Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Moderní javascriptovací jazyky a frameworky v dynamickém webu

Bakalářská práce

Autor: Eva Pustowková
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Jiří Cabal

Hradec Králové Listopad 2017

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně a s použitím uvedené literatury.

V Hradci Králové dne 14.11.2017 Eva Pustowková

Poděkování:

Ráda bych poděkovala vedoucímu bakalářské práce Ing. Jiřímu Cabalovi za odborné vedení práce, cenné rady a ochotu při zpracování této bakalářské práce.

Anotace

Tématem této bakalářské práce je problematika týkající se webových technologií. Práce se zaměřuje na JavaScript, alternativy k javascriptu a na webové frameworky. Hlavním cílem této práce je objektivní nalezení nejpoužívanějších frameworků. Práce je určena především pro webové programátory, kterým může posloužit jako prvotní a ucelený zdroj informací pro výběr javascriptového frameworku.

Annotation

Title: Modern Script Languages and Frameworks in Dynamic Web

The topic of this bachelor thesis is concerned with web technologies. The work focuses on JavaScript, alternatives to JavaScript and web frameworks. The main aim of this thesis is to objectively identify the most commonly used frameworks. The work is primarily intended for web developers as a primary and comprehensive source of information for the choice of JavaScript framework.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Tématem bakalářské práce je problematika spadající do oblasti webových technologií. Ať si to uvědomujeme či ne, většina z nás každodenně používá webové stránky a aplikace pro různé účely. Za krásným vzhledem těchto stránek a aplikací se skrývá ne jeden programovací jazyk. Jedním z nich, používaný ve většině moderních internetových aplikacích, je JavaScript. JavaScript je skriptovací programovací jazyk, jehož kód je vykonáván webovými prohlížeči na straně klienta. Tento jazyk zajišťuje ověřování formulářů, komunikaci se serverem, animaci interaktivních prvků a mnohé další.

V úvodní kapitole se tato práce zabývá JavaScriptem samotným, který je jedním ze základních prvků moderních webových aplikací. Tato kapitola je zaměřena na historii tohoto jazyka, jeho klíčové vlastnosti a základní syntaxi. Část této kapitoly je věnována technologii AJAX, která využívá JavaScript. Dále jsou specifikovány alternativní jazyky k JavaScriptu. V této částí jsou popsány základní informace o těchto jazycích a porovnání mezi sebou dle subjektivního pohledu uživatelů používajících tyto alternativy.

Další kapitola je věnována javascriptovým frameworkům, hlavnímu tématu této bakalářské práce. Tato kapitola přibližuje význam frameworku jako takového. Jsou zde vysvětleny architektury, na kterých může být framework založen. V této kapitole jsou řešeny také možnosti kombinace jednotlivých javascriptových frameworků a jazyků.

Praktická část je zaměřena na konkrétní příklady nalezených javascriptových frameworků. Na tyto frameworky je pohlíženo z různých úhlů pohledu. Frameworky jsou seřazeny dle jednotlivých kritérií a statistickou metodou jsou konečně seřazeny do žebříčku, ze kterého se dá určit, které z vybraných frameworků jsou v těchto dnech nejoblíbenější, nejvíce používané, či podporované.

Podle sestaveného žebříčku javascriptových frameworků jsou vybrány právě tři frameworky. V závěrečné části bakalářské práce dochází k analýze výkonu vybraných javascriptových frameworků podle veřejně dostupných výkonnostních testů.

2 Cíl práce

Cílem této bakalářské práce je nahlédnutí do světa webových technologií, a to konkrétně na javascriptové jazyky, jejich alternativy a javascriptové frameworky. Všechny uvedené technologie jsou používány při vytváření moderních webových a mobilních aplikací.

Hlavní otázkou této práce je, jaké jsou možnosti při výběru javascriptových frameworků a které z nich jsou nejvíce používány a podporovány, jelikož za kvalitním javascriptovým frameworkem musí vždy stát silná komunita vývojářů a technologických firem, které daný framework podporují, používají a nadále rozvíjí. Odpověď na tuto otázku dá vývojářům přehled o tom, kolik informací a jak silnou podporu mohou očekávat k dané technologii.

Cíle této práce jsou sestavení základního seznamu dnes nejužívanějších frameworků, objektivní určení 10 nejoblíbenějších a nejpoužívanějších frameworků a jejich následné představení čtenáři. K dosažení tohoto cíle budou použity především dostupné internetové zdroje. K určení výsledného žebříčku budou použity také základní statistické metody, které určí na základě vybraných kritérií, které frameworky jsou v dnešní době nejvíce používané, a tedy i nejvíce oblíbené.

3 JavaScript

JavaScript je programovací jazyk s objektově orientovanými schopnostmi. Zapisuje se přímo do HTML kódu. Je to klientský skript. To znamená, že se skript odesílá se stránkou na klienta (do prohlížeče) a teprve tam je vykonáván.

3.1 Úvod do JavaScriptu

Univerzální jádro jazyka bylo vloženo do webových prohlížečů a rozšířeno přidáním objektů reprezentujících okno webového prohlížeče za účelem programování na webu. Tato „klientská" verze JavaScriptu poskytuje možnost vkládat do webových stránek „proveditelný obsah" – což znamená, že stránka nemusí být pouze statický dokument HTML, ale může být dynamická, kdy obsahuje dynamické skripty komunikující s prohlížečem a tímto vytváří dynamický obsah HTML. Jádro jazyka JavaScript připomíná Javu, C a C++, a to svými programovými konstrukcemi, jako jsou příkazy if, while a operátor &&. Tímto však podobnost končí. JavaScript je jazyk bez typové kontroly. To znamená, že proměnné nemusí mít striktně specifikovaný tys. Také je tento jazyk čistě interpretovaný jazyk v porovnání s jazyky C a C++, které jsou kompilovány, a jazykem Java, který je kompilován do bajtového kódu ještě, než je interpretován. [1]

3.2 Historie JavaScriptu

JavaScript od svého úplného vzniku prošel mnohými změnami. Na začátku byl odsuzován za mnoho nedostatků. Podpora prohlížečů nebyla dostatečná. V prvních verzích nenabízel plnou kompatibilitu s tímto jazykem, a to také vedlo k tomu, že se mnoho vývojářů snažilo minimalizovat používání tohoto jazyka. Někteří ho dokonce považovali za „hračičkovský" jazyk pro grafické designéry. Nicméně postupem času se JavaScript stal fenoménem, který posunul web o pořádný kus vpřed tím, že dal do rukou vývojářů silný nástroj pro realizaci jejich představ. Podkapitola historie JavaScriptu čerpá z [2].

3.2.1 Původ JavaScriptu

Javascript je marketingový název pro specifikaci formálně nazývanou ECMAScript. Odkud vlastně Javascript vzešel?

JavaScript byl původně vyvíjen Brendanem Eichem ve společnosti NetScape v rozmezí let 1995 až 1996. První prototyp se nazýval Mocha, později LiveWire a LiveScript. Ty se ale neujaly jako vhodné názvy nového jazyka, a tak si lidé z marketingu „nešťastným" rozhodnutím postavili hlavu a jazyk pojmenovali JavaScript. Zmatení přišlo záhy a to proto, že jazyk Java byl v té době velkou novinkou a byl velmi populární. JavaScript byl neúprosně spojován s Javou. Avšak později toto propojení obou jazyků JavaScript setřásl.

JavaScript není kompilovaným (překládaným) jazykem, což může vytvářet pocit, že není dostatečně výkonný. Na konci devadesátých let minulého století byl tento jazyk zatracován mnoha webovými vývojáři, ale záhy objevili jeho užitečnost a sílu. Díky ní lze vytvářet interaktivní World Wide Web. Do té doby se webové stránky vytvářely pomocí hypertextového značkovacího jazyka s grafikou, která postrádala vizuální dojem a schopnost ovlivňovat obsah stránek.

Cílem JavaScriptu bylo zaměřit se na ověřování obsahu formulářů na straně klienta a na práci s obrázky. Tímto JavaScript poskytl užitečnou interaktivitu a také zpětnou vazbu návštěvníkům webových stránek. V případě, že uživatel vyplnil formulář, JavaScript se mohl použít ke kontrole jeho obsahu, aniž by se musela data odesílat webovému serveru. Toto vedlo k pozitivnímu urychlení aplikací, a tak k ušetření času, který byl nutný ke komunikaci se serverem, a to obzvláště v době, kdy ještě nebyl rozšířen širokopásmový internet neboli Broadband. Tento jazyk je tedy velmi užitečným nástrojem pro vytváření dynamických webových aplikací.

3.2.2 JScript

Kolem roku 1996 na hrací plochu vstupuje velký hráč Microsoft, který v témže roce zahrnul do své třetí vydané verze Microsoft Internet Exploreru podporu jádra JavaScriptu, známou jako JScript, s podporou svého skriptovacího jazyka VBScriptu. Tyto jazyky si jsou velmi podobné, avšak liší se ve své implementaci. Také proto byly vyvinuty metody, které detekovaly, jaký webový prohlížeč používá návštěvník webových stránek a podle výsledku mu byl poskytnut takový skript, který v daném prohlížeči fungoval. Tomuto procesu se říká detekce prohlížeče, který se ještě stále používá, i když v mnohem menší míře.

3.2.3 ECMAScript

Obě verze JavaScript od společnosti NetScape i JScript od společnosti Microsoft procházely vývojem. Pomalu začal vznikat stav narůstající nekompatibility, a proto zasáhla mezinárodní standardizační průmyslová asociace se zaměřením na informační a komunikační systémy ECMA a v roce 1997 vznikala první verze ECMAScriptu, na které spolupracovaly společnosti Microsoft a NetScape. Od této chvíle všechny prohlížeče od firem Microsoft a NetScape implementují jednotlivé verze ECMAScriptu. Některé z populárních prohlížečů, jako jsou například Firefox a Opera, také implementují standart ECMAScriptu. Poté přišla poslední verze známá jako standart ECMA-262, která byla vydána v roce 1999. Tato novinka přinesla jednu dobrou a jednu špatnou zprávu. Tou dobrou zprávou je, že prohlížeče jako je Internet Explorer 4 a NetScape 4.5 tento standart podporovaly. Od té doby každý důležitý prohlížeč podporuje verzi JavaScriptu přijatou ve standardech ECMA-262. Ta špatná zpráva je, že každý prohlížeč si tento standart vykládá trochu jinak, a to vyvolává situaci, že i dnes se mnoho programátorů v JavaScriptu setkává s nekompatibilitou.

3.2.4 DOM

Dalším standardem, kterého využívají programátoři v JavaScriptu, je objektový model dokumentu neboli DOM (Document Object Model). Ten byl vyvinut a standardizován konsorciem World Wide Web Consortium (W3C). Suering ve své knize [2] zmiňuje: „*W3C popisuje DOM jako na platformě a jazyku nezávislé rozhraní umožňující programům a skriptům dynamicky přistupovat a aktualizovat obsah, strukturu a styly dokumentů* “.

DOM je možno chápat jako prohlížeči používanou specifikaci, která pomáhá pracovat s webovou stránkou v dynamickém slova smyslu. Objektový model reprezentuje dokumenty HTML a XML ve stromové struktuře a umožňuje dynamický přístup k objektům této struktury. Javascript a DOM byly úzce propojeny, postupně se ale vyvinuly v oddělené koncepty. DOM představuje plně objektově orientovanou reprezentaci stránky, která může být modifikována skriptovacím jazykem, jako je například JavaScript nebo VBScript. Postupem času vznikaly jednotlivé úrovně objektového modelu dokumentu. [2] [3]

3.3 Základy JavaScriptu

V této podkapitole je popsána základní struktura JavaScriptu, jeho funkce, ale také nedostatky, základ syntaxe.

3.3.1 Obsah programu v JavaScriptu

Základní kameny programu v JavaScriptu jsou, ostatně jako ve všech programovacích jazycích, příkazy. Tyto příkazy jsou tvořeny znaky, operátory a identifikátory umístěné za sebou v řadě. Pro představu jednoduchý příkaz muže vypadat takto:

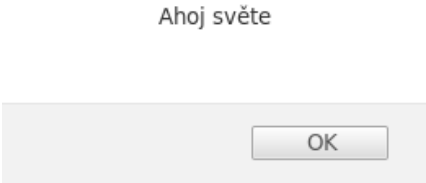
```
var myNumber = 1;
```

Tento příkaz se skládá ze znaků, a to rezervované slovo var, identifikátor (myNumber), operátor (=) a číslo (1). Cílem tohoto příkazu je nastavit proměnnou myNumber na hodnotu 1. Příkazy se uspořádávají za sebe. Program vykoná jednu nebo více funkcí. JavaScript má vlastní způsob, jímž funkce definuje a umožňuje použití řadu vestavěných funkcí.

Dalším příkladem může být použití pseudoprotokolu a funkce javascript. Po otevření jakéhokoliv prohlížeče uživatel zadá následující kód do adresního řádku a zmáčkne klávesu Enter.

```
javascript:alert("Ahoj světe");
```

Po stisku klávesy uživatel uvidí podobný dialog:



Obrázek 1: Javascriptový dialog (zdroj: vlastní zpracování)

Tento dialog je kousek programového kódu v JavaScriptu, i když nepříliš užitečný. Skrývá v sobě ale dvě důležité věci, a to identifikátor pseudoprotokolu JavaScript a funkci alert, která zajišťuje zobrazení vyskakovací okna. [2]

3.3.2 Jak vložit JavaScript do webové stránky

V HTML se vkládají do webové stránky elementy pomocí párových značek neboli tagů, uzavřených do ostrých závorek, kdy uzavírací začíná s lomítkem (/). Jednotlivé elementy se dají vnořovat do sebe. Kód v JavaScriptu se vkládá do těla tagu *<script>*, jež se vkládá dovnitř tagu *<head>*, a (ta) zase do *<body>*. Příklad takového kódu:

```
<html>
        <head>
                <title> Titulek webové stránky</title>
                <script type="text/javascript">
                        //zde patří libovolný programový kód v JavaScriptu
                </ script>
        </head>
        <body>
                <script type="text/javascript>
                        //zde patří libovolný programový kód v JavaScriptu
                </script>
        </body>
</html>
```

Kód JavaScriptu vložený do tagu *<body>* se spouští při čtení dokumentu, jakmile na něj dojde řada. V následující ukázce se využívá případ, kdy je potřeba psát do dokumentu pomocí některé z funkcí JavaScriptu. [2]

```
<html>
        <head>
                <title> Titulek webové stránky</title>
                <script type="text/javascript">
                        //zde patří libovolný programový kód v JavaScriptu
                </ script>
        </head>
        <body>
                <script type="text/javascript>
                document.write("Nějaký text");
                </script>
</body>
</html>
```

3.3.3 Co umí Javascript

JavaScript umí různými způsoby přidat na stránky dynamiku. Tento jazyk patří mezi univerzální a to znamená, že je možné v něm psát programy pro provádění různých výpočtu, jak už psaní jednoduchých počítadel, tak nesčetné množství funkcí ve webových aplikacích. Síla tohoto jazyka spočívá v prohlížeči a v dokumentově založených objektech, s kterými jazyk úzce souvisí a podporuje je.

