

Střední průmyslová škola strojní   
a elektrotechnická a Vyšší odborná škola, Liberec 1, Masarykova 3

Možnosti stylování v React

Maturitní práce

Autor **Štěpán Klein**

Obor **Technické lyceum**

Vedoucí práce **Ing. Tomáš Kazda, DiS.**

Školní rok **2021/2022**

Anotace

Práce se zabývá tvorbou dokumentace procesu tvorby uživatelského rozhraní v aplikaci na platformě React. Zaměřuje se na porovnání nejčastějších způsobů stylování. Tyto přístupy jsou prezentovány na čtyřech variantách té samé ukázkové aplikace.

Summary

The work deals with the creation of documentation of the user interface creation process in the application on the React platform. It focuses on comparing the most common ways of styling. These approaches are presented on four variants of the same sample application.

Čestné prohlášení

Prohlašuji, že jsem předkládanou maturitní práci vypracoval sám a uvedl jsem veškerou použitou literaturu a bibliografické citace.

V Liberci dne 24.02.2022

Štěpán Klein

Obsah

[Úvod 1](#_Toc96622721)

[1 React 2](#_Toc96622722)

[1.1 Komponenty 2](#_Toc96622723)

[1.1.1 Stav komponenty 2](#_Toc96622724)

[2 Stylování v React 3](#_Toc96622725)

[2.1 External css 3](#_Toc96622726)

[2.1.1 Styl psaní 3](#_Toc96622727)

[2.2 CSS Modules 4](#_Toc96622728)

[2.2.1 Styl psaní 4](#_Toc96622729)

[2.3 Styled Components 4](#_Toc96622730)

[2.3.1 Styl psaní 5](#_Toc96622731)

[3 Tvorba demo aplikace 6](#_Toc96622732)

[3.1 Návrh 6](#_Toc96622733)

[3.2 Stylování jednotlivých verzí 7](#_Toc96622734)

[3.2.1 External css 7](#_Toc96622735)

[3.2.2 Css modules 8](#_Toc96622736)

[3.2.3 Styled components 8](#_Toc96622737)

[4 SharePoint 9](#_Toc96622738)

[Závěr 10](#_Toc96622739)

[Seznam zkratek a odborných výrazů 11](#_Toc96622740)

[Seznam obrázků 12](#_Toc96622741)

[Použité zdroje 13](#_Toc96622742)

[A. Seznam přiložených souborů I](#_Toc96622743)

Úvod

V rámci této se soustředím na zdokumentování a předvedení stylování v React. Zaměřuji se hlavně na čtyři odlišné a široce používané metody. Sám jsem chtěl prohloubit své znalosti v práci s reactem. To se povedlo a díky dokumentaci ve které své poznatky prezentuji chci poskytnout jako další zdroj informací pro lidi kteří již mají nějáké základní zkušenosti s tvorbou webů. V této práci se věnuji jak teorii spojené se stylováním v React, tak i popisu tvorby samotné dokumentace a demo aplikací.

# React

React je JavaScriptová knihovna pro tvorbu uživatelského rozhraní webových aplikací. React mění strukturu psaní webových stránek html a JS spojuje v JSX a samotný web následně rozděluje do komponent.

## Komponenty

Existují dva hlavní typy React komponent. První je definována pomocí třídy, což využívá nativní objektové orientace jazyka JavaScript. Druhý rozšířenější způsob je komponenta definovaná pomocí funkce. Tato funkce má parametr „props“ a vrací html kód který se bude renderovat uživateli. Ve zbytku dokumentu budu referovat hlavně na komponenty definovane funkcemi, ale principy spojené se stylováním jsou téměř stejné v obou případech.

