Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Porovnání vybraných moderních nástrojů pro tvorbu webových aplikací

Nikita Yarosh

Bakalářská práce
2022

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Nikita Yarosh**
Osobní číslo:	**I18211**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Porovnání vybraných moderních nástrojů pro tvorbu webových aplikací**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je porovnat vybrané modemí technologie pro vývoj webových aplikací. Porovnání bude provedeno na vybraných příkladech využívajících např. připojení pomocí Microsoft Graph (API pro přístup k Microsoft Cloud service resources). Pro autorizaci uživatelů bude využita technologie Microsoft Authentication Library (MSAL). Jedním z možných příkladů je přístup k uživatelskému kalendáři.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. Front-End Development Projects with Vue.js: Learn to build scalable web applications and dynamic user interfaces with Vue 2. Packt Publishing; 1st edition, 2020, 774 pp. ISBN 978-1838984823.
2. React Explained: Your Step-by-Step Guide to React. Independently published, 2019, 366 pp. ISBN 978-1798752982.

Vedoucí bakalářské práce: **doc. Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **17. prosince 2021**
Termín odevzdání bakalářské práce: **13. května 2022**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2022

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 11. 5. 2022

Nikita Yarosh

PODĚKOVÁNÍ

Chtěl bych podekovat vedoucímu mojí bakalářské práce doc. Ing. Michaelu Bažantu, Ph.D. za odbornou pomoc a poradenství při napsaní bakalářské práce. Také bych chtěl poděkovat Bc. Viktorii Savinove za pomoc s formátováním a dodržením stylu v dokumentu při psaní bakalářské práce.

ANOTACE

Cílem bakalářské práce je porovnat vybrané moderní technologie pro vývoj webových aplikací. Porovnávání bude probíhat mezi JavaScriptovým frameworkem Vue.js a knihovnou React a bude provedeno na vybraných příkladech využívajících připojení pomocí Microsoft Graph (API pro přístup k Microsoft Cloud service resources). Pro autorizaci uživatelů bude využita technologie Microsoft Authentication Library (MSAL). Jedním z možných příkladů je přístup k uživatelskému kalendáři, profilu atd. Teorie obsahuje popis použitých technologií, praktická část implementaci výše popsané úlohy. V závěru bakalářské práce je provedena analýza použitých nástrojů, jejich výhody, nevýhody a nejlepší případy využití.

KLÍČOVÁ SLOVA

MS Graph API, MSAL, HTML, CSS, SASS, SCSS, JavaScript, Vue.js, Vuex, Vue Router, React, Redux, React Redux, React Router, Redux Thunk, webová aplikace

TITLE

Comparison of selected modern tools for creating web applications.

ANNOTATION

The aim of the bachelor thesis is to compare selected modern technologies for web development. The comparison will be between JavaScript's framework the Vue.js and the React library and will be performed on selected examples using connections via Microsoft Graph (API for accessing Microsoft Cloud servíce resources). Microsoft authentication library (MSAL) technology will be used to authorize users in the system. One of the possible examples is accessing to the user's calendar, profile etc. The theory contains a description of the technologies which have been used, in turn, the practical part has the implementation of the task described above. The conclusion of the bachelor thesis is an analysis of the tools used, their advantages, disadvantages, and best use cases.

KEYWORDS

MS Graph API, MSAL, HTML, CSS, SASS, SCSS, JavaScript, Vue.js, Vuex, Vue Router, React, Redux, React Redux, React Router, Redux Thunk, web application

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

API	Application Programming Interface
CLI	Command Line Interface
CSS	Cascading Style Sheets
DOM	Document object model
DRY	Don’t repeat yourself
HTML	HyperText Markup Language
JS	JavaScript
JSON	JavaScript Object Notation
MS	Microsoft
MSAL	Microsoft Authentication Library
NPM	Node Package Manager
PHP	PHP: Hypertext Preprocessor
SASS	Syntactically Awesome Style Sheets
SCSS	Sassy CSS
SEO	Search Engine Optimization
SPA	Single-page application
URI	Uniform Resource Identifier

ÚVOD

Každý den více a více různých firem transformuje své obchody do online formátu, a proto je pro komerční použití nejdůležitějším úkolem automatizovat a urychlit vývoj svých produktů. V tom jim pomůže jedna ze trojice technologií určených pro tvorbu webových stránek používajících JavaScript (JavaScript v nativní podobě již delší dobu není používán). V dnešní době nelze si představit vývoj internetových stránek bez použití JavaScript frameworků, které můžou nabídnout široké spektrum různých instrumentů programátorovi: kratší kód, automatické připojení závislostí atd. Takových nástrojů existuje spousta, každý s vlastními plusy a minusy. Z toho důvodu autor bakalářské práce řešil výběr mezi dvěma nejpopulárnějšími JavaScript frameworky, přičemž pro výběr posloužila analýza z pohledu dostupných pracovních nabídek v České republice, počet dotazů na populární platformě Stack OverFlow, jednoduchost učení, výkon, srozumitelnost dokumentace atd.

Cílem teoretické části bakalářské práce je se seznámit s vybranými a podporujícími technologiemi pro tvorbu webových aplikací. Podrobnější popis lze najít na oficiálních stránkách každého frameworku[1,2].

Cílem praktické části je ukázka možností obou frameworků na základě webové aplikace na platformě Microsoft Outlook, která umožňuje autorizace pomocí MSAL a vytvoření událostí ve vlastním kalendáři, a proto pro využiti funkcionality je třeba mít zaregistrovaný uživatelsky účet u společnosti Microsoft.

[1] Dostupné z: https://vuejs.org/
[2] Dostupné z: https://reactjs.org/

1 POUŽITÉ ZÁKLADNÍ WEBOVÉ TECHNOLOGIE

Tato část bude obsahovat krátký pohled na použité technologie v praktické části bakalářské práce pro vytváření webových aplikací pomoci moderních nástrojů. Za základ byly vzaty Vue.js a React, které poskytují dynamiku na stránkách. CSS bude zodpovědný za stylistický design a HTML bude hlavním skeletem, nad kterým budou uvedené technologie použity. Pomocí JavaScript knihoven společnosti Microsoft bude zajištěno spojení s konkrétním rozhraním API.

1.1 HTML

HyperText Markup Language (HTML) dokument je obyčejný textový soubor, který může být vytvořen a editován jak v běžném textovém editoru, tak i ve specializovaném editoru s vhodnými tipy a zvýrazněním kódu (Notepad++, WebStorm, Visual Studio Code atd.). HTML dokument má příponu *.html* nebo *.htm* a skládá se ze stromu prvků HTML (tzv. tagů), do kterých je uvnitř uložen kontent. Tagy existují párové a jednotlivé. Příkladem jednotlivých tagů jsou např. značky `<link atributy>`, `<meta atributy>` nebo `<br>`. Každá položka párového tagu je v dokumentu označena počátečním (otvíracím) a koncovým (uzavíracím) tagem. Uzavírací značka je vytvořena přidáním speciálního symbolu „/“ (lomítko) před názvem druhého tagu: `<název tagu>…</název tagu>`. Mezi počátečním a uzavíracím tagem je obsah položky. Každá značka obsahuje také vlastní atributy, které mají vliv na zobrazení a chování HTML prvků. [1]

Příklad HTML kódu:

```
<!doctype html>
<html lang="cs">
<head>
    <meta charset="UTF-8">
    <title>Vlastní název</title>
</head>
<body>
    <p style="color: blue; text-decoration: underline; font-weight: bold;">
        Tento odstavec bude modrý, tučný a podtržený
    </p>
    <hr /><!--Toto je dlouhá čárka-->
    <a href="#odkaz_na_stranku">Odkaz</a>
    <!-- neuspořádaný seznam -->
    <ul>
        <li>1</li><li>2</li><li>3</li>
    </ul>
</body>
</html>
```

Zdroj: vlastní

HTML prvky uvnitř značky `<html>` tvoří strom dokumentu, tzv. objektový model dokumentu (DOM, document object model). Element `<html>` je kořenový prvek.

Sekce `<head>…</head>` obsahuje technické informace o stránce: titulek, popis, klíčová slova pro vyhledávače, kódování a tak dále. Uvedená data nebudou zobrazena v okně prohlížeče. Tato data určují prohlížeči, jak by měl zpracovat stránku.

Volitelnou součástí oddílu `<head>` je tag `<meta>`. Pomocí něj lze určit popis obsahu stránky a klíčová slova pro vyhledávače, autora dokumentu HTML a další vlastnosti metadat.

Položka `<head>` může obsahovat více položek `<meta>`, protože v závislosti na použitých atributech nesou různé informace. [1]

Párová značka `<title>` v HTML se používá k definování názvu dokumentu HTML a jeho hodnotu můžeme najít v titulku prohlížeče, při přepínání oken přes shortcut *Alt+Tab*, v názvu webové stránky, když je přidána do oblíbených pomocí *Ctrl+D* atd. Titulek je také užitečný pro prohlížeč např. při uložení stránky přes zkratku kláves *Ctrl+S* se objeví jako automaticky nabízené jméno souboru nebo, co je důležitější, pro SEO (search engine optimization). Název stránky také používají algoritmy vyhledávače k rozhodování o pořadí při výpisu stránek ve výsledcích vyhledávání. [2]

Uvnitř tagu `<body>` se nachází veškerý obsah dokumentu kam lze ukládat ostatní značky. Celý seznam tagů najdete na w3schools.com[3].

1.2 CSS

CSS (Cascading Style Sheets) vznikl někdy kolem roku 1997. Je to kolekce metod pro grafickou úpravu webových stránek. Kaskádové, protože se na sebe mohou vrstvit definice stylu, ale platí jenom ta poslední.

Styl se můžete deklarovat třemi způsoby. Lze používat jeden ze tří možných způsobů:

1. Přímo v textu zdroje u formátovaného elementu pomocí atributu `style="…"`. Tomu se říká přímý styl.

```
<p style="color: blue; text-decoration: underline; font-weight: bold;">
    Tento odstavec bude modrý, tučný a podtržený
</p>
```

[3] Dostupné z: https://www.w3schools.com/tags/default.asp

2. Pomocí „stylopisu“ (angl. „stylesheet“) kdekoliv v tělu značky `<head>` nebo `<body>`. Stylopis je jakýsi seznam stylů. Je v něm obecně napsáno, co má být a jak zformátováno. Do stránky se stylopis píše mezi párovým tagem `<style>`.

```
<style>
    p {
       color: blue;
       text-decoration: underline;
    }
</style>
```

3. Použitím externího stylopisu – to je soubor s příponou *.css*, který lze připojit tagem `<link>` v hlavičce *.html* souboru. V souboru je umístěný stylopis, který můžete nalinkovat na několika HTML stránkách, čím zabezpečíte zachování koncepce DRY (don’t repeat yourself).

```
<link rel="stylesheet" href="style.css">
```

Z výše uvedeného důvodu tento způsob je nejčastěji používán. [3]

CSS není HTML a má svou vlastní syntaxi (uvozovky, dvojtečky, složené závorky, středníky, mřížky a čárky). Místo tagů říkáme těmto prvkům selektory. Selektorů existuje několik druhů:

- *.class* – vybírá všechny elementy s uvedenou určitou třídou v atributech značky.
- *#id* – vybírá všechny elementy s uvedeným určitým identifikátorem v atributech značky.
- *element* – vybírá všechny tagy uvedeného elementu.

  Samostatně zvýrazníme znak „*“ (asterisk), protože vybírá všechny elementy. [4]

1.2.1 Preprocesory CSS

Syntactically Awesome Style Sheets (SASS) – je jazyk založený na CSS, který je navržen tak, aby zvýšil úroveň abstrakce CSS kódu a zjednodušil soubory kaskádových stylů. Oproti CSS SASS může pracovat s matematickými operacemi, podmínkami, cykly, mixiny (používá se aby nedocházelo k duplicitnímu opakování vlastností), poskytuje vytváření vlastních proměnných, což velmi pomáhá při dlouhodobé podpoře kódu. Soubory SASS mají příponu *.sass*, ale pro úspěšné využiti je nutné provést kompilaci do CSS a pak spojit zkompilovaný soubor s HTML přes tag `<link>`.

Existuje ještě i jiná varianta preprocesoru SASS a je známa pod zkratkou **SCSS** (Sassy CSS). Rozdíl je jenom v povinnosti využívat složené závorky a středníky na konci jako v obyčejném CSS. [5]

1.2.2 Práce s preprocesory ve Vue

Jako jednou z variant použití CSS v bakalářské práci byl zvolen preprocesor SCSS. Pro začátek práce s ním ve Vue nainstalujte jeho pomocí následujícího příkazu v terminálu:

```
npm install -D sass-loader sass
```

Poté v tagu `<style>` v komponentě lze použít bez žádných chyb atribut `lang="scss"` a psát kód s využitím syntaxe SCSS. [15]

Pozn.: při získání následující chyby „*TypeError: this.getOptions is not a function*“ v konzole zkuste následující řešení[4]: smažte **sass-loader** a stáhnete jej znovu s verzi **10.1.1**. [16]

Pro globální sdílení proměnných použijte následující řešení[5]:

1. Vytvořte nový soubor (jestliže Vue CLI neudělal to před tím) v kořenu projektu a pojmenujte jeho jako *vue.config.js*.
2. Vložte následující konfigurační kód uvnitř:

```
module.exports = {
    css: {
        loaderOptions: {
            scss: {
                additionalData: `
                @import "cesta_k_hlavnímu_stylovému_souboru";
                `
            }
        }
    }
}
```

Zdroj: [17]

3. Poté zastavte server a spusťte jej znovu, aby načetl novou konfiguraci ze souboru *vue.config.js*. [17]

[4] Dostupné z: https://stackoverflow.com/questions/66082397/typeerror-this-getoptions-is-not-a-function
[5] Dostupné z: https://stackoverflow.com/questions/54158207/how-do-i-add-a-global-scss-file-in-vue-js-that-will-compile

1.2.3 Práce s preprocesory v Reactu

Pro práci s preprocesory v Reactu není nutné k projektu instalovat žádné další prvky. Jen je vhodné používat takové IDE (Visual Studio Code, WebStorm atd.), které umí pracovat se soubory s příponou *.scss*, *.sass* atd.

1.3 JavaScript

JS (JavaScript) je programovací jazyk, který na naše stránky přidává interaktivitu a vlastní chování. Je to *client-side* skriptovací jazyk, což znamená, že běží na straně uživatele, a ne na serveru oproti jiným programovacím jazykům např. PHP nebo Ruby. To znamená, že JavaScript (a způsob, jakým jej používáme) závisí na možnostech a nastavení prohlížeče.

Jako dynamický programovací jazyk nemusí být JavaScript spuštěn prostřednictvím jakékoli formy kompilátoru, který interpretuje náš lidsky čitelný kód do něčeho, co prohlížeč může pochopit. Prohlížeč efektivně čte kód stejně jako my a interpretuje to za běhu.

