ROZLOŽENÍ

Úvod do jednoduchého rozložení (tabulky, listy, ...) a pokročilého rozložení pomocí moderních technologií (flexbox, grid, ...).

WEBOVÉ FONTY

Pokud chceme aktuálně používat nějaký font na webové stránce, musí být nainstalován na uživatelově zařízení a to ještě pod nějakým generalizovaným jménem. To je pro většinu stránek šílený problém a proto si fonty můžeme na webové stránce načítat z jiných zdrojů.

Fonty načítáme uvnitř CSS a v prohlížeči se na daný font načte a uloží do cache. V cache je po nějakou dobu a nemusí se po danou dobu stahovat.

Fonty jsou relativně velké soubory, musí uchovávat spoustu informací a proto bych jich na webových stránkách nemělo být spousta. Obecně limit samozřejmě daný není, ale doporučuje se mít max. 3 různá písma s různými variantami. Mezi varianty se počítá tloušťka písma a například i styl kurzívy.

My se nebudeme zaobírat tím, jak se přímo fonty registrují v CSS, ale ukážeme si službu Google Fonts, která nám velmi jednoduše přidá jakýkoliv font z jejich velké kolekce, kde jsou všechny k dispozici zdarma. Na stránkách Google Fonts naleznete jazyky označené Latin Extended, které mohou obsahovat znaky, které Český jazyk používá.

