Standardising the approach to CSS

Hlavní důvody

- Nejednotnost v pojmenování jednotlivých částí komponenty, která zapříčiňuje komplikovanou orientaci v kódu
- Chybějící metodologie pro návrh komponent
- Mnohonásobné zanořování a vysoká specifičnost selektorů
- Dědičnost proměnných, které jsou na sebe navázány a ovlivňují ostatní komponenty
- Slučování layoutu s komponentami
- Rozšiřitelnost a udržitelnost
- Velké množství „ohnutých“ komponent pro účel zjednodušení

Implementace

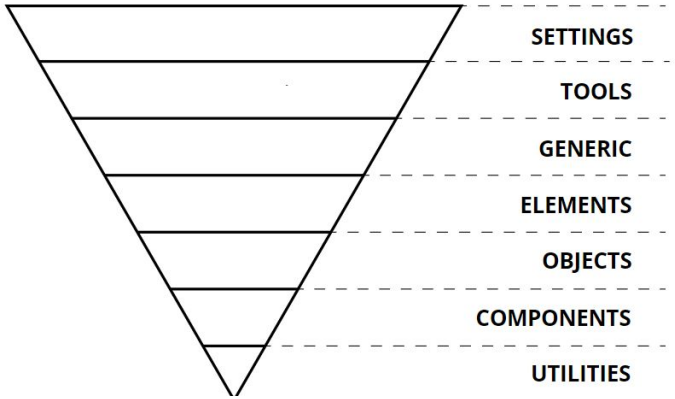
- Implementace CSS architektury ITCSS
- Převod komponent do BEM
- Redukce globálních proměnných, které jsou vázané na ostatní a umístění do jednotlivých stylů komponenty
- Případná restrukturalizace jednotlivých komponent v rámci převodu komponent do BEM a úprava logiky, pokud to bude nutné
- Sjednocení obsahu kódu stylů s popisky jednotlivých bloků

ITCSS

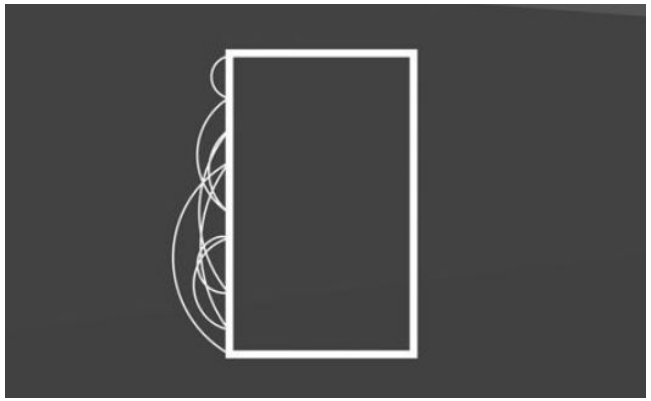
Popis architektury:
https://www.creativebloq.com/web-design/manage-large-css-projects-itcss-101517528

Příklad použití:
https://gist.github.com/iamryanyu/3bf1a3c4d23cbe4e407c1dd95358d9b6

Popis jednotlivých struktur:
https://www.hweaver.com/using-itcss-for-optimized-css-performance/



Hlavní myšlenka této struktury spočívá v tom, že se CSS codebase rozdělí do několika částí a styly se zapisují od obecných po ty nejvíc specifické.



Každá vrstva může obsahovat jen konkrétní selektory a to zabrání vzniku problémů s přetěžování.

ITCSS – výhody

- Správné zařazení souborů do předem definovaných úrovní nám zajistí, že se vyhneme vysoké specifičnosti selektorů a předejdeme tak řešení konfliktů https://stuffandnonsense.co.uk/archives/css_specificity_wars.html
- Přehled a orientace v kódu - vrstvy můžeme libovolně přidávat nebo ubírat podle potřeb. Vždy ale budeme mít přehled o tom, co se v nich nachází. Naše codebase bude čistá, systematická a přehledná
- Závislost - vrstvy nejsou na sobě závislé a můžeme s nimi libovolně manipulovat
- Kombinace s jinými metodikami - tento přístup můžeme sloučit s dalšími metodikami nebo s vlastním hybridním konceptem (BEMIT, CSS Modules, SUITCSS, OOCSS a další).
- Škálovatelnost - píšeme CSS v pořadí podle specifičnosti od nízké k vysokým. Specifičnost se pomalu zvyšuje podle konkrétních potřeb a mnohem snadněji tak můžeme škálovat naše CSS soubory.
- Přehlednost, udržitelnost, rozšiřitelnost, orientace – tato struktura a správné zařazení částí kódu do jednotlivých vrstev nám umožní vytvářet smyslupnou a udržitelnou codebase, ve které se bude snadné orientovat i po stálém rozšiřování
- Zmenšení velikosti výsledného minifikovaného souboru

01 - Settings

Globální nastavení projektu. Mělo by zde být umístění jen globálních proměnných systematicky rozdělených do více vrstev (**colors.scss**, **headings.scss** apod.). Jedná se o velikosti písma, barevné palety nebo konfigurace apod.

