SLEZSKÁ UNIVERZITA V OPAVĚ
Filozoficko-přírodovědecká fakulta v Opavě

# BAKALÁŘSKÁ PRÁCE

Opava 2024 Lukáš Sukeník

SLEZSKÁ UNIVERZITA V OPAVĚ
Filozoficko-přírodovědecká fakulta v Opavě

Lukáš Sukeník

Studijní program: Moderní informatika
Specializace: Informační a komunikační technologie

# Porovnání SPA frontend frameworků

# Comparison of SPA frontend frameworks

Bakalářská práce

Opava 2024

Vedoucí bakalářské práce:
doc. RNDr. Lucie Ciencialová, Ph.D.

**Abstrakt**

Text abstraktu v češtině. Rozsah by měl být 50 až 100 slov. Abstrakt není cíl práce, zde stručně popište, co čtenář má na následujících stránkách očekávat. Typické formulace: „V práci se zabýváme...“, „Tato bakalářská práce pojednává o...“, „součástí je“, „je provedena analýza“, „praktickou částí práce je aplikace xxx“ ... Prostě napište stručný souhrn či charakteristiku obsahu práce.

**Klíčová slova**

Napište 5–8 klíčových slov v českém jazyce (v jednotném čísle, první pád atd.), měla by vystihovat téma práce. Slova oddělujte čárkou. Snažte se vystihnout nejdůležitější pojmy vystihující práci.

**Abstract**

Anglická verze abstraktu by měla odpovídat české verzi, třebaže nemusí být úplně doslova. Když nutně potřebujete automatický překlad, použijte raději
https://www.deepl.com/cs/translator,
je lepší než Google Translator. Není nutno překládat doslova.

**Keywords**

Anglická obdoba českého seznamu klíčových slov.

**Čestné prohlášení**

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškerou literaturu a další zdroje, z nichž jsem při zpracování čerpal, v práci řádně cituji a jsou uvedeny v seznamu použité literatury.

V Opavě dne 18. října 2023

.....

Lukáš Sukeník

**Poděkování**

Rád bych poděkoval za odborné vedení, rady a cenné poznatky k danému tématu vedoucímu práce ..... Také bych rád poděkoval mé rodině a přátelům za podporu a pomoc během mého studia.

# Obsah

**Úvod** **1**

**1 Webové aplikace** **2**

**2 Analýza frameworků** **3**

2.1 React . . . 3

2.1.1 Komponenty . . . 4

2.1.2 JSX . . . 4

2.1.3 Správa stavů . . . 5

2.1.4 Hooky . . . 5

2.1.5 Životní cyklus . . . 5

2.1.6 State management . . . 6

2.1.7 Routování . . . 6

2.1.8 Ekosystém . . . 6

2.2 Svelte . . . 6

2.3 Vue . . . 6

2.4 Porovnání . . . 6

**3 Testování frameworků** **8**

**4 Ukázková kapitola** **9**

4.1 Struktura a formát . . . 9

4.1.1 Jak strukturovat práci . . . 9

4.2 Obrázky a tabulky . . . 9

4.2.1 Vkládání ukázkového kódu . . . 10

4.3 Vyznačování pojmů v textu . . . 11

4.4 Odrážky, číslování, pojmenované odstavce . . . 11

**5 Práce se zdroji** **13**

5.1 Seznam použité literatury . . . 13

5.2 Citace . . . 13

5.3 Parafráze . . . 14

**Závěr** **16**

**Seznam použité literatury** **17**

**Seznam obrázků** **19**

**Seznam tabulek** **20**

**Seznam zkratek** **21**

**Přílohy** **22**

# 2 Analýza frameworků

Text druhé kapitoly.