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

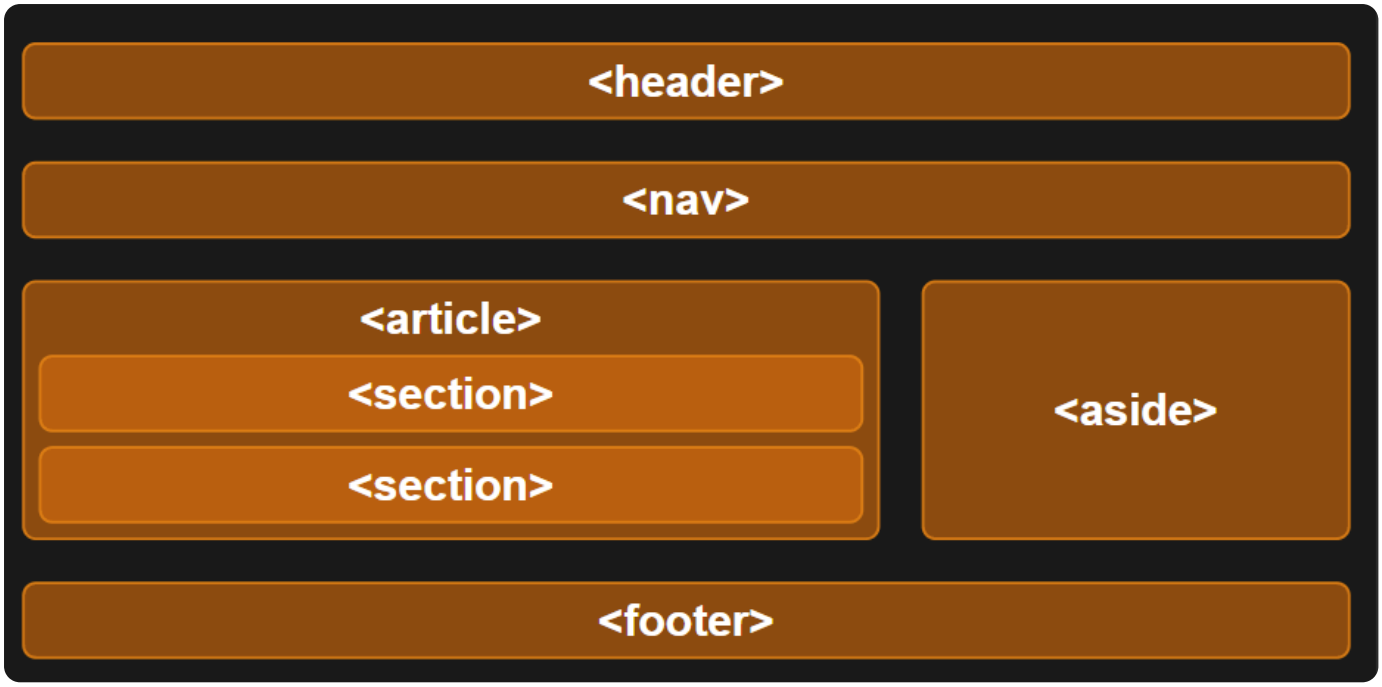
TAGY HTML ROZLOŽENÍ

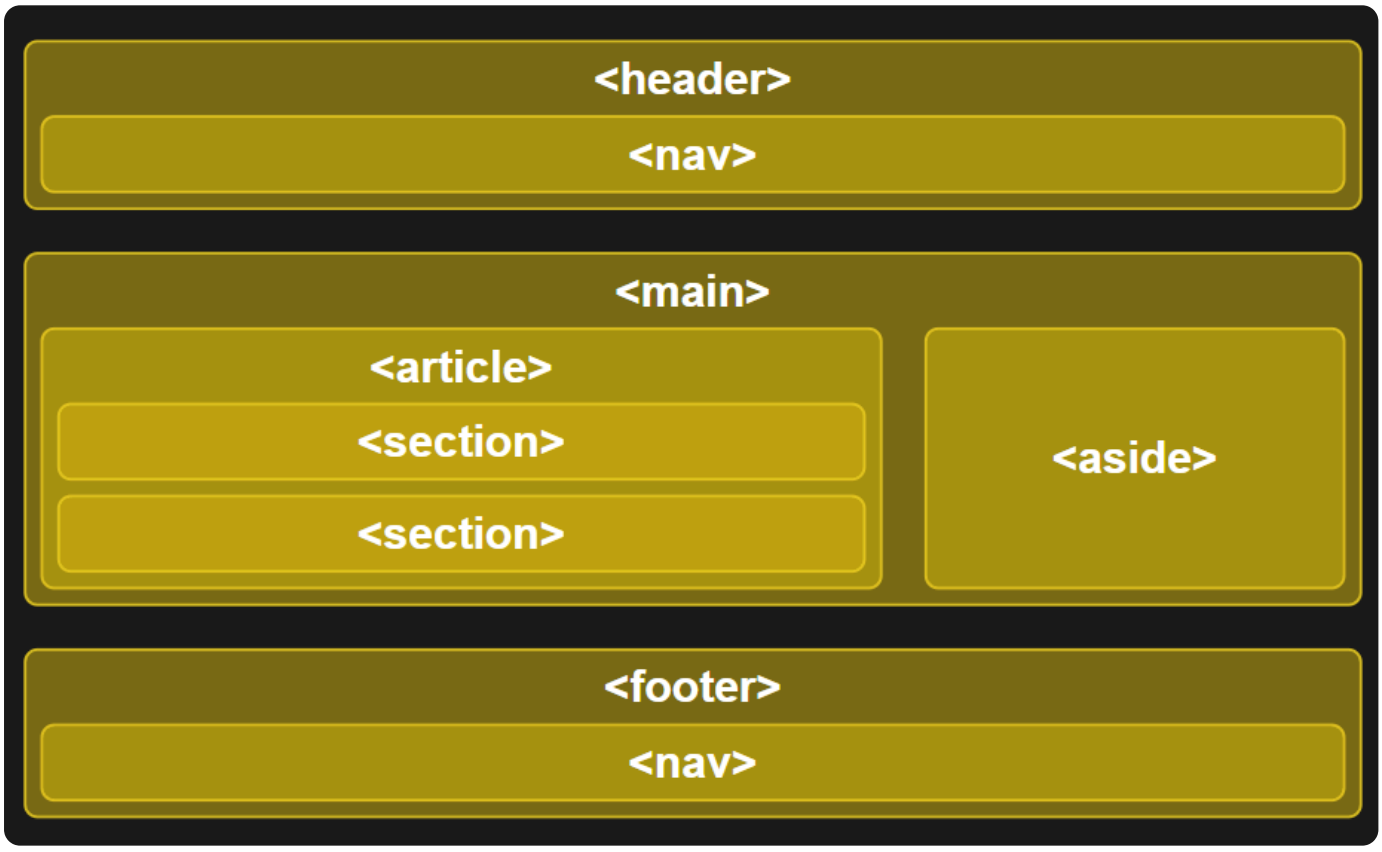
Jedním z cílů HTML5 bylo přidat tzv. sémantické tagy, které popisují různý obsah na stránce. Používání divu a podobných blokových prvků je fajn, ale velmi to limituje to, co může robot přečíst ze stránky. Když stránku rozdělíme pomocí níže uvedených tagů, tak může robot jednodušeji rozdělit stránku tak a vypreparovat to, co potřebuje.