Kompletní průvodce JavaScriptem [1] uvádí, že tento jazyk v sobě skrývá velkou řadu schopností a některé z důležitých schopností jsou popsány níže.

- **Vzhled a obsah řídícího dokumentu**

Javascriptový „object document“ pomocí metody *write()*, umožňuje zapisovat libovolný text HTML do dokumentu v době, kdy dokument zpracovává prohlížeč. To například umožní zahrnovat do dokumentu aktuální datum, který, má na správu objekt **Date** nebo zobrazovat různý text na různých platformách. „Object document“ je možné využít ke generování dokumentů od samotného začátku. Jeho vlastnosti dokonce dovolují specifikovat například barvu pozadí dokumentu, textu a jiné.

- **Řízení prohlížeče**

Existuje několik objektů JavaScriptu, které umožňují řízení chování prohlížeče. Objekt **Window** spolupracuje s metodami dialogů. Tyto metody umožní zobrazit jednoduché zprávy pro uživatele a jednoduchý uživatelský vstus. Také tento objekt definuje metodu pro vytváření nových oken prohlížeče, kterým lze nastavit jakoukoliv danou velikost a jakoukoli kombinaci uživatelského ovládání. To vede k tomu, že je možné otevřít několik oken, a umožnit uživateli několik pohledů na webový prostor. Jedna z dalších metod tohoto objektu dovoluje JavaScriptu zobrazovat zprávy uživateli na stavovém řádku okna prohlížeče.

Javascript poskytuje možnost kontroly nad tím, které webové stránky jsou zobrazeny v prohlížeči. Object *Location* zajišťuje načtení a zobrazení obsahu URL adresy v jakémkoliv okně nebo rámečku prohlížeče. Další objekt *History* umožňuje pohyb vpřed a zpět v uživatelské historii.

- **Interakce s obsahem dokumentu**

„Object document“ JavaScriptu a jeho objekty poskytují programům možnost číst a ovlivňovat části dokumentu. Jednu z nejdůležitějších schopností, schopnost ovlivňovat obsah dokumentů, poskytuje objekt *Form* a jeho prvky objektů, obsahující: *Button, Checkbox, Hidden, Password, Radio, Reset, Select, Submit, Text a Textarea*. Pomocí těchto prvkových objektů je možné číst a zapisovat hodnoty

vstupních prvků do dokumentu. Obvyklým využitím schopnosti číst uživatelský vstup z prvků formuláře se nachází v kontrole formuláře před jeho odesláním. V případě schopnosti JavaScriptu provést veškerou potřebnou kontrolu chyb uživatelského vstupu na straně klienta, detekci a informování uživatele o chybách, není zapotřebí běžná okružní cesta k serveru. Toto vede k redukci dat, které jsou odesílané na server.

- **Interakce s uživatelem**

Další ze schopností, co ovládá JavaScript, je schopnost nastavovat „ovladače událostí". Jsou to libovolné části kódu, které se vykonají ve chvíli, kdy nastane určitá událost. Takovou událostí může být například zápis hodnot uživatelem do formuláře nebo uživatelem zmáčknuté tlačítko *Submit* pro odeslání formuláře. Javascript umí spustit jakoukoliv akci jako odezvu na nějakou událost. Příkladem je zobrazení potvrzovacího dialogového okna, když uživatel odesílá důležitý formulář.

- **Čtení a zápis klientského stavu pomocí Cookies**

Termín „Cookies" používá firma NetScape pro malé množství stavových dat. Jsou uložena dočasně nebo trvale uživatelem. Zasílají se zpět a vpřed na a ze serveru a umožňují webové stránce či lokalitě zapamatovat si údaje o klientovi. Takový údaj může být registrace daného klienta a následné přidělení hesla nebo jiné nastavení. Cookies slouží k obstarání stavové informace, která chybí v nestavovém webovém protokolu HTTS. [1]

3.3.4 Co neumí JavaScript

JavaScript se může chlubit seznamem mnoha schopností, avšak je používán v omezeném kontextu. Následně jsou popsány úkoly, které JavaScript neumí splnit a také některé, které by ani plnit neměl.

- **JavaScript nemůžete klientovi vnutit**

Tento jazyk svou funkcionalitu opírá o jiné rozhraní nebo hostitelský program. Většinou takovýmto hostitelem je webový prohlížeč na straně klienta,

známý jako uživatelský agent (user agent). Stále ještě existují starší verze prohlížečů, které jej vůbec nepodporují. Někteří uživatelé dokonce použití JavaScriptu v prohlížečích úplně zakážou, kvůli problémům se zabezpečením, či kvůli slabé reputaci JavaScriptu, vzniklé například obtěžováním vyskakovacích oken s reklamou.

- **Není garantováno, že JavaScript proběhne**

Program napsaný v JavaScriptu běží celý na straně klienta, a proto je nezbytné, aby se programátor naučil jej tak používat. Provádění kódu na straně klienta může vést k vážným bezpečnostním důsledkům. Program napsaný JavaScriptem nemusí vůbec proběhnout. Lze říci, že klíčová funkcionalita webové aplikace nemůže spoléhat na JavaScript. Jedním ze základních pravidel je nevěřit jakýmkoli datům přicházejícím od klienta, i když by byla použita funkce pro kontrolu formuláře. Je nutné provést kontrolu těchto vstupů znovu ve chvíli, kdy se data dostanou na server.

- **JavaScript nemůže překročit hranice domény**

Každý programátor v JavaScriptu si musí být vědom tzv. politiky stejného původu. Podle této politiky je nutností, aby skripty v jedné doméně nesměly mít přístup k prostředkům jiné domény, a také nesmí ovlivňovat skripty a data jiné domény. Tato politika je často problematická při použití rámců nebo objektu XMLHttpRequest používaného v AJAXu, kdy je nezbytné pomocí JavaScriptu poslat více požadavků několika různým serverům. Technologie AJAX je popsaná v kapitole č. 4.6.

- **Nepracuje na straně serveru**

Při vývoji kódu na straně serveru, jako v případě jazyků Visual Basic.ne3t nebo PHP, server implementuje celou řadu funkcí, jako je komunikace s databází nebo přístup k modulům důležitým pro chod webové aplikace. JavaScript například nemůže přímo přistupovat k databází umístěné na serveru. Javascriptový kód

je omezen tím, co se může udělat v rámci platformy, na které skript běží, což je obvykle prohlížeč. [2]

3.4 Syntaxe a příkazy

Základní strukturou JavaScriptu je jeho syntaxe, plná termínů jako: proměnné, funkce, podmínky, operátory a další. Pomocí těchto prvků je možné sestavovat různé příkazy, které jsou schopny, po spuštění provádět různé operace. Vysvětlením všech pojmů s kódovými ukázkami se věnuje Suehring [2, kas. 3].

3.4.1 Zdrojový kód

Následující pojmy vytváří strukturu zdrojového kódu. Vývojář se při vývoji řídí pravidly jazyka, jimž se zpravidla říká syntaxe.

- **Citlivost na velikost písma**
- **Prázdná mezera**

JavaScript v mnoha případech prázdnou mezeru mezi příkazy ignoruje. Lze ji tedy použít pro docílení lepší čitelnosti kódu. Ale existují i některé výjimky, jako třeba klíčová slova. Takovým příkladem může být return. V případě, že je na řádku samotný, interpreter pro JavaScript může příkaz špatně vyhodnotit.

- **Komentáře**

Komentáře slouží především jako pomůcka pro vývojáře. Cílem je vytvářet čitelný programový kód a udržovat jej tak po delší dobu. Hlavním důvodem je srozumitelnost kódu, určená pro každého, kdo se k němu může dostat. Lze je zapisovat dvěma způsoby: na jeden řádek a více řádků. Jednořádkový komentář začíná lomítky (//), jako například:

```
// Toto je jednořádkový komentář
```

Komentář na více řádku začíná a končí znaky /* a */. Tento případ ukazuje následující kód:

```
/* Toto je víceřádkový komentář
Napsaný v JavaScriptu. */
```

Další možností je používat jednořádkové komentáře na více řádcích pod sebou.

- **Středníky**

Středníky se využívají v JavaScriptu pro stanovení rozsahu příkazů. Je vhodnější je používat, i když ve většině případů nejsou třeba. V případě nepoužití středníků mohou v určitých případech vznikat zbytečné chyby a následné nutné ladění problému.

- **Zalomení řádku**

Zalomení řádku se podobá mezeře a středníku. Jsou to znaky separující jeden řádek programového kódu od druhého za účelem zlepšit čitelnost kódu.

3.4.2 Příkazy

Program napsaný v JavaScriptu, podobně jako v jiných jazycích, se skládá z příkazů, které se zapisují za sebe, načež interpreter JavaScriptu vykoná jednu či více akcí.

- **Co obsahuje příkaz**

Příkaz se skládá ze znaků, operátorů a identifikátorů. Zpravidla příkaz končí středníkem, výjimkou jsou podmíněné bloky a cykly (*if*, respektive *while* a *for*). Příklad příkazu je ukázán v podkapitole č. 4.3.1.

- **Dva typy příkazů**

Příkazy JavaScriptu mohou mít dvě formy, jednoduchou a složenou. Jednoduchý příkaz vypadá například takto:

```
x = 4;
```

Ten složený je většinou kombinací více úrovní logiky. Podmíněný blok if/ else je vhodným příkladem pro ukázku složeného příkazu:

```
if(variable = 1) {
  // zde vložíte programový kód
}else{
```

```
// zde vložíte další programový kód
}
```

3.4.3 Vyhrazená slova

V JavaScriptu existují vyhrazená slova a nemohou se používat pro názvy proměnných, identifikátorů ani konstant v programech, neboť by programový kód při spuštění zahlásil chybu, jako například takovýto příkaz:

```
var var = 4;
```

Existuje seznam slov, která jsou vyhrazená specifikací ECMA 262. Jsou to například slova jako *break*, *case*, *catch*, *continue* a další. [2]

3.5 Proměnné a data

Kód napsaný v JavaScriptu by se neobešel bez datových typů a proměnných. Zdroj informací, s jakými proměnnými a daty se můžeme v JavaScriptu setkat je uveden v [2].

3.5.1 Datové typy

Datové typy mají za úkol popsat základní elementy tohoto jazyka. JavaScript pracuje se třemi až šesti datovými typy a těmi jsou:

- **Čísla**

V JavaScriptu neexistuje žádný speciální typ pro celá čísla a pro čísla s pohyblivou desetinnou čárkou. Například čísla 4, 51.50, -14, 0xd připouští jako správná. JavaScript obsahuje vestavěné číselné funkce, které pracují s číselnými hodnotami a číselné konstanty.

- **Textové řetězce**

Tento datový typ se skládá z jednoho či více znaků uzavřených do uvozovek. Například takovýmto řetězcem je: "Ahoj Světe!"

- **Pravdivostní hodnoty (Boolean)**

Pravdivostní hodnota je výsledkem vyhodnocení nějakého výrazu a může nabývat dvou hodnot a to pravda (true) a nepravda (false). V praxi se většinou nedefinuje proměnná tohoto typu, ale používají se výrazy jako podmínky bloků *if/else*, jejichž výsledkem je pravdivostní hodnota.

- **Array**

Datový typ Array neboli pole je proměnná, která umožňuje uchovávat více než jednu hodnotu. Hodnoty jsou v poli indexovány od nuly.

- **Null**

Dalším speciálním datovým typem JavaScriptu je *null* a znamená nic. Logicky odpovídá nepravdě (*false*). Pokud se pracuje s typem *null*, pracuje se s ničím. Nesmí se tato hodnota zaměňovat s prázdnou hodnotou nějaké proměnné, neboť prázdná hodnota je stále existující hodnota, ale *null* je prostě nic.

- **Undefined**

Undefined (v překladu nedefinováno) je stav hodnoty proměnné, které žádná hodnota nebyla přidělena. Jedná se o něco jiného než v případě *null*, i když logicky se vyhodnocují stejně. Rozdíl mezi *undefined* a *null* je v tom, že *null* naznačuje, že příslušná proměnná je z nějakého důvodu sice deklarovaná, ale zatím nemá přiřazenou žádnou hodnotu, zatímco *undefined* znamená, že se s touto proměnnou dosud nepracovalo a datový typ *undefined* nelze do proměnné přiřadit.

- **Objekty**

JavaScript je objektově orientovaný skriptovací jazyk, který implementuje specifickou funkcionalitu založenou na objektech. Objekty v JavaScriptu se dělí do čtyř skupin a to:

  - Uživatelsky definované objekty (*user defined objects*) vytvořené programátorem

- Vestavěné objekty (*built-in objects)* samotným JavaScriptem
- Objekty prohlížeče (*browser object)*
- DOM objekty (*document object*)

V JavaScriptu všeobecně platí, že všechno, co není primitivní datový typ, jimiž jsou číslo, pravdivostní hodnota a řetězec, je objekt. Objekty JavaScriptu jsou kolekcí atributů a metod. Přes názvy atributu můžeme přistupovat k jejich hodnotám a každá hodnota atributu může být jak běžnou hodnotou, nebo dalším objektem nebo také dokonce funkcí.

Objektově orientované jazyky, například Java a C++, jsou *class-based* jazyky, tedy jazyky založené na objektech a jejich třídách. V těchto jazycích jsou vlastnosti objektů definovány jako třídní popis. Na rozdíl od těchto jazyků je JavaScript *prototype-based* a to znamená, že každý objekt v JavaScriptu má prototyp, ze kterého může dědit metody a vlastnosti. Této schopnosti dědit se někdy také říká prototypová dědičnost. [6]

Objekty se tvoří několika způsoby. Jedním z nich je použití *new Object()*, který vytvoří prázdný objekt. Proměnné a metody do něj můžeme přidávat následujícím způsobem:

```
var article= new Object();
article.caption = "Objekty v JS";
article.text = "Objekty v JavaScriptu";
article.published = false;
publish = function (){
this.published = true;
}
```

Tento kus kódu vytváří objekt článek, která má titulek Objekty v JavaScriptu a také obsahuje metodu, která publikuje daný článek.

