**ŽILINSKÁ UNIVERZITA V ŽILINE**

FAKULTA RIADENIA A INFORMATIKY

BAKALÁRSKA PRÁCA

MARTIN KUMECKÝ

**Návrh off-line webovej aplikácie pre riadenie ventilačnej techniky**

Vedúci práce: Ing. Matej Meško, PhD.

Registračné číslo: 57/2017

Ministerské číslo: 28360720181057

Žilina, 2018

**ŽILINSKÁ UNIVERZITA V ŽILINE**

FAKULTA RIADENIA A INFORMATIKY

BAKALÁRSKA PRÁCA

ŠTUDIJNÝ ODBOR: Informatika

MARTIN KUMECKÝ

**Návrh off-line webovej aplikácie pre riadenie ventilačnej techniky**

Žilinská univerzita v Žiline

Fakulta riadenia a informatiky

Žilina, 2018

#### [Čestné vyhlásenie](Pokyny_pre_vypracovanie_ZP.docx#Čestné_vyhlásenie)

Čestne prehlasujem, že som bakalársku prácu *Návrh off-line webovej aplikácie pre riadenie ventilačnej techniky* vypracoval samostatne na základe vlastných teoretických a praktických poznatkov, pod odborným vedením vedúceho bakalárskej práce a používal som len uvedenú odbornú literatúru nachádzajúcu sa v zozname použitej literatúry.

V Žiline Dňa:................................. Podpis:................................

#### [Poďakovanie](Pokyny_pre_vypracovanie_ZP.docx#Poďakovanie)

Chcem sa poďakovať ...

[ABSTRAKT V ŠTÁTNOM JAZYKU](Pokyny_pre_vypracovanie_ZP.docx" \l "Abstrakt)

KUMECKÝ, Martin: *Návrh off-line webovej aplikácie pre riadenie ventilačnej techniky*. [Bakalárska práca]. – Žilinská univerzita v Žiline. Fakulta riadenia a informatiky; Študijný program: Informatika. – Vedúci práce: Ing. Matej Meško, PhD. – Stupeň odbornej kvalifikácie: Bakalár informatiky – Mesto: FRI ŽU v Žiline, 2018. Počet strán (napr. 35 s.)

[Vysvetlenie a príklad.](Pokyny_pre_vypracovanie_ZP.docx#Príklad_abstrakt)

[**Kľúčové slová**](Pokyny_pre_vypracovanie_ZP.docx#Kľúčové_slová)**:** aplikácia, web

[ABSTRAKT V CUDZOM JAZYKU](Pokyny_pre_vypracovanie_ZP.docx" \l "Abstrakt_CJ)

[Vysvetlenie.](Pokyny_pre_vypracovanie_ZP.docx#Abstrakt_CJ)

**Key words:**

[Obsah](Pokyny_pre_vypracovanie_ZP.docx#Obsah)

Zoznam obrázkov 9

Zoznam tabuliek 10

Zoznam skratiek 11

Úvod 12

1 Najpoužívanejšie reaktívne frameworky 15

1.1 Angular 15

1.1.1 História verzií 15

1.1.2 TypeScript 16

1.1.3 Inštalácia a používanie 18

1.2 React 21

1.3 Vue 21

1.4 Porovnanie 21

2 Zber požiadaviek 22

3 Analýza požiadaviek 23

4 Implementácia 24

Záver 25

Zoznam použitej literatúry 26

Zoznam príloh 27

Prílohy 28

Príloha A: Názov prílohy 29

Príloha B: Obsah DVD 30

# [Zoznam obrázkov](Pokyny_pre_vypracovanie_ZP.docx" \l "Zoznam_obrázkov)

[Obrázok 1 Ukážka kódu v jazyku TypeScript 17](#_Toc507328376)

[Obrázok 2 Ukážka rovnakého kódu v jazyku JavaScript 17](#_Toc507328377)

# [Zoznam tabuliek](Pokyny_pre_vypracovanie_ZP.docx" \l "Zoznam_obrázkov)

[Tabuľka 1 Základné príkazy v príkazovom riadku 18](#_Toc508368481)