JavaScript je *dynamically typed language*. To znamená, že nemusíte nutně říct JavaScriptu, jakého datového typu je proměnná. Pokud nastavujeme proměnnou na hodnotu 5, nemusíme programově specifikovat tuto proměnnou jako číslo; to znamená, že 5 je číslo a JavaScript jej jako takové rozpozná.

Nejčastěji se setkáme s JavaScriptem pro přidaní interaktivity do stránky. Zatímco „strukturální“ vrstva stránky je naše HTML značkování a „prezentační“ vrstva stránky je tvořena CSSem, třetí „behaviorální“ vrstvou se skládá z našeho JavaScriptu. Ke všem prvkům, atributům a textům na webové stránce lze přistupovat pomocí speciálních funkcí, které budou pracovat s DOM.

Stručně řečeno, JavaScript umožňuje vytvářet vysoce responsivní rozhraní, které zlepší uživatelský komfort a poskytne dynamické funkce bez čekání na načtení nové stránky ze serveru. [6]

Skript se zapisuje do HTML mezi párovou značkou `<script>`. Program v JavaScriptu se dá uložit i do jiného souboru a potom ho do stránky načítat přidáním atributu *src*, kde bude uvedena cesta k externímu soubory s příponou *.js*. Tím dodržíte koncepce DRY.

Pozor, pokud zkusíte jedním tagem `<script>` spustit externí i interní skripty, tak se to nepodaří. [7]

Poslední věc, která je důležitá v JS je deklarativní programování pomocí funkcí `filter()`, `map()`, `reduce()` atd. Stručně řečeno, existují imperativní a deklarativní přístupy programovaní v JS. V prvním případě je třeba uvést všechny kroky jak bude pracovat např.

funkce a v druhém jenom říct co potřebujete. JavaScript podporuje obě dvě varianty, ale v dnešní době první programovací přístup dokonale doplňuje ještě i deklarativní. Ukázka obou případů programování je uvedena na následujícím příkladu:

```
const numbers = Array.from({length: 10}, (v, k) => k + 1);
function filterImperative(array, divider) {
    let filteredResult = [];
    for (let i = 0; i < array.length; i++) {
        if (array[i] % divider === 0) {
            filteredResult.push(array[i]);
        }
    }
    return filteredResult;
}

function filterDeclarative(array, divider) {
    return array.filter(number => number % divider === 0);
}

console.log(filterImperative(numbers, 2));
console.log(filterDeclarative(numbers, 2));
```

Zdroj: vlastní

Při imperativním programování je třeba realizovat každý krok algoritmu: vytvořit nové pole, v cyklu provést ověření podmínky či je `array[i]` dělitelným na `divider` a pak vložit hodnotu v pole `filteredResult`. V druhém případě stačí jenom popsat co chcete a vás nezajímá kterým způsobem bude získán výsledek. [54]

1.4 Vue.js

Vue.js uvedl na trh stejnojmenná společnost v roce 2014. Díky svým vlastnostem je řazen mezi progresivní frameworky, což znamená, že umožňuje webovou aplikaci rozdělit na několik částí a ty pak vyvíjet nezávisle na sobě. Pokud tak chcete Vue použít v rámci již existující aplikace, nemusíte nutně znovu kódovat všechny stránky, případně komponenty aplikace.

Jedná se o univerzální a výkonné řešení, jež napomáhá udržování pořádku v kódu. Mezi jeho největší výhody patří jeho „čerstvost“. Pozdější uvedení na trh tvůrcům umožnilo začít s čistým štítem, poučit se z chyb či nedostatků předchozích řešení a reagovat na aktuální požadavky vývojářů i uživatelů.

Stejně jako React využívá model založený na reaktivních komponentech, který usnadňuje použití často opakovaných částí aplikace, jako je např. zápatí nebo navigační lišta. Toto se typicky využívá zejména u *single-page aplications* (SPA). Pokud chcete vytvořit nový komponent, tak Vue nabízí několik způsobů, ale nejčastěji použitelnou variantou je vytvoření

zvláštního souboru s příponou *.vue* a proto z toho důvodu další ukázky budou zaměřené na tento způsob. [8]

Základní tři části v komponentu Vue vypadají následovně:

```
<template>…</template>

<script>…</script>

<style>…</style><!-- Tato část není povinná -->
```

Zdroj: vlastní

Značka `<template>` slouží k popsaní vzhledu komponenty, kde se píše už známý HTML kód, `<script>` a `<style>`. To jsou sekce podobné stejnojmenným tagům v HTML - Vue je tedy přátelský pro ty, kteří již znají základy web programování. Pro ukázku níže je uveden jednoduchý příklad.

```
<template>
  <div>
    <p>{{ counter }}</p>
    <button class="button plus" v-on:click="increment">+</button>
    <button class="button minus" @click="counter--">-</button>
    <!--reprezentace jednořádkové funkce-->
  </div>
</template>

<script>
export default {
  name: "Test",
  data() {
    return {
      counter: 0
    }
  },
  methods: {
    increment() {
      this.counter++;
    }
  }
}
</script>

<!-- scoped slouží pro lokalizace stylu -->
<style scoped lang="scss">
$side : 30px;

.button {
  width  : $side;
  height : $side;
  color  : white;
  margin : 5px;

  &.plus {
    background-color : blue;
  }
```

```
  &.minus {
    background-color : red;
  }
}
</style>
```

Zdroj: vlastní

1.4.1 Direktivy Vue.js

Nyní se pojďme podívat na výše napsaný kód blíže. V jádru Vue.js je systém, který nám umožňuje deklarativně vykreslit data do DOM pomocí jednoduché syntaxe šablony. Jsou to dvojité složené závorky:

```
<p>{{ counter }}</p>
```

Obsah této značky bude reaktivně nahrazen při změně stavu proměnné `counter`, která byla deklarována ve funkci *data*.

Direktivy Vue jsou speciální atributy HTML se speciálním prefixem „`v-`“ , které nám umožňují manipulovat s DOM. Pro zpracování různých událostí Vue.js nabízí `v-on` direktivu:

```
<button class="button plus" v-on:click="increment">+</button>
```

Zde je předpona `v-`, což je výchozí. ID direktivy je `on` a výrazem je funkce `increment()`, která je popsaná v tělu objektu *methods*. Tato direktiva nařizuje Vue.js spustit přiřazenou funkci vždy, když bude stisknuto tlačítko `+`. Taky existuje možnost nahradit celou direktivu `v-on` pro kratší syntaxi zavináčem („@“, místo `v-on:click` – `@click`) nebo lze psát tělo metody přímo uvnitř závorek. [18]

Další důležitou direktivou pro vázaní javascriptových výpočtů s atributy tagů je `v-bind`.

```
<!-- Correct -->
<p v-bind:title="`counter: ${counter}`">{{ counter }}</p>
<!-- Incorrect -->
<p title="`counter: ${counter}`">{{ counter }}</p>
```

Zdroj: vlastní

Stejně jako s direktivou `v-on, v-bind` lze vynechat a nepsat se jej vůbec (místo `v-bind:název="…"` nechat jenom dvojtečku („`:`“)).

Další důležitou direktivou je `v-if`, která zabezpečuje zobrazení prvků podle stavu v závislosti na pravdivosti uvedeného výrazu. Při přepínání bude položka a všechny direktivy/komponenty v ní obsažené zničeny a znovu vytvořeny. Tato direktiva spouští

přechody, když se změní její stav. `v-else` a `v-else-if` fungují za stejným principem jako v libovolném jinem programovacím jazyce.

V případě práce se seznamy vám pomůže direktiva `v-for`. Opakovaně vykresluje prvek nebo blok šablony na základě zdrojových dat. Hodnota direktivy musí používat speciální syntaxi *alias* **in** *expression*, aby deklarovala proměnnou pro aktuální iterační prvek:

```
<ul>
  <li v-for="i in 10" :key="i">{{ i }}</li>
</ul>
```

Zdroj: vlastní

Vždy používejte klíč s `v-for`!

Klíč je povinný pro komponenty, aby se zachoval jeho vnitřní stav a stav podstromu. Dokonce i u prvků je to dobrá praxe pro udržení předvídatelného chování, jako je konzistence objektu v animaci.

Nepoužívejte `v-if` s `v-for`!

Nikdy nepoužívejte `v-if` na stejném prvku jako `v-for`. Když Vue zpracovává cyklus, `v-for` má vyšší prioritu než `v-if`. Takže když zobrazujeme položky pouze pro sudá čísla, měli bychom při každém opětovném vykreslení procházet celý seznam.

Špatně

```
<ul>
  <li v-for="i in 10" :key="i" v-if="i % 2 === 0">{{ i }}</li>
</ul>
```

Zdroj: vlastní

Správně

```
computed: {
  evenNumbers() {
    return this.numbers.filter(num => num % 2 === 0);
  }
}
```

Zdroj: vlastní

Tato možnost je zde uvedena v předstihu, neboť je nutné využít vlastnost Vue komponenty s klíčovým slovem *computed*. Na první pohled se může zdát, že vypočtená vlastnost je funkce, protože její tělo vypadá podobně, ale není to tak. Detailněji *computed* a ostatní vlastností budou rozebrány v následujících kapitolách.

```
<ul>
  <li v-for="i in evenNumbers" :key="i">{{ i }}</li>
</ul>
```

Zdroj: [20]

Plný seznam direktiv s popisem a příklady lze získat na oficiálních dokumentačních stránkách Vue.js v oddílu API reference, directives[6]. [19]

Pokud chcete celou aplikaci vyvíjet jen a pouze v jednom frameworku, Vue obsahuje celý ekosystém (jádro, Vuex, Vue-router, Vue CLI a další), o kterém budou uvedeny bližší informace v další části dokumentu. [9]

1.4.2 Z čeho se skládá komponent ve Vue.js?

Dříve jsme se zabývali některými složkami součásti ve Vue.js. Teď je na řadě čas se s nimi blíže seznámit.

Jak bylo uvedeno v úvodu ke Vue.js, nejpohodlnějším a nejoblíbenějším druhem vytváření komponent je *Single File Component*. Skládá se ze tří částí: šablona, skript, styl. Poslední část není povinná.

V tagu šablony stojí za to popsat skelet budoucí části webové aplikace pomocí pravidelného HTML kódu a v-direktiv, aby byla zajištěna reaktivita. Důležitou zmínkou je umístění pouze jednoho kořenového prvku HTML do tagu šablony.

Hlavním cílem skriptovací značky je exportovat náš komponent pro další použití. Na tomto místě lze psát libovolný validní JS kód.

Komponenty lze také globálně nebo lokálně (což se doporučuje v oficiální dokumentaci[7]) stylizovat pomocí CSS. Lokální použití vyžaduje pouze přidání vlastnosti `scoped` do značky stylů. Pokud chcete použít preprocesor CSS, musíte ho nejprve nainstalovat a poté přidat atribut `lang` s požadovanou hodnotou: *scss*, *sass*, *less* atd. [21]

Nejčastěji používanými prvky komponenty jsou: *name*, *data*, *props*, *computed*, *methods*, *filters* a *lifecycle hooks*. O všech ostatních částech si můžete přečíst v oficiální dokumentaci Vue v sekci API[8].

[6] Dostupné z: https://v3.vuejs.org/api/directives.html
[7] Dostupné z: https://vuejs.org/v2/style-guide/#Component-style-scoping-essential
[8] Dostupné z: https://vuejs.org/api/

Name

Povolí komponentu, aby se rekurzivně vyvolal ve své šabloně. Všimněte si, že když je komponent registrován globálně s `Vue.component()`, globální ID se automaticky nastaví jako jeho název.

Další výhodou zadání možnosti názvu je debugging. Pojmenované komponenty vedou k užitečnějším varovným zprávám. Také při kontrole aplikace ve *vue-devtools* se nejmenované komponenty zobrazí jako `<AnonymousComponent>` a proto poskytnutím možnosti *name* získáte mnohem informativnější strom komponent. [22]

Data

Při použití vlastnosti *data* v komponentu ona musí být hodnotou funkce, která vrací objekt (je to syntaktická podmínka frameworku Vue.js), jehož vlastnostmi jsou jednotlivé proměnné, které potřebujeme zobrazit v šabloně nebo dále s nimi pracovat v metodách daného komponentu. Více detailů k vysvětlení proč je to tak najdete v oficiální dokumentaci v oddílu Style Guides, Rule A, Component data[9].

Props

Props je speciální klíčové slovo, které je zkratkou pro *properties* (česky – vlastností). Může být registrován v komponentu pro předání dat z rodičovského komponentu do jedné z podřízených komponent. Data v *props* mohou posílat údaje pouze v jednom směru – z nadřazeného komponentu do podřízeného komponentu. To jednoduše znamená, že nemůžete předat data z dítěte rodiči. Další věc, kterou je nutné mít na paměti je, že *props* jsou pouze pro čtení a nemůžou být změněny dítětem, protože rodičovský komponent vlastní hodnotu.

Podle Vue.js Style Guides[10] ve finálním kódu definice *props* by měly být vždy co nejpodrobnější a specifikovat alespoň datový typ. Komponenty taky mohou specifikovat další požadavky na své *props*, například výchozí hodnotu. Detailnějším nastavením si pomůžete při vývoji, protože Vue vás upozorní v konzoli prohlížeče, pokud komponent někdy poskytnul nesprávná *props*, čímž se lze vyhnout potenciálním chybám. To je zvláště užitečné při vývoji komponentů, které jsou určené k použití ostatními.