Další způsob, jak vytvářet objekty je přímo definovat atributy a metody do složených závorek. Zápis vytvoření objektu může vypadat takto:

```
var article= {
caption: "Objekty v JS",
text : "Objekty v JavaScriptu",
published: false,
publish: function(){
this.published= true;
}
```

```
};
```

Klíčové slovo *this* v metodě publikovat zastupuje objekt, ve kterém se nachází daná funkce. To se nazývá kontext funkce. [2]

3.5.2 Proměnné

Proměnné v sobě ukládají data, která se mohou měnit v průběhu a díky nim je možné vnést na webovou stránku dynamiku.

- **Deklarace proměnné**

Proměnné se v JavaScriptu deklarují prostřednictvím klíčového slova *var*. Takové příklady proměnných jsou již uvedeny v podkapitole 5.3.1 a v předchozí podkapitole. Název proměnné nemůže začínat číslicí a nesmí obsahovat mezery, či jiné speciální znaky. Je možné deklarovat více proměnných v jednom řádku a zároveň i přiřadit hodnoty.

- **Datový typ proměnné**

U proměnných JavaScriptu není nutné dopředu deklarovat, zda proměnné bude přiřazena celočíselná hodnota, desetinná nebo řetězec, protože je možné změnit datový typ jednoduše novým přiřazením hodnoty. Následující příklad tomu odpovídá a to tak, že nejdříve je hodnota proměnné celé číslo a po druhém příkazu je jeho hodnota již řetězec. [2]

```
var x = 5;
var x = " A teď je to řetězec.";
```

3.6 AJAX

AJAX (Asynchronous JavaScript and XML) je nástroj pro vytváření interaktivních webových aplikací. Tato podkapitola o AJAXu čerpá z více zdrojů. [2,3,4,5]

3.6.1 Obecně o AJAXu

Technologie AJAX umožňuje posílat a přijímat data asynchronně bez nutnosti znovunačtení celé stránky. Umožňuje odeslat a přijmou data ze serveru bez vyrušení

uživatele v prohlížení stránky. Je tak možné na stránce provádět více úkonů najednou, aniž by bylo nutné čekat na to, až bude předchozí požadavek vyřešen.

V aplikaci, která AJAX využívá, se odešle požadavek na server, kde se provede, aniž by se musela celá stránka opětovně načítat, dojde k načtení pouze potřebných dat, které byly vyžádány AJAX požadavkem. Stránka se celá kompletně nepřekresluje, vždy se překreslí jenom konkrétní prvek.

Primární komponentou AJAXu je JavaScript. V názvu se vyskytuje zkratka XML, protože je to jeden z formátů dat, ve kterém umí AJAX obdržet informaci ze serveru. Nejčastěji používané jsou JSON, XML, text a binární data, přičemž každý z nich se používá na něco jiného a trochu jinak. Velkým pozitivem je, že AJAX nevyžaduje žádný další software nebo hardware a jeho použití je ve většině dnešních moderních prohlížecích.

3.6.2 Objekt XMLHttpRequest

Některé aplikace využívající AJAX mohou vypadat velmi složitě, ale samotný proces odeslání požadavku a zpracovaní odpovědi je jednoduchý a funguje za pomoci objektu *XMLHttpRequst*, jenž je stavebním kamenem pro vytváření aplikací v AJAXu. Tento objekt nikdy neprošel žádnou standardizací, ale přesto s příchodem Internet Exploreru 7 jeho použití je ve všech používaných prohlížečích stejné.

- Vytvoření objektu *XMLHttpRequest* se může lišit podle daného prohlížeče, ale v novějších verzích NetScape Navigator, Apple Safari, Mozilla a Firefox je umožněno vytvoření objektu následujícím způsobem:

```
var xmlHttp = new XMLHttpRequest();
```

3.6.3 Odeslání a zpracování požadavku

Dle [2] je odeslání požadavku umožněno kombinací volání metod *open* a *send* objektu *XMLHttpRequest*. Způsoby, jak odesílat požadavky na server pomocí AJAXu, jsou dvojí, a to synchronně a asynchronně. Požadavek poslaný synchronně je součástí blokující operace, která zabraňuje provádění dalšího programového kódu v JavaScriptu, dokud nedorazí odpověď od serveru. Tento proces má

nedostatky v rychlosti nebo dokonce se může požadavek či odpověď po cestě ztratit, proto se ve většině případů doporučuje ho nepoužívat.

Prvním krokem je inicializace požadavku, která připraví požadavek a nakonfiguruje ho pro použití se serverem pomocí metody open, jejíž syntaxe je následující (položka v hranaté závorce je nepovinná):

```
open("method", "URL" [, signOfAsynchronousCommunication]);
```

Význam jednotlivých parametrů je následující:

- method: Metoda komunikace protokolem HTTP, jako například GET, POST, PUT, HEAD nebo PROFIND.
- URL: Požadované URL
- signOfAsynchronousCommunication: Booleanovský výraz indikující, zda se jedná o asynchronní volání. Výchozí hodnota je *true*.

Dalším důležitým krokem je odeslání požadavku pomocí metody *send*. Následující ukázka kódu shrnuje tyto vysvětlené metody a pro otestování funkčnosti kódu je nutné jej zadat do konsole v browseru:

```
var xmlHttp = new XMLHttpRequest();
xmlHtts.open("GET","https://jsonplaceholder.typicode.com/posts/1",true);
xmlHtts.onreadystatechange = function() { console.log(this.responseText)};
xmlHtts.send();
```

Atribut *onreadystatechange* dává možnost specifikovat funkci, která je zavolána po změně stavu požadavku. Doba změny stavu požadavku je dána tím, jak dlouho trvá zpracování požadavku na serveru. Mezi tím je možné zpracovávat další požadavky, čímž je zajištěna asynchronnost zpracování. Změna stavu požadavku je dána hodnotou *readyState*, která může nabývat pěti hodnot (neinicializováno, inicializováno, přičemž požadavek nebyl odeslán, požadavek odeslán, probíhá přijímání odpovědi, odpověď přijata). Proměnná *responseText* obsahuje odpověď ze serveru v textové podobě. Další ukázka kódu je zpracování odpovědi prohlížeče v konzoli:

```
XHR finished loading: GET "https://jsonplaceholder.typicode.com/posts/1"
        {
                "userId": 1,
                "id": 1,
                "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
```

```
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
}
```

3.6.4 Metody GET a POST v AJAXu

Použití AJAXu vyžaduje odesílání kontextových dat serveru. Běžné metody požadavků jsou metody GET a POST.

Metoda GET předává páry argumentů název/hodnota jako součást URL. Konec vlastního URL se skládá z otazníku a po něm páry název/hodnota. Krátký příklad:

http://localhost/myLibrary?authorName=David&authorSurname=Flanagan

Metoda POST pro odesílání argumentů serveru je velmi identická metodě GET. Rozdíl spočívá v tom, že metoda POST odesílá řetězec s argumenty (řetězec s dotazem) v těle požadavku.

Použití metody GET by mělo být převážně pro získávání dat. Metoda POST je doporučená pro operace, které mění stav datového modelu, a to ukládáním nebo aktualizací dat. Obě tyto metody mají své výhody a nevýhody. Argumenty GET požadavku jsou zakódovány v URL požadavku, a to není příliš užitečné v kontextu asynchronního požadavku. Metoda POST je flexibilnější a lze odeslat jakékoli množství dat.

3.6.5 Komunikace se serverem pomocí XML

Pro odesílání většího počtu změn modelu na server je nedostačující jednoduchý řetězec s dotazem tvořený páry název/hodnota. Mnohem lepší řešení nabízí odeslání změn modelu serveru ve formě XML. Jazyk XML je jazyk tvořený značkami definovanými uživatelem. Kód XML lze odesílat serveru jako součást těla požadavku. Poté server může z těla požadavku dokument XML načíst a dále s ním pracovat. Ukázka přenosu dat pomocí XML:

```
<book>
<title>Book</title>
<publisher>Publisher</publisher>
```

```
<author>Author</author>
</book>
```

3.6.6 Komunikace se serverem pomocí JSON

JSON (JavaScript Object Notation) je způsob, jak předávat objekty JavaScriptu jako data bez nutnosti kódovat data do dokumentů XML. Vstupními hodnotami může být libovolná datová struktura, výstupem je vždy řetězec. Efektivnost tohoto způsobu předání dat spočívá v usnadnění komunikace server-klient, protože není nutné zpracovávat odpovědi pomocí DOM a data jsou dostupné k použití ihned, bez nutnosti explicitní konverze na objekty JavaScriptu. Přístup k danému elementu je mnohem jednodušší v případě formátu JSON než XML. Menší velikost objektu v JSON může být velmi užitečná, například při odesílání většího množství dat po síti. Ukázka přenosu dat pomocí JSONu:

```
{"book":
 {
 "title":"Book1",
 "publisher":"Publisher1",
 "author":"Author1",
 }
}
```

3.6.7 Použití AJAXu

Jedním z prvních osvojitelů této technologie je Google. Příklady využití v aplikacích jsou Google Maps, Google Suggest, Gmail a další. AJAX je také zabudován do samotného vyhledavače Google (např. našeptávač, zobrazení výsledků před stisknutím enteru). Použití AJAXu není omezeno pouze na známé „.com'' stránky, ale stále více členů vývojářské společnosti používá AJAX pro řešení problematických validací nebo pro získávání dat za běhu. Rozsáhlé použití AJAXu přispělo k úspěchu mnoha stránek. Například ovládací prvky založené na AJAXu využil Amazon, který má díky této technice skvělý vyhledávací nástroj. Dalším příkladem je Facebook, ve kterém AJAX umožňuje vyjádřit se k danému tématu, aktualizují se jednotlivá oznámení a udržují se okamžité zprávy chatu na jedné stránce a další. Existuje mnoho dalších aplikací postavených na této technice, které pomáhají vytvářet všestranné interaktivní webové aplikace. [2][3][4][5]

3.7 Alternativní jazyky k JavaScriptu

JavaScript je velmi používaný klientský jazyk určený pro tvorbu moderních webových aplikací. Přesto na základě některých nedostatků začaly postupem času vznikat různé alternativy k tomuto jazyku. Tyto jazyky jsou v určité míře odlišné od JavaScriptu, a proto se jim také někdy říká „compile-to-JS“ jazyky. V této podkapitole jsou popsány alternativní programovací jazyky k jazyku JavaScript a důvody, proč alternativní jazyky vznikají.

3.7.1 Proč vznikají alternativní jazyky k jazyku JavaScript

Jádro jazyka JavaScript není samozřejmě dokonalé a přichází s různými výzvami pro programátory, kteří se ho rozhodnou použít. Jedna z největších výzev je otázka udržitelnosti kódu samotného. Jádro jazyka JavaScript nedává mnoho prostoru psát udržovatelný a škálovatelný kód. Z tohoto důvodu mimo další nedostatky začaly vznikat různé alternativní jazyky k JavaScriptu.

3.7.2 Kompilace do JavaScriptu

Webové prohlížeče (Chrome, IE, Firefox, Safari, Opera) umí v současné době pracovat pouze s JavaScriptem, a tak se vlastně ve většině případů takzvaných „alternativních“ jazyků o alternativě nedá ani mluvit, protože výsledné kódy jsou kompilovány zpět do jazyka JavaScript tak, aby byly spustitelné na výše zmíněných webových prohlížečích.
V této době je nemyslitelné vyjmout jádro JavaScriptu ze všech prohlížečů a nahradit ho jádrem pro jiný standard. Z tohoto důvodu je nutné brát níže zmíněné alternativy k jazyku JavaScript spíše jako pokusy o jiný přístup k JavaScriptu samotnému. Každá tato „alternativa“ se snaží přinést své výhody k jazyku JavaScript, jako jsou například:

- snadnější údržba kódu
- využití jiných syntaxí založených na různých jazycích Ruby, C, Java a dalších
- optimalizace javascriptového kódu použitím vlastních source-to-source kompilátorů

Pro kompilaci těchto jazyků do JavaScriptu se používají transpilery. Transpiler je označení pro source-to-source kompilátor, který čte zdrojový kód v jednom programovacím jazyce a přetváří ho do zdrojového kódu v jiném jazyce. Takovýchto transpilerů do JavaScriptu existuje velké množství, neboť i nových alternativních jazyků k JavaScriptu vzniklo a stále vzniká velký počet. Vybrané alternativní jazyky jsou blíže popsány v následujících podkapitolách. [7]

3.7.3 Výpis alternativ k JavaScriptu

V tabulce níže je vypsáno několik alternativních jazyků, které jsou seřazeny sestupně na základě počtu tagů na webové stránce stackoverflow.com (výsledky byly získány dne 19. 11. 2016). Tato webová stránka se stala největším online komunitním webem pro programátory. Je to především Q&A software (question and answer), sloužící jako alternativa ke klasickým diskuzním fórům. Dle počtu tagů pro jednotlivé jazyky se dá usuzovat o jejich oblibě mezi programátory. [7,8]

Tabulka je navíc doplněna o statistiku komitů a počtu přispěvatelů na webu github.com (výsledky byly získány dne 19. 11. 2016). GitHub je webová služba, která podporuje vývoj softwaru pomocí verzovacího systému Git. GitHub nabízí svým uživatelům bezplatný webhosting pro open source projekty.

Alternativa k JavaScriptu	**Počet tagů na stackoverflow.com**	**Počet komitů na github.com**	**Počet přispěvatelů na github.com**
TypeScript	18687	15295	168
CoffeeScript	9076	4148	193
Dart	6190	Není hostován na github.com	Není hostován na github.com
Haxe	1104	14164	116
ELM	669	Není hostován na github.com	Není hostován na github.com
JSX	584	3863	18
Babel	576	7403	265
EMScripten	369	15907	220
PureScript	143	3335	100
Traceur	95	1730	57
LiveScript	62	4194	57

Tabulka 1: Seznam javascriptových alternativ (zdroj: vlastní zpracování)

Z této tabulky vyplývá, které jazyky jsou více populární a které méně. Dále jsou popsány ty alternativy, které se dle počtu tagů na stackoverflow.com umístily na prvních třech místech.

3.7.4 TypeScript

TypeScript vytvořil Anders Hejlsberk ze společnosti Microsoft, a tak není divu, že TypeScript přišel ze světa jazyka C#. Syntaxe jazyka C# jsou zřetelné i v jazyce TypeScript. Tento jazyk byl vytvořen jako jednodušší a spolehlivější nástroj pro vývoj webu. Syntaxe TypeScriptu je navržena tak, aby se co nejvíce podobala syntaxi ECMAScript 6.