[Tabuľka 2 Životný cyklus komponentu 19](#_Toc508368482)

# [Zoznam skratiek](Pokyny_pre_vypracovanie_ZP.docx" \l "Zoznam_skratiek)

SPA Single page application

# [Úvod](Pokyny_pre_vypracovanie_ZP.docx#Úvod)

Kedysi, pri začiatkoch samotného internetu boli webové stránky jednoduché dokumenty, zobrazujúce poväčšine statický text, doplnený obrázkami. No postupom času sa tento statický text zmenil na dynamický, vznikli prehrávače videí, do stránok bolo možné vložiť najrôznejšie animácie a celkovo komplexnosť webovej stránky rástla. Avšak neskôr, webový prehliadač už nebol iba prostriedok, pomocou ktorého sa dostaneme k informáciám na internete. Stal sa aj zobrazovacím nástrojom pre rôzne aplikácie, využívané v rôznych odvetviach priemyslu, účtovníctve, štátnej správe a to práve z dôvodu, že aplikácie nebolo nutné inštalovať jednotlivo na každú pracovnú stanicu, ale aplikácie boli dostupné všade, kde ich bolo potreba.

Vývoj takýchto webových aplikácií bolo možné dosiahnuť aj za pomoci skriptovacieho jazyka JavaScript, ktorého popularita od svojho vzniku neustále rástla. Taktiež rástla aj komplexnosť webových aplikácií, a preto boli nároky na vývojárov, či už časové alebo vedomostné, čoraz väčšie. To v súčasnosti vyústilo do vzniku mnohých JavaScriptových frameworkov, ktoré majú za úlohu zjednodušiť a urýchliť vývoj webových stránok.

V prvej kapitole preskúmam možnosti v súčasnosti najpoužívanejších reaktívnych frameworkov. Pri skúmaní sa zameriam hlavne na zložitosť daného frameworku, jeho silné a slabé stránky. V závere kapitoly pomocou tabuľky zhrniem a porovnám hlavné poznatky, ako aj výhody jednotlivých frameworkov.

Hlavnou úlohou tejto práce však je vytvoriť funkčnú webovú aplikáciu pre diaľkové ovládanie ventilačného systému za pomoci jedného z reaktívnych frameworkov. Okrem základného ovládania je aplikácia rozšírená aj o možnosť sledovania dôležitých atribútov, rovnako ako aj ich štatistických vyhodnotení, hlavne v podobe grafu. Nasadenie výslednej aplikácie v praxi bude mať za následok zvýšenie komfortu pri prevádzke objektu.

Úvodný text sa píše v štýle odseku textu Normálny. Ten je v tejto šablóne nastavený a neodporúčame ho meniť.

Zásady spracovania záverečnej práce sú uvedené [TU](Pokyny_pre_vypracovanie_ZP.docx#Zásady_spracovania_ZP).

Formálna úprava záverečných prác je uvedená [TU](Pokyny_pre_vypracovanie_ZP.docx#Formálna_úprava_ZP)

#### Pokyny pre používanie šablóny

V šablóne sú pripravené štyri typy nadpisov:

* **Nadpis 1** je číslovaný a začína vždy na novej stránke (dá sa priradiť klávesom **F5**)
* **Nadpis 2** nadväzuje číslovaním na Nadpis 1 (**F6**)
* **Nadpis 3** nadväzuje číslovaním na predchádzajúce nadpisy (**F7**)
* **Nadpis 4** je bez čísla (**F8**)

V šablóne sú pripravené ďalšie štýly odseku textu, pri ktorých odporúčame nemeniť ich nastavené vlastnosti:

* **Normálny** je určený na písanie bežného textu (**F4**)
* **Obrázok** je zarovnaný na stred a určený pre obrázky a ich popis (**F10**)
* **Skratky** je určený na písanie zoznamu skratiek použitých v dokumente
* **Literatúra** je určený na písanie zoznamu literatúry (ešte nie je pripravený)
* **Tabuľka** je určený na vkladanie textu do tabuliek
* **Typ práce** je určený na vypísanie typu práce do hlavičky

#### Pokyny pre prácu s ilustráciami

Ilustrácie sú obrázky obsahujúce grafy, diagramy, mapy, schémy a pod. Nie je potrebné rozlišovať rozličné typy ilustrácií, stačí, ak sa všetky označia ako „Obrázok”.