### Stav komponenty

Stavem komponenty referujeme na určitý typ proměnných v komponentě. Pokud se některé z těchto proměnných změní hodnota ergo změní se stav komponenty, komponeta se znovu vyrendruje. Pokud chceme vytvořit tento typ proměnné ve komponentě využijeme React hook funkce useState. Hook useState přímá jako parametr výchzí hodnotu stavu a vrací pole se dvěma prvky. První prvek nese sučasnou hdnotu stavu a druhý je funkce která slouží ke změně stavu. Tato funkce přímá pouze jeden parametr a to novou hodnotu funkce.

const [variable, setVariable] = useState(0);

# Stylování v React

React je JavaScriptová knihovna pro tvorbu uživatelského rozhraní webových aplikací. Ve svém řešení ale tak troch zapomíná na css. Pro React neexistuje jednotný způsob jak přistoupit k stylování. Existuje však mnoho odělených proudů které tuto problematiku řeší zcela odlišnými způsoby.

## External css

Prvním způsobem, jak stylovat webovou aplikaci napsanou v React je vytvořit jeden nebo více externích css souboru které se naimportují přímo do app.js. A to takto „import './App.css';“ Samotná aplikace se pak stylujete jako běžná webová stránka. Na každý html tag se odkazuje převážně pomocí atributu class (v React classname) nejlépe s pomocí nějaké stylovací konvence jako například BEM (Block-Element-Modifier). Styl je tedy celý oddělený od samotné aplikace.

### Styl psaní

Při tomto způsobu stylování React aplikací se finální css soubory stávají velkými a často i nepřehlednými. Protože se veškeré styly aplikují globálně muže docházet k neočekávanému chování a chybám které je těžké najít. Proto je dobré při pojmenovávat třídy pomocí některé z široce používaných konvenčních metod jako je například bem.

#### BEM v React

Bem staví na OOCSS tedy odděluje styl od struktury, a proto by se mělo dbát na nízkou specifičnost selektorů. Rozděluje webovou stránku na bloky a elementy ze kterých se skládají. Syntaxe pojmenování tříd vypadá takto pro blok „.nazev\_bloku“ a takto pro element „.nazev\_bloku\_\_nazev\_elementu“. Při pojmenování elementu tedy vždy přidáme i název jeho rodičovského bloku oddělený dvěma podtržítky. Jak blok, tak element mohou mít i vlastní modifikátory které mění jejich chování. Název modifikátoru se přidává oddělený dvěma spojovníky. Například tedy „.nazev\_bloku\_\_nazev\_elementu—nazev\_modifikatoru“.

## CSS Modules

Další nativní možností, jak stylovat React aplikaci je pomocí css modules. Css modules nám umožnuje aplikovat styl pouze lokálně na komponenty v souboru na který je importován. React css modules se importují do každého souboru komponenty, na kterou se styl má aplikovat pomocí import, from statementu takto „import styles from "../styles/NazevSouboru.module.css";“. samotné styly se píší jako klasické css, ale do atributu classname v reactu odkazujeme na blok stylu pomocí „className={styles.nazevAtributuClass}“. tento zápis se ve finálním html interpretuje jako „NazevSouboru\_nazevAtributuClass\_\_hash“.

### Styl psaní

Abychom dokázali využít všech výhod které css modules nabízejí, je potřeba dodržovat pár základních pravidel. Soubory css, které jsou určeny požít jako modules se běžně pojmenovávají nazev.modules.css, aby bylo jasné že se nejedná o klasické css určené k aplikování globálně. Dále je vhodné využít komponentové struktury reactu a ke každému komponentu přiřadit právě jedem css module styl. Tak je potom jasné, jaký styl patří, k jakému komponentu. Při pojmenovávání class není potřeba dodržovat BEM vzhledem k tomu, že pro každý komponent máme vlastní soubor a lokální scope. problémy, které bem řeší nikdy nenastanou a je proto redundantní. Naopak při pojmenovávání nedoporučuji používat podtržítko pro čistotu zápisu ve konečné interpretaci css modulu.

## Styled Components

Trochu jiný přístup ke stylování React aplikací je takzvané „css in js“ které spojuje css se zbytkem React. Tento způsob není nativně podporovaný js ani reactem jako předchozí dva způsoby proto je potřeba využít externí knihovny která pomůže styl implementovat do aplikace tímto způsobem. Existuje mnoho různých knihoven. Já jsem vybral tu nejoblíbenější, a to Styled Components, které plně podporují veškeré funkce css.