Zavedení tříd a modulů výrazně usnadňuje vývoj velkých aplikací. Zavedení generik a rozhraní do typového systému umožňuje snadné vytváření komponent a knihoven, které lze využít s celou řadou objektů a pomáhají tak psát udržitelný a rozšiřovatelný kód. TypeScript přidává do JavaScriptu volitelnou statickou kontrolu datových typů, která se spouští přes interpreter. S konstrukty pro členění kódu mohou automatizované nástroje dobře rozumět kódu. Největší výhodou TypeScriptu je to, že umožňuje, nad již existujícím kódem, napsaném v JavaScriptu, postavit své vlastní vylepšené či rozšířené knihovny s využitím možností TypeScriptu. Použití TypeScriptu tak posunulo vývoj JavaScriptu na další úroveň. TypeScript je open source a podporují ho mnohá vývojová prostředí. Kromě doplňku pro Visual Studio je k dispozici například pro JetBrains WebStorm a PhpStorm pro Mac OS X i Linux. [9,10]

3.7.5 CoffeeScript

CoffeeScript vytvořil v roce 2009 Jeremy Ashkenas a byl uvolněn pod MIT licencí jako open source. Tento jazyk byl vyvinut s účelem využít dobré části JavaScriptu a vyhnout se těm špatným.

Podobně jako většina programovacích jazyků poskytuje kontrolní struktury k popisu logiky aplikace, jednoduché typování dat k uchování a nakládání s informacemi a funkce k izolaci sekcí spouštění programů. CoffeeScript byl inspirován jazyky Ruby a Python. CoffeeScript je kompilován do jazyka JavaScript a samotný jazyk JavaScript může být použit v kódu CoffeeScriptu. Výstupem

z kompileru CoffeeScriptu je tedy javascriptový kód, který je možné dále přesunout všude tam, kde je JavaScript podporovaný, např. prohlížeče, nebo samostatného javascriptového překladače. [11]

3.7.6 Dart

Dart je programovací jazyk vyvinutý společností Google určený k vytváření webových, serverových a mobilních aplikací a také v neposlední řadě pro nově rozmáhající se fenomén „Internet věcí“ (Internet of Things – IoT). Jde o open-source software dostupný pod BSD licencí. Dart byl dle autorů navržen tak, aby bylo snadné pro ně připravit vývojová prostředí vhodná k vývoji moderních aplikací. Dart zkompilovaný do jazyka JavaScript by měl také běžet rychleji než totožná aplikace napsaná v samotném JavaScriptu. Dart je plně objektově orientovaný jazyk, který je možné spustit například:

- Zkompilovaný jako JavaScript – ke spuštění Dartu v mainstreamových webových prohlížečích je nutné kód vytvořený v jazyku Dart zkompilovat do jazyka JavaScript. K tomu je použit dart2js compiler.
- Ve webovém prohlížeči Dartium. Dartium je webový prohlížeč výhradně určený k vývoji a testování aplikací. Dartium nevyžaduje kompilaci do JavaScriptu. [12]

Hlavní myšlenkou jazyka DART je podpora varianty volitelného typování, které bylo do jazyku zavedeno již od úplného začátku. Jazyk Dart byl vyvinut, aby zachytil typické chyby při kompilaci. Jedná se o jazyk orientovaný čistě na objekty. Mezi hlavní výhody patří jeho jednoduché naučení a obsah celé řady vestavěných knihoven. K tomuto jazyku je k dispozici také řada užitečných nástrojů jako jsou Dart Editor (s jeho pomocí lze vytvářet, modifikovat a spouštět aplikace Dartu), Dartboard (s jeho pomocí lze psát a spouštět Dart kód pomocí prohlížeče) nebo SDK (obsahuje nástroje na vytvoření JavaScriptu, který je možný vložit do kteréhokoli moderního prohlížeče). [13,14]

4 Javascriptové frameworky

Základním účelem každého frameworku je zobecnění řešení problému, který se často opakuje, nebo který potřebuje řešit většina programátorů. Hlavním cílem

frameworku je usnadnit práci vývojáři a zajistit pokrytí základních funkcionalit tvořené aplikace. Usnadněním se myslí to, že se tvůrce aplikace může zabývat složitostmi a rutinní úkoly nechat na frameworku. [15]

Jinými slovy, pod pojmem framework si můžeme představit kostru, kterou je potřeba obalit tak, aby mohla začít fungovat. Tento obal je výsledná aplikace, která má za základ právě onu kostru. Vývojáři se mohou soustředit na aktuální úkoly a nemusí se zaobírat věcmi, jako je například persistence dat, routování, session management apod.

4.1 Proč používat javascriptové frameworky

Hlavní argumenty pro používání javascriptových frameworků jsou:

- Nevynalézat kolo – není potřeba psát kód, který byl již mnohokrát napsán a nejspíše i lépe.
- Dělat více práce s méně řádky zdrojového kódu – většina javascriptových frameworků nabízí funkci řetězení. Řetězení umožňuje udělat více práce s méně řádky kódu. Méně kódu znamená menší čas potřebný na udržování kódu, kratší čas stahování a kratší čas samotného vytvoření kódu.
- Výkon –Javascriptový framework bude ve většině případů efektivnější než kód účelově vytvořený vývojářem. [92]

4.2 Běžné úkony zajištěné frameworkem

Mezi obecné úkony, které zajišťují webové frameworky patří:

- Podpora DOM manipulace
- Session management
- Podpora stránkového layoutu
- Podpora widgetů
- Podpora generování HTML
- Podpora validací formulářů
- Podpora web socketů
- A mnohé další…

4.3 Architektura javascriptových frameworků

Javascriptové frameworky jsou nejčastěji založeny na architekturách, které jsou popsány v následující části.

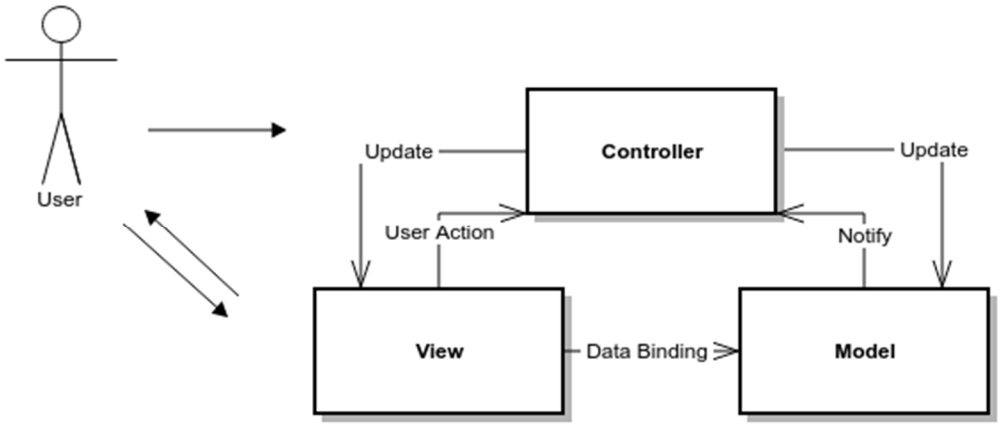
4.3.1 MVC architektura

MVC (Model-View-Controller) je návrhový vzor (design pattern) k oddělení aplikační logiky od té prezentační. Architektura MVC dělí aplikace do tří logických částí, tak aby je šlo upravovat jednotlivě a dopad změn byl na ostatní co nejmenší.

Model reprezentuje data samotná, pravidla určující přístup a update těchto dat. Model také zahrnuje samotnou business logiku, práci s daty.

View zobrazuje obsah Modelu a specifikuje, jak budou data zobrazena koncovému uživateli. V případě, že dojde ke změně dat, View na tuto změnu patřičně zareaguje.

Controller zpracovává vstupy od View a Modelu a překládá je do příkazů právě View nebo Modelu. Controller může odeslat příkazy Modelu k aktualizaci stavu modelu. Stejně tak odesílá příkazy příslušnému View ke změně prezentace modelu. [16,17]



Obrázek 2: MVC Diagram (zdroj: vlastní zpracování)

MVC diagram znázorňuje princip této architektury. Výstup aplikace má na starost vždy View, ale s uživatelským vstupem je to komplikovanější. Existují dvě principiálně odlišné možnosti:

- U „widgetových“ systémů (např. Java Swing, Silverlight) uživatelský vstup zachycuje View, kde si tento vstup ošetřují samotné komponenty v reakci na událost click a další.
- U systémů „newidgetového“ typu neexistují žádné komponenty, a tak je ošetření uživatelského vstupu zařízeno controllerem.

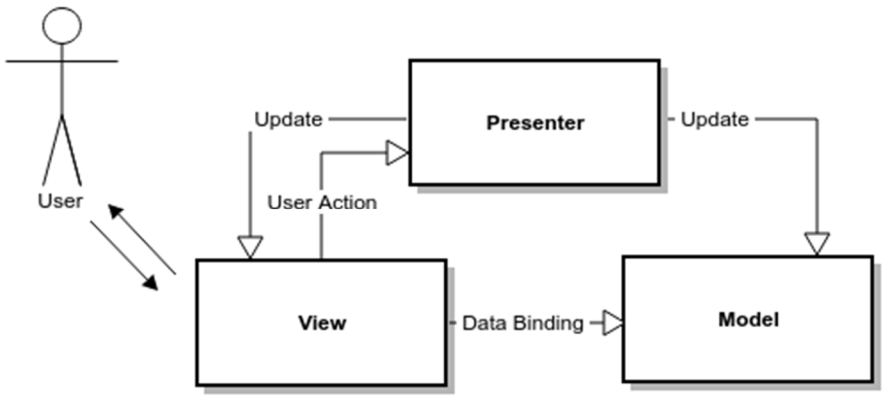
4.3.2 MVP architektura

MVP (Model-View-Presenter) je odvozen od návrhového vzoru MVC. Nicméně se od něj určitým způsobem liší. MVP určuje princip rozdělení kódu do tří vrstev a byl navržen především pro usnadnění implementace automatizovaných unit testů.

Model opět reprezentuje data k zobrazení a obsahuje business logiku.

View plně kontroluje uživatelský vstup i výstus.

Presenter v MVP je něco jako prostředník, který spolupracuje jak s Modelem, tak s view. Presenter načítá data z modelu a formátuje je pro zobrazení ve view. Obsahuje prezentační i aplikační logiku. [58,59, 60]



Obrázek 3: MVP diagram (zdroj: vlastní zpracování)

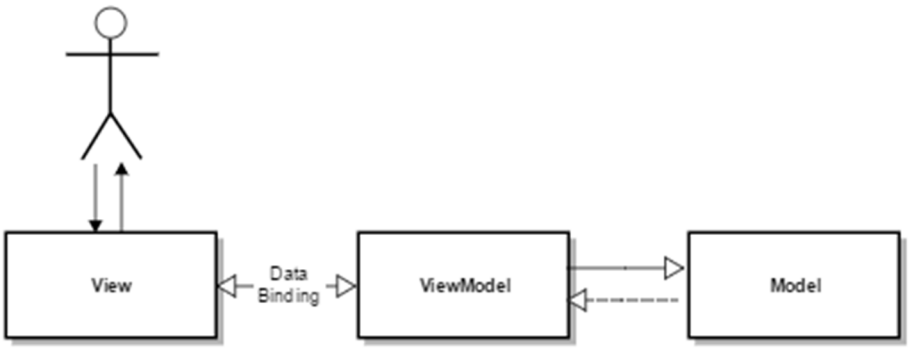
4.3.3 MVVM architektura

MVVM (Model-View-Viewmodel) je návrhový vzor odvozen opět od MVC. MVVM je také nazýván jako PM (Presentation Model). Obvykle se MVVM řadí mimo MVC/MVP vzory, ale stále se řídí podobnými principy jako MVC. [61,87,88,89]

Model jednoduše uchovává data a nemá nic společného s business logikou.

ViewModel vystavuje data a příkazy, které view potřebuje k zobrazení dat modelu.

View zobrazuje data, které získává pomocí ViewModel.



Obrázek 4: MVVM diagram (Zdroj: vlastní zpracování)

4.3.4 Reaktivní programování

Reaktivní programování není samo o sobě architekturou frameworků. Spíše jde o programovací paradigma, tedy o jakýsi základní programovací styl a také o motivaci stojící za rozhodnutí se pro tento programovací „styl“ rozhodnout.

Reaktivní programování deklaruje čtyři základní principy, které by měly programátora vést při psaní „reaktivní“ aplikace.

- Responsivita
- Škálovatelnost
- Odolnost proti výpadkům systému
- „Message-driven“ architektura

Responsivní aplikace je hlavním cílem. Responsivní systém dostatečně rychle reaguje na požadavky všech uživatelů tak, aby dlouhodobě zajistil jejich pozitivní zkušenost. Rychlost a pozitivní uživatelská zkušenost za různých provozních podmínek, jako je například selhání externího systému nebo ve špičce, závisí na dvou vlastnostech: škálovatelnost a odpovědnost. „Message-driven“ architektura poskytuje celkovou podporu pro responzivní systém. [18,19]

4.4 Využití javascriptových frameworků

Využití javascriptových frameworků je velmi široké a každým dnem se zvětšuje především díky širokému spektru problémů, které lze s jejich pomocí vyřešit. Je využíván pro psaní efektivního zdrojového kódu a udržení vzájemné kompatibility mezi webovými prohlížeči a napsaným kódem. Využití jednotlivých prvků šetří programátorův čas. Díky spolupráci s technologii AJAX se frameworky stávají mnohem silnějšími a nacházejí větší využití. Velká výhoda těchto frameworků spočívá v jednoduchém a dynamickém přístupu k jednotlivým elementům stránky (DOM model) a GUI prvkům, kterým může přiřazovat události a vytvářet animace. Některé frameworky mohou obsahovat již předpřipravené GUI komponenty. [15]

4.5 Možnosti kombinace frameworků – technologické stacky

Při vývoji komplexnějších informačních systémů a webových aplikací dochází k situacím, kdy vývojový tým zjistí, že daný framework či technologie nevyhovuje pro potřeby nějakého specifického případu užití (např. zvolený framework je pomalý při vykreslování velké množiny dat). Pokud takovýto problém nebyl při prvotní softwarové analýze odhalen a připravovaná aplikace je již v takové fázi vývoje, že by bylo velmi nákladné přejít kompletně na jinou technologii, je možnost vypomoci si jednoduše kombinací více technologií při řešení daného problému. Takovéto kombinaci technologií se říká technologický stack. Samozřejmě nejlepší variantou je zvolit správný technologický stack již při prvotní analýze softwaru. [90]

4.5.1 Co je to technologický stack

„Tech stack“ slangové označení pro kombinaci softwarových produktů a programovacích jazyků použitých k vytvoření webové nebo mobilní aplikace. Aplikace mají standardně dvě softwarové komponenty, client-side a server-side neboli front-end a back-end. Tato kapitola se zabývá problematikou technologických stacků na straně klienta.