Tagy jsou blokové a nemají většinou žádné základní styly. Tagy jsou následující:

- `header` — hlavička — obsahuje logo, navigaci a další
- `nav` — navigace — odkazy, mapy a další
- `article` — příspěvek — ucelený celek informací
- `section` — sekce — sekce příspěvku (můžeme však využívat i bez `article`)
- `main` — hlavní část — obsahuje informace, které se na stránkách často mění
- `aside` — boční část — nejsou hlavní informace, např. autor článku a podobně
- `footer` — patička — navigace, copyright a další

Často jsou stránky rozděleny například následovně:





Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

HISTORIE ROZLOŽENÍ

Jak jsem si dříve říkali, tak se zejména pro rozložení v minulosti používali tabulky. Práce s tabulkami je špatná, protože nejsou responzivní a v CSS neumíme přesměrovávat různé věci. Obecně seřazení je v CSS špatné a proto vznikli nové technologie - flexbox a grid. A náhodou se obě naučíme!

Stránky se často taktéž umisťovali na levou část stránky a úplně se ignorovala šířka zařízení.

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

FLEXBOX

Flexbox se rozděluje na kontejner a předměty. Kontejner je takový obal okolo prvků a určuje jim právě to, jak se budou umisťovat. Tam taktéž zapínáme flexbox. Předměty jsou uvnitř flexboxu a můžeme jim určovat nějaké vlastnosti, které bude flexbox respektovat.

Flexbox zapneme tedy tím, že mu v CSS určíme:

```
.container {
    display: flex;
}
```

Flexbox se dělí na dvě různé osy a to hlavní a sekundární osu. V základu je hlavní osa ta horizontální a sekundární vertikální. Flexbox však není dvourozměrné rozložení a osy slouží spíše pouze k zarovnání.

Na hlavní ose můžeme prvky zarovnávat pomocí vlastnosti `justify-content`. Na vedlejší pomocí `align-items`. Na hlavní ose máme k dispozici hodnoty:

- `center` - zarovnání doprostřed
- `flex-end` - zarovnání na konec flexboxu
- `flex-start` - zarovnání na začátku flexboxu
- `space-between` - zarovnání tak, že prvky budou mít co nejvíce místa mezi sebou
- `space-evenly` - zarovnání tak, že prvky budou mít mezi sebou stejné místa a to i okolo
- `space-around` - zarovnání tak, že prvky mezi sebou budou mít stejné místa a okolo budou poloviční

Na vedlejší ose můžeme zadat:

- `center` - zarovnání doprostřed
- `flex-end` - zarovnání na konec flexboxu
- `flex-start` - zarovnání na začátku flexboxu
- `stretch` - natáhne všechny prvky na danou dimenzi flexboxu

Jednotlivé osy můžeme prohodit pomocí vlastnosti `flex-direction` a hodnot `row` a `column`. Jednotlivé osy však můžeme i zalomit, to znamená, že když se přidá prvek, který se na osu nevejde, tak bude přesunut na "další řádek" osy. To ale neznamená, že máme dvourozměrné rozložení (přemýšlejte proč). Zalomení uděláme pomocí vlastnosti `flex-wrap` a hodnot `nowrap` a `wrap`.

Při tvorbě flexu je důležité přemýšlet jinak a v začátcích silně doporučuji jak jsou jednotlivé flexy propojené. Často zjistíte, že vlastně máte velmi mnoho flexů uvnitř sebe.

Pokud tato lekce není dostatečná pro nacvičení si flexboxu, můžete použít stránku ZDE, kde je flexbox velmi hezky vysvětlen.

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

NOVÉ VLASTNOSTI

Přetékaní

Novou vlastnost, kterou si ukážeme, je vlastnost `overflow`, která slouží k určení toho, jak se bude prvek chovat při tom, kdy jeho obsah je tak velký, že se do něj nevejde a přeteče. Vlastnosti `overflow` můžeme nastavit hodnoty:

- `hidden` - přetečený obsah nebude viditelný
- `visible` - přetečený obsah bude viditelný
- `auto` - zapne se skrolovací bar právě tehdy, když obsah přeteče
- `scroll` - skrolovací bar bude vždy viditelný

Stín

Vlastnost `box-shadow` slouží k vytvoření stínů okolo prvku. Jednotlivé stíny oddělujeme čárkou a každému stínu nastavujeme (v tomto pořadí):

- horizontální posun (povinné)
- vertikální posun (povinné)
- velikost rozmazání
- šířka nebo šíření stínu
- barva (povinné)
- pozice (pouze hodnota `inset`)

```
.fancy {
    box-shadow: 0 0 0 2em #b0f4aa,
                0 0 0 4em #66CCFF;
}
```