Ve styled component místo vytváření klasických css souborů a aplikování stylu pomocí atributu classname, vytváříme úplně novou komponentu na základě jednoho html tagu. Této nové komponentě definujeme styl již v js. To nám dává možnost css zkombinovat s js a vytvořit tak dynamické chování stylu přímo v css kódu. Tak komponenta muže měnit styly na základě atributů a jejich hodnot, které se komponentě přiřadí. tímto způsobem se nahradí všechny tagy ke kterým chceme přiřadit styl.

Styled components přinášejí nový způsob zápisu. Nejprve se vytvoří nová konstanta, která bude držet komponentu. Samotná komponenta se tvoří pomocí styled.[HTML tag]`[klasické css deklarace]`;. Kde si určíme na základě jakého tagu chceme komponentu stavět a string s již klasickými css deklaracemi které se na komponentu aplikují. Pomocí ${} můžeme do tohoto stringu psát další js. Atributy zpřístupňujeme klasicky pomocí parametru props Css z funkcí se vrací pomocí „return css `[klasické css deklarace]`;“.

styled components můžeme exportovat importovat pracovat s nimi jako z běžnými react komponenty. To nám dává možnost vytvořit si Styled components které následně použijeme na více místech v kodu.

### Styl psaní

Styled components přinášejí mnoho řešení na problémy spojené se stylováním React aplikace ale také mnoho způsobů jak dělat věci špatně. Existuje pár obecně uznávaných příkladů jak na to.

Jedním z prvních pravidel je vyhnout se psaní selektorů do styled components a prokaždý element který chceme stylovat napsat vlastní styl. Tyto nastylované komponenty poté píšeme do stejného souboru jako React komponent který stylujeme. Pokud budeme chtít styled component použít ve více souborech(React komponentech) vytvoří se soubor styles.jsx ve kterém budou napsany pouze všechny tyto styly. Když se tyto styly importují, lze použít klasické import { nazevkomponenty } from './styles'; avšak obecně se preferuje import \* as S from './styles'; aby se styled components lépe odlyšily od React komponentů.

## Bootstrap

Bootstrap je knihovna stylů a komponent k vytvoření webových stránek bez nutnosti psaní vlastního css. Tato knihovna obsahuje vše od nastylovaných tlačítek, navbarů, tabulek, karet atd… . Samotný bootstrap se na web aplikuje pomocí tříd. Pro použití bootstrapu v reactu je možné využít knihovny Reactstrap, která staví na bootstrapu již předem vytvořené nastylované React komponenty. Ty dále můžeme nastavit pomocí atributů a nebo využít bootstrap napřímo pomocí classnames.

Všechny reactstrap komponenty a jejich použití naleznete na webové stránce <https://reactstrap.github.io/?path=/story/home-installation--page>

a dokumentaci k bootstrapu zde: <https://getbootstrap.com/docs/5.1/getting-started/introduction/>