4.5.2 Front-end technologický stack

Obecně se technologický stack na straně klienta skládá z:

- HTML
- CSS
- JavaScriptu, nebo jeho alternativy (TypeScript, CoffeeScript, Dart)
- Javascriptového frameworku nebo knihovny
- Presentačního frameworku.

4.5.3 Jak zvolit front-end technologický stack

Na otázku „jak zvolit front-end technologický stack“ neexistuje žádná konkrétní odpověď. Při volbě front-end stacku je nutné si nejprve položit několik základních otázek.

První z nich je, zda má být výsledná aplikace určena pro mobilní zařízení či webové prohlížeče. V těchto dnech se ale uplatňuje spíše pravidlo „mobile first“ což znamená, že výsledná aplikace by měla splňovat požadavky responsivního designu tak, aby byla použitelná jak na mobilních zařízeních, tak na webu.

Další otázkou je, jaké technologie jsou v dané době dostupné a jak moc se používají. Čím více organizací a vývojářů danou technologii používá či rozšiřuje, tím spíše se najde řešení pro technické otázky, které zcela jistě přijdou. Značná rozšířenost technologie také poukazuje na fakt, že jde zřejmě o kvalitní technologii, za kterou stojí velcí hráči.

Jednou z mnoha dalších otázek může být, jak schopný máte vývojový tým a jak danou technologii zvládá nebo zda je daná technologie open-source či nikoliv.

Tak či tak softwarový architekt je vždy po konzultaci se zadavatelem zodpovědný za správný výběr technologického stacku, a to platí obecně, ne jenom pro front-end stacky. [90]

4.5.4 Příklady front-end technologických stacků

Zde je uvedeno několik technologických stacků, které byly použity při implementaci konkrétních webových aplikací. [91]

Webová aplikace	Programovací Jazyky	Frameworky/Knihovny
Freeletics	JavaScript, Node.js	AngularJS, React
PlanYourTris.travel	Java	AngularJS, jQuery
Dorm Room Tycoon	CoffeeScript, JavaScript, Python	AngularJS, Gulp
Trello	CoffeeScript, Node.js	Backbone.js
Airbnb	Ruby, CoffeeScript, Node.js	Backbone.js
appknox	Python, CoffeeScript, Node.js, JavaScript	Ember.js
Tumblr	PHP, Scala, Ruby	Gulp, Webpack, React
Uber	Python, Node.js, Java, Go, JavaScript	React
Instagram	Python	React
Newspaper Club	Ruby, Objectiv-C, JavaScript, CoffeeScript, Go	Backbone.js, React, jQuery

Tabulka 2: Příklady front-end technologických stacků (zdroj: vlastní zpracování)

5 Nejpoužívanější javascriptové frameworky

Během procházení dostupných zdrojů na internetu nebyl nalezen jediný ucelený seznam javascriptových frameworků, který by obsahoval výpis všech dostupných frameworků.

Otázka, jak najít všechny javascriptové frameworky byla zúžena na výpis 39 nejpoužívanějších javasriptových frameworků, protože snaha o výpis všech javascriptových frameworků a následné srovnání těchto frameworků podle různých kritérií by značně přesáhla rozsah této práce.

5.1 Výchozí seznam javascriptových frameworků

V této kapitole je uvedeno 39 javascriptových frameworků. Všechny další kapitoly uvedené v této práci pracují s tímto, nebo dále pak se zúženým seznamem. Seznam těchto 39 javascriptových frameworků byl sestaven následujícím způsobem:

1. Na internetu byla vyhledána fráze „best JavaScript frameworks“.

2. Pro další zpracování bylo použito 10 nalezených odkazů, obsahujících subjektivní autorův výpis nejoblíbenějších javascriptových frameworků. [20-29]
3. Podle nalezených odkazů byl vytvořen následující seznam 39 frameworků: AdonisJs, AngularJS, Aurelia, Backbone.js, Babylon, Backend, D3.js, Durandal, Deku, Ember.js, Famous, Fligt, jQuery, KendoUI, Keystone, Knockout.js, Matreshka, Mercury, Meteor, Mithril, Mobservable, Nodal, Node.js, Omniscient, Polymer, Ractive, React, Responsive, Riot, scaleApp, Skel, Socket, Spine, Stapes.js, Strapi, Trails, VanilaJS, Vue.js, WebRx. [20-29]

5.2 10 frameworků dle výskytu

V předchozí podkapitole 5.1 bylo použito 10 odkazů, ve kterých se jednotlivé javascriptové frameworky z výchozího seznamu vyskytly alespoň jednou. V následující tabulce jsou vypsány frameworky s přiřazenými hodnotami výskytu.

Pořadí	Javascriptový framework	Počet výskytů
1.	AngularJS	10
2.	React	9
3.	Ember.js	9
4.	Backbone.js	8
5.	Meteor	7
6.	Aurelia	6
7.	Vue.js	5
8.	Polymer	4
9.	Knockout.js	3
10.	Mercury	3

Tabulka 3: Nejčastěji citované frameworky (zdroj: vlastní zpracování)

Stapes.js, AdonisJs a jQuery se vyskytly dvakrát a zbylých 26 frameworků se vyskytlo pouze jedenkrát, a to jsou: Matreshka, Socket, Strapi, Mobservable, Fligt, Kendo UI, scaleApp, Trails, Deku, VanilaJS, Famous, Ractive, Responsive, Skel, WebRx, Riot, Nodal, Keystone, Spine, Node.js, D3.js, Mithril, Durandal, Omniscient, Babylon a Backend. [20-29]

Již z tohoto seznamu je možné udělat si rychlý obrázek o tom, které javascriptové frameworky se v dnešní době nejvíce používají, o kterých se nejvíce

mluví a píše. Mezi tyto frameworky jsou zařazeny také jQuery a Node.js, které nelze kvůli své povaze přímo zařadit mezi javascriptové frameworky, ale s ohledem na jejich oblíbenost a to, že se zakládají na JavaScriptu, nemohou býti v této práci opomenuty. Rozdíl mezi frameworkem, knihovnou a běhovým prostředím je popsán dále v kapitole 5.7.3. Tímto také vzniká zúžený seznam 10 nejpoužívanějších frameworků dle počtu výskytu na internetu.

5.3 10 frameworků dle počtu tagů na stackoverflow.com

Popularita javascriptového frameworku je při výběru pro vývojáře důležitá, protože pro populární frameworky je online k dispozici nejvíce různých návodů a rad. Vhodný framework je tak často vybírán v závislosti na charakteristice projektu. Pomoc při výběru a hodnocení různých frameworků je možné hledat na různých specializovaných webových stránkách, např. StackOverflow.com. [64]

V tabulce níže je vybráno (z výchozího seznamu 39 frameworků) 10 frameworků, které mají na stránce stackoverflow.com nejvíce tagů (výsledky byly získány dne 22. 11. 2016). Dle tohoto parametru lze usoudit, které frameworky jsou vývojáři nejčastěji vyhledávány.

Tabulka je doplněna o počet komitů a počet přispěvatelů na github.com. Z doplňující statistiky lze vyčíst, že ač nejoblíbenějšími nástroji dle této statistiky jsou jQuery AngularJS a Node.js, nejživějšími projekty jsou Meteor, Node.js a Ember.js.

Pořadí	Framework	Počet tagů na stackoverflow.com	Počet komitů na github.com	Počet přispěvatelů na github.com
1.	jQuery	792576	6166	255
2.	AngularJS	208572	8226	1548
3.	Node.js	148672	15431	1090
4.	React	27435	7648	866
5.	Meteor	24103	18041	321
6.	D3.js	23150	4051	117
7.	Ember.js	20178	13711	625
8.	Backbone.js	20168	3328	288
9.	Knockout.js	17130	1464	62
10.	Kendo UI	12254	6832	166

Tabulka 4: Oblíbenost JS frameworků dle počtu tagů na stackoverflow.com (zdroj: vlastní zpracování)

Podle získaných výsledků z tabulky 3 lze popsat několik poznatků:

- Na prvním místě se umístila javascriptová knihovna jQuery, která se používá již 10 let, z čehož lze usuzovat, že je velmi oblíbená, a proto má tak vysokou hodnotu počtu tagů.
- Z výsledků získaných ze stackoverflow.com je zřejmé, že jeden z nejpoužívanějších frameworků je AngularJS, který má také nejvíce přispěvovatelů na github.com. Zatímco na stackoverflow.com má Ember.js 10x méně tagů v porovnání s AngularJS, počet komitů je takřka 2x vyšší u Ember.js. Z tohoto faktu je patrné, že Ember.js nabírá na popularitě a je mohutně hnán dopředu.
- Node.js je velmi používaná platforma, která se sice v některých zdrojích zařazuje mezi frameworky, ale nedá se zcela říct, že je to framework. Přesto do hodnocení byla zařazena a umístila se na přední příčce ve všech třech sledovaných hodnotách.
- Na čtvrtém místě v hodnocení podle počtu tagů se umístil React. Zjištěné hodnoty nejsou tak vysoké jako u předcházejících, ale důvodem může být to, že byl vytvořen až v roce 2013.

Pro srovnání tabulka níže obsahuje stejné frameworky a k nim příslušná data, získaná ze stránek stackoverflow.com a github.com, aktualizovaná ke dni 9. 4. 2017. Je možné porovnat s předchozími výsledky a zjistit, zdali s odstupem 5 měsíců došlo k nějaké změně v oblíbenosti těchto frameworků.

Pořadí	Framework	Počet tagů na stackoverflow.com	Počet komitů na github.com	Počet přispěvatelů na github.com
1.	jQuery	828950	6216	261
2.	AngularJS	227111	8445	1584
3.	Node.js	167138	16965	1310
4.	React	38077	8409	961
5.	Meteor	25353	18509	340
6.	D3.js	25161	4086	119
7.	Ember.js	20956	14223	659
8.	Backbone.js	20331	3338	293
9.	Knockout.js	17734	1498	64
10.	Kendo UI	12792	7559	179

Tabulka 5: Oblíbenost JS frameworků dle počtu tagů na stackoverflow.com (zdroj: vlastní zpracování)

Podle získaných výsledků z tabulky 4 v porovnání s tabulkou 3 lze popsat několik poznatků:

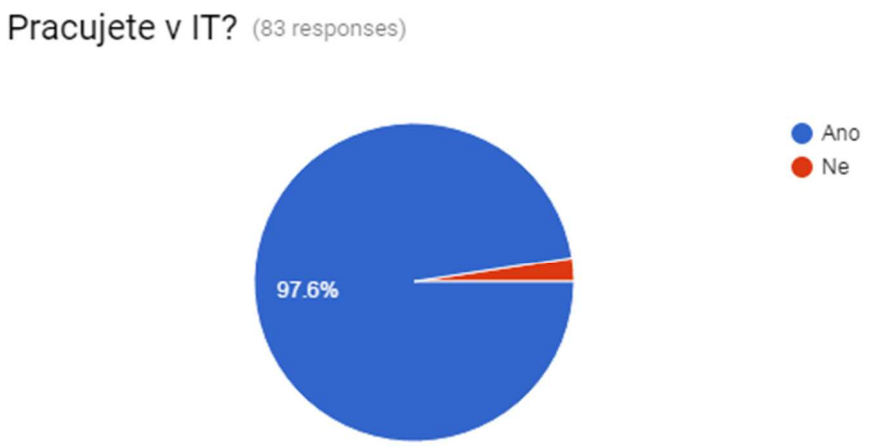
- U všech frameworků lze pozorovat navýšení.
- Počet tagů na stackoverflow.com u frameworku React výrazně vzrostl. Z toho lze usuzovat, že jde stále dopředu a na poli webových technologií hraje velkou roli.

5.4 10 frameworků dle dotazníku

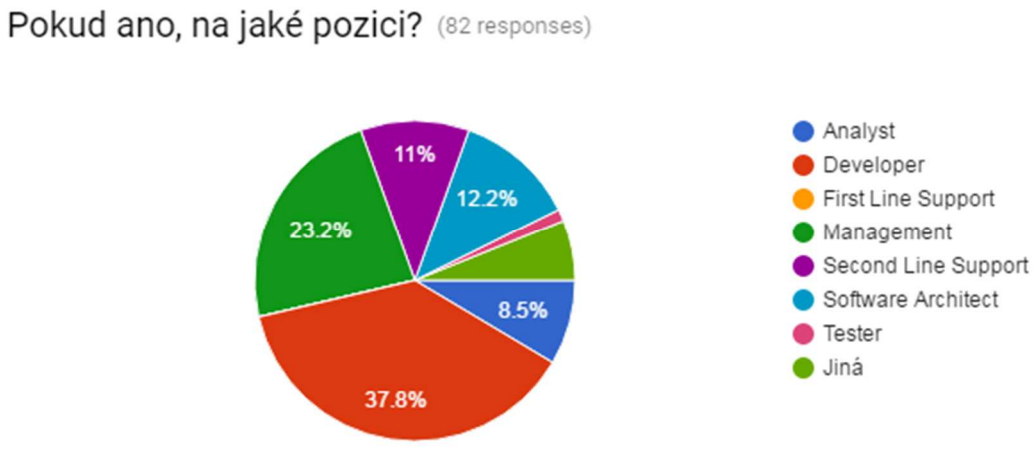
Součástí této práce je také anonymní dotazník, který byl vyplněn především pracovníky společnosti Unicorn a.s. Dotazník byl sestaven tak, aby jeho vyplnění zabralo minimální úsilí. Dotazníku se jmenuje „I like JavaScript Framework“ a byl vytvořen pomocí nástroje Google Form (vytvořeno dne 24. 11. 2016). Dotazník byl vyplněn 83 respondenty.

5.4.1 Pracujete v IT? Pokud ano, na jaké pozici?

Jelikož jde o téma široké veřejnosti nic neříkající, výsledek prvního dotazu není nijak překvapivý. 97.6 % dotázaných odpovědělo, že pracuje v IT. Touto otázkou bylo ověřeno, že většina respondentů má základní přehled o IT, a tak odpovědi na následující otázky budou relevantní.