#### Vloženie ilustrácie

Obrázky a tabuľky vkladáme v tejto šablóne takto:

1. Karta **Vložiť** → **Obrázok**
2. Na obrázku klikneme na pravé tlačidlo myši → **Vložiť popis**
3. Do poľa **Popis** dopíšeme názov obrázku
4. Na obrázok a jeho popis použijeme štýl **Obrázok** (**F10**)

V texte sa na vytvorené obrázky a tabuľky odkazujeme krížovými odkazmi. Krížový odkaz na obrázky a tabuľky vytvoríme takto:

1. Nastavíme sa kurzorom tam, kde sa má objaviť odkaz,
2. Na karte **Vložiť** → **Krížový odkaz**
3. V dialógovom okne pre **Krížový odkaz** v poli **Typ odkazu** vyberieme **Tabuľka** alebo **Obrázok**
4. V poli **Vložiť odkaz** vyberieme **Iba menovka a číslo**
5. Zo zoznamu popisov vyberieme tabuľku alebo obrázok a zaškrtneme **Hypertextový odkaz**
6. Klikneme na **Vložiť**

#### Aktualizácia krížových odkazov a popisov obrázkov

Pokiaľ v texte vymažete alebo pridáte ilustráciu, tak je potrebné prečíslovanie ilustrácií a krížových odkazov:

1. Vyberte celý dokument (**Ctrl + A**)
2. Stlačte kláves **F9**

Obrázky by mali byť kreslené v rovnakom štýle s popisom v jazyku práce. Podrobnejší popis pre prácu s ilustráciami nájdete [tu](Pokyny_pre_vypracovanie_ZP.docx#Ilustrácie).

#### Pokyny pre prácu s tabuľkami

Každá tabuľka musí mať poradové číslo a titulok, umiestnený zvyčajne nad tabuľkou, pričom titulok je zarovnaný na pravú stranu tabuľky. Tabuľky by mali byť rovnako naformátované s rovnakým vzhľadom buniek.

Podrobnejší popis pre prácu s tabuľkami nájdete [tu](Pokyny_pre_vypracovanie_ZP.docx#Tabuľky).

#### Zoznam použitej literatúry

Bibliografické odkazy vkladajte cez kartu **Referencie,** zoskupenie nástrojov **Citácie a bibliografia**.

# [Najpoužívanejšie](Pokyny_pre_vypracovanie_ZP.docx#Súčasný_stav_riešenia) reaktívne frameworky

V súčasnosti existuje nespočetné množstvo frameworkov, ktorými je možné uľahčiť si prácu s front-end časťou webových aplikácií. Ich hlavnou výhodou oproti používaniu obyčajnej kombinácie HTML, JavaScriptu a CSS je rýchlosť vývoja aplikácií.

## Angular

Angular je JavaScriptový framework, vyvíjaný a podporovaný spoločnosťou Google. V súčasnosti ide o plnohodnotný framework, navrhnutý podľa architektúry service/controller, čo znamená, že na rozdiel od knižníc, ako napr. jQuery, vie ponúknuť presmerovania pomocou vlastného router modulu. Pomocou Agularu je možné vytvoriť rôzne SPA a na dnešnom trhu je vývoj práve týchto aplikácií žiadaný. Pre svoj objektovo orientovaný prístup je vo veľkej miere využívaný vo väčších projektoch, kde je potrebné, aby na vývoji pracoval tím viacerých vývojárov. Dôležité je tiež spomenúť fakt, že spomedzi vybraných frameworkov je Angular najzložitejší z pohľadu učenia sa.