Filtr

Vlastností `filter` můžeme na celý prvek aplikovat nějaký styl. Jako ukázku můžeme použít rozmazání. Do hodnoty dáme funkci blur s hodnotou, o kolik se má prvek rozmazat. Hodnot existuje velmi mnoho, veškeré naleznete ZDE.

```
.blur { /* just visually! user can still copy the text and informations */
    filter: blur(20px) grayscale(1);
}
```

Transformace

Vlastností `transform` slouží k různým posunům a obecně úpravám prvků. Můžeme tedy prvky natáčet, zvětšovat, naklonit a podobné. Jednotlivé hodnoty můžeme spojovat a aktivovat zároveň. Veškeré hodnoty naleznete ZDE.

```
.item.transformed-2 {
    transform: rotate(1deg) scale(0.4) skew(30deg) scaleY(4);
}
```

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

MEZERY

V rozložení často potřebujeme, aby jednotlivé prvky měly různé mezery mezi sebou. V CSS máme proto k dispozici vlastnost `gap`, které v rozložení flexbox (a nám neznámem gridu) dovoluje nastavit mezery mezi prvky jak v hlavní tak i vedlejší ose. Dovnitř můžeme dát jakoukoliv hodnotu a jednotku.

Mimo těchto vlastností můžeme mezery na jednotlivých osách ovlivňovat rozdělně a to díky vlastnosti `row-gap` a `column-gap`.

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

FORMÁTOVÁNÍ KÓDU

Pro jakékoliv programy a soubory v informačních technologiích je důležité, aby nám psaný kód či program byl zcela čitelný i pro ostatní lidi. Správný kód, syntaxe a návyky si může však určovat každá skupina lidí (open-source, práce, projekty, ...) určovat sama a je důležité dodržovat jejich pravidla. V zásadě však platí níže uvedené.

Každý soubor by měl použít pouze jeden typ odsazení a to buď mezery (většinou 4 na tab) a nebo jenom taby. Tímto způsobem by měl být celý soubor odsazen. K formátování můžeme používat např. vestavěnou funkci VSCode F1 > Format document a nebo si nainstalovat jiný plugin.

Pro WBA platí, že by třídy a identifikátory měly být anglicky, krátce a výstižně. Podporují se různé formáty, nejčastěji se však používá camelCase, snake_case a PascalCase. HTML by mělo být tak, aby bylo velmi sémanticky čitelné a HTML by se nemělo ohýbat kvůli formátování v CSS. V CSS používáme co nejvíce volné selektory, které půjdou dobře editovat. Např. tedy neděláme třídu pro každou věc, ale používáme vnořování a další selektory.

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

POMOCNÉ VLASTNOSTI PRO FLEXBOX

Pořadí prvků

Každému prvku ve flexboxu můžeme nastavit pořadí, v jakém se vykreslí nehledě na to, kdy byl definován v HTML. Čím větší hodnota, tím "později" budou. Vlastnost se jmenuje `order`. Pokud mají prvky stejnou hodnotu, tak má přednost prvek takový, který byl definovaný jako první (tj. první až posledně definovaný).

```
.first {
    order: -1;
}
```

Zarovnání na sekundární ose

Každý prvek v flexu poslouchá svého rodiče k tomu, jak se má na sekundární ose zarovnat pomocí vlastnosti align-items. Každému prvku však můžeme říct, aby toto nastavení ignoroval a pozicoval se dle sebe. Vlastnost se jmenuje `align-self` a hodnoty má stejné jako jeho ekvivalent v rodiči.

```
.special {
    align-self: center;
}
```

Zvětšení/zmenšení prvku

Každý prvek se ve flexu zvětšuje a zmenšuje stejně dle toho, co mu určí rodič. Tuto vlastnost zvětšení a zmenšení můžeme nastavovat každému prvku rozdílné a to pomocí vlastnosti `flex-grow` a `flex-shrink`. První určuje jako moc (kolikrát více) se bude moci prvek zvětšovat a druhý jak zmenšovat. Základní hodnotou je hodnota 1. Hodnota 0 říká, že se prvek nesmí zvětšovat a/nebo zmenšovat.

```
.donottouch {
    flex-grow: 0;
    flex-shrink: 0;
}
```

Základní velikost

Často potřebujeme ve flexu nastavovat velikost před tím, než na ni "někdo šáhne". Na to používáme vlastnost `flex-basis`. Vlastnost taktéž určuje automaticky, zda se nastavuje velikost na horizontální či vertikální ose. Flexbox se při rozdělování místa podívá na tuhle vlastnost a pomocné `flex-grow` a `flex-shrink`. Hodnota 0 na `flex-basis` znamená nejmenší možnou velikost.

```
.largeboi {
    flex-basis: 1000px;
}
```

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.