- settings.scss
- colors.scss
- typography.scss
- animation.scss

02 - Tools

Sem patří hlavně **mixiny** a **funkce**. Tato vrstva je umístěna za vrstvou Settings, protože může využívat některé definice z předchozí vrstvy. Privátní mixiny u jednotlivých komponent (pokud takové budou) můžeme držet v jednotlivých souborech u komponent, ale ty globální by se udržovali tady.

- functions.scss
- placeholders.scss
- mixins.scss
- media-queries.scss

03 - Generic

Tato vrstva je první, která generuje nějaké CSS. Měla by obsahovat high-level a dalekosáhlé styly. Nebude se skoro vůbec editovat. Typicky to je **Normalize.css**, **Reset.css** nebo globální pravidla pro určování velikostí boxů box-sizing apod.

Taky ovlivňuje mnoho DOM prvků a proto je nastavená v hierarchii takto vysoko. Můžeme pak třídami přepisovat již definované třídy ve vyšší úrovni, pokud je to třeba. Např. ***** selector, s definicí **box-sizing: border-box;**

- reset.scss
- normalize.scss
- bootstrap.scss
- fancybox.scss

04 – Elements (Base)

Jedná se o čísté a nezařazené prvnky DOM. Typicky se jedná o seznamy nadpisů **<h1>**, anchorů **<a>** a dalším HTML značek. Je něco explicitnější než předchozí vrstva a nachází se v ní bližší určení nastavení prvků - jakou mají nadpisy velikost nebo barvu.

Je to stále nízko specifická vrstva, ale ovlivňuje o něco méně DOM. Tato vrstva je poslední, ve které bychom našli holou definici HTML. Samostatné úpravy těchto prvků bychom měli pak upravovat jedině pomocí tříd.

- h1
- a
- article

05 - Objects

V této vrstvě najdeme první definice pomocí tříd. V této vrstvě se objevují selektory od základních jako **.container** element, přes selektory layoutu **.grid** až po media objects. Má vyšší specifičnost a je o něco explicitnější v tom, že cílíme na sekce DOM třídami.

Měli bychom sem dávat převážně definice **containers**, **rows**, **grids**, **colums** atd. Jedná se teda převážně o "non-cosmetic structural design patterns". Patří sem tedy spíše návrhové vzory (design patterns), které ovlivňují spíš rozvržení než vizuál.

- grid.scss
- content.scss
- .container
- .grid
- .row
- Spacers, Pushes/Pullers

06 - Components

Do této vrstvy patří jednotlivé rozpoznatelné komponenty UI. Tato vrstva je explicitnější než ta předchozí. Jedná se části UI od těch nejmenší až po ty velké.

Jedná se komponenty, které jsou rozšiřitelné a měly by fungovat kdekoliv napříč webem. Komponenty by měli být psány mobile-first a uvnitř komponent (což platí u všech vrstev) by se měli používat mediaqueries.

- Accordions
- Search-form
- Button
- Breadcrumbs
- Tooltip

07 - Patterns

V této vrstvě bychom mohli uchovávat větší komponenty, které bychom si pojmenovali patterny. Jednalo by se o kolekce „komponent“ jako jsou vetší vizuální bloky.

- Header
- Footer
- IamPatternMoreThanComponent

08 – Utilities (Trumps)

Tato vrstva přebíjí všechny předchozí vrstvy a dokáže přebít všechno, co je předtím. Tato třída obsahuje utility classes, help classes, hacks a overrides. Mnoho deklarací bude mít **!important** a jedná se o vrstvu s nejvyšší specifičností.

Využití této vrstvy by mělo být pro případy, když potřebujeme udělat změny pouze na určitém místě - např. zvětšit na jednom místě velikost nadpisu u komponenty, která má jinak všude definován nadpis menší.