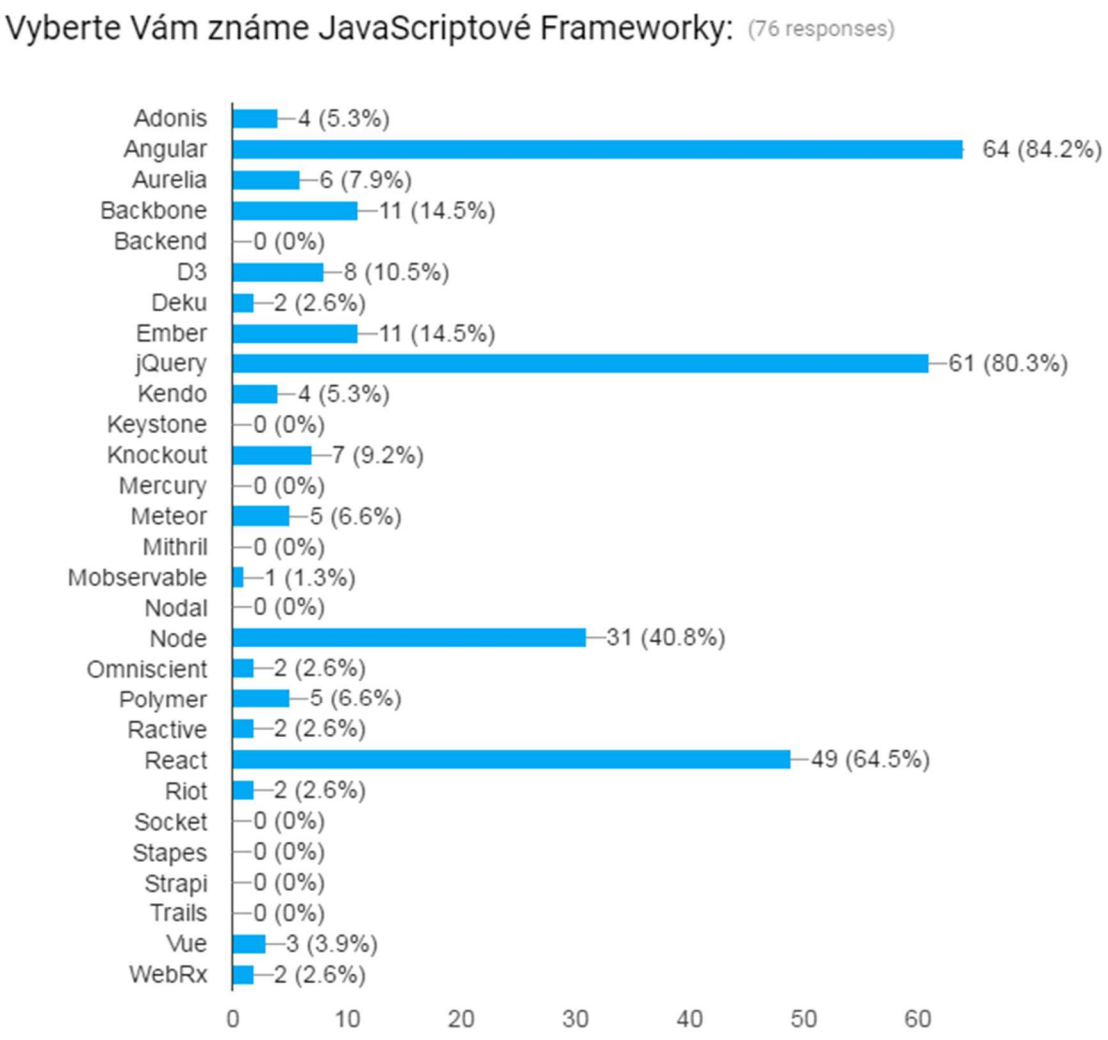
Obrázek 5: Grafické znázornění odpovědí na dotaz "Pracujete v IT? " (zdroj: vlastní zpracování, Google Form)



Obrázek 6: Grafické znázornění odpovědí na dotaz "Pokud ano, na jaké pozici? " (zdroj: vlastní zpracování, Google Form)

Z koláčového grafu výše vyplývá, že více než polovina respondentů pracuje na pozici, kde může reálně přijít do styku s javascriptovými frameworky. A to sice ze skupin: developer, second line support a software architect. Respondenti ze skupiny management a analyst mají také přístup k popisovaným technologiím. Software Architektem se na základě diskuzí právě s projekt managerem, analytikem, zákazníkem a dalšími musí rozhodnout, které technologie budou nakonec při implementaci softwarového řešení použity.

5.4.2 Vyberte Vám známé Javascriptové Frameworky



Obrázek 7: Grafické znázornění odpovědí na dotaz "Vyberte Vám známé javascriptové frameworky " (zdroj: vlastní zpracování, Google Form)

Z grafu uvedeného výše je patrné, že mezi nejznámější javascriptové frameworky ve společnosti Unicorn a.s. patří všechny očekávané frameworky. Níže je uvedena tabulka 10 nejznámějších frameworků v rámci dotazníku.

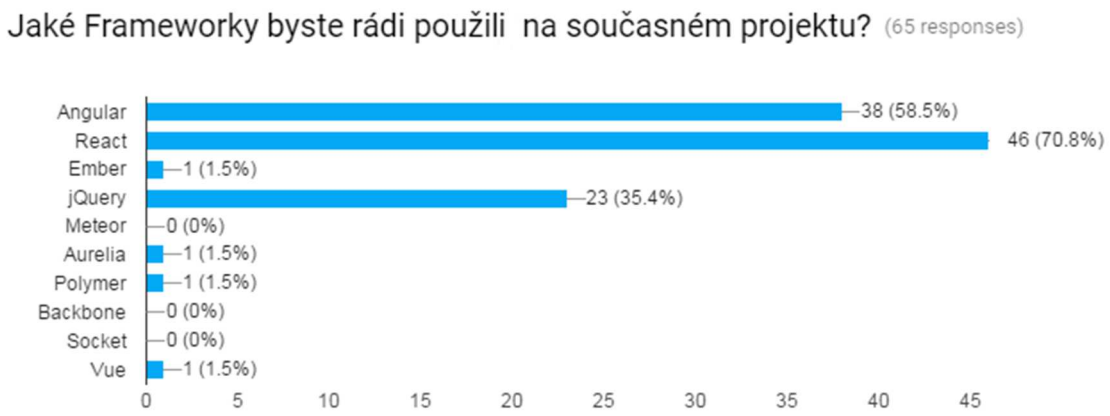
Pořadí	Framework	Počet respondentů znajících tento framework (%)
1.	AngularJS	64 (84.2%)
2.	jQuery	61 (80.3%)
3.	React	49 (64.5%)
4.	Node.js	31 (40.8%)
5.-6.	Backbone.js	11 (14.5%)
5.-6.	Ember.js	11 (14.5%)
7.	D3.js	8 (10.5%)
8.	Knockout.js	7 (9.2%)
9.	Aurelia	6 (7.9%)
10.	Meteor	5 (6.6%)

Tabulka 6: 10 nejznámějších frameworků v rámci dotazníku (zdroj: vlastní zpracování)

K této otázce patřila také doplňující otázka, „Pokud znáte jiný javascriptový framework, uveďte který“. 9 respondentů uvedlo, že v nabídce chyběl javascriptový framework „UU5 UI HTML5“ vyvíjený společností Unicorn. Tento framework je vyvíjen za podpory majitele Unicorn Vladimíra Kováře.

5.4.3 Jaké Frameworky byste rádi použily na současném projektu?

Pro tuto otázku byl výběr zúžen na 10 frameworků, které byly vybrány na základě průběžných výsledků této práce. Nejde o finální výběr 10 nejlepších frameworků.



Obrázek 8: Grafické znázornění odpovědí na dotaz "Jaké Frameworky byste rádi použili na současném projektu?" (zdroj: vlastní zpracování, Google Form)

Jak je z grafu patrné, React, AngularJS a jQuery jsou opět na vrcholu. V doplňující otázce „Pokud byste rádi použili jiný framework, prosím uveďte který“ byl opět několikrát zmíněn UiHtmlV5 javascriptový framework.

5.4.4 Postrádáte v nabídce jakýkoliv javascriptový framework?

Poslední otázkou byla respondentům dána možnost uvést jakýkoliv framework, který nebyl v dotazníku uveden. 16 respondentů opět uvedlo UiHtmlV5.

5.4.5 UiHtmlV5

Tento front-endový framework zde musí být zmíněn, jelikož byl v provedeném dotazníku několikrát citován.

Tento framework je vytvářen a podporován výhradně společnostmi v holdingu Unicorn a je podporován Vladimírem Kovářem, zakladatelem Unicornu. Tento framework není nikde veřejně dostupný, nejde tedy o klasický open-source projekt a je výhradním majetkem společnosti Unicorn.

Jelikož dotazník vyplňovali především zaměstnanci nebo spolupracovníci společnosti Unicorn, dá se usuzovat, že jde především o subjektivní ovlivnění dotazníku z řad zástupců společnosti Unicorn. Z tohoto důvodu a z důvodu uzavřenosti platformy se o tomto frameworku tato práce dále nezmiňuje.

5.5 Výsledné seřazení frameworků

V předchozích kapitolách došlo k zúžení původního seznamu 39 frameworků na 10 frameworků podle nejvyšších hodnot získaných v různých kategoriích, a to dle počtu výskytu na internetu 5.2, dle počtu tagů na stackoverflow.com 5.3 a dle dotazníku 5.4. Výsledně vznikl seznam s 16 frameworky, ze kterého se každý vyskytl nejméně jednou v deseti nejlepších dané kategorie. Stále je ale nutné vyhodnotit, které z frameworků jsou podle subjektivního pohledu oblíbenosti nejpoužívanějšími pro případné porovnávání, a to pomocí základních statistických výpočtů.

5.5.1 Shrnutí získaných výsledků

Následující tabulka obsahuje 16 frameworků a k nim přiřazená data, získaná v předchozích kapitolách. Každý sloupec v této tabulce odpovídá získaným výsledkům z několika měření, jako jsou počty výskytů na webu, počet tagů na stacoverflow.com, počet komitů a počet přispěvovatelů na github.com a také nejznámější frameworky podle dotazníků. Tabulka je rozšířená o maximální hodnotu z každého sloupce.

Frameworky	Výskyty	Počet tagů na stackoverflow	Počet komitů na github	Počet přispěvatelů na github	Dotazník
AngularJS	10	227111	8445	1584	64
React	9	38077	8409	961	49
Ember.js	9	20956	14223	659	11
Backbone.js	8	20331	3338	293	11
Meteor	7	25353	18509	340	5
Aurealia	6	2079	560	86	6
Vue.js	5	4943	1851	90	3
Polymer	4	6611	5183	110	5
Knockout.js	3	17734	1498	61	7
Mercury	3	30	589	30	0
Stapes.js	2	3	345	12	0
AdonisJs	2	24	377	13	4
jQuery	2	828950	6216	261	61
Node.js	1	167138	16965	1310	31
D3.js	1	25161	4086	119	8
Kendo UI	1	12792	7559	179	4
MAX:	10	828950	18509	1584	64

Tabulka 7: Shrnutí dosažených výsledků (zdroj: vlastní zpracování)

Cílem následujících výpočtů je seřadit tento seznam frameworků s ohlédnutím na všechny výsledky získané v jednotlivých sloupcích. Nejprve je potřeba znázornit procentuální zastoupení hodnot z maximální hodnoty každého sloupce. Výsledné hodnoty jsou v následující tabulce. Poslední řádek této tabulky obsahuje přiřazené váhy k daným sloupcům. Největší váha 50 % je přidělena sloupci s počty tagů na stackoverflow.com. 20 % váha je přidělena výskytům v 10 odkazech na internetu a zbylé sloupce jsou ohodnoceny 10 % celkové váhy.

Frameworky	Výskyty	Počet tagů na stackoverflow	Počet komitů na github	Počet přispěvatelů	Dotazník
AngularJS	100	27,40	45,63	100,00	100,00
React	90	4,59	45,43	60,67	76,56
Ember.js	90	2,53	76,84	41,60	17,19
Backbone.js	80	2,45	18,03	18,50	17,19
Meteor	70	3,06	100,00	21,46	7,81
Aurealia	60	0,25	3,03	5,43	9,38
Vue.js	50	0,60	10,00	5,68	4,69
Polymer	40	0,80	28,00	6,94	7,81
Knockout.js	30	2,14	8,09	3,85	10,94
Mercury	30	0,00	3,18	1,89	0,00
Stapes.js	20	0,00	1,86	0,76	0,00
AdonisJs	20	0,00	2,04	0,82	6,25
jQuery	20	100,00	33,58	16,48	95,31
Node.js	10	20,16	91,66	82,70	48,44
D3.js	10	3,04	22,08	7,51	12,50
Kendo UI	10	1,54	40,84	11,30	6,25
Váhy:	**20 %**	**50 %**	**10 %**	**10 %**	**10 %**

Tabulka 8: Procentuální zastoupení jednotlivých výsledků (zdroj: vlastní zpracování)

Dalším krokem výpočtu je vynásobení hodnoty ze sloupců váhou přiřazenou k danému sloupci.

Frameworky	Výskyty	Počet tagů na stackoverflow	Počet komitů na github	Počet přispěvatelů	Dotazník
AngularJS	20	13,70	4,56	10,00	10,00
React	18	2,30	4,54	6,07	7,66
Ember.js	18	1,26	7,68	4,16	1,72
Backbone.js	16	1,23	1,80	1,85	1,72
Meteor	14	1,53	10,00	2,15	0,78
Aurealia	12	0,13	0,30	0,54	0,94
Vue.js	10	0,30	1,00	0,57	0,47
Polymer	8	0,40	2,80	0,69	0,78
Knockout.js	6	1,07	0,81	0,39	1,09
Mercury	6	0,00	0,32	0,19	0,00
Stapes.js	4	0,00	0,19	0,08	0,00
AdonisJs	4	0,00	0,20	0,08	0,63
jQuery	4	50,00	3,36	1,65	9,53
Node.js	2	10,08	9,17	8,27	4,84
D3.js	2	1,52	2,21	0,75	1,25
Kendo UI	2	0,77	4,08	1,13	0,63

Tabulka 9: Procentuální zastoupení jednotlivých výsledků k frameworkům s přirazenou váhou důležitosti (zdroj: vlastní zpracování)

Posledním výpočtem pro získání výsledného seřazeného seznamu frameworků je suma hodnot z daných kategorií k jednotlivým frameworkům. Frameworky jsou seřazeny sestupně od nejvyšší hodnoty.