# Tvorba demo aplikace

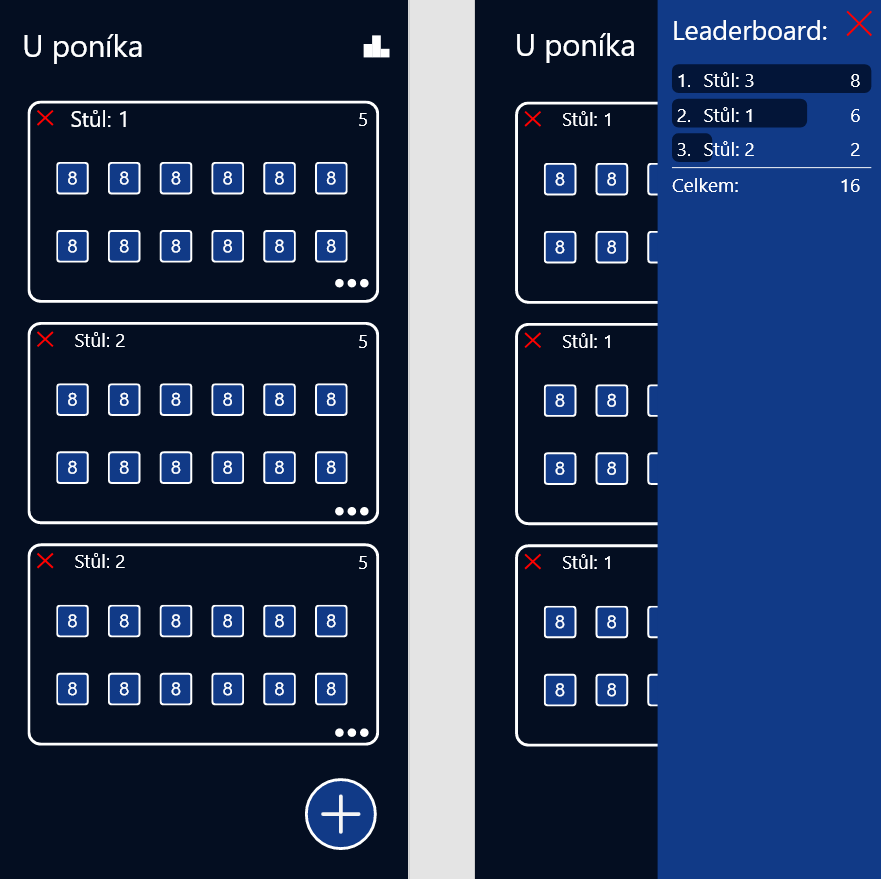
Pro ukázku stylovaní v praxi jsem si zvolil nastylovat již hotovou aplikaci která byla vyžívana při výuce na škole <https://github.com/TomasKazda/react-sandwich-feast>. Tato aplikace byla již nastylována pomocí knihovny Bootstrap. Vzhledem k tomu že aplikace již byla nastylovana jední ze zpusobů který v práci pokrývám a že její rozsah mi nedovoloval použít všechno co jsem potřeboval při stylování vyzkoušet, rozhodl jsem se k úpravám nad rámec stylování. A to konkrétně jsem nahradil komponent Navigation.jsx komponenty Leaderboard.jsx a Placement.jsx dále jsem upravil ostatní komponenty aby je byli v souladu s návrhem.

## Návrh

Při tvorbě návrhu jsem využil program Adobe XD. Při tvorbě návrhu jsem dbal na to aby se dalo využít co nejvíce různých vlastností css abych si udělal představu jak vhodně půjdou použít jednotlivé přístupy stylování v různých situacích.



Obrázek 1 Návrh designu pro desktop



Obrázek 2 návrh designu pro mobil

## Stylování jednotlivých verzí

### External css

Jako první jsem styloval aplikaci jako klasický web tedy pomocí externího css souboru s dodržením konvence bem.

I přestože demo aplikace, kterou jsem styloval byla velmi jednoduchá ke konci bylo někdy těžké se v celém css souboru vyznat. Bem se pro React velmi hodí a funguje v něm celkem intuitivně vzhledem k tomu, že se pracuje už tak se stránkou rozdělenou na komponenty. Interaktivní prvky se daly krásně zvládnout pomocí knihovny Classnames, ale někdy jsem byl nucen napsat i inline styl. Osobně si myslím že tento přístup je nejvhodnější při práci na velmi malých projektech vzhledem k tomu, že se nemusíte učit nic nového a dají se tak dobře zvládnout media queries. I tak bych doporučil alespoň CSS modules. Ukázka kodu:

return (

<div className='tables'>

<h1 className='tables\_\_h1'>{state.name}</h1>

<div className='tables\_\_grp'>

{tablesGrp}

</div>

<button className='tables\_\_btn' onClick={e =>

dispatch({type: ADD\_TABLE})