Platné typy, které můžete použít, jsou: String, Number, Boolean, Array, Object, Date, Function, Symbol.

```
export default {
  props: ['propA', 'propB'], //Bad
  props: {
```

[9] Dostupné z: https://vuejs.org/v2/style-guide/#Component-data-essential
[10] Dostupné z: https://vuejs.org/v2/style-guide/#Prop-definitions-essential

```
    propA: Number, //Better way to define props
    propB: String,
  },
  props: {
    propA: { //Perfect!
      type: Number,
      required: true,
      default: 100
    }
  }
}
```

Zdroj: [23]

A poslední věc, kterou rozebereme a bez čeho se neobejit při práci s komponenty, je předání dat přes *props* ve Vue. Hodnotu můžete poslat jako datovou vlastnost pomocí `v-bind`:

```
<ComponentName :propA="10" :propB="'Value'"/>
```

Zdroj: [25]

Nebo jako statickou hodnotu:

```
<ComponentName propA="10" propB="Value"/>
```

Zdroj: [25]

[23], [24], [25]

Computed

Další je vlastnost konfiguračního objektu s názvem *computed*. Jejím obsahem musí být objekt obsahující pouze „metody“. Každá tato „metoda“ musí povinně vracet hodnotu a může pracovat s *data* nebo *props* daného komponentu. Vypadají podobně metodám, ale to je opravdu hodnota. K té lze přistupovat pomocí klíčového slova `this`, které v tomto případě referuje samotnou komponentu. [26]

Proč nepoužít metodu? Namísto vypočtené vlastnosti můžete definovat stejnou funkci. Pro konečný výsledek jsou tyto dva přístupy skutečně úplně stejné. Rozdíl je však v tom, že vypočtené vlastnosti jsou ukládány do mezipaměti na základě jejich reaktivních závislostí. Vypočítaná vlastnost se přehodnotí pouze tehdy, když se některé z jejích reaktivních závislostí změnily. [27]

Níže je uvedena ukázka *computed* vlastnosti z komponentu *Calendar* z praktické části bakalářské práce. `isHidden()` bude vracet vždy na začátku *true*, pokud proměnná `this.currentOption` nebude změněná. Pak Vue provede rerender všech komponentů závislých na této proměnné.

```
computed: {
  isHidden() {
    return this.currentOption === CalendarTable.name;
  }
}
```

Zdroj: vlastní

Methods

Obyčejná metoda v JS při volání, kdy se provede její tělo. K těmto metodám můžete přistupovat přímo na instanci Vue nebo je použít v *directive expressions*. Všechny metody budou mít jejich `this` kontext automaticky vázán na instanci Vue. Všimněte si, že byste neměli používat *arrow functions* k definování metody (např. `plus: () => this.a++`). Důvodem je, že takové funkce váží nadřazený kontext, takže to nebude instance Vue, jak očekáváte, a `this.a` bude `undefined`. [28]

Filters

Vue.js umožňuje definovat filtry, které lze použít k běžnému formátování textu. Filtry jsou použitelné na dvou místech: *mustache interpolations* a *`v-bind` expressions*. Filtry by měly být připojeny ke konci výrazu JavaScript, označeného symbolem *pipe* („|“). [29]

Filtry ve Vue nemění data přímo tam, kde je ukládáte, vztahuje se pouze na formátování našich dat. Data zůstávají stejná, změní se pouze výstup dat do prohlížeče. Vue.js tyto filtry ve výchozím nastavení nedává, takže je musíme vytvořit. S Vue.js můžeme použít filtry dvěma různými způsoby, tj. **globální** filtr a **lokální** filtr. Globální filtr poskytuje přístup pro všechny složky, zatímco lokální filtr nám pouze umožňuje použít náš filtr uvnitř komponenty kde byl definován. [30]

```
<EventsDetailField :field-name="'Attendees:'"
                   :field-data="event.attendees | trimAttendees"/>
```

Zdroj: vlastní

Níže je uvedena demonstrace lokálního filtru z komponenty *EventsDetail* v praktické ukázce.

```
filters: {
  trimAttendees: value => {
    if (value.length) {
      let attendees = [];
      for (let i = 0; i < value.length; i++) {
        attendees.push(value[i].emailAddress.name);
      }
      return attendees.join(", ");
```

```
    }
    return '-';
  }
}
```

Zdroj: vlastní

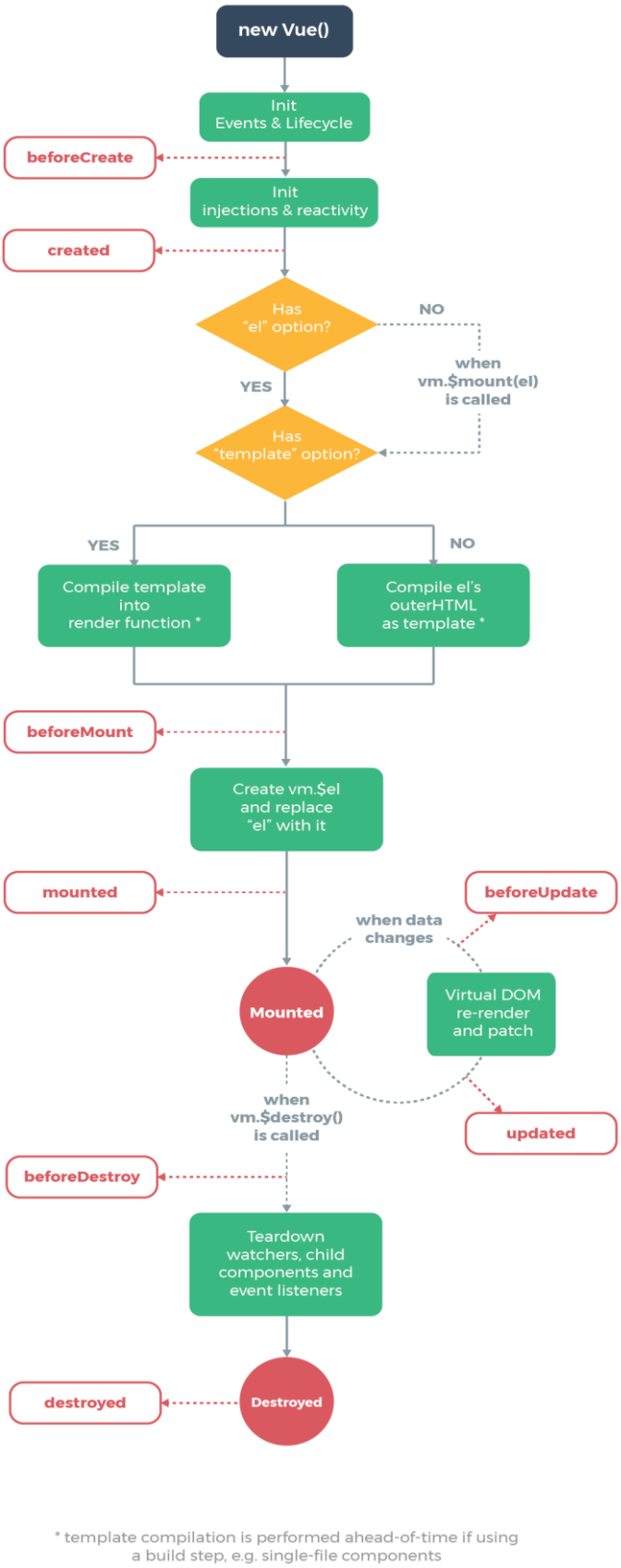
Pro globální vytvoření filtru potřebujete použít *Global API* v hlavním souboru (typicky je to *main.js*) `Vue.filter()`, kam dáte prvním argumentem název filtru a druhým – funkce. Potom můžete používat jeho v libovolném komponentu přes svislou čáru.

```
Vue.filter("capitalizer", value =>
    value ? value.replace(/\b\w/g, character => character.toUpperCase()) : "");
```

Zdroj: vlastní

Lifecycle hooks

Každá instance Vue při vytváření prochází posloupnost kroků inicializace – například nastaví sledování dat, zkompiluje šablonu, připojí kopii v DOM, aktualizuje DOM při změně údajů. Mezi těmito kroky jsou volány funkce nazývané hooky životního cyklu, pomocí kterých můžete v určitých fázích provádět svůj kód. [31]



Obrázek 1: schéma životního cyklu Vue komponentu. Zdroj: [31]

1.4.3 Vue CLI

Vue CLI si klade za cíl být standardním nástrojovým základem pro ekosystém Vue. Zajišťuje, že různé nástroje pro sestavení fungují hladce společně s rozumnými výchozími hodnotami, takže se můžete soustředit na psaní své aplikace, místo abyste trávili dny nastavením a konfigurací. Současně stále nabízí flexibilitu pro vyladění konfigurace každého nástroje.

CLI (@vue/cli) je globálně nainstalovaný balíček npm a poskytuje příkaz `vue` ve vašem terminálu. Taky nabízí možnost rychle vytvořit nový předkonfigurovaný projekt (ale při generování lze vybírat co přesně chcete vidět ve svém projektu) prostřednictvím `vue create` *`název`*. Projekty lze také spravovat pomocí grafického uživatelského rozhraní prostřednictvím Vue UI. [10]

1.4.4 Vue Router

Vue Router je oficiální router pro Vue.js. Hluboce se integruje s jádrem Vue, aby bylo vytváření jednostránkových aplikací s tímto frameworkem jednoduché. Mezi funkcemi patří:

- Vnořené mapování trasy/zobrazení.
- Modulární konfigurace směrovače na základě komponent.
- Dynamické přiřazování rout pomoci parametrů.

Když přidáte Vue Router do Vue, vše, co musíme udělat, je namapovat komponenty na trasy a dát Vue Routeru vědět, kde je má vykreslit. [11]

Příkaz pro instalace routeru vypadá následovně:

```
npm install vue-router
```

Při použití s modulovým systémem musíte směrovač explicitně nainstalovat pomocí `Vue.use()`. Následující ukázka je z praktické části ze souboru *routes.js*.

```
import Vue from 'vue';
import VueRouter from 'vue-router';
import store from '@/store';
import Navbar from "@/components/Navbar";
…
Vue.use(VueRouter);
…
export default new VueRouter({
    mode: 'history',
    routes: [
        {
            path: '*',
            component: () => import('@/components/NotFound'),
        },
        {
            path: '/',
```

```
            component: Navbar,
            name: 'Navbar',
            children: […]
        }
    ]
});
```

Zdroj: vlastní

Potom budete jenom potřebovat přidat tento modul v hlavním souboru (obvykle to je *main.js* nebo *app.js*) jako parametr v konstruktoru se vztahem klíč-hodnota. Když bude název importovaného souboru shodný s názvem klíče, tak nám JS umožňuje předat jako parametr jenom jednu hodnotu.

```
import Vue from 'vue';
import App from '@/App.vue';
import routes from "@/router/routes";
…
new Vue({
    router: routes,
    …
    render: h => h(App),
}).$mount('#app');
```

Zdroj: vlastní

Více mozností instalací a nastavení cest lze najít v oficiální dokumentaci[11]. [12], [13]

1.4.5 Vuex

Vuex je *state management pattern* + knihovna pro Vue.js aplikace. Slouží jako centralizované uložiště pro všechny komponenty v aplikaci s pravidly zajišťujícími, že stav může být změněn pouze předvídatelným způsobem.

Myšlenka na vytvoření pro Vuex byla vypůjčena z mnoha řešení, včetně Redux. Vuex používá **single state tree**: když jeden objekt obsahuje všechny globální stavy aplikace a slouží jako jediné místo odkud lze čerpat data.

Jakékoliv změny určitého stavu v úložišti by vedly k přepočtu a aktualizaci pouze těch komponent, jejichž hodnoty jsou spojeny s vypočtenou vlastností.

Příkaz pro instalace Vuex vypadá následovně:

```
npm install vuex --save
```

Stejně jako se směrovačem potřebujeme explicitně přidat modul do projektu.

[11] Dostupné z: https://router.vuejs.org/installation.html

```
import Vue from 'vue';
import Vuex from 'vuex';
import auth_module from '@/store/modules/auth.module';
import graph_module from '@/store/modules/graph.module';

Vue.use(Vuex);

export default new Vuex.Store({
    modules: {
        auth_module,
        graph_module
    }
})
```

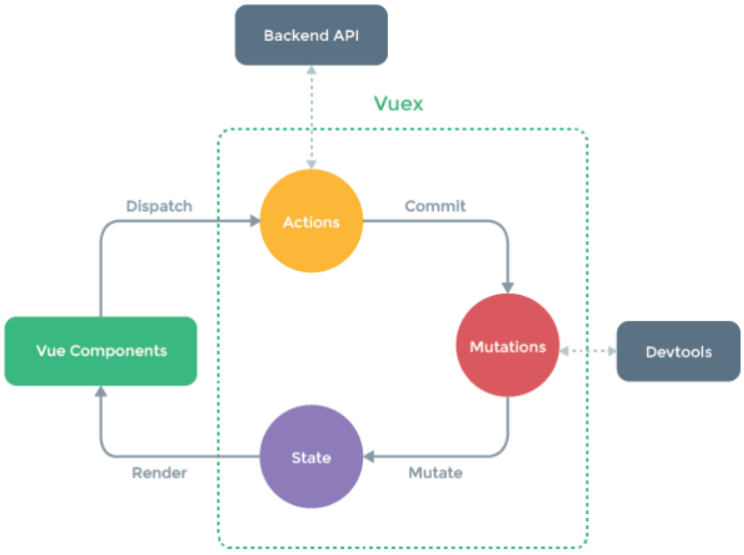
Zdroj: vlastní

Teď znovu budete jenom potřebovat přidat tento modul v hlavním souboru (obvykle to je main.js) jako parametr v konstruktoru se vztahem klíč-hodnota.

```
import Vue from 'vue';
import App from '@/App.vue';
import store from '@/store/index';
…
new Vue({
    store: store,
    …,
    render: h => h(App),
}).$mount('#app');
```

Zdroj: vlastní

Konečně máte připravený projekt s Vue Routerem a Vuexem pro další realizace vlastních záměrů. Projekt připravený ke spuštění a testování projekt je k dispozici v praktické části bakalářské práce.



Obrázek 2: architektura Vuex. Zdroj: [14]

State

Zde jsou uloženy všechny proměnné celé aplikace. Pro práci s těmito proměnnými existují mutace a gettery, lze k nim ale přistupovat i přímo pomocí *$store.state.variable_name*. Jakékoliv změny *$store.state.variable_name* způsobí přepočet závislých elementů a spuštění souvisejících aktualizací DOM.

Getters

Někdy je nutné provést některé výpočty založené na stavu úložiště, například filtrování seznamů. Pokud bude takový výpočet probíhat více než v jedné složce, je nutné vytvořit duplicitní funkce nebo ji umístit do obecné metody, která pak bude importovaná ve všech místech – oba přístupy ale nejsou ideální. Vuex umožňuje definovat gettery v úložišti. Můžete je považovat za vypočtené vlastnosti úložiště. Stejně jako vypočtené vlastnosti výsledky getterů jsou ukládány do mezipaměti a přepočítány pouze při změně jedné ze závislostí. Stojí za připomenutí, že gettery dostávají stav úložiště jako první argument.

Mutations

Jediným způsobem, jak změnit stav úložiště ve Vuexu, jsou mutace. Chcete-li zahájit zpracování mutace, musíte volat *context.commit* s označením jejího typu. Například:

```
mutations: {
    changeAuthState(state) {
        state.isAuthenticated = !state.isAuthenticated;
    },
    setAuthResult(state, authResult) {
        state.authResult = authResult;
    }
},
actions: {
    async signIn({commit, state}) {
        state.msalInstance = new PublicClientApplication(state.config);
        await state.msalInstance.loginPopup(state.loginRequest)
            .then(loginResponse => {
                commit('setAuthResult', loginResponse);
                commit('changeAuthState');
                commit('setGraphClient', loginResponse.accessToken);
            })
            .catch(err => console.log(err));
    },
    …
},
…
```

Zdroj: vlastní

Vzhledem k tomu, že stav úložiště Vuex je reaktivní proměnná Vue, pokud dojde ke změně, součásti Vue závislé na tomto stavu se automaticky aktualizují. Také je třeba si uvědomit jedno důležité pravidlo: zpracovatelé mutací musí být synchronní. Stojí za

připomenutí, že mutace dostávají stav úložiště jako první argument a další paramenty navíc, které se nazývají *payload*.

Actions

Akce je podobná jako mutace, ale s několika odlišnostmi:

- Spíše než přímo změnit stav, akce iniciují mutace.
- Akce mohou být použity pro asynchronní operace.

Obslužné rutiny získají jako první parametr objekt kontextu obsahující stejné metody a vlastnosti jako samotná instance úložiště, a proto můžete vyvolat *context.commit*, který zahájí mutaci nebo se obrátit na stav a gettery prostřednictvím *context.state* a *context.getters*. [14]

1.5 React

React je JavaScriptová *open-source* knihovna od společnosti Facebook pro tvorbu uživatelského rozhraní (UI). Pro vývoj UI se React často používá s jinými knihovnami, jako jsou MobX, Redux, Material-UI, React-Bootstrap atd. React může také pracovat na serveru Node.js v kombinaci např. s knihovnou Next.js a na mobilních platformách pomocí React Native. Základním stavebním kamenem jsou znovupoužitelné zapouzdřené komponenty s vlastním stavem, které pak lze kombinovat do více složitých komponentů. Takové rozdělení dělá kód předvídatelnější a snadnější k jeho ladění. Vzhledem k tomu, že logika komponentu je napsána v JavaScriptu a není obsažena v šablonách, je možné snadno přenášet různé údaje v celé aplikaci a udržovat stav mimo DOM. [42]

1.5.1 Class komponent vs. Functional komponent v Reactu

Oproti Vue React nabízí dvě možnosti pro tvorbu komponent. Podívejme se na ně blíže a najdeme hlavní rozdíly. Pro ukázku bude vytvořen jednoduchý čítač jako v příkladu ve Vue.js.

Class komponent

```
import React from 'react';
import './test.css';

export default class TestClass extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            counter: 0
        }
    }

    increment = () =>
        this.setState(prevState =>
```

```
            ({ counter: prevState.counter + 1 }));

    render() {
        return (
            <>
                <p>{this.state.counter}</p>
                <button className={'button plus'}
                    onClick={() => this.increment()}>+
                </button>
                <button className={'button minus'}
                    onClick={() => this.setState(prevState =>
                        ({ counter: prevState.counter - 1 }))}>-
                </button>
            </>
        );
    }
}
```

Zdroj: vlastní

Teď pojďme se podívat na výše napsaný kód blíže. Nejprve je třeba vytvořit třídu, která bude dědit z `React.Component`, a realizovat metodu `render()`.

Komponenta přijímá parametry v konstruktoru, nazývané stejně jako ve Vue: *props*. Když není potřeba přijímat parametry od rodiče nebo definovat *state* tak není nutné konstruktor používat. `this.props` je obyčejným objektem JavaScriptu a obsahuje vlastnosti, které byly definovány tím, kdo volá tento komponent.

Metoda `render()` je jediná **povinná** metoda ve třídě. Vrací popis toho, co chcete vidět na obrazovce. Většina vývojářů React používá speciální syntaxi JSX, která usnadňuje popis těchto struktur. Jedná se o rozšíření syntaxe JavaScriptu. React lze použít i bez JSX, ale takový kód bude do značné míry těžší číst, rozšiřovat, upravovat apod.

Chcete-li vložit JavaScript do HTML nebo jako atribut značky, musíte jej zabalit do jediných složených závorek, tím řeknete, že zde chcete použít JS.

```
…
<p>{this.state.counter}</p>
<button className={'button plus'}
    onClick={() => this.increment()}>+
</button>
…
```

Po kliknutí na tlačítko `+` obsah tagu `<p>` bude reaktivně nahrazen při změně stavu proměnné `counter`, která byla deklarována v konstruktoru třídy jako pole objektu *state*.

State jako *props* je jednoduchým objektem JavaScriptu, který obsahuje specifická data pro tento komponent. Mohou se libovolně měnit a jsou definována uživatelem. Metoda `setState()` plánuje asynchronní změnu stavového objektu komponentu. Když se *state* změní,

komponent se znovu vykreslí. *State* aktuálního komponentu můžete zobrazit pomocí klíčového slova `this`.

Místo direktiv React používá už známe událostí z JavaScriptu pro nastavení posluchačů na tagy a proto nenajdete docela užitečné direktivy `v-if`, `v-for` atd. Ale jak vykreslit pohled s podmínkou nebo v cyklu? V Reactu je to trochu komplikovanější a zabírá více řádků kódu.

Podmíněný render

Místo direktivy `v-if` v Reactu existují dvě varianty pro zobrazení podmíněného pohledu. Jak bylo už zmíněno výše pro programovaní v JavaScriptu, stačí otevřít složené závorky přímo uvnitř HTML tagu, ale je možný i opačný přístup, což lze vidět na druhém přikladu.

```
render() {
    return (
        <div>
            {this.state.counter % 2 === 0
                ? <div>Číslo je děleno dvěma</div>
                : <div>Číslo není dělitelné dvěma</div>
            }
        </div>
    );
}
```

JSX je taky výrazem!

Po kompilaci se každý výraz JSX stává běžným voláním funkce JavaScriptu, jehož výsledkem je objekt JavaScriptu.

Z toho vyplývá, že JSX lze použít v příkazu *if* a cyklech *for*, přiřadit proměnné, předat funkce jako argument a vrátit se z funkce. [43]

```
render() {
    let text;
    if (this.state.counter % 2 === 0) {
        text = <div>Číslo je děleno dvěma</div>;
    } else {
        text = <div>Číslo není dělitelné dvěma</div>;
    }
    return (
        <div>{text}</div>
    );
}
```

Zdroj: vlastní

Cyklický render

Pro následující ukázku vytvoříme ještě jedné pole *numbers* typu *Array* v objektu *state* a vyplníme jeho v speciální metodě životního cyklu React komponenty: `componentDidMount()`. Pro změnu stavu voláme pomocí klíčového slova *this* asynchronní

metodu `setState()`, se kterou jako parametr vložíme pár klíč-hodnota. V takovém jednoduchém přikladu všechno bude pracovat správně, ale při složitější logice může dojit k chybám, ke kterým by nikdy nedorazilo ve Vue, protože tam si potřebujete jenom přivlastnit novou hodnotu k proměnné z části *data* bez volání pomocné funkce. Jedním řešením je přidání funkce druhým parametrem metody `this.setState()`, která bude fungovat pouze po aktualizaci stavu - tedy pokud potřebujete získat novou hodnotu založenou na aktuálním stavu.

Pak pomocí metod `filter()` a `map()` ve složených závorkách provedeme vykreslení sudých nebo lichých čísel podle podmínky. Výsledek lze taky uložit do proměnné pro další zpracování. [44]

```
componentDidMount() {
    const numbers = Array.from({length: 10}, (v, k) => k + 1);
    this.setState({numbers: numbers});
}

render() {
    return (
        <div>
            {this.state.numbers
                .filter(num => num % 2 === this.state.counter % 2)
                .map(num => <div key={num}>{num}</div>)
            }
        </div>
    )
}
```

Zdroj: vlastní

Nyní se pojďme podívat na komponenty-funkce. Na první pohled je vidět, že je zápis kratší, není obsažena metoda `render()` a místo konstruktoru se objeví nová konstrukce, ale většina kódu se zůstává bez změn. Funkcionální komponenty přijímají *props* jako obyčejný parametr funkce. Ale kde se nachází *state*? Pro toto řešení existují tzv. **React Hooks**.

Functional komponent

```
import React, {useState} from 'react';
import './test.css';

export default function TestFunction() {
    const [counter, setCounter] = useState(0);
    const increment = () => setCounter(counter + 1);
    return <>
        <p>{counter}</p>
        <button className={'btn plus'} onClick={() => increment()}>+</button>
        <button className={'btn minus'} onClick={() => setCounter(counter-1)}>-
        </button>
    </>;
};
```

Zdroj: vlastní

1.5.2 React Hooks

Hooky je inovace v React 16.8, která umožňuje použití stavu a dalších funkcí Reactu bez psaní tříd. Hooky umožňují rozdělit jednu složku na malé funkce podle jejich účelu, nikoliv na základě metod životního cyklu. Pomocí hooků můžete extrahovat *state* logiku z komponentů abyste ji otestovali nebo znovu použili. Hooky vám umožňují znovu používat stavovou logiku bez ovlivnění na strom komponent. Díky tomu jsou hooky snadno použitelné v různých složkách, ale hooky nefungují uvnitř tříd. [45]

Pravidla hooků

Hooky jsou funkce JavaScriptu, ale mají dvě další pravidla:

- Volejte pouze hooky na nejvyšší úrovni. Nevolejte hooky uvnitř cyklů, podmínek nebo vnořených funkcí.
- Volejte pouze hooky z funkcionálních komponentů. Nevolejte háčky z obyčejných funkcí JavaScriptu. Existuje jen jedno další platné místo pro volání háčků: vaše vlastní hooky. [46]

Hook useState()

V přikladu výše hook `useState()` je volán, aby náš funkční komponent získal vnitřní stav. React bude ukládat tento stav mezi rendery. Volání `useState()` vrátí pole se dvěma položkami, které obsahuje: aktuální hodnotu stavu a funkci pro jeho aktualizaci. Jediným argumentem hooku `useState()` je počáteční stav. V příkladu výše je to číslo 0, ale může být jako pole, řetězec, objekt atd.

Hook useEffect()

Výše byla zmíněna metoda životního cyklu React komponentu `componentDidMount()`. K nahlédnutí na oficiálních stránkách[12] najdete nápovědu s diagramem životního cyklu. Také mezi nejběžnější metody životního cyklu patří: `componentDidUpdate()` a `componentWillUnmount()`. Hook `useEffect()` provádí stejnou roli jako `componentDidMount()`, `componentDidUpdate()` a `componentWillUnmount()` dohromady v React třídách a kombinuje je do jediného API rozhraní pro funkcionální komponenty. Když chcete získat data ze serveru nebo ručně měnit DOM, tak se tyto děje nazývají „efekty“. Ty mohou ovlivnit práci jiných komponentů a nemohou být provedeny během renderu, ale pomocí hooku `useEffect()` lze provádět různé „efekty“ v funkcionálních komponentách.

[12] Dostupné z: https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

Když zavoláte hook `useEffect()`, React dostane pokyn k spuštění funkce s „efektem“ po odeslání změn v DOM. Vzhledem k tomu, že efekty jsou deklarovány uvnitř komponentu, mají přístup k jeho *props* a *state*. Ve výchozím nastavení React spustí efekty po každém renderu, včetně prvního. V případě potřeby lze z efektu vrátit funkci, ve které bude popsáno jak provést „clean up“ podobně metodě `componentWillUnmount()`. [47]

Seznam všech důležitých hooků s popisem a příklady je k dispozici na oficiálních dokumentačních stránkách[13] Reactu v oddílu Docs, Hooks, Hooks API Reference. [48]

Po všech získaných informacích lze vše podstatné shrnout do porovnání níže.

Class komponenty:

- Je třeba dědit z React.Component a implementovat metodu `render()`, která se vrátí JSX.
- Používá pro kontrolování životního cyklu speciální metody.
- Používá konstruktor pro *props* a speciální proměnnou `this.state` pro spravování vnitřního stavu komponentu.
- Je starší způsob tvorby komponentů a je méně populárnější z toho důvodu, že funkcionální komponenty jsou menší a snadnější pro čtění a testování.

Functional komponenty:

- Jsou obyčejnou JS funkcí, která přijímá *props* jako argument a vrátí se JSX.
- Místo speciálních metod pro kontrolování životního cyklu používá jenom jeden hook.
- Používá hook `useState()` pro práci se stavem komponentu.

1.5.3 Stateful a Stateless komponenty

V této časti budeme přezkoumávat co jsou stateful a stateless komponenty v Reactu. Na začátku připomínáme co je *state*. Vnitřním stavem komponentu je uživatelsky definovány objekt *state*, který voláte pomocí klíčového slova `this` v třídních komponentech nebo pomocí hooků `useState()` ve funkcionálních komponentech.

Stateful a Stateless komponenty mají ještě několik ostatních názvů: Container/Presentational nebo Smart/Dumb.

Základní myšlenkou paradigmatu Stateless komponentů je odmítnutím použití vnitřního stavu a získání dat jenom od rodičovského komponentu pomocí *props*. Tím můžete zajistit znovupoužitelnost komponentů. [67]

[13] Dostupné z: https://reactjs.org/docs/hooks-reference.html

Níže je uvedena krátká ukázka možného scénáře využití Stateless komponentů pomocí připojení k bezplatnému falešnému API pro testování a prototypování z webu *jsonplaceholder*. V rodičovskému komponentu

```
import React from 'react';
import ListItem from "./ListItem";

class List extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            users: [],
            posts: []
        }
    }

    fetchData = url =>
        fetch(url).then(response => response.json());

    async componentDidMount() {
        const posts =
            await this.fetchData('https://jsonplaceholder.typicode.com/posts');
        this.setState({posts});

        const users =
            await this.fetchData('https://jsonplaceholder.typicode.com/users');
        this.setState({users});
    }

    render() {
        return <>
            <ListItem title={'Posts'} array={this.state.posts}/>
            <ListItem title={'Users'} array={this.state.users}/>
        </>;
    }
}

export default List;
```

Zdroj: vlastní

```
import React from 'react';

const ListItem = ({title, array}) => {
    return <>
        <h1>{title}</h1>
        <ul>
            {array.map(data =>
                <li key={data.id}>{JSON.stringify(data)}</li>)}
        </ul>
    </>;
};

export default ListItem;
```

Zdroj: vlastní

1.5.4 React Router

Jak pro Vue, tak i pro React existuje vlastní router. Tři hlavní úkoly React Routeru:

- Přihlášení k odběru a manipulace se stackem historie.
- Přiřazení adresy URL k vašim trasám.
- Vykreslení vnořeného uživatelského rozhraní ze shody trasy. [49]

Příkaz pro instalaci routeru vypadá následovně:

```
npm install react-router-dom
```

Jakmile projekt ukončil build a React Router je nainstalován jako závislost, otevřete *src/index.js* v IDE nebo jinem textovém editoru. Importujte Browser Router z balíčku *react-router-dom* v horní části souboru a zabalte aplikaci do `<BrowserRouter>`:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import {BrowserRouter} from "react-router-dom";