Pořadí	Frameworky	Výsledná suma
1.	jQuery	68,54
2.	AngularJS	58,26
3.	React	38,56
4.	Node.js	34,36
5.	Ember.js	32,83
6.	Meteor	28,46
7.	Backbone.js	22,60
8.	Aurealia	13,91
9.	Polymer	12,67
10.	Vue.js	12,34
11.	Knockout.js	9,36
12.	Kendo UI	8,61
13.	D3.js	7,73
14.	Mercury	6,51
15.	AdonisJs	4,91
16.	Stapes.js	4,26

Tabulka 10: Výsledné seřazení frameworků (zdroj: vlastní zpracování)

5.6 Popis 16 nejoblíbenějších frameworků

Tato kapitola stručně popisuje frameworky, porovnávané v předchozí podkapitole 5.5 podle oblíbenosti. Mezi tyto frameworky jsou zařazeny také jQuery a Node.js, které jak již bylo řečeno výše nelze považovat za frameworky.

5.6.1 AngularJS

AngularJS patří mezi nejpoužívanější frameworky pro vývoj moderních jednostránkových webových aplikací. Tyto aplikace se tvoří pomocí HTML kódu, do něhož jsou vkládány speciální formátovací značky. Tyto značky určují, jaké operace nebo data mají být na daném místě vloženy. Tomuto způsobu se říká *data-binding*. AngularJS je navržený pro podporu dynamického zobrazování v aplikacích, procházení stránky je pak stejně hladké jako v původní aplikaci. AngularJS využívá MVC přístus. Framework rozšířil stránku o skupinu šablon, kterým lze přiřadit funkčnost, tzv. direktivy. Ty slouží k zobrazení dynamického obsahu pomocí

dvoucestné synchronizace dat (mechanizmus, díky kterému se náhled změní v závislosti na změně modelu). Další užitečnou funkcí AngularJS je *dependancy injection,* tj. mechanizmus který umožňuje AngularJS načíst všechny služby ještě před jakýmkoli zpracováním. Mezi popisované nevýhody pak patří častá tvorba vlastních direktiv, detaily řešení problémů s degradací výkonu. [30,31,32]

5.6.2 React

Další z možností pro vytváření jednostránkových aplikací je JS Framework React. React je javascriptová knihovna a byl zpřístupněn Facebookem v roce 2013. V architektuře MVC bývá často zařazován pod View. Tento Framework využívá myšlenky, že manipulace s DOM je náročná operace a měla by se minimalizovat. U JS frameworků je běžné při aktualizaci části DOM překreslit DOM jako celek. React využívá k řešení tohoto problému tzv. virtuální DOM. Jakékoliv změny DOM jsou renderovány do tohoto virtuálního DOMu, namísto reálného DOMu. React porovná nový virtuální DOM s reálným DOMem a provede minimální počet DOM operací pro dosažení nového stavu. Aktualizují se tedy pouze změněné části. React je také deklarativní. Jakmile dojde ke změně dat, React jakoby stiskne tlačítko Obnovit a aktualizuje pouze změněné části. Další významnou funkcí je využití souborů JSX. S jejich pomocí je možné propojit JavaScript a HTML do jediného souboru. Pro použití Reactu pro vývoj mobilních aplikací je v současnosti vyvíjen React Native. [33]

5.6.3 Ember.js

Ember.js je dalším z JS frameworků s otevřeným zdrojovým kódem pro tvorbu náročných webových aplikací využívající model MVC. Framework je ještě rozšířený o routery (reprezentují stav aplikace v závislosti na určité URL) a komponenty (pro tvorbu vlastních elementů). Patří mezi oblíbené frameworky, který využívá řada oblíbených stránek, včetně Yahoo, LinkedInu nebo Netflixu. Tento framework je výborně navržený pro vývoj jednostránkových aplikací. [34,35,36]

Ember.js je postavený na těchto komponentách:

- Routes – Stav aplikace v Ember.js je reprezentována pomocí URL. Každá URL má odpovídající route objekt, který kontroluje, co je zobrazeno uživateli.
- Models – Každý route objekt má přiřazený model, obsahující data odpovídající současnému stavu aplikace.
- Controllers – Využívají se k výměně dat mezi modelem a view.
- Templates – V Ember.js jsou používány k vytvoření HTML.
- Components – Komponenty slouží k tvorbě vlastních elementů, které se dají využít v šablonách.
- Services – Jde o singleton objekty (možné vytvořit pouze jedinou instanci v rámci celé aplikace), které na sobě drží dlouho trvající. [37,38]

5.6.4 Backbone.js

Backbone.js vytvořil Jeremy Ashkenas a první verze byla zveřejněna v r. 2010. Jedná se o další z řady frameworků s architekturou Model-View-Presenter (MVP). Komunikuje skrze RESTful JSON rozhraní a od ostatních frameworků se odlišuje tím, že zde nejsou žádné controllery. Součástí Backbone.js je i router, umožňující z uživatelského rozhraní měnit URL ve webovém prohlížeči podle vyvolaných událostí. V podstatě se jedná o knihovnu, která umožňuje psaní čistého a opět použitelného kódu. Backbone.js je závislý na jediné externí JS knihovně (Underscore.js), která se snadno instaluje i používá. Také tento framework je obzvláště užitečný pro vývoj jednostránkových aplikací. [35,36]

Backbone.js je znám především pro svůj úsporný a čitelný kód. Mezi klíčové vlastnosti Backbone.js patří:

- Zjednodušuje vytváření aplikací a front-endů. K tomu využívá javascriptových funkci.
- Poskytuje komponenty jako models, views, events, routes a collections k sestavení klientských webových aplikací.
- Při změně modelu se automaticky updatuje HTML aplikace. [39,40]

5.6.5 Meteor

Meteor je open-source platforma postavená nad Node.js a MongoDB databází. V tomto případě tedy nejde pouze o framework, ale můžeme hovořit o platformě. Zatímco například AngularJS slouží pouze k vývoji front-end částí aplikace, Meteor lze použít jak k vývoji front-end části, tak k vývoji back-end (server) části aplikace.

Mezi klíčové vlastnosti Meteoru patří například:

- JavaScript na serverové i klientské straně
- Sdílení kódu mezi serverovou a klientskou částí aplikace
- Live reload – v případě změny dat se stránka překreslí automaticky bez nutnosti vyvolání reload akce ze strany uživatele
- Minimongo – Jde o klientskou implementaci MongoDB [41,42]

5.6.6 Aurelia

Aurelia patří mezi novější frameworky s řadou podobností s AngularJS 2.0. Aurelia je framework s MVVM architekturou a vytvořený pro vysokou modularitu s cílem zakomponovat pouze potřebné části. Využívá Polyfills, který umožňuje použití novějších modulů/invencí i na starších webových prohlížečích. Stejně jako v Angularu využívá two-way data binding, práce s daty je proto jednoduchá. K dispozici je celá řada zásuvných modulů a možností uživatelské konfigurace, práce s tímto frameworkem je tedy vysoce flexibilní. Na oficiální stránce frameworku je popisován jako nejpokročilejší a nejlépe uživatelsky ovládatelný frontendový framework. Je také jediným frameworkem, se kterým je možné vytvářet komponenty pomocí prostého JavaScriptu. [43,44]

5.6.7 Vue.js

Vue.js patří mezi progresivní javascriptové frameworky podobné AngularJS 2.0 pro tvorbu interaktivních webových rozhraní, což je pravděpodobně jeden z důvodů jeho popularity. Syntaxe je velmi podobná standardnímu javascriptovému kódu a je jednodušší na použití, protože jsou jasněji odděleny direktivy a komponenty. Zdrojová knihovna je zaměřená pouze na vrstvu View a lze snadno

zakomponovat do dalších knihoven nebo existujících projektů. V kombinaci s moderními nástroji a podpůrnými knihovnami lze použít také pro tvorbu jednostránkových aplikací. Možná nevýhoda je viděna v podpoře pouze prohlížečů IE9+. [45,46]

5.6.8 Polymer

Knihovna Polymer poskytuje řadu funkcí pro tvorbu požadovaných HTML elementů. Pomocí těchto funkcí lze požadované elementy, které pracují stejně jako standardní elementy DOM, vytvořit snadněji a rychleji, znovu je používat, nebo využívat elementy vytvořené jinými uživateli. Takové elementy pak lze nastavit pomocí atributů nebo vlastností. Elementy mohou reagovat na změny těchto atributů, vlastností, nebo metod, které ovládají jejich vnitřní stav. Polymer staví na webových standardech „Web Components“ zabudovaných do prohlížeče, elementy tak lze využívat i s jinými frameworky nebo knihovnami, využívajícími moderní prohlížeče. Jednou z mnoha možností využití požadovaných elementů je tvorba znovupoužitelných UI komponent. Namísto opakované tvorby určité navigační lišty nebo tlačítka v různých frameworcích, lze nadefinovat takový element pomocí Polymeru a nadále ho využívat i pro další budoucí projekty. Vedle možnosti tvorby požadovaných elementů obsahuje projekt Polymer také sadu přednastavených elementů, které lze přesunout na stránku a okamžitě používat. [47,48]

5.6.9 Knockout.js

Knockout.js se také řadí mezi frameworky s MVVM architekturou, kde ViewModel sleduje data buď z View nebo Modelu. Jedná se o javascriptovou knihovnu pro tvorbu bohatých, responsivních, zobrazovacích a editačních uživatelských rozhraní. Mezi hlavní výhody zařazují tvůrci na oficiální stránce frameworku snadnou asociaci DOM elementů s daty modelu za použití stručného a čitelného syntaxu. Dále automatické obnovení *User Interface*, *Dependency tracking* (implicitní nastavení řetězce vzájemných vztahů mezi daty modelu) a rychlou tvorbu různých šablon. Velmi působivá je také podpora starších prohlížečů (až IE6). [49,50]

5.6.10 Mercury

Mercury je podobné Reactu, ovšem ve větším měřítku. Mercury také využívá DOM, ovšem v aplikačním kódu se vůbec nemanipuluje s DOM, což znamená, že při renderování nebo spravovaní eventů není potřeba se odkazovat na DOM elementy. Mercury sestává z řady modulů, kterými lze řešit různé front-endové problémy. Pokud není některý z modulů vyhovující, lze jej díky vysoké modularitě výhodně kombinovat s moduly jiných frameworků. Obecně jsou aplikace psány jako série komponent, které vrátí stav (data komponent v jakémkoliv daném bodě) a poskytnou funkci render. Samotný zdrojový kód Mercury je velmi dobře udržovatelný, používané moduly jsou malé, dobře testované a výborně zdokumentované. [51]

5.6.11 jQuery

jQuery je javascriptová knihovna s jednoduchou syntaxí a zakládá se na programovacím jazyku JavaScript. Sama si dokáže vyřešit problémy s nekompatibilitou mezi jednotlivými prohlížeči. jQuery byla však na rozdíl od JavaScriptu, který využívá často dlouhých kódů, vyvinuta s minimálním a snadným kódováním. Mezi další výhody patří dobrá čitelnost kódu díky jednoduché syntaxi a jednodušší práce s CSS. Častým důvodem volby jQuery bývá správa animací. jQuery totiž poskytuje mnoho metod podporujících výborné animace, např. *animate*, *fadeIn*, *fadeOut*, *fadeTo*, ad., s jejichž pomocí lze animovat jakékoli elementy. Ke knihovně jQuery je k dispozici řada zásuvných modulů, které umožňují vývojářům přidávat do jQuery funkce stejným způsobem, jako kdyby přidávali tyto funkce přímo do zdrojového kódu jQuery. [52,53]

Jedinečnost jQuery lze nalézt ve způsobu jakým jsou vybírány jednotlivé elementy v rámci modelu DOM. Ty jsou vybírány pomocí selektorového jádra Sizzle. [54]

5.6.12 Node.js

Jedná se o platformu pro vývoj vysoce škálovatelných webových aplikací, zejména webových serverů. Node.js je jedním z událostmi řízených frameworků s asynchronním zpracováním, všechny I/O operace tedy neblokují vlákno

a probíhají v pozadí. Existuje zde jakási fronta událostí (event loop) a ty jsou pak zpracovány, jakmile na ně dojde řada. Uživatelské rozhraní pro Node.js je prostý JavaScript v jediném vlákně, jinak by mohlo při zpracování fronty událostí docházet k problémům se synchronizací. Skládá se z V8 JavaScript engine vytvořeného společností Google a z několika standartních knihoven. [55,56,57]

5.6.13 Stapes.js

Stapes.js je minimalistický javascriptový framework, komprimovaný gzip technologií o výsledné velikosti 1.8 Kb. Stapes.js umožňuje vyvíjet komponenty typu module, view, controller nebo pouze moduly, které jsou jednoduše znovupoužitelné. Zároveň umožňuje integrovat různé technologie, jako například jQuery, React, Vue.js a další. Díky jeho minimální velikosti je ideálním frameworkem pro vývoj mobilních aplikacích. [65,66]

5.6.14 AdonisJs

AdonisJs je MVC framework pro platformu Node.js. AdonisJs zapouzdřuje všechny nudné části vývoje Webových aplikací a nabízí pěkné a čisté API pro práci s ním. AdonisJs nabízí místo nástrojů již hotová řešení. Jsou to například:

- ORM (Objektově Relační Mapování) pro vytváření bezpečných SQL dotazů
- API a Session Based autentizační systém
- Jednoduchá podpora pro odesílání emailů pomocí SMTP nebo web services (Mailgun, Madril etc.)
- Jednoduchá validace všech vstupů od uživatele [67]

5.6.15 D3.js

D3.js je JavaScriptová knihovna určená pro manipulaci dokumentů založených na datech. D3.js používá k zobrazení dat technologie HTML, SVG a CSS. D3.js klade důraz na dodržování webových standardů. Umožňuje využití všech schopností moderních prohlížečů bez nutností svázání vývojáře s konkrétním frameworkem. Kombinuje schopné vizualizační komponenty spolu s „data-driven“ přístupem při DOM manipulaci s daty. [68]

5.6.16 Kendo UI

Kendo UI je javascriptová knihovna pro jQuery, která umožňuje vytváření moderních webových aplikací ve velmi krátkém čase. Mezi hlavní přínosy Kendo UI patří:

- 70+ UI komponent
- Možnost práce s daty v online nebo offline módu
- Integrace s AngularJS
- Kompatibilita se všemi prohlížeči [69]

5.6.17 Reference

V následující tabulce jsou přehledně uvedeny nejčastější reference výše zmíněných technologií.