}>+</button>

</div>

);

A css

.table{

position: relative;

width: 375px;

height: 220px;

border: 3px solid rgb(255, 255, 255);

border-radius: 15px;

color: rgb(255, 255, 255);

margin: 10px auto;

}

.table\_\_head{

display: flex;

height: 32px;

justify-content: space-evenly;

margin-bottom: 20px;

margin-top: 5px;

}

### Css modules

Druho verzi jsem styloval pomoci css modules které řeší mnoho problémů vznikalé se stylováním pomocí externího css. Aplikují styl pouze na komponent, do kterého se importují a pracuje se především s menšími soubory. Obecně má tento přístup lepší škálovatelnost a nebál bych se ho použít i ve větších projektech. Vzhledem k tomu že se stále píše klasické css je velmi jednoduché si tento způsob stylování osvojit a pokud nechcete věnovat čas učit se knihovnu Styled Components jedná se určitě o způsob, jak stylovat React Aplikaci. Ukázka kodu:

return (

<div className={styles.tables}>

<h1 className={styles.h1}>{state.name}</h1>

<div className={styles.grp}>

{tablesGrp}

</div>

<button className={styles.btn} onClick={e =>

dispatch({type: ADD\_TABLE})

}>+</button>

</div>

);

A css

.tables{

margin: 0px;

margin-bottom: 80px;

margin-top: 20px;

}

.grp{

width: 100%;

}

### Styled components

Styled components je velmi zajímaví přístup k řešení celého problému stylování. Jedná se o nejpokročilejší metodu s uplně jinou filozofii a přístupem k problému. Pokud se člověk chce věnovat tvorbě aplikací v React myslím že se vyplatí tuto metodu naučit. Sc celkově navazuje na filozofii React a rozšiřuje ji i do css což se velmi hodí hlavně pokud chcete mít velmi interaktivní styly a nebo při práci na rozsáhlých projektech. Ukázka kodu:

const Rectangle2 = styled.rect`

width: 9.076px;

height:23px;

${(props) => {

if(props.cross){

return css`

transform: translate(388.882px, 20px) ;

fill: rgba(0,0,0,0);

`;

}

else{

return css`

transform: translate(388.882px, 40px);

fill: #fff;

`;

}

}}`;

### Bootstrap

Velmi vhodný pro tvorbu jednoduchých prototypů, nebo pro aplikace kde nepotřebujete originální stylování. Jedna se o knihovnu komponent to znamená že u výsledné aplikace člověk nemá kontrolu nad jejím výsledkem. Samozřejmě v některých případech je potřeba použít inline styl nebo trochu improvizovat, aby člověk docílil požadovaného vzhledu.

return (

<Container >

<h1 >{state.name}</h1>

<CardGroup>

{tablesGrp}

</CardGroup>

<Button onClick={e =>

dispatch({type: ADD\_TABLE})

}>+</Button>

</Container>

);

# SharePoint

Závěr

Tak jsem se dostal až na konec.

Seznam zkratek a odborných výrazů

HTML

HyperText Markup Language – značkovací jazyk používaný pro tvorbu webových stránek.

Seznam obrázků

[Obrázek 1 Návrh designu pro desktop 6](#_Toc96551900)

[Obrázek 1 návrh designu pro mobil 7](#_Toc96551901)

Použité zdroje

1. **Meta Platforms, Inc.** *React.* [Online] [Citace: 20. Únor 2022.] https://reactjs.org/.

2. **Wikimedia Foundation, Inc.,.** React (JavaScript library). *Wikipedia.* [Online] [Citace: 20. Únor 2022.] https://en.wikipedia.org/wiki/React\_(JavaScript\_library).

3. **Michálek, Martin.** BEM: Pojmenovávací konvence pro třídy v CSS. *Vzhůru dolů.* [Online] [Citace: 22. Únor 2022.] https://www.vzhurudolu.cz/prirucka/bem.

1. Seznam přiložených souborů