ReactDOM.render(
    <BrowserRouter>
        <App/>
    </BrowserRouter>,
    document.getElementById('root')
);
```

Zdroj: [50]

Teď je možné používat React Router kdekoliv v celé web aplikaci. Více možností instalace a nastavení cest lze najít v oficiální dokumentaci[14]. [50]

1.5.5 Redux

Redux je předvídatelný stavový kontejner pro aplikace JavaScript. Pomáhá psát aplikace, které se chovají konzistentně, běží v různých prostředích (klient, server a nativní) a snadno se testují. Celý globální *state* aplikace je uložen ve stromu objektů uvnitř jednoho *store*. Jediným způsobem, jak změnit stavový strom, je vytvořit *action*, objekt popisující, co se stalo, a odeslat ji do *store*. Chcete-li určit, jak se *state* aktualizuje v reakci na *action*, je třeba napsat *pure reducer* funkcí, které počítají nový *state* na základě starého *state* a *actions*.

Místo toho, abyste přímo mutovali *stav*, určíte mutace, ke kterým chcete dojít, s prostými objekty nazývanými *action*. Pak napíšete speciální funkci nazvanou *reducer*, která rozhodne, jak každá akce transformuje stav celé aplikace.

[14] Dostupné z: https://reactrouter.com/docs/en/v6/getting-started/installation

V typické aplikaci Redux existuje pouze jeden *store* s *reducer* funkcemi jednoho kořene. Jak aplikace roste, rozdělíte kořenový *reducer* na menší *reducery* nezávisle pracující na různých částech stavového stromu. To je přesně tak jako je jen jedna kořenová složka v aplikaci React, ale je složena z mnoha malých komponentů. [51]

Pro práci s Reduxem v Reactu je třeba zpočátku nainstalovat Redux Core a pak oficiální vazbu Reactu pro Redux:

```
npm i redux react-redux
```

React Redux obsahuje komponentu `<Provider>`, díky němuž je Redux *store* dostupný pro zbytek vaší aplikace:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import {Provider} from "react-redux";
import {rootReducer} from "./store/rootReducer";
import {createStore} from "redux";
const store = createStore(rootReducer);

ReactDOM.render(
    <Provider store={store}>
        <App/>
    </Provider>,
    document.getElementById('root')
);
```

Zdroj: [53]

Tím spojíme Redux a React, ale jak využívat *state* v React komponentech? React Redux poskytuje dvě moznosti, ale rozebereme jenom jednu přes hooky. React Redux má pár vlastních hooků, které umožňují React komponentům komunikovat s Redux *store*. `useSelector()` přečte hodnotu ze stavu obchodu a přihlásí se k aktualizací, zatímco `useDispatch()` se vrátí *store's dispatch method*, aby mít možnost odesílat akcí. [53]

React Redux kromě hooků nabízí ještě jednou možnost pro práci s ním pomocí metody-obalky `connect()`. Metoda přijímá 4 parametry, ale nás budou zajímat jenom první dva: *mapStateToProps* a *mapDispatchToProps*, které jsou ekvivalentní hookům `useSelector()` a `useDispatch()`.

Rozebereme je podrobněji. Prvním parametrem je funkce *mapStateToProps*, která je volána po každé aktualizaci *store*. Výsledkem této funkce je obyčejný JavaScriptový objekt, který bude sloučeny do *props* komponentu.

Druhým parametrem může být funkce nebo objekt *mapDispatchToProps*. Jestliže je to funkce, tak prvním parametrem *mapDispatchToProps* dostane *dispatch* našeho *store*. V jinem případě *mapDispatchToProps* je objektem, ve kterém každé pole je *action creatorem*. [68]

Praktická ukázka práci s hooky a metodou `connect()` je představena v praktické části bakalářské práce.

```
import React from 'react';
import {connect} from "react-redux";
import {callMsGraphApi} from "../../../store/graph/graphReducer";
…

class CalendarTable extends React.Component {
…
}

const mapStateToProps = state => ({
    graphClient: state.graphReducer.graphClient
})

const mapDispatchToProps = {
    callMsGraphApi
}

export default connect(mapStateToProps, mapDispatchToProps)(CalendarTable);
```

Zdroj: vlastní

Bohužel, Redux Core neumí pracovat s asynchronními funkcemi. Ve výchozím nastavení jsou akce Reduxu odesílány synchronně, což je problém pro jakoukoliv netriviální aplikaci, která potřebuje komunikovat s externím API nebo provádět vedlejší účinky. Redux také umožňuje middleware, který sedí mezi odeslanou akcí a akcí, která dosáhne určitého *reduceru*.

Existují dvě velmi populární middlewarové knihovny, které umožňují zpracovávat *side effects* a asynchronní akce: Redux Thunk a Redux Saga. V této bakalářské práce prozkoumáme Redux Thunk, který nainstalujete následujícím příkazem.

```
npm install redux-thunk
```

Pak budete potřebovat přidat Thunk do našeho *store*.

```
import thunk from "redux-thunk";
import {applyMiddleware, …} from "redux";
…
const store = createStore(rootReducer, applyMiddleware(thunk));
…
```

Zdroj: [52]

Thunk je programovací koncept, kde se funkce používá ke zpoždění hodnocení/výpočtu operace.

Redux Thunk je *middleware*, který umožňuje volat *action creators*, které vracejí funkci místo objektu akce. Tato funkce přijímá *store's dispatch method*, která se poté použije k odeslání pravidelných synchronních akcí uvnitř těla funkce po dokončení asynchronních operací. [52]

Níže je zobrazena základní struktura *thunk funkce*, která přijímá dva argumenty: metodu Redux *store dispatch* a metodu pro získání stavu *getState*. Thunk můžete odeslat voláním akce, je to stejný způsob, jako byste odeslali jakoukoli jinou akci Redux.

```
export const fetchExample = (var1, var2, …) =>
    async (dispatch, getState) => {
        //code inside
    }
…
dispatch(fetchExample);
```

Zdroj: [52]

2 POUŽITÉ DALŠÍ TECHNOLOGIE

Kromě hlavních nástrojů bylo použito mnoho dalších pomůcek pro usnadnění a urychlení práce autora bakalářské práce. Níže bude uveden jejich seznam s krátkým popisem.

2.1 npm

Balíčkovací ekosystém *npm* není jediný správce balíčků pro Node.js, ale je opravdu nejpoužívanější a je součástí základní instalace Node.js. Pomocí něho můžete obecně instalovat i spravovat závislosti, které se obvykle nacházejí v souboru *package.json*, ve svých projektech. S *npm* se pracuje přes klasický příkazový řádek a všechny moduly se instalují jednoduchým příkazem `npm install` *`název`* (nebo zkráceně `npm i` *`název`*), takže chcete-li nainstalovat např. balíček *moment* pro práci s daty a s časem, nainstalujete jeho takto:

```
npm install moment
```

Pokud si otevřete adresář projektu, uvidíte, že byl vytvořen nový adresář *node_modules*. Zde se instalují všechny moduly, pokud je nechcete instalovat globálně do jednoho hlavního adresáře pro všechny projekty. V takovém případě je potřeba provádět instalaci s parametrem *-g*. [32]

2.2 Komunikace s Microsoft

Microsoft Authentication Library (MSAL)

Microsoft Authentication Library (MSAL) umožňuje vývojářům získat tokeny z *Microsoft Identity Platform* za účelem ověření uživatelů a přístupu k zabezpečeným webovým API. Může být použitá k zajištění bezpečného přístupu k *Microsoft Graph*, dalším API společnosti Microsoft, webovým API třetích stran nebo k vlastnímu webovému API. MSAL podporuje mnoho různých aplikačních architektur a platforem včetně .NET, JavaScript, Java, Python, Android a iOS. [33]

@azure/msal-browser

Existují zvláštní knihovny MSAL pro Vue a pro React, ale bylo řešeno použít jedinou universální variantu pro obou dva případy.

Ale před použitím *@azure/msal-browser v* aplikaci budete muset zaregistrovat *Single Page Application* v *Azure Active Directory* podle oficiálního tutoriálu pro získání platného *clientId* pro nastavení konfigurace. Po vytvoření aplikace přidáte knihovnu k projektu pomocí následujícího příkazu:

```
npm install @azure/msal-browser
```

Všechna nastavení pro autorizaci se nacházejí v souborech *auth.module.js* (Vue projekt) a *authReducer.js* (React projekt) v praktické části bakalářské práce. [34]

MS Graph API

Microsoft Graph API je RESTful web API, které vám umožní přístup ke zdrojům cloudových služeb společnosti Microsoft. Po registraci aplikace a získání autentizačních tokenů pro uživatele nebo služby, můžete posílat dotazy do aplikace přes *Microsoft Graph API*.

Microsoft Graph vystavuje REST API a klientské knihovny pro přístup k datům v následujících cloudových službách společnosti Microsoft:

- Microsoft 365 servíces: Delve, Excel, Microsoft Bookings, Microsoft Teams, OneDrive, OneNote, Outlook/Exchange, Planner, a SharePoint.
- Enterprise Mobility a Security servíces: Advanced Threat Analytics, Advanced Threat Protection, Azure Active Directory, Identity Manager a Intune.
- Windows 10 servíces: activities, devíces, notifications.
- Dynamics 365 Business Central. [35]

@microsoft/microsoft-graph-client

```
npm install @microsoft/microsoft-graph-client
```

Příklady volání MS Graph API se nacházejí v souborech *graph.module.js* (Vue projekt) a *graphReducer.js* (React projekt) v praktické části bakalářské práce a na oficiálních stránkách Microsoft Graph REST API v1.0 reference[15]. [36]

2.3 Bootstrap

Bootstrap je *open source toolkit* a nejpopulárnější CSS framework pro vývoj responzivních a mobilních webů, který obsahuje obrovskou kolekci různých pluginů a stylů.

Bootstrap pro Vue

```
npm install bootstrap@4.5.3 bootstrap-vue
```

Poté zaregistrujte *BootstrapVue* ve vstupním bodě aplikace (obvykle *app.js* nebo *main.js*):

```
import Vue from 'vue';
import {BootstrapVue} from 'bootstrap-vue';

import 'bootstrap/dist/css/bootstrap.css';
import 'bootstrap-vue/dist/bootstrap-vue.css';
…
Vue.use(BootstrapVue);
```

Zdroj: [37]

[15] Dostupné z: https://docs.microsoft.com/en-us/graph/api/overview

Bootstrap pro React

```
npm install bootstrap@5.1.3 react-bootstrap
```

Po instalaci se zůstává přidat následující řádek do vstupního bodu aplikace (obvykle *index.js* nebo *App.js*).

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

Zdroj: [38]

2.4 Moment.js

Knihovna JavaScriptu pro analýzu, ověřování, manipulaci a formátování datumů.

Moment.js pro Vue

```
npm install vue-moment
```

Po instalaci můžete nastavit globálně ve vstupním bodě aplikace (obvykle *app.js* nebo *main.js*) jako plugin a používat přes svislou čáru podobně filtru ve Vue:

```
Vue.use(require('vue-moment'));
```

Nebo importovat v komponentě v bloku script:

```
import moment from "moment";
```

Pak vytvořit filtr a používat pro vlastní účely:

```
Vue.filter('dateFormatter', value => moment(value).format('DD.MM.YYYY HH:mm'));
```

Zdroj: [40]

Moment.js pro React

```
npm install --save moment react-moment
```

Po ukončení instalace můžete klidně používat ve svých React komponentech. Stačí importovat metodu `moment()`:

```
import moment from "moment";
```

Nebo použít speciální React komponent *Moment*:

```
import Moment from 'react-moment';
…
<Moment format="DD.MM.YYYY HH:mm">{date}</Moment>
```

Zdroj: [41]

3 POROVNÁNÍ

3.1 Velikost frameworku a knihovny

Na stránce CDN vue@2.6.14[16] ve složce *dist* můžete najít spoustu různých souborů, ale nás většinou budou zajímat jenom dva: *vue.js* (vývojová verze) a *vue.min.js* (produkční verze). Velikost vývojové verze je 336 kB a velikost produkční verze je 92 kB. Pro porovnání s react@18.0.0[17] soubor *react.development.js* má velikost 110 kB a *react.production.min.js* má 11 kB. Jak je to vidět React je menší v obou dvou případech, ale s Reactem dost často je používán React DOM jako neoddělitelná součást. Velikost souboru *react-dom.development.js* je více než jeden megabajt a rovná se 1,06 MB. Produkční verze je mnohem menší a činí 131 kB. [55], [56], [57]

3.2 Složitost učení Vue a Reactu

React má velmi specifickou syntaxi, která je jádrem obrovského nepořádku a nedává absolutně žádný smysl. Pro práci s Reactem potřebujete se naučit JSX, taky je to dobry znát standard JavaScriptu ES6, aby se psát jednoduché věcí složitým a nepřehledným způsobem. Stačí jenom porovnat syntaxe podmíněného renderu, cyklického renderu nebo aktualizace stavu přirazením nových hodnot ve Vue a Reactu. Při práci nad projektem přechod z jiného frameworku/knihovny na Vue probíhá rychle jak pro jednoho člověka, tak i pro cely tým. [58]

Vuex jako *state manager* ve Vue oproti Reduxu má velmi snadnou koncepci pro učení, oficiální podporu pro Vue a zahrnuje v sobě po instalaci všechno co potřebujete pro práci, když pro vývoj v Reactu je třeba nainstalovat balíčky *redux*, *react-redux* (pro vázání stavu jsou dvě možností: hooky `useDispatch()` a `useSelector()` nebo metoda-obálka pro celý komponent `connect()`) a *redux-thunk* nebo *redux-saga* pro asynchronní dotazy, ale při existenci různých variant jednoho nástroje potřebujete vědět všechny, abyste při setkání s nimi v praxi mohli umět pracovat. V takovém případě je nutné si navíc pamatovat informace, které nebudou využity příliš často.

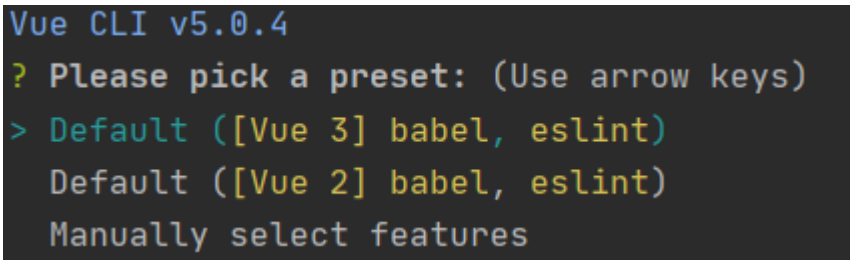
Pro začátek práce v obou frameworcích stačí spustit příkazy pro generaci základní hotové šablony pro spuštění a programování.

Pro React existuje následující příkaz:

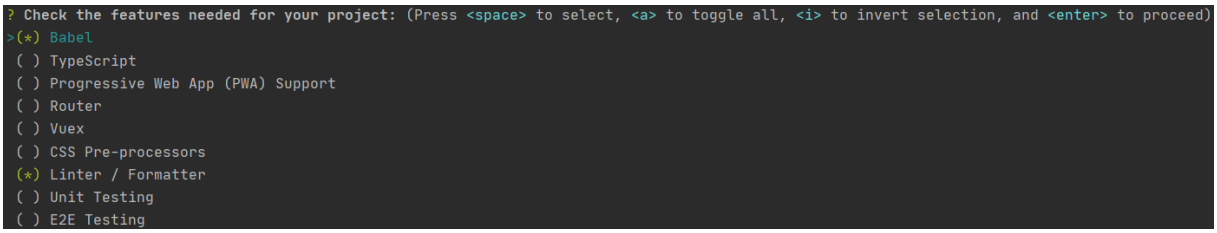
```
npx create-react-app name
```

[16] Dostupné z: https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/
[17] Dostupné z: https://unpkg.com/browse/react@18.0.0/umd/

Pro Vue existuje Vue CLI, který může vygenerovat předkonfigurovaný projekt nebo můžete ručně zvolit vyžadované nástroje.



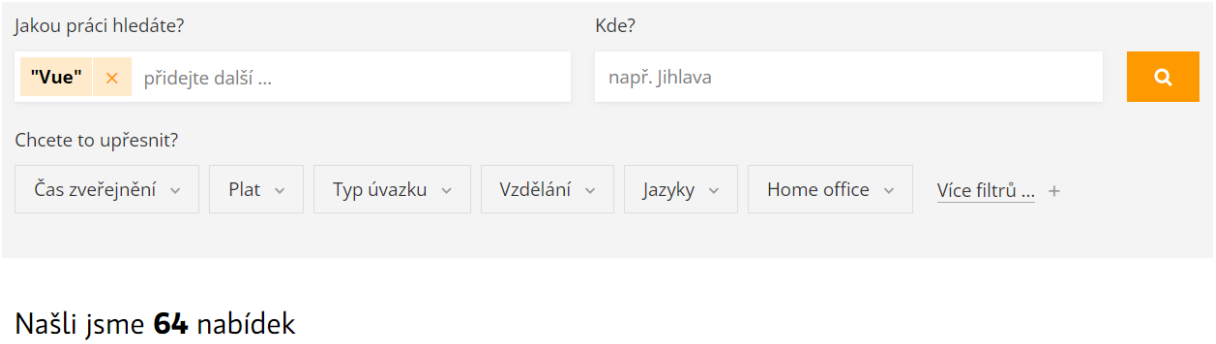
Obrázek 3: předkonfigurované možností pro vytvoření projektu. Zdroj: vlastní



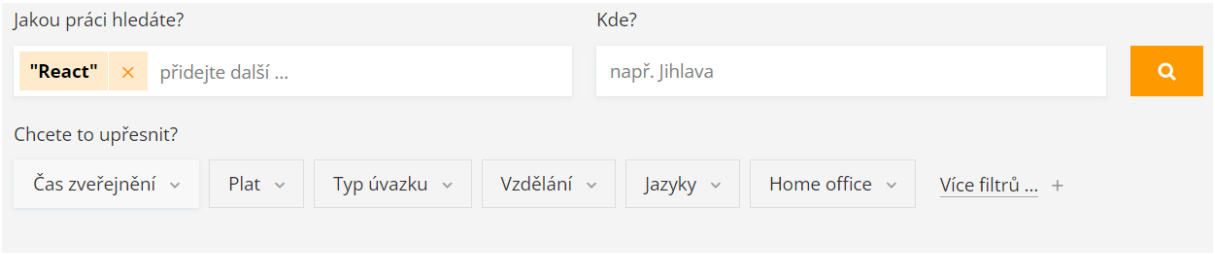
Obrázek 4: ruční nastavení projektu. Zdroj: vlastní

3.3 Pracovní nabídky v ČR

Pro sběr dat byl použít web *jobs.cz*. Stavem na 2022-04-19 vydávají následující výsledky:



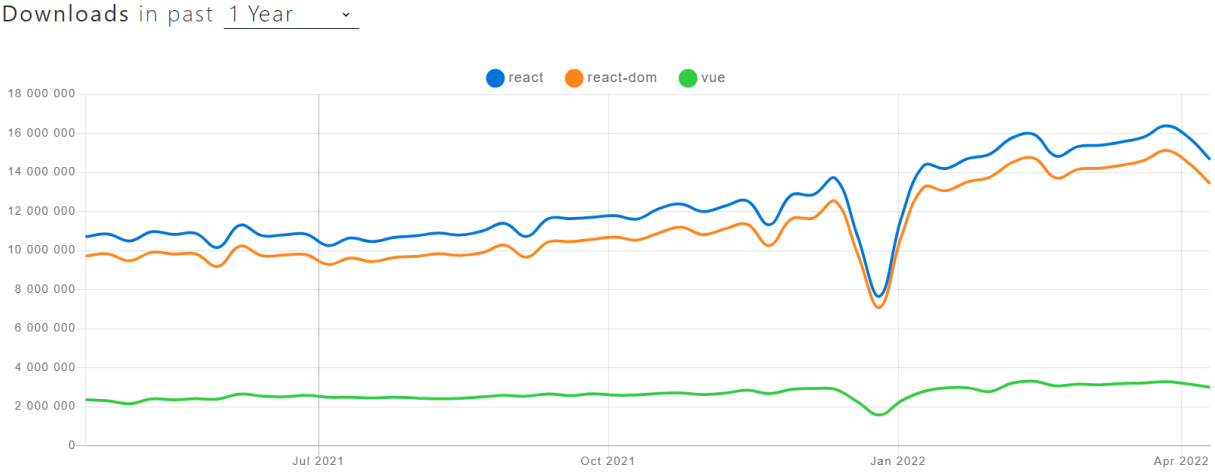
Obrázek 5: počet nabídek pro Vue na *jobs.cz*. Zdroj: vlastní



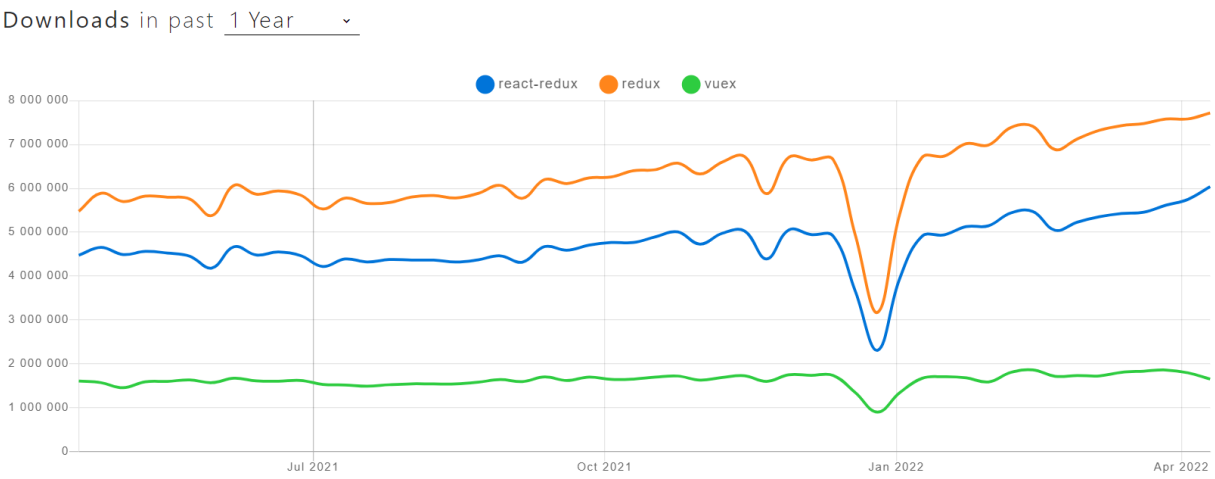
Obrázek 6: počet nabídek pro React na *jobs.cz*. Zdroj: vlastní

3.4 npm trends

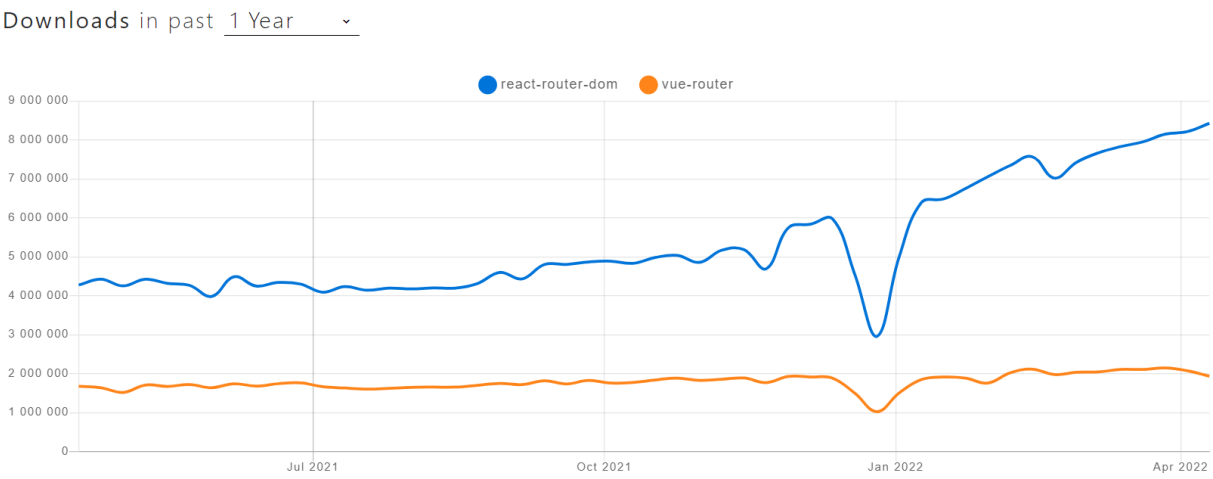
Pro sběr frekvence stahování stažených balíčků pomocí *npm* je nejlepší využít web *npmtrends.com*. Níže budou představeny srovnávací grafy ve třech nejdůležitějších kategoriích.



Obrázek 7: počet stažených core balíčků za rok. Zdroj: vlastní



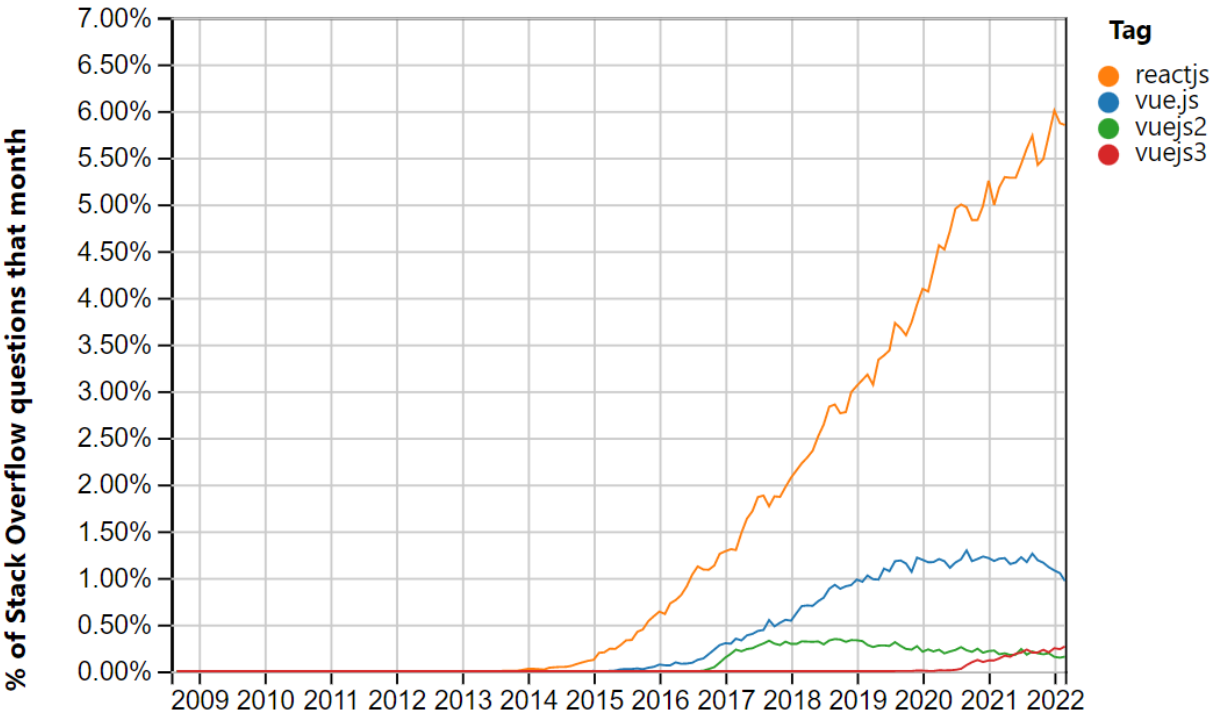
Obrázek 8: počet stažených state managerů za rok. Zdroj: vlastní



Obrázek 9: počet stažených routerů za rok. Zdroj: vlastní

3.5 Stack Overflow

Stack Overflow taky nabízí vlastní nastroj pro sběr počtu dotazů na Stack Overflow Trends. Níže je představen graf s výsledkem jak často lidí se ptají na různé Vue a React.

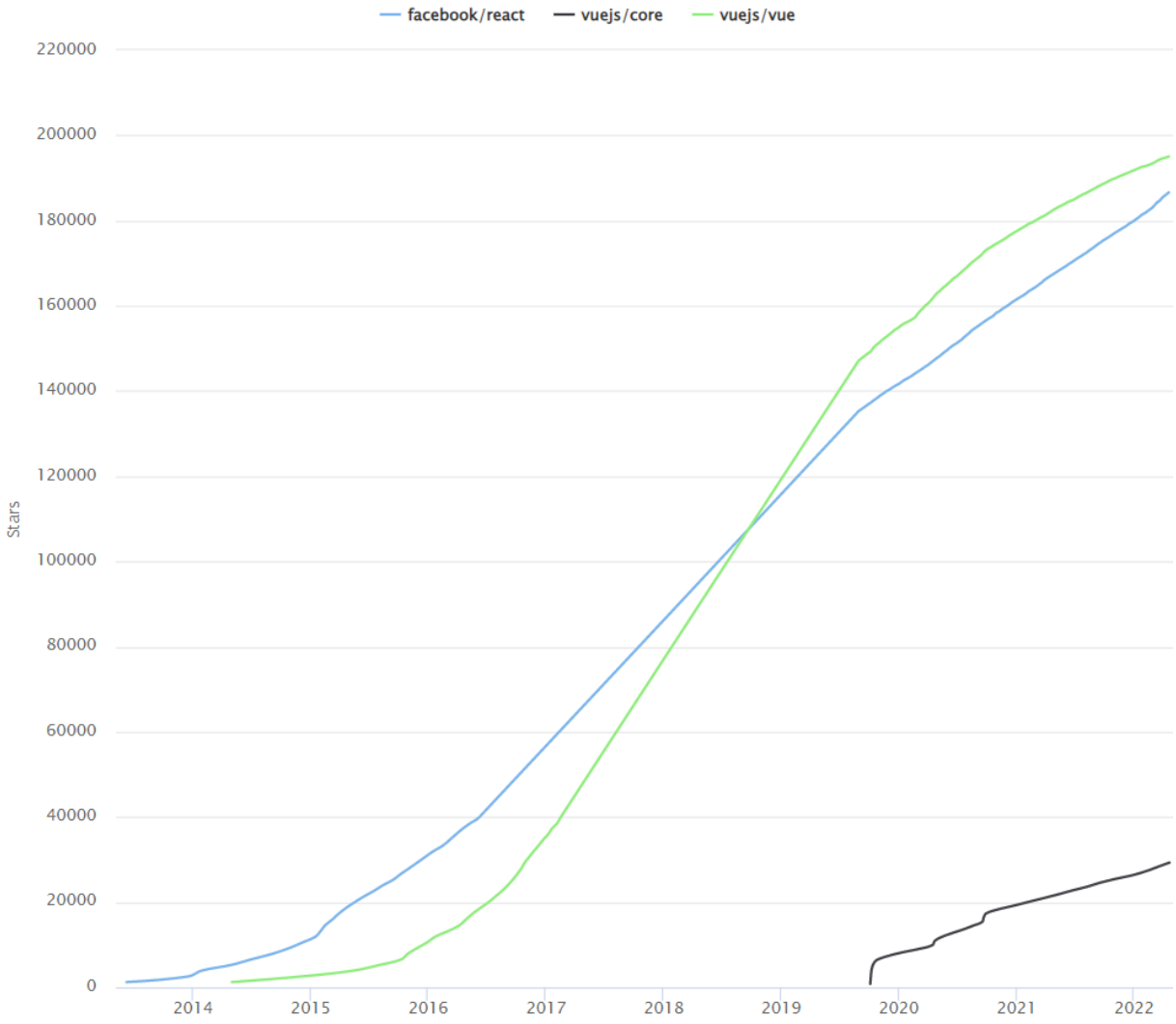


Obrázek 10: procent dotazů Vue a Reactu ode všech Stack Overflow dotazů. Zdroj: vlastní

3.6 GitHub

Stavem na 2022-04-21 oddíl GitHub topics mají následující výsledky:

- React: 203 559 odpovídajících veřejných repositářů[18].
- Vue: 41 563 odpovídajících veřejných repositářů[19].