### História verzií

Angular vznikol v roku 2009 pod názvom AngularJS. Bol v podstate úvodným frameworkom pre SPA, teda rozšíril HTML o možnosť vývoja aplikácií, ktoré by sa za iných okolností programovali pre desktopové operačné systémy. Do vývojárskeho sveta priniesol nové pojmy, ako napríklad Two Way Data-Binding, vlasntú implementáciu Depencency Injection alebo znovupoužiteľnosť komponentov pri vývoji. Zároveň bol však navrhnutý podľa archtektúry MVC – model, view, controller. Uvedené pojmy si vysvetlíme neskôr v tejto kapitole.

Jeho nástupcom bol Angular 2, ktorý sa od svojho predchodcu významne líšil. V tejto verzii už naďalej nebola používaná architektúra MVC, ale nahradila ju doteraz používaná service/controller architektúra. Zmena architektúry zapríčinila aj nemožnosť spätnej kompatibility medzi verziou 1 a 2, čo vo väčšine prípadov zapríčinilo, že vývojári boli nútení prepísať aplikačný kód odznova. Ďalšia veľká zmena nastala aj v používanom programovacom jazyku – JavaScript bol nahradený jazykom TypeScript, ktorý má pomáhať pri písaní robustného a štruktúrovaného kódu. Vzhľadom ku rozrastajúcemu sa počtu mobilných zariadení, Angular 2 sa tiež prispôsobil tejto platforme vylepšeným renderovaním komponentov.

Po verzii 2 nasledovala Angular verzia 4. Toto zdanlivo nelogické číslovanie je spôsobené tým, že už Angular 2 používal komponent Angular Router s verziou 3.x a teda vývojári chceli jednoducho zosynchronizovať číslovanie. Verzia 4 priniesla zmeny vo View Engine, čo viedlo k redukcii veľkosti generovaného kódu pre jednotlivé komponenty až okolo 60%. Druhou veľkou zmenou bolo osamostatnenie animácií, ktoré boli vytiahnuté z hlavného balíčka @anglular/core a bol im priradený balík @angular/platform-browser/animations. Ak teda vývojár nechcel poúžívať v aplikácii animácie, nemuseli sa zbytočne k nej pripájať. Uvedené zmeny praxi znamenali zrýchlenie celého frameworku. Výhodou bolo tiež zachovanie spätnej kompatibility s verziou 2.

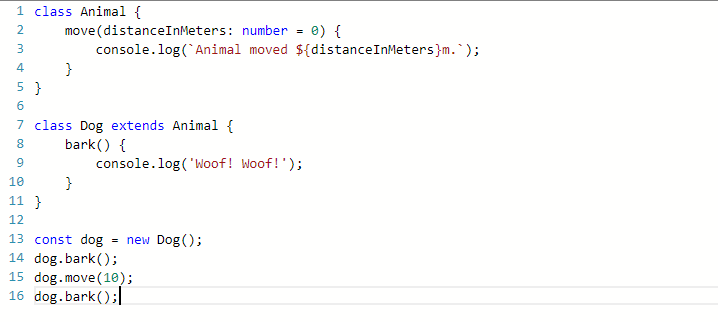
V čase písania tejto práce existuje najnovšia verzia s verziovým číslom 5.2. Medzi najdôležitejšie zmeny, ktoré priniesol Angular 5 patrí aj možnosť viacnásobného pomenovania komponentov aplikácie. Výhodu tohto spôsobu pomenúvania zdôvodňuje Otemuyiwa vo svojom článku: „Export komponentov s viacerými názvami môže používateľom pomôcť pri migrácii bez porušenia zmien.“[[1]](#footnote-1). Okrem toho Angular 5 obsahuje medzinárodné formáty čísel alebo dátumov a ako aj u predošlých verzií prebehla aj optimalizácia frameworku.