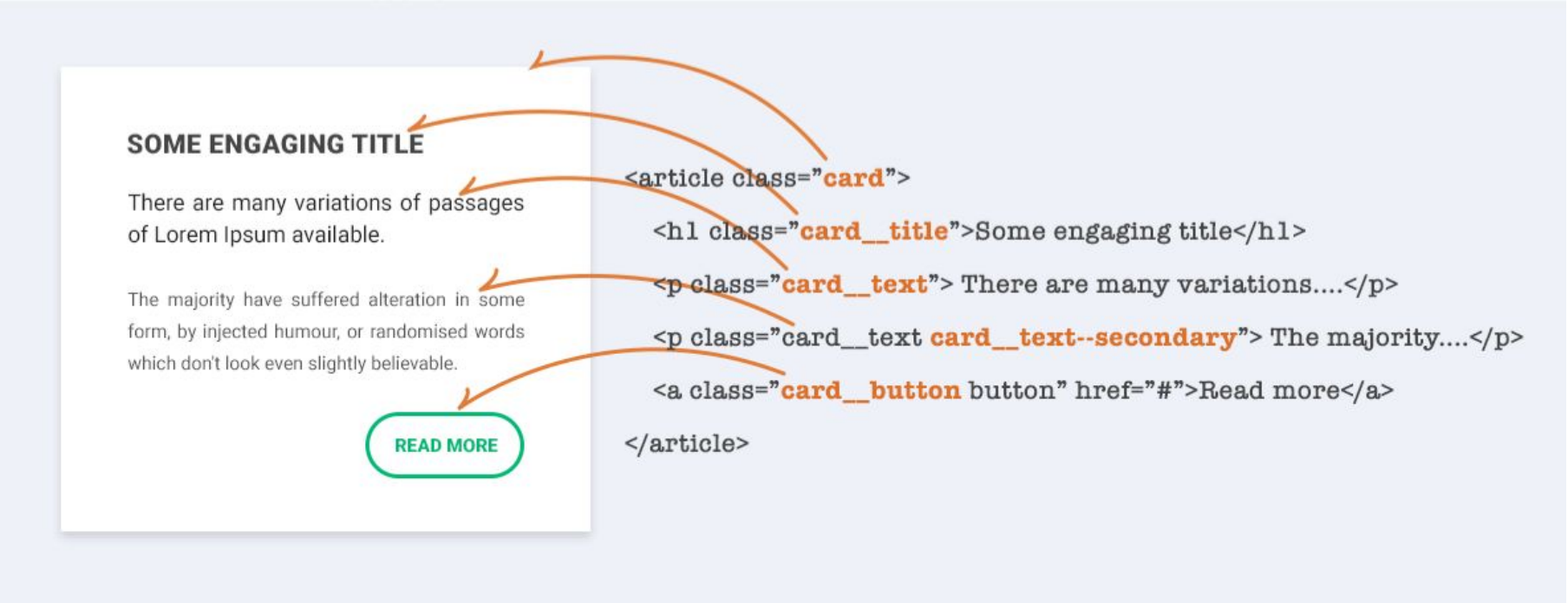
- helpers.scss
- utilities.scss
- overrides.scss

BEM

Jedná se o naming convency převzatou z plnohodnotné metodiky a organizace CSS.

Oficiální stránka:
http://getbem.com/naming/

Stručné vysvětlení:
https://codeburst.io/understanding-css-bem-naming-convention-a8cca116d252



BEM - výhody

- Rozhodovat se o tom, jak přistoupíme k pojmenování komponent bychom měli vždy na základé designu a vždy pomůže prvek probrat s UX týmem a sjednotit si tak názor s nimi.
- Pokud budeme chtít rozšířit nový styl pro danou komponentu, na první pohled je zřejmé jaké modifikátory a elementy už existují.
- Pokud čteme CSS, měli bychom získat rychlou představu o tom, který prvek závisí na jiném.
- Designéři a vývojáři mohou důsledně pojmenovat komponenty pro snazší komunikaci mezi členy týmu. Prostě dává všem zúčastněním syntaxi, kterou mohou sdílet a být nalazeni na stejné myšlení.

BEM - výhody

- Větší jistota vývojářů/kodérů – jedním z velkých problémů je, že se codebase zvětšuje a je plná kódu, který se už dávno nepoužívá. Kodéři se bojí něco upravit nebo předělat ze strachu z toho, že to bude mít negativní efekt napříč celým webem.
- Pokud budeme držet 1:1 s design systémem, vyhneme se nadměrnému naboptnání codebase a budeme mít přehled o tom co a jak funguje. Taky nebudou v kódu zastaralé a neznámé CSS, což je největší problém kaskád.
- Ne vždy jsme schopni napsat 100% předvídatelný kód ale pomocí BEM se dají pochopit kompromisy, které provedeme.
- Standard pro psaní smysluplnějších, dobře udržitelných a lehce rozšiřitelných komponent.

BEM- zanořování

Zanořovat maximálně do druhé úrovně – tzn. pokud se nabízí úroveň třetí, pokusit se nahlédnout na komponentu z jiného úhlu a sestavit ji tak, abychom se nedostávali nad tuto úroveň.

Nejlépe však zanořovat jen tak, aby bylo na první pohled zřejmé **(&:hover**, **&:focus**, **&::after**). Jednotlivé elementy však vždy psát na nový samostatný řádek pod definici blocku.

Učelem omezení této techniky je vyhnout se Nesting HELL:
https://gist.github.com/strann/9900620