- odůvodnit výběr frameworků:
  - https://survey.stackoverflow.co/2023/
- analýza frameworků:
  - co to je
  - stručná historie
  - kdo jej vytvořil
  - kdo jej používá
  - jak se používá
  - popularita ve společnosti
- srování frameworků dle:
  1. Component-Based architektura (HTML, CSS, JS/TS)
  2. Šablony
  3. Předávání vlastností (props)
  4. Správa stavů
  5. Life-cycle
  6. State management
  7. Reaktivita ???
  8. DOM
  9. Routování
  10. Server-Side Rendering
  11. Kompilace zdrojových souborů
  12. Ekosystém

## 2.1 React

Pod pojmem React rozumíme open-source JavaScript framework, který vyvinula a dále vyvíjí společnost Meta (dříve Facebook). Podle [2] jde spíše o knihovnu funkcí, než-li o komplexní nástroj pro tvorbu webových aplikací (framework). Tato technologie se používá pro vývoj interaktivních uživatelských rozhraní a webových aplikací.[5]

První kořeny Reactu sahají až do roku 2010, kdy tehdejší společnost Facebook přidala novou technologii XHP do PHP. Jde o možnost znovu použít určitý blok kódu, stejného principu posléze využívá i React. Následně Jordan Walke vytvořil FaxJS, jenž byl prvním prototypem Reactu. O rok později byl přejmenován na React a začal jej využívat Facebook. V roce 2013 byl na konferenci JS ConfUS představen široké veřejnosti a stal se open-source.

Od roku 2014 vývojáři představují nespočet vylepšení samotné knihovny, stejně jako spoustu rozšíření pro zlepšení vývojových procesů. Kolem roku 2015 postupně React nabývá na popularitě i celkové stabilitě. Následně je představen také React Native, což je framework pro vývoj nativních aplikací. V dnešní době je React využíván společnostmi každého rozsahu po celém světě. Z těch největších jde například o Metu, Uber, Twitter a Airbnb.[2, 4]

### 2.1.1 Komponenty

Hlavním stavebním kamenem Reactu jsou komponenty, jež představují nezávislé, vnořitelné a opakovaně použitelné bloky kódu. Komponentu v Reactu tvoří JavaScript funkce a HTML šablona. Validně seskládané komponenty poté tvoří webovou aplikaci. V Reactu se můžeme setkat s funkčními a třídními komponentami. Vytváření třídních komponent oficiální dokumentace nedoporučuje. Pro komunikaci mezi komponentami se používá předávání vlastností (props), přes které je možné předávat hodnoty jakýchkoli datových typů. Výstup komponent tvoří elementy ve formě JSX. Tyto elementy obsahují informace o vzhledu a funkcionalitě dané komponenty.[2, 8]

### 2.1.2 JSX

Název JSX kombinuje zkratku jazyka JavaScript – JS a počáteční písmeno ze zkratky XML. Konkrétně jde o syntaktické rozšíření, které vývojářům umožňuje tvořit React elementy pomocí hypertextového značkovacího jazyku přímo v JavaScriptu. V rámci JSX pak je možné dynamicky vykreslovat obsah na základě logiky definované pomocí JavaScriptových hodnot. Při kompilaci se JSX překládá do JavaScriptu pomocí nástroje Babel.[2, 8]

### 2.1.3 Správa stavů

Stav lze definovat jako lokální vnitřní vlastnost či proměnnou dané komponenty, jež představuje základní mechanismus pro uchovávání a aktualizaci dat. Pro aktualizaci komponenty je tedy nutné stav změnit. React pak na tuto skutečnost zareaguje a vyvolá tzv. re-render neboli překreslení komponenty s novými daty.