### TypeScript

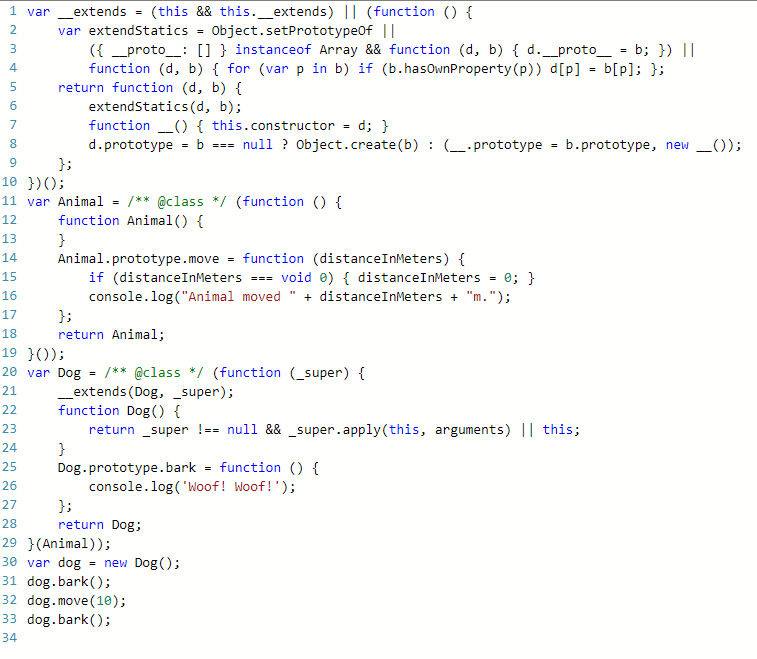
Ako som už vyššie spomenul, Angular využíva TypeScript ako svoj hlavný programovací jazyk. TypeScript funguje ako nadstavba JavaScriptu a jeho kód sa do JavaScriptu aj kompiluje. Vznikol pod záštitou spoločnosti Microsoft a od svojej prvej verzie prešiel viacerými zmenami. Pre jednoduchší vývoj aplikácií, Microsoft pridal podporu jazyka TypeScript aj do svojho vývojového prostredia Visual Studio.

Angular ho však využíva pre jeho objektový prístup. TypeScript pozná pojmy ako trieda, rozhranie alebo dedičnosť. Rovnako sa stretneme aj s typovou kontrolou, ktorá v JavaScripte chýba.

Nižšie je ukážka zdrojového kódu, písaného jazykom TypeScript a pre porovnanie, uvádzame aj ukážku rovnakého kódu v jazyku JavaScript.



Obrázok 1 Ukážka kódu v jazyku TypeScript



Obrázok 2 Ukážka rovnakého kódu v jazyku JavaScript

### Inštalácia a používanie

Inštalácia frameworku prebieha pomocou príkazového riadku. Nevyhnutným základom pre používanie Angularu je inštalácia Node.js a npm, čo je vlastne veľký register balíčkov od rôznych vývojárov, ktorí chcú zdieľať svoje open-source projekty. V nasledujúcej tabuľke sú uvedené základné príkazy pre používanie Angularu.

Tabuľka 1 Základné príkazy v príkazovom riadku

|  |  |
| --- | --- |
| npm install –g @angular/cli | Inštalácia balíka Angular globálne na pevný disk. Príkaz sa zadáva iba raz pri samotnej inštalácii frameworku |
| ng new nazov-aplikacie | Vytvorenie nového projektu, kde nazov-aplikacie je ľubovoľný názov novej aplikácie. |
| ng serve | Spustenie servera. Častý parameter príkazu je –open (skrátene -o). Server je spúšťaný na porte 4200, pokiaľ nie je určené inak. |
| ng generate component nazov | Vygenerovanie nového komponentu. Názov môže byť ľubovoľný |
| ng generate service nazov | Vygenerovanie nového servisu. Názov môže byť ľubovoľný |

Základnými časťami Angular aplikácie sú component a service. Component je základným stavebným blokom aplikácií. Pomocou neho je možné zobraziť dáta na obrazovku, získavať vstupy od používateľa a následne može component na tieto vstupy reagovať. Component je definovaný nasledujúcimi súbormi:

1. app.component.ts – kód tried componentu, napísaný v TypeScripte. Obsahuje logiku komponentu alebo výpočty
2. app.component.html – šablóna komponentu, písaná v HTML
3. app.component.css - štýlovanie, týkajúce sa len daného komponentu

Vzhľad HTML a CSS dokumentov je takmer identický s ostatnými dokumentami, avšak TypeScript dokument komponentu vyzerá nasledovne:

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-heroes',

templateUrl: './heroes.component.html',

styleUrls: ['./heroes.component.css']

})

export class HeroesComponent implements OnInit {

constructor() { }

ngOnInit() {

}

}

Poďme si vyššie uvedený kód postupne rozobrať. Ako prvé sa nachádzajú v dokumente importujú komponenty z jadra Angularu. Ďalej je definovaný komponent samotný pomocou povinných parametrov selector, templateUrl a styleUrls. Parameter selector určuje názov selektora komponentu, ktorý bude použitý v HTML časti iných komponentov. TemplateUrl definuje cestu ku HTML dokumentu pre daný komponent a syleUrls obsahuje pole ciest k dokumentom určených pre štýlovanie. K jednému komponentu je teda možné priradiť viac štýlovacích dokumentov.

Komponent má aj svoj vlastný životný cyklus definovaný metódami:

Tabuľka 2 Životný cyklus komponentu

|  |  |
| --- | --- |
| **Metóda** | **Význam** |
| ngOnChanges() | Je volaná pred ngOnInit() a vždy ak sa zmení niektorý z data-bound prvkov |
| ngOnInit() | Inicializácia komponentu |
| ngDoCheck() | Volaná počas každej kontroly zmien hneď po ngOnChanges() a ngOnInit() |
| gnAfterContentInit() | Volaná raz po prvom ngDoCheck() keď vlastnosti zobrazovaného komponentu sú inicializované |
| ngAfterContentChecked() | Volaná po ngAfterContentInit() a každom nasleduúcom ngDoCheck() |
| ngAfterViewInit() | Volaná raz hneď po ngAfterContentInit() keď vlastnosti child templatu boli inicializované |
| ngAfterViewChecked() | Volaná ak sa zmenia vlastnosti child komponentu |
| ngOnDestroy() | Volaná tesne pred zničením komponentu. Najlepšie miesto pre uvoľnenie premenných alebo udalostí, aby nevznikli úniky pamäte. |

Vzhľadom k faktu, že pracujeme s komponentmi na objektovo orientovanej úrovni, máme k dispozícii aj konštruktor triedy. Avšak podľa odporúčaných postupov by sa konštruktor mal používať hlavne kvôli dependency injection, kedy Angular analyzuje parametre konštruktora a na základe nich vloží do komponentu potrebné inštancie objektov.

Na zobrazovanie obsahu premenných využíva Angular tzv. data-binding. V rámci práce s premennými je možné využiť direktívy ako ngIf – podmienka, ngFor – cyklus, ktoré sú určené pre ľahšiu prácu s premennými. Data-binding môže fungovať jednosmerne, kedy je iba vypísaný obsah premennej v šablóne komponentu alebo obojsmerne, pomocou direktívy ng-model, kedy sa aj v šablóne, napr. pomocou selektora input vo formulári dá meniť hodnota premennej za behu aplikácie. Rovnako je možné pomocou direktív preniesť hodnotu premennej z rodiča na potomka.

Ako ďalší stavebný prvok si opíšeme Service. Komponenty by nemali pristupovať priamo k dátam. Preto je táto úloha prenechaná Servisu. Servis teda zbiera informácie zo servera a predáva ich komponentom pomocou dependency injection. Servis zároveň funguje aj ako entita, ktorá uchováva dáta počas behu aplikácie a môže ich poskytnúť každému komponentu, ktorý si o ne požiada.