Obrázek 11: počet hvězd na GitHubu. Zdroj: vlastní

Na webu *star-history.com* nebo *stars.przemeknowak.com* lze získat celkový počet hvězd konkrétních repositářů. Hvězdy na GitHubu jsou obdobou lajků v sociálních sítích, které zobrazují hodnocení vývojářů a v nějaké podobě vyjadřují popularitu. Je ale třeba vzít do vědomí skutečnost, že vývojářů Vue je významně méně. V daném grafu balíček *vuejs/core* představuje poslední verzi Vue 3 a *vuejs/vue* je pro starší Vue 2.

[18] Dostupné z: https://github.com/topics/react
[19] Dostupné z: https://github.com/topics/vue

3.7 Výkon

React a Vue jsou ve mnoha ohledech podobné:

- Používají virtuální DOM.
- Poskytují reaktivitu a komponentní strukturu.
- Zaměřují se na kořenovou knihovnu a přináší další otázky, jako routing nebo správa globálního stavu aplikace, do dalších knihoven.

V Reactu, když se změní stav komponenty, spustí se rerender celého podstromu komponentů, začíná sebou samým. Abyste se vyhnuli zbytečnému opětovnému vykreslování podřízených komponent, měli byste použít *PureComponent* nebo implementovat `shouldComponentUpdate()`, kdykoli je to možné.

V případě Vue jsou závislosti na komponentu automaticky sledovány během vykreslování, takže systém přesně ví, které komponenty je třeba při změně stavu znovu vykreslit. Každá komponenta může být viděna jako `shouldComponentUpdate()`, automaticky implementovaná pro vás, bez omezení pro vnořené komponenty.

Zajímavé je také porovnání výkonu[20], které se zaměřuje na výkon vykreslování stromu velmi jednoduchých komponentů. [66]

Name	vue-v2.5.16-keyed	react-v16.4.1-keyed
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,252.7 ± 0.2 (0.01%)	2,477.6 ± 0.6 (0.02%)
script bootup time the total ms required to parse/compile/evaluate all the page's scripts	55.1 ± 1.5 (2.74%)	65.6 ± 2.3 (3.56%)
main thread work cost total amount of time spent doing work on the main thread. includes style/layout/etc.	420.6 ± 67.5 (16.05%)	466.0 ± 3.9 (0.85%)
total byte weight network transfer cost (post-compression) of all the resources loaded into the page.	215,445.0 ± 0.0 (0.00%)	251,915.0 ± 0.0 (0.00%)

Obrázek 12: Startup metriky (s klíči). Zdroj: vlastní

[20] Dostupné z: https://stefankrause.net/js-frameworks-benchmark8/table.html

Name	vue-v2.5.16-keyed	react-v16.4.1-keyed
create rows Duration for creating 1000 rows after the page loaded.	182.1 ± 7.6 (4.17%)	180.5 ± 7.3 (4.04%)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	158.8 ± 2.7 (1.68%)	157.3 ± 2.0 (1.26%)
partial update Time to update the text of every 10th row (with 5 warmup iterations) for a table with 10k rows.	156.4 ± 9.8 (6.25%)	81.9 ± 2.7 (3.33%)
select row Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	10.6 ± 2.0 (19.12%)	10.3 ± 2.1 (20.10%)
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	20.0 ± 2.9 (14.34%)	106.5 ± 1.9 (1.79%)
remove row Duration to remove a row. (with 5 warmup iterations).	54.2 ± 2.2 (4.12%)	49.6 ± 0.8 (1.69%)
create many rows Duration to create 10,000 rows	1,603.2 ± 34.8 (2.17%)	1,935.4 ± 33.6 (1.74%)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	342.5 ± 6.0 (1.76%)	268.6 ± 6.9 (2.58%)
clear rows Duration to clear the table filled with 10.000 rows.	191.9 ± 6.1 (3.20%)	175.4 ± 4.1 (2.35%)

Obrázek 13: Doba trvání v milisekundách ± směrodatná odchylka (s klíči). Zdroj: vlastní

3.8 Licence

Vue stejně jako React používají MIT Licenci. [63], [64]

3.9 Dokumentace

V této části není vše tak jednoznačné. Dokumentace Vue je mnohem lepší než dokumentace Reactu. Je pochopitelnější, zahrnuje řešení hodně problémů a vysvětlení, se kterými se vývojáři mohou setkat při napsání kódu. Komunita Vue je ale menší a proto může být těžší získat odpověď na otázku, která nebyla jasně vysvětlena. Takže dokumentace Vue se tyká souvisejících témat ekosystému takových jako Vuex, Vue Router a nevyžaduje hlubokých znalostí web programování, zejména v JavaScriptu. [59], [62]

React zahrnuje méně témat a popisuje je hůře bez zahrnutí ostatních nástrojů. Je tedy vytvořena pro již zkušenější vývojáře, ale pro začátečníky je Vue přátelštější. Taky v tutoriálu na začátku je upozornění, že by bylo dobry vědět ES6 standard JavaScriptu, protože některé vlastnosti budou použité z ES6. [65]

React vývojářů je více, protože tato knihovna se stala populární podstatně dříve a proto pro React existuje spousta externích materiálů pro doučení.

3.10 Komunita a popularita

Jak je vidět z obrázků React je populárnější než Vue, ale existuje několik příčin, proč je to tak. Datum vydání Reactu je květen 2013 roku a Vue je o rok mladší. Takže je důležitým důvodem, že React je navržen a podporován společností Facebook, která jej používá ve vlastních aplikacích a vydává aktualizace. To znamená, že React má neustálou podporu komunity, která stabilně vytváří a vyvíjí nové nástroje a knihovny. Mezitím byl Vue zahájen samostatným vývojářem jménem Evan You, nikoliv společností a proto se nestal populárním jako React. Navíc, když byl poprvé vydán, mnozí vývojáři to považovali za nespolehlivé a váhali s přijetím. Od té doby se Vue stal mnohem populárnější díky pokračující podpoře a úsilí komunity uživatelů. [59]

Ale i když React má více stažených balíčků než Vue, má více dotazů na Stack Overflow a taky má více repositářů na GitHubu, počet hvězd se zůstává za Vue při menším počtů vývojářů, což říká nám o růstu popularity a největší spokojenosti mezi vývojářů (až 91,2 % spokojenosti stavem na 2018 rok). Toto není nijak zvláštní, protože tento výsledek je spojen s tím, že téměř každý kdo pracoval s Vue pozitivně reaguje na intuitivnost frameworku a nízkou křivku učení. [60]

Počet hvězd není jediným argumentem pro Vue. Dotazníky Stack Overflow říkají, že oproti 2019 roku, kde Vue.js má jenom 15,2 % popularity, už v roce 2021 bylo dosaženo 18,97 %. Společnost JetBrains s. r. o., která vyvíjí software pro různé platformy

a programovací jazyky, prezentuje, že v roce 2016 byl pravidelně využíván Vue 20 %, ale v roce 2021 již Vue má 43 % a zaostává od Reactu jenom o 5 %. [61]

V tuto chvíli nezbývá nic než přiznat, že Vue nabývá popularitu a rozšiřuje se.

Firmy, které využívají **Vue**: Gitlab, Euronews, Adobe Portfolio, Behance, Alibaba, Trustpilot, Více, Nintendo, BMW.

Firmy, které využívají **React**: BBC, Airbnb, Facebook, PayPal, The New York Times, Netflix, Instagram, Twitter, WhatsApp. [59]

3.11 Vlastní příklady porovnání

V tomto oddílu bude provedeno obecné porovnání kódů Vue a Reactu na zajímavých příkladech.

Prvním bude představen kus z komponentu *Calendar.vue*. Je to hlavní komponent pro práci s kalendářem a formulářem pro přidání nové události. Pro dynamické zobrazení různých komponentů byl použit speciální tag `<component>` s atributem `is`, kde `currentOption` je jménem registrovaného komponentu. Princip fungování je docela jednoduchý. Po každém přechodu Vue maže komponent a pak jej vykreslí znovu. To není vždy dobré. Někdy je třeba cachovat údaje jako např. zde, protože není užitečné po každém přepnutí na tabulku s událostmi posílat dotaz na server pro získání dat a proto ve Vue je speciální tag `<keep-alive>`. Bohužel, React nenabízí takový užitečný nastroj.

```
<template>
  <div>
    <div class="table-wrapper">
      <h1>Calendar</h1>
      <transition name="fade" appear>
        <b-button v-show="isHidden" variant="primary" class="my-1"
                  @click="showAddingNewEventForm">
          Add new event
        </b-button>
      </transition>

      <transition name="fade" mode="out-in" appear>
        <keep-alive>
          <component :is="currentOption"
                     @show-calendar-events="showCalendarEvents"/>
        </keep-alive>
      </transition>
    </div>
  </div>
</template>
…
```

Zdroj: vlastní

Následující ukázka bude se týkat konfiguračního souboru směrování ve Vue. V daném frameworku základní myšlenka je založena na vytváření celého pole cest pro každý komponent. Navíc pomocí metody `beforeEnterToComponent()` probíhá ověření autorizace uživatele. Když není, tak uživatel bude přesměrován na hlavní stránku aplikace.

```
import Vue from 'vue';
import VueRouter from 'vue-router';
import store from '@/store';
import Navbar from "@/components/Navbar";
import Calendar from "@/components/user_components/calendar/Calendar";
import Home from "@/components/Home";

Vue.use(VueRouter);

function beforeEnterToComponent(to, from, next) {
    if (store.getters.getAuthenticationState) {
        next();
    } else {
        next({name: 'Home'});
    }
}
export default new VueRouter({
    mode: 'history',
    routes: [
        {
            path: '*',
            component: () => import('@/components/NotFound'),
        },
        {
            path: '/',
            component: Navbar,
            name: 'Navbar',
            children: [
                {
                    path: 'calendar',
                    component: Calendar,
                    name: 'Calendar',
                    beforeEnter: (to, from, next) =>
                        beforeEnterToComponent(to, from, next),
                    components: {
                        calendar: Calendar
                    }
                },
                {
                    path: '',
                    component: Navbar,
                    name: 'Home',
                    components: {
                        home: Home
                    }
                }
            ]
        }
    ]
});
```

Zdroj: vlastní

Oproti tomu React Router od verze 6.0.0 už nepoužívá podobný způsob konfigurace pro směrování. Teď je třeba ve vstupním bodu po nainstalování React Routeru zabalit všechny cesty v komponent `Routes`.

```
import React from 'react';
import RNavbar from "./components/navbar/RNavbar";
import Jumbotron from "./components/jumbotron/Jumbotron";
import {Routes, Route} from 'react-router-dom';
import Calendar from "./components/calendar/Calendar";
import PageNotFound from "./components/notfoundpage/PageNotFound";
import RequireAuth from "./components/RequireAuth";

const App = () => {
    return (
        <>
            <Routes>
                <Route path={'/'} element={<RNavbar/>}>
                    <Route index element={<Jumbotron/>}/>
                    <Route path={'/calendar'} element={
                        <RequireAuth>
                            <Calendar/>
                        </RequireAuth>
                    }/>
                    <Route path={'*'} element={<PageNotFound/>}/>
                </Route>
            </Routes>
        </>
    );
};

export default App;
```

Zdroj: vlastní

Pro zajištěni ověření uživatele potřebné komponenty jsou zabalené ve speciální komponent `RequireAuth`. V něm probíhá stejná kontrola je-li uživatel autorizován s následným přesměrováním na hlavní stránku v negativním případě.

```
import React from 'react';
import {useLocation, Navigate} from "react-router-dom";
import {useSelector} from "react-redux";

const RequireAuth = ({children}) => {
    const location = useLocation();
    const isAuthenticated = useSelector(state =>
        state.authReducer.isAuthenticated);
    if (!isAuthenticated) {
        return <Navigate to={'/'} state={{from: location}} replace={true}/>
    }
    return children;
};

export default RequireAuth;
```

Zdroj: vlastní

ZÁVĚR

Závěrem je třeba říct, že výkon, využití paměti a škálovatelnost jsou pro obě řešení přibližně stejné, z tohoto důvodu nebudou tyto parametry brány jako hlavní důvod při finálním výběru.

Programovat ve Vue můžete začít po přečtení dokumentace nebo po sledování krátkých video lekcí bez nutných znalostí standardu JavaScriptu ES6 a JSXu (Vue ho taky podporuje, ale to není populární). Direktivy ve Vue pomáhají psát kód kratší a čitelnější. HTML šablony jsou přirozenější pro čtení, psaní a taky zahrnují HTML vzor, CSS styly s volitelným omezením pro tento komponent a JS logiku v jednom souboru. V dnešní době je růst popularity Vue zřejmý.

React oproti tomu všemu má větší ekosystém, velkou komunitu a podporu ze strany velké společnosti. V Reactu je absolutně všechno JavaScriptem a proto můžete použít všechny možnosti toho programovacího jazyka. Trh práce nabízí pozice pro více zkušených vývojářů a obecně pracovních nabídek pro React.

Finální výběr použité technologie je na každém uživateli, a nakonec zřejmě rozhodne sympatie k dané knihovně a produktivita při práci s knihovnou, která je také velmi důležitá.

ZDROJOVÉ KÓDY PRAKTICKÉ ČÁSTI BAKALÁŘSKÉ PRÁCE

Zdrojové kódy jsou dostupné na stránce GitHubu autora bakalářské práce[21]. Pro zpuštění je třeba mít naistalovány *Node.js*.

[21] Dostupné z: https://github.com/cjOne7/Bachelor-work-Yarosh

POUŽITÁ LITERATURA

[1] NAZAROVA, Elena. Основы HTML. *HTML5BOOK* [online]. HTML5BOOK.RU, 2014-10-22, 2021-01-01 [cit. 2021-10-19]. Dostupné z: https://html5book.ru/osnovy-html/

[2] JANOVSKÝ, Dušan. Titulek stránky a proč je důležitý. *Jak psát web* [online]. ©1999-2021 [cit. 2021-10-19]. ISSN 1801-0458. Dostupné z: https://www.jakpsatweb.cz/titulek.html

[3] JANOVSKÝ, Dušan. CSS styly – úvod. *Jak psát web* [online]. ©1999-2021 [cit. 2021-10-19]. ISSN 1801-0458. Dostupné z: https://www.jakpsatweb.cz/css/css-uvod.html

[4] CSS Selector Reference. *W3schools* [online] Refsnes Data, ©1999-2021 [cit. 2021-10-19]. Dostupné z: https://www.w3schools.com/cssref/css_selectors.asp

[5] Sass Basics. *Sass* [online]. Sass team, ©2006-2021 [cit. 2021-10-19]. Dostupné z: https://sass-lang.com/guide

[6] ROBBINS, J. N. *Learning Web Design: a Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics.* 5. přeprac. vyd. Sebastopol, USA: O'Reilly Media, 2018-05-29. 790 s. ISBN: 978-1-491-96020-2.

[7] JANOVSKÝ, Dušan. Začlenění skriptu do stránky. *Jak psát web* [online]. ©1999-2021 [cit. 2021-10-20]. ISSN 1801-0458. Dostupné z: https://www.jakpsatweb.cz/javascript/zacleneni.html

[8] KOĎOUSKOVÁ, B. VUE JS: VÝHODY, NEVÝHODY a MOŽNOSTI VYUŽITÍ. *Rascasone* [online] Rascasone s.r.o., ©2021, 2021-07-15 [cit. 2021-10-20]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-framework-vuejs

[9] BARÁŠEK, J. Úvod do frameworku Vue.js. *Vue.js manuál* [online] ©2009-2021, 2020-08-07 [cit. 2021-10-20]. Dostupné z: https://vue.baraja.cz/uvod-do-vue

[10] Overview. *Vue CLI* [online]. Evan You, ©2018, 2019-07-10 [cit. 2021-10-20]. Dostupné z: https://cli.vuejs.org/guide/#overview

[11] Vue Router. *Vue Router* [online] Evan You, Eduardo San Martin Morote, ©2014 [cit. 2021-10-20]. Dostupné z: https://router.vuejs.org/

[12] Installation. *Vue Router* [online]. Evan You, Eduardo San Martin Morote, ©2014 [cit. 2021-10-20]. Dostupné z: https://router.vuejs.org/installation.html

[13] API Reference. *Vue Router* [online]. Evan You, Eduardo San Martin Morote, ©2014 [cit. 2021-10-20]. Dostupné z: https://router.vuejs.org/api/

[14] What is Vuex? *Vuex* [online]. Evan You, ©2015 [cit. 2021-10-20]. Dostupné z: https://vuex.vuejs.org/

[15] Using Pre-Processors. *Vue loader* [online]. Evan You, ©2015 [cit. 2021-10-20]. Dostupné z: https://vue-loader.vuejs.org/guide/pre-processors.html#sass

[16] TypeError: this.getOptions is not a function. *Stack Overflow* [online]. Stack Exchange Inc., 2021-02-06 [cit. 2021-10-22]. Dostupné z: https://stackoverflow.com/questions/66082397/typeerror-this-getoptions-is-not-a-function

[17] How do i add a global scss file in Vue.JS that will compile? *Stack Overflow* [online]. Stack Exchange Inc., 2019-01-12 [cit. 2021-10-22]. Dostupné z: https://stackoverflow.com/questions/54158207/how-do-i-add-a-global-scss-file-in-vue-js-that-will-compile

[18] Directives. *Vue.js* [online]. Evan You, ©2013 [cit. 2021-11-02]. Dostupné z: https://012.vuejs.org/guide/directives.html

[19] Directives. *Vue.js* [online]. Evan You, ©2013 [cit. 2021-11-02]. Dostupné z: https://v3.vuejs.org/api/directives.html

[20] Style Guide: Avoid v-if with v-for. *Vue.js* [online]. Evan You, ©2013 [cit. 2021-11-02]. Dostupné z: https://vuejs.org/v2/style-guide/index.html#Avoid-v-if-with-v-for-essential

[21] GLABAZŇA, Tomáš. Lekce 2 - Komponenty ve Vue.js. *itnetwork.cz* [online]. itnetwork.cz, ©2021 [cit. 2021-11-03]. Dostupné z: https://www.itnetwork.cz/javascript/vuejs/komponenty-ve-vuejs

[22] API: Options/Misc. *Vue.js* [online]. Evan You, ©2013 [cit. 2021-11-04]. Dostupné z: https://vuejs.org/v2/api/#name

[23] Style Guide: Prop definitions. *Vue.js* [online]. Evan You, ©2013 [cit. 2021-11-04]. Dostupné z: https://vuejs.org/v2/style-guide/#Prop-definitions-essential

[24] Props: Prop Validation. *Vue.js* [online]. Evan You, ©2013 [cit. 2021-11-04]. Dostupné z: https://vuejs.org/v2/guide/components-props.html#Prop-Validation

[25] OLAWANLE, Joel. How to Use Props in Vue.js. *freeCodeCamp.org* [online]. 2021-08-11 [cit. 2021-11-04]. Dostupné z: https://www.freecodecamp.org/news/how-to-use-props-in-vuejs/

[26] GLABAZŇA, Tomáš. Lekce 5 - Computed properties a dynamické styly ve Vue.js. *itnetwork.cz* [online]. itnetwork.cz, ©2021 [cit. 2021-11-04]. Dostupné z: https://www.itnetwork.cz/javascript/vuejs/computed-properties-a-dynamicke-styly-ve-vuejs

[27] Computed Properties and Watchers: Computed Caching vs Methods. *Vue.js* [online]. Evan You, ©2013 [cit. 2021-11-04]. Dostupné z: https://vuejs.org/v2/guide/computed.html#Computed-Caching-vs-Methods

[28] API: methods. *Vue.js* [online]. Evan You, ©2013 [cit. 2021-11-04]. Dostupné z: https://vuejs.org/v2/api/#methods

[29] Filters. *Vue.js* [online]. Evan You, ©2013 [cit. 2021-11-04]. Dostupné z: https://vuejs.org/v2/guide/filters.html

[30] Adding Filters in VueJS. *GeeksforGeeks* [online]. 2020-11-20 [cit. 2021-11-04]. Dostupné z: https://www.geeksforgeeks.org/adding-filters-in-vuejs/

[31] The Vue Instance: Instance Lifecycle Hooks. *Vue.js* [online]. Evan You, ©2013 [cit. 2021-11-04]. Dostupné z: https://vuejs.org/v2/guide/instance.html#Instance-Lifecycle-Hooks

[32] MROZEK, Jakub. JavaScript na serveru: moduly a npm. *zdrojak.cz* [online]. Devel.cz Lab s.r.o., 2012-10-12 [cit. 2021-11-06]. ISSN 1803-5620. Dostupné z: https://zdrojak.cz/clanky/javascript-na-serveru-moduly-a-node-package-manager/

[33] Overview of the Microsoft Authentication Library (MSAL). *Microsoft* [online]. Microsoft, ©2021, 2021-08-19 [cit. 2021-11-06]. Dostupné z: https://docs.microsoft.com/en-us/azure/active-directory/develop/msal-overview

[34] Microsoft Authentication Library for JavaScript (MSAL.js) 2.0 for Browser-Based Single-Page Applications. *npm* [online]. 2021-11-02 [cit. 2021-11-06]. Dostupné z: https://www.npmjs.com/package/@azure/msal-browser?activeTab=readme

[35] Microsoft Graph API. *Microsoft* [online]. Microsoft, ©2021, 2021-10-14 [cit. 2021-11-06]. Dostupné z: https://docs.microsoft.com/en-us/azure/active-directory/develop/microsoft-graph-intro

[36] Microsoft Graph JavaScript Client Library. *npm* [online]. 2021-08-09 [cit. 2021-11-06]. Dostupné z: https://www.npmjs.com/package/@microsoft/microsoft-graph-client

[37] Getting Started. *BootstrapVue* [online]. ©2016-2020 [cit. 2021-11-06]. Dostupné z: https://bootstrap-vue.org/docs

[38] Introduction. *React Bootstrap* [online]. ©2014 [cit. 2021-11-06]. Dostupné z: https://react-bootstrap.github.io/getting-started/introduction

[39] Install Font Awesome 5 with NPM. *Stack Overflow* [online]. Stack Exchange Inc., 2018-09-22 [cit. 2021-11-06]. Dostupné z: https://stackoverflow.com/questions/52455614/install-font-awesome-5-with-npm

[40] Vue-moment. *npm* [online]. 2019-12-22 [cit. 2021-11-06]. Dostupné z: https://www.npmjs.com/package/vue-moment

[41] React-moment. *npm* [online]. 2020-12-19 [cit. 2021-11-06]. Dostupné z: https://www.npmjs.com/package/react-moment

[42] React: a JavaScript library for building user interfaces. *React* [online]. Meta Platforms, ©2022 [cit. 2022-04-13]. Dostupné z: https://reactjs.org/

[43] Introducing JSX. *React* [online]. Meta Platforms, ©2022 [cit. 2022-04-14]. Dostupné z: https://reactjs.org/docs/introducing-jsx.html

[44] React.Component. *React* [online]. Meta Platforms, ©2022 [cit. 2022-04-15]. Dostupné z: https://reactjs.org/docs/react-component.html

[45] Introducing Hooks. *React* [online]. Meta Platforms, ©2022 [cit. 2022-04-15]. Dostupné z: https://reactjs.org/docs/hooks-intro.html

[46] Hooks at a Glance: Rules of Hooks. *React* [online]. Meta Platforms, ©2022 [cit. 2022-04-15]. Dostupné z: https://reactjs.org/docs/hooks-overview.html#rules-of-hooks

[47] Hooks at a Glance: Effect Hook. *React* [online]. Meta Platforms, ©2022 [cit. 2022-04-15]. Dostupné z: https://reactjs.org/docs/hooks-overview.html#effect-hook

[48] Hooks API Reference. *React* [online]. Meta Platforms, ©2022 [cit. 2022-04-15]. Dostupné z: https://reactjs.org/docs/hooks-reference.html

[49] Main Concepts. *React* [online]. Remix, ©2022 [cit. 2022-04-16]. Dostupné z: https://reactrouter.com/docs/en/v6/getting-started/concepts

[50] Installation. *React* [online]. Remix, ©2022 [cit. 2022-04-16]. Dostupné z: https://reactrouter.com/docs/en/v6/getting-started/installation

[51] ABRAMOV, Dan. Getting Started with Redux: Basic Example. *Redux* [online]. ©2015-2022 [cit. 2022-04-16]. Dostupné z: https://redux.js.org/introduction/getting-started#basic-example

[52] Understanding Asynchronous Redux Actions with Redux Thunk. *DigitalOcean* [online]. 2018-06-19, 2020-10-08 [cit. 2022-04-16]. Dostupné z: https://www.digitalocean.com/community/tutorials/redux-redux-thunk

[53] ABRAMOV, Dan. Getting Started with React Redux. *React Redux* [online]. ©2015-2022 [cit. 2022-04-16]. Dostupné z: https://react-redux.js.org/introduction/getting-started

[54] Принципы функционального программирования в JavaScript. *Habr* [online]. ©2006-2022, 2018-10-26 [cit. 2022-04-17]. Dostupné z: https://habr.com/ru/company/ruvds/blog/434112/

[55] vue CDN files. *jsdelivr* [online]. ©2012-2022 [cit. 2022-04-19]. Dostupné z: https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/

[56] UNPKG: react. *UNPKG* [online]. ©2022 [cit. 2022-04-19]. Dostupné z: https://unpkg.com/browse/react@18.0.0/umd/

[57] UNPKG: react-dom. *UNPKG* [online]. ©2022 [cit. 2022-04-19]. Dostupné z: https://unpkg.com/browse/react-dom@18.0.0/umd/

[58] HEMANT, Rai. Moving From Angular to Vue : a vuetiful journey. *Medium* [online]. 2017-05-01 [cit. 2022-04-19]. Dostupné z: https://medium.com/@Hemantisme/moving-from-angular-to-vue-a-vuetiful-journey-c29842ab2039

[59] Лучший JavaScript-фреймворк 2021: React или Vue? *Evrone* [online]. ©2008-2022 [cit. 2022-04-19]. Dostupné z: https://evrone.ru/react-vs-vue

[60] Why does Vue have more GitHub stars than React? *Quora*. [online]. ©2022 [cit. 2022-04-23]. Dostupné z: https://www.quora.com/Why-does-Vue-have-more-GitHub-stars-than-React

[61] Front-end frameworks popularity (React, Vue, Angular and Svelte). *GitHub*. [online]. ©2022 [cit. 2022-04-23]. Dostupné z: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190

[62] NOWAK, Maja. Reasons Why Vue.js Is Getting More Traction Every Month. *monterail* [online]. 2018-12-19 [cit. 2022-04-23]. Dostupné z: https://www.monterail.com/blog/reasons-why-vuejs-is-popular

[63] Facebook/React: License. *GitHub* [online]. ©2022 [cit. 2022-04-23]. Dostupné z: https://github.com/facebook/react#license

[64] Vuejs/Vue:License. *GitHub* [online]. ©2022 [cit. 2022-04-23]. Dostupné z: https://github.com/vuejs/vue#license

[65] Tutorial: Intro to React: Prerequisites. *React* [online]. Meta Platforms, ©2022 [cit. 2022-04-23]. Dostupné z: https://reactjs.org/tutorial/tutorial.html#prerequisites

[66] Сравнение с другими фреймворками. *Vue.js* [online]. Evan You, ©2013 [cit. 2022-04-23]. Dostupné z: https://ru.vuejs.org/v2/guide/comparison.html

[67] HUERGO, Francisco. Stateful and Stateless Components in React. *Programming with Mosh* [online]. 2019-04-10 [cit. 2022-05-06]. Dostupné z: https://programmingwithmosh.com/javascript/stateful-stateless-components-react/

[68] connect(). *React Redux* [online]. ©2015-2022 [cit. 2022-05-06]. Dostupné z: https://react-redux.js.org/api/connect