Za účelem ukládání stavu se využívá hook (funkce) useState. Ten poskytuje stavovou proměnnou, přes kterou se dostaneme k aktuálnímu stavu. Dále useState poskytuje state setter funkci, díky které můžeme stav aktualizovat. Jediný argument useState definuje počátační hodnotu daného stavu.[6, 8]

### 2.1.4 Hooky

Specifickou funkcionalitou pro React jsou tzv. hooky, které byly do Reactu přidány až ve verzi 16.8.0.[9] Hook je definován jako funkce, která obohacuje komponenty pomocí předdefinovaných funkcionalit. Jedním z nejpoužívanejších hooků je useState. Vývojáři mohou používat již zabudované hooky, nebo si vytvářet své vlastní s pomocí předdefinovaných hooků. Mezi zabudované hooky patří např. useEffect, useMemo, useCallback, useRef, useContext.[8]

### 2.1.5 Životní cyklus

Životní cyklus komponenty je sekvence událostí, jež nastanou mezi vytvořením a zničením komponenty. Ve třídních komponentách existovaly speciální metody, tzv. lifecycle metody, starající se o provedení určité části kódu při daném okamžiku v životě komponenty. Nyní React disponuje pár hooky, které umožňují provádět side-effects podobně jako lifecycle metody.

O momentu, kdy je komponenta přidána na obrazovku, mluvíme jako o namontování (mount) komponenty. Při změně stavu či obdržení nových parametrů hovoříme o aktualizaci (update) komponenty. A v neposlední řadě okamžik, kdy je komponenta odstraněna z obrazovky, nazýváme odmontování (unmount) komponenty.[7, 8]

### 2.1.6 State management

Základní práce se stavy spočívá v lokálních stavech komponent a následným předáváním stavu do potomků či rodičů. V případě, že potřebujeme sdílet stav mezi komponentami, měli bychom zvážit odlišné řešení. React sám o sobě disponuje pouze základním řešením, kterému říká Context API. Context umožňuje sdílet data celému podstromu dané komponenty. To se může hodit například při vytváření barevných módů aplikace, sdílení informace o přihlášeném uživateli, anebo routování.[8]

Správa stavů v komplexních aplikacích se stává výzvou. Problémy začínají při potřebě sdílení identických dat mezi větším množstvím konzumentů. Existuje však mnoho knihoven třetích stran, které vývojáři využívají pro usnadnění manipulace se stavy. Společné cíle state management knihoven spočívají v ukládání a získávání globálního stavu, jednodušší správě stavů a rozšiřitelnosti aplikace. Mezi tyto knihovny patří kupříkladu Redux, MobX, Recoil nebo Jotai.[1, 3]

### 2.1.7 Routování

React nemá žádný nativní standard pro routování. Podle [2] je React Router jedním z nejvíce populárních řešení pro React. Knihovna React Router umožňuje nastavení jednotlivých cest aplikace. Zajišťuje tedy routování na straně klienta. Klasické webové stránky při změně URL pokaždé žádají o nové dokumenty. Routování na straně klienta může provést aktualizaci stránky bez dalších duplikátních requestů. Při vyžádání dané cesty pak můžeme na stránce vykreslit požadovaný obsah a zažádat pouze o data potřebné pro vykreslení. Výstup činí rychlejší uživatelskou zkušenost, jelikož prohlížeč nevyžaduje nové dokumenty a nemusí vyhodnocovat kaskádové styly či JavaScript.[2, 10]

### 2.1.8 Ekosystém

## 2.2 Svelte

## 2.3 Vue

## 2.4 Porovnání

- co zjistím na první pohled, platforma,

- jako bych si četl reklamu ...,
- prvotní srovnání.

# Seznam obrázků

1 Ukázka vložení titulku s označením zdroje . . . . . 10

# Seznam tabulek

1 Ukázka tabulky . . . . . 10

# Přílohy

Do tohoto seznamu napište přílohy vložené přímo do této práce a také seznam elektronických příloh, které se vkládají přímo do archivu závěrečné práce v informačním systému zároveň se souborem závěrečné práce. Elektronickými přílohami mohou být například soubory zdrojového kódu aplikace či webových stránek, předpřipravený produkt (spustitelný soubor, kontejner apod.), vytvořená metodická příručka, tutoriál... (tento text odstraňte)

- Přílohy v souboru závěrečné práce:
  - Příloha A xxxx
  -
- Elektronické přílohy:
  - Příloha A xxxx
  -