Framework/Knihovna	**Reference**
AngularJS	Youtube, weather.com, netflix.com
React	Facebook, Airbnb, BBC
Ember.js	Yahoo, Zendesk, Tidle
Backbone.js	Adobe, e-bay, Twitter
Meteor	Ikea, Mazda, PGA Tour
Aurelia	GistRun, BeatMaker, Ployst
Vue.js	Codeship, Storyblok, Collate
Polymer	Google, Coca-Cola, ING
Knockout.js	Azure, Pillsbury.com, Dominos.jp
Mercury	Reference nebyly nalezeny
jQuery	Nike, Porsche, Uber
Node.js	Paypal, LinkedIn, Yahoo
Stapes.js	Reference nebyly nalezeny
AdonisJs	Reference nebyly nalezeny
D3.js	NewYork Times, BA Apps
Kendo UI	crm.int.godaddy.com, hosting.timeweb.ru, suite.seozoom.it

Tabulka 11: Seznam referencí (zdroj: vlastní zpracování)

U některých frameworků nebyly nalezeny žádné reference. To může poukazovat na to, že daný framework nemá za sebou silnou komunitu, nevyvíjí se a není tudíž ani používaný silnými hráči na trhu. [41,48,73-83]

5.7 Kategorizace frameworků

Průzkumem stránek o javascriptových frameworcích bylo zjištěno, že je těchto frameworků velké množství. Předchozí podkapitoly naznačují, že některé z nich jsou velmi známé a používané, na druhou stranu některé nejsou tak známé. Seznam byl zúžen podle oblíbenosti, avšak je potřeba těmto frameworků přidělit určité vlastnosti, neboť nejsou identické a každý je něčím speciální. Proto se tato podkapitola zabývá bližšími poznatky o nich.

5.7.1 Rozdělení podle typu architektur

Následující tabulka obsahuje frameworky rozdělené podle architektur, na kterých jsou založeny.

MVC	**MVP**	**MVVP**	**Reaktivní programování**
jQuery	Backbone.js	Aurelia	Vue.js
AngularJS		Knockout.js	Mercury
React		Kendo UI	
Node.js			
Ember.js			
Meteor			
Polymer			
D3.js			
AdonisJs			
Stapes.js			

Tabulka 12: Frameworky dle typu architektury (zdroj: vlastní zpracování)

Z tabulky je zřejmé, že nejvíce frameworků je založeno na MVC architektuře, která umožňuje, aby webové stránky obsahovaly minimální skriptování. To znamená, že je prezentační logika oddělena od PHP skriptování. Ostatní architektury jsou odvozené od MVC a liší se v různých detailech.

5.7.2 Rozdělené podle typu licence

Velkou výhodou javascriptových frameworků je, že jsou skoro všechny jejich verze open source. To znamená, že jejich zdrojový kód je otevřený uživatelům a je volně dostupný ke stažení. Otevřenost kódu je vždy omezena legální licencí, která specifikuje, jaká práva uživatel získá k danému otevřenému kódu frameworku. Následně jsou frameworky rozděleny podle typu licence a popsány tyto jednotlivé licence.

MIT licence	BSD licence	Apache licence 2
AngularJS	React	Kendo UI Core
Ember.js	Polymer	
Backbone.js	D3.js	
Meteor		
Aurelia		
Vue.js		
Knockout.js		
Mercury		
AdonisJs		
Node.js		
jQuery		

Tabulka 13: Frameworky dle typu licencí (zdroj: vlastní zpracování)

Jediná verze Kendo UI Proffessional je zpoplatněnou verzí frameworku. Všechny ostatní zmíněné frameworky a knihovny používají jednu z open-source licencí.

- MIT Licence – MIT licence je jedna z nejotevřenějších licencí vůbec. Uživatel smí kód pod touto licencí užívat komerčně, upravovat, distribuovat, vytvářet pod-licence a využívat pro vlastní účely. Autor softwaru pod MIT licencí nedrží žádnou zodpovědnost za poskytnutý kód. Uživatel musí vždy uvést Copyright a do zdrojových kódů přiložit licenční znění. [84]
- BSD Licence – BSD licence je další licence hojně využívaná v open-source komunitě. BSD existuje ve třech variantách a to: 4 bodová, 3 bodová, 2 bodová. V této době je připravována i 0 bodová varianta. V 3 bodové variantě uživatel smí použít software pod BSD licencí pro komerční účely, modifikovat ho, distribuovat a umísťovat záruku. Naopak nesmí používat

obchodní značku a vyžadovat záruku po autorovi. Uživatel musí používat copyright a vkládat licenční znění. [85]

- Apache Licence 2 – I tato licence patří především do světa open-source technologií. Uživatel smí používat software komerčně, modifikovat, distribuovat, sublicencovat, používat pro privátní účely, využívat patentové nároky a umisťovat záruku. Uživatel naopak nesmí požadovat záruku po autorovi a používat obchodních značek. Uživatel musí vkládat copyright, licenci, uvádět změny v kódu a vkládat „NOTICE“ soubor, pokud je používán přispěvovateli. [86]

5.7.3 Rozdělení podle designu

Rozdělení frameworků dle designu není vždy jednoznačná záležitost. Neexistuje zřejmě žádný jasný řez, podle kterého by se dalo jednoznačně říci, zda jde o framework, platformu nebo knihovnu. V následující tabulce jsou javascriptové frameworky rozděleny podle designu.

Framework	Knihovna	Platforma	Runtime Environment
AngularJS	React	Meteor	Node.js
Ember.js	Polymer		
Backbone.js	Knockout.js		
Aurelia	jQuery		
Vue.js	D3.js		
Mercury	Kendo UI		
Stapes.js			
AdonisJs			

Tabulka 14: Frameworky rozdělené dle designu (zdroj: vlastní zpracování)

- **Knihovna** je kolekce objektů, funkcí a metod. Jsou to jen části kódů, které jsou znovupoužitelné. Knihovny na rozdíl od frameworků neřeší komplexně otázku persistence dat, session managementu a apod.
- **Platforma** integruje více technologií dohromady za účelem zajištění životního cyklu aplikace. Od programovacího jazyka a knihoven, přes vývojové prostředí až po runtime prostředí, ve kterém může být výsledná aplikace spuštěna.

- **Runtime environment** neboli běhové prostředí, tedy prostředí, ve kterém může býti výsledná aplikace spuštěna. Node.js je například takovým runtime prostředím. [70,71,72]

5.8 Analýza výkonu vybraných frameworků

V této kapitole je na tři nejvýše umístěné frameworky (AngularJS, React, Ember.js) z tabulky 10 pohlédnuto z hlediska výkonu. jQuery a Node.js nebyly vybrány, protože se nejedná o frameworky.

Analýza poskytuje zevrubný pohled na výkonnost testovaných frameworků. K analýze výkonu jsou použity Dostupné z webové stránkydroje z internetu.

5.8.1 Stefan Krause benchmark test

Nejrozsáhlejší zdroj, co se týče testování výkonu javascriptových frameworků je webový portál Stefana Krause.

V úvodním článku „Benchmarking JS-Frontend Frameworks“ z 25. ledna 2016 se autor zabývá, jakým způsobem porovnat výkon javascriptových frameworků. Z článku vyplývá, že objektivně porovnat výkon jednotlivých frameworků může býti složité, jelikož každý framework přistupuje k zobrazení finální podoby stránky jinak.

Jednou z možností je použití nástrojů zabudovaných přímo v prohlížečích, jako je například „chrome timeline tool“. Hlavní nevýhodou těchto nástrojů je to, že nejsou automatizované a zpracování výsledků testu může být časově náročnější.

K zautomatizování testu výkonu jednotlivých frameworků mohou sloužit například tzv. „Hooks“, což jsou programátorem přidané „berličky“ do kódu, které jsou následně odchyceny automatizovaným nástrojem na měření výkonu. Například AngularJS nabízí metodu $postdigest, React metodu componentDidMount. Tyto metody jsou volány ihned po manipulaci s DOM elementy. Měření pomocí těchto „hooku“ pak v sobě nezahrnuje renderování stránky a vykreslení stránky, což už není v režii frameworku, ale o tyto dvě akce se stará webový prohlížeč. Jedním z automatizovaných nástrojů pro test výkonu frameworků může být browser-perf.

Další možností, jak testovat výkon javascriptových frameworků je „vlastní řešení“. Autor tohoto článku použil nástroj „selenium webdriver“, který umí reportovat výkonnostní report z nástroje „chrome timeline“.

V článku „JS web frameworks benchmark – Round 1“ z 26. ledna 2016 se autor pokusil změřit výkon několika populárních javascriptových frameworků. Mezi nimi jsou i AngularJS, React a Ember.js. K měření použil selenium benchmark test, který si sám připravil. Podrobný popis, jak tento test funguje, může být nalezen již ve zmíněném článku „Benchmarking JS-Frontend Frameworks“. Níže seznam akcí, které byly v testu měřeny:

- Vytvoření 1000 řádků
- Update 1000 řádků
- Částečný update – update každého 10. řádku
- Výběr řádku
- Odebrání řádku

Výsledky testu jsou shrnuty v tabulce níže. Hodnoty v tabulce byly přibližně odečteny z grafů dostupných v uvedeném zdroji (nižší hodnota je lepší).

Kategorie	AngularJS 1	AngularJS 2	Ember.js	React
Vytvoření 1000 řádků čas v ms	250	180	630	530
Update 1000 řádků čas v ms	295	185	590	250
Částečný update čas v ms	20	18	160	25
Výběr řádku čas v ms	8	6	60	8
Odebrání řádku čas v ms	80	140	102	80

Tabulka 15: Stefan Krause benchmark test (zdroj: vlastní zpracování)

V tomto výkonnostním testu je vítězem AngularJS 2 s 21 body. Za ním se umístil AngularJS 1 s 18 body, následuje React s 16 body a poslední skončil Ember.js s pouhými 7 body (body byly rozdány v každé kategorii podle tohoto klíče: 1. místo 4 body, 2. místo 3 body, 3. místo 2 body, 4. místo 1 bod).

Z testů dále vyplývá, že AngularJS 1 v porovnání s Ember.js a React si nevede nijak špatně a AngularJS 2 dosáhl zlepšení výkonu od své první verze. Ember.js dosáhl absolutně nejhorších výsledků, a to nejen v porovnání s AngularJS a React. [62]

5.8.2 Sebastian Peyrott benchmark test

Tento benchmark test vyšel 7. února 2016 a mimo jiných obsahuje porovnání třech vybraných frameworků AngularJS, React a Ember.js. Autor v tomto testu výkonnosti použil nástroj „browser-perf“. Pro účel tohoto testu byl tento nástroj navíc rozšířen o generování statistik využití operační paměti.

Zde jsou vypsány sledované parametry tohoto testu:

- Major GC (Garbage Collection) – čištění JVM (Java Virtual Machine) „Tenured“ části
- Minor GC – čištění JVM „Young“ částí (Eden, Survivor 1 a Surviver 2)
- Layout – čas strávený prohlížečem na úpravu vnitřní reprezentace změn v DOM stromu
- Vykreslení – čas strávený prohlížečem při rasterizaci elementů
- Snímky za sekundu – počet vykreslených snímků za sekundu
- Počet snímků vykreslujících se déle než 16.6ms
- Javascript – celkový čas, který prohlížeč strávil vykonáním javascriptového kódu
- Použitá velikost paměti – průměrné využití paměti prohlížečem během výkonnostního testu

Výsledky testu jsou shrnuty v tabulce níže. Hodnoty v tabulce byly přibližně odečteny z grafů dostupných v uvedeném zdroji.

	AngularJS 1	AngularJS 2	Ember.js	Ember.js 2	React
Major GC čas v ms (nižší je lepší)	270	30	28	25	20
Minor GC čas v ms (nižší je lepší)	310	73	230	149	25
Layout čas v ms (nižší je lepší)	170	220	670	790	100
Vykreslení čas v ms (nižší je lepší)	70	95	300	350	60
Snímky za sekundu (vyšší je lepší)	38	50	47	46	54
Počet snímků vykreslujících se déle než 16.6ms (nižší je lepší)	470	240	525	526	180
Javascript čas v ms (nižší je lepší)	4800	1600	6400	6200	800
Průměrná použitá velikost JS Heap v megabytech (nižší je lepší)	92,5	48,8	48,6	42	17,2

Tabulka 16: Sebastian Peyrott benchmark test (zdroj: vlastní zpracování)

V tomto výkonnostním testu je absolutním vítězem React, který dopadl nejlépe ve všech měřených statistikách a získal celkem 40 bodů. Za ním se umístil AngularJS 2 s 26 body a na 3. – 5. místě se shodně umístily frameworky Ember.js, Ember.js 2 a AngularJS s 18 bodů (body byly rozdány v každé kategorii podle tohoto klíče: 1. místo 5 bodů, 2. místo 4 body, 3. místo 3 body., 4. místo 2 body, 5. místo 1 bod)

Zatímco u AngularJS se při přechodu mezi verzí 1 a verzí 2 podařilo některé výkonnostní problémy odstranit, o Ember.js se totéž říci nedá a stále patří mezi výkonnostně nejhorší frameworky.

Autor článku zmiňuje, že React je volbou jedna především z důvodu jeho skvělé výkonnosti, lehké integraci a čistému, čitelnému kódu. [63]

5.9 Instalace vybraných frameworků

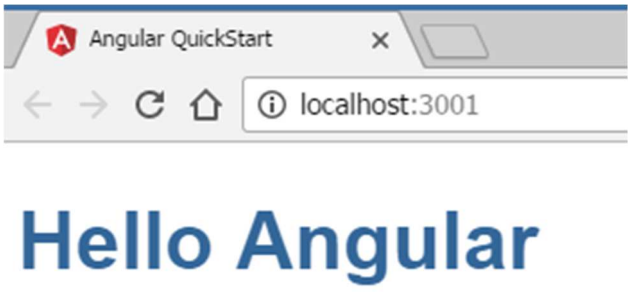
V této kapitole je popsána instalace 3 nejvýše postavených frameworků (AngularJS, React a Ember.js) z tabulky 10. jQuery a Node.js nebyly vybrány, protože se nejedná o frameworky.

5.9.1 Instalace AngularJS 2

Při instalaci AngularJS 2 je čtenář dle domovské stránky nejprve vyzván k instalaci Node.js a npm. Node.js se instaluje především z důvodu, že AngularJS 2 je založen na Typescriptu a ten vychází z ECMAScript 6. Tyto technologie vyžadují kompilaci do ECMAScript 5, který je podporován naprostou většinou dnešních webových prohlížečů na rozdíl od ECMAScript 6. K této kompilaci je používán právě Node.js. Nástroj npm je takzvaný balíčkovací nástroj.

Instalace AngularJS 2 netrvá déle než 30 minut a skládá se z těchto kroků:

1. Instalace Node.js a npm
2. Vytvoření projektové složky
3. Zkopírování QuickStart projektu „Hello Angular“
4. Instalace npm balíčků
5. