IT doupě

# Jak se tvoří web – TypeScript, Sass a CSS moduly

Minulý měsíc jsme úspěšně přidali React a díky JSX se tak zbavili HTML. Stejně jako v Argentině se ale naše motorová pila stále nezastavila, a proto dnes skrouhneme další dvě technologie – JavaScript a CSS – aby se dal projekt do budoucna rozšiřovat ještě lépe než doposud.

Proč se ale vlastně chceme zbavit JavaScriptu? Dám vám příklad. Už se vám někdy stalo, že jste komponentě přidali propu, která měla jiný typ, než který komponenta požadovala? Ne? Ale jistě jste programovali v Pythonu a pak hodinu hledali, proč vám program padá. A chyba nejspíš měla něco společného s dynamickým typováním proměnných. Přidali jste řetězec do čísla, neiniciovali hodnotu, a tak podobně.

Pokud ale použijete TypeScript, bez otypovaných proměnných se vám program odmítne poskládat. Až když je přidáte, proběhne kontrola a vy můžete mít téměř stoprocentní jistotu, že se s podobnou chybou už nesetkáte.

TypeScript toho ale umí víc, než jen kontrolovat správné typy. Třeba doteď jste museli všechny argumenty pro komponenty vypisovat na řádek, což přestane být zábavné, až jich bude víc než tři. Nyní je můžete shlukovat do interfaců, a z těch dále dědit nebo je používat na více místech zároveň. Také můžete využít enum (seznam s přednastavenými hodnotami) a naplno používat objektově orientované programování díky dekorátorům a chráněným nebo abstraktním funkcím.

Teď se zbavíme toho style.css souboru v rootu stránky. Jakmile ho ale rozkouskujeme tak, že každá komponenta má svůj styl u sebe, narážíme na problém. Názvy tříd nejsou lokální. Pokud pojmenujete třídu v jedné komponentě layout, bude ovlivňovat jinou třídu layout definovanou v úplně jiné komponentě v úplně jiné části stránky. Abychom nemuseli dokola vymýšlet unikátní jména, přidáme do webpacku CSS moduly, které při buildu všechny názvy vezmou a přejmenují. Schválně se podívejte třeba na Instagram a zmáčkněte F12, i tam místo normálních názvů uvidíte jen shluk náhodných znaků. A kromě lokálních jmen jste právě spoustě web scraperům znepříjemnili život, protože bez konzistentních názvů tříd se na stránkách špatně sbírají data.

A když už jsme v tom, rovnou můžeme místo CSS začít používat Sass – konkrétně jeho podtyp s příponou .scss. Má úplně stejnou syntax jako CSS, ale s několika vychytávkami. Například podporují proměnné. Ty hned můžeme využít při standardizaci celého webu, například se dají použít pro vlastní barvy nebo velikosti fontu, které pak můžeme importovat z jednoho místa. Pokud chceme recyklovat více řádků stylů najednou, můžeme je zabalit do mixinu – takové Sass obdoby funkcí, dokonce do něj můžeme vkládat i parametry. A v neposlední řadě dokáže zpracovávat podmínky a cykly. Při buildu se pak všechny styly převedou zpět na CSS, aby je prohlížeč uměl číst.

A aby kód po těchto změnách přečetl nejen prohlížeč, ale i my, naučím vás ještě používat ESLint. Jedná se o knihovnu, kterou po propojení s IDE můžete používat na formátování kódu. Podle pravidel, které jí nastavíte v konfiguračním souboru, vás může upozornit na nepoužívané proměnné, chyby v kódu, a jedním kliknutím opraví všechna odsazení řádků.

A to je vše. Kód opět najdete u mě na GitHubu (<https://github.com/dlabaja/IT_doupe>), a pokud to ten jeden jediný čtenář dočetl až sem, tak se dozví, že příště zprovozníme Storybook pro lepší vývoj a testování komponent. Tak zase za měsíc, Informatice Zdar!

Jan Dlabaja, 4L