicováním

- `width`, `height` – šířka, výška prvku

- `min-width`, `max-width` – minimální a maximální šířka
- `min-height`, `max-height` – minimální a maximální výška

- `top`, `right`, `bottom`, `left` – vzdálenost od příslušného okraje



Tip:

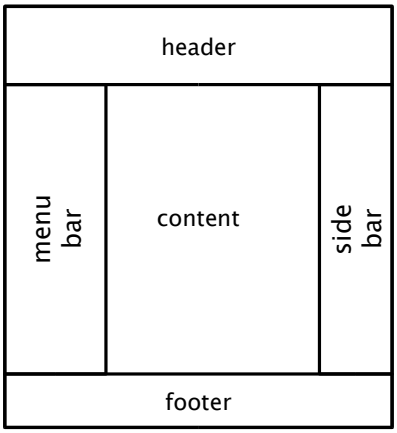
Pokud nastavíte velikost obrázku pomocí

```
max-width: 100%,
```

nezvětší se víc, než je jeho skutečná velikost. (Výška se dopočítá automaticky.)

## Layout webové stránky

- =rozložení jednotlivých prvků na stránce
- Web designer musí vyřešit:
  - Scrollování stránky a jejích částí
  - Co se stane, když obsah nějakého prvku “přetéká”
  - …






Obrázek vpravo zachycuje typické rozmístění prvků na stránce, s postranními panely, záhlavím a zápatím, tzv. “svatý grál”, kterého dlouhou dobu nebylo snadné dosáhnout.

## Layout webové stránky – přístupy

- (Historicky) existují různé způsoby, jak vytvořit rozložení prvků s postranními panely
  - Tabulky – NEPOUŽÍVAT
  - Využití CSS stylů
    - Pomocí vlastnosti float
    - Pomocí pozicování
    - Pomocí flexbox a grid – moderní přístup – ANO, používat (mimo rozsah této prezentace)



- Tabulky
  - Proti filosofii HTML a sémantického významu tagů
  - Velmi neflexibilní i při jednoduchých úpravách
  - Fixní rozměry stránky
- Pomocí float
  - Poměrně jednoduché vytvořit
  - Není úplně snadné nastavit výšku postranních panelů
- Pomocí pozicování
  - Může nastat problém s překrýváním obsahu postranními panely
  - Není problém umístit postranní panely téměř kamkoliv
- Flexbox, grid
  - Moderní přístup - používat

## @media

- Umožňují developerovi reagovat na typ zobrazení stránky a další vlastnosti zařízení, na kterém si uživatel stránku prohlíží
  - Typ zobrazení: all, screen, print, speech

```
@media print { styly pro tisk }
@media screen { styly pro zobrazení na obrazovce }
```

  - Další vlastnosti zařízení:
    - Velikost displeje (viewportu), orientace, barevná hloubka, ...
- Použití
  - Přímo v CSS souboru
  - Při připojování externího .css souboru v `<link>` uvnitř `<head>`



Pomocí `@media` je možné deklarovat zvlášť styly pro různé typy zobrazení webové stránky, např. styly pro obrazovku a speciální styly pro tisk

## @media

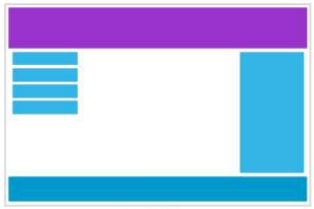
- Podmínky je možné spojovat pomocí and nebo čárky (která funguje jako or)

```
@media screen and (min-width: 480px) {
    CSS rules...
}
```

- Styly uvnitř složených závorek se aplikují, pokud je šířka okna na obrazovce, na které se stránka zobrazuje, větší než 480px

## Responsivní design

- Cíl: aby stránka vypadala dobře na všech zařízeních (počítač, notebook, tablet, telefon)
- Jak? Stejný obsah, změna velikostí prvků, přesun prvku na jiné místo
- Jak konkrétně?
  - Rozměry zadané relativně, v %, nebo pomocí jednotek `vh`, `vw`
  - Různá rozložení prvků (CSS styly) pro různá zařízení (pomocí `@media`)








Na malém displeji mají typicky všechny části stránky šířku 100%.

Moderní přístup je navrhovat rozložení stránky pro mobilní telefony, a teprve potom pomocí `@media` deklarovat styly pro větší displeje. Takto se styly pro telefony, kde uživatel má často pomalejší připojení, načtou jako první.

Jednotky `vh` a `vw` zamenají výšku a šířku tzv. viewportu.

## Viewport

- Oblast pro webovou stránku na zařízení uživatele
  - Např. velikost displeje telefonu, velikost okna prohlížeče

- V HTML5 můžeme viewport nastavit v hlavičce stránky

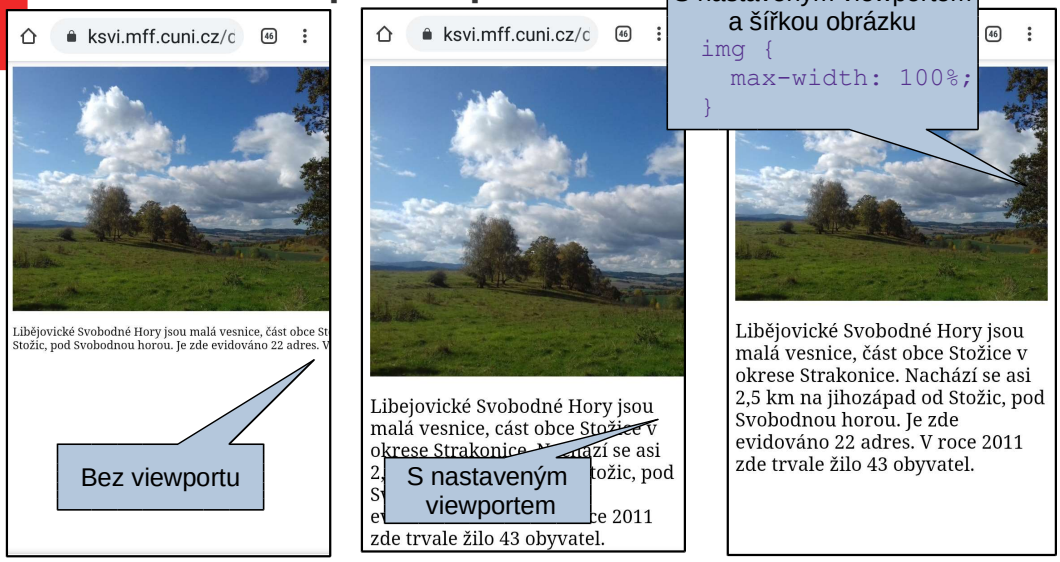
```
<meta name="viewport" content="width=device-width,
                               initial-scale=1.0">
```

- `width=device_width` nastaví stránce stejnou šířku, jako má zařízení, na kterém se stránka zobrazuje
- `initial-scale` nastaví počáteční přiblížení



Dříve bylo běžné, že stránky měly statický design a pevnou velikost – prohlížeče např. na telefonu stránku zmenšily (lepší než nic).

# Nastavení viewportu – příklad



# Otázky...

?