Angular ako plnohodnotný framework obsahuje aj Routing modul, slúžiaci na navigáciu medzi jedotlivými komponentmi. Pre lepšiu organizáciu zdrojových kódov sa odporúča, aby bol Routing modul implementovaný ako samostatný top-level modul, ktorý je následne importovaný do hlavného modulu aplikácie. Samotný modul obsahuje definíciu ciest, ktorá sa skladá z povinných parametrov path a component a ďalších nepovinných parametrov, ako napr. children alebo data. V module hlavnej aplikácie je možné routing modul vložiť do šablóny pomocou selektora <routing-outlet>.

## React

React nie je síce framework v pravom slova zmysle, je iba JavaScriptovou knižnicou, avšak má vlastnosti reaktívnych frameworkov, preto je zahrnutý aj v tejto bakalárskej práci.

React je vyvíjaný a udržiavaný hlavne spoločnosťou Facebook, avšak rovnako ako pri Angulari, aj tu majú vývojári možnosť prispieť k obohateniu frameworku.

Od svojho prvého vydania v roku 2013 React prešiel viacerými verziami a zmenami, ktoré neboli veľmi výrazné, ako na príklad v prípade Angularu a v súčasnosti je jeho verziové číslo v16.x.x. Zmeny vo frameworku sa týkali hlavne zrýchľovania renderovania aplikácie a do frameworku bolo pridaných viacero nových elementov (dialog, picture, atď.).

### JSX

Podobne ako Angular, aj React má svoje vlastné rozšírenie JavaScriptu v podobe JSX. Avšak na rozdiel od TypeScriptu JSX funguje skôr ako spojenie JavaScriptu s HTML, teda v jednom dokumente je možné písať aj logiku programu a zároveň aj spôsob, akým sa budú komponenty renderovať na obrazovke. Nie je nutnosťou toto rozšenie používať v aplikáciách, avšak ako uvádzajú samotní autori frameworku React, „Veľa ľudí ho považuje za užitočný pri práci s používateľským rozhraním vo vnútri JavaScript kódu.“.[[2]](#footnote-2) Deklarácia premennej pomocou JSX môže vyzerať nasledovne:

const element = (

<h1>

Hello, {formatName(user)}!

</h1>

);

V prípade, že sa vo výraze bude nachádzať aj kód JavaScriptu (zvyčajne funkcia), je možné ho vložiť medzi množinové zátvorky a tým sa zabezpečí zároveň aj dynamickosť renderovaného komponentu. Ďalšou výhodou využívania JSX je spôsob zabezpečovania vstupov používateľa, aby sa zabránilo injection útokom.

## Vue

## Porovnanie

# [Zber](Pokyny_pre_vypracovanie_ZP.docx#Ciele_práce) požiadaviek

# Analýza požiadaviek

# Implementácia

# [Záver](Pokyny_pre_vypracovanie_ZP.docx#Záver)

# [Zoznam použitej literatúry](Pokyny_pre_vypracovanie_ZP.docx#Zoznam_použitej_literatúry)

<https://www.angularminds.com/blog/article/comparison-difference-between-angular1-vs-angular2-vs-angular4.html>

https://blog.angularjs.org/2017/03/angular-400-now-available.html

<https://blog.angular.io/version-5-0-0-of-angular-now-available-37e414935ced>

https://auth0.com/blog/whats-new-in-angular5/

<https://www.gurustop.net/newsletter/19>

https://www.typescriptlang.org/docs/handbook/classes.html

# [Zoznam príloh](Pokyny_pre_vypracovanie_ZP.docx" \l "Prílohy)

**Príloha A** Názov

**Príloha B** Názov

# 

# Prílohy

## Príloha A: Názov prílohy

Každá ďalšia príloha začína na novej strane.

## Príloha B: Obsah DVD

Priložené DVD obsahuje:

* Práca v elektronickej podobe (formát PDF)

1. https://auth0.com/blog/whats-new-in-angular5/ [↑](#footnote-ref-1)
2. https://reactjs.org/docs/introducing-jsx.html [↑](#footnote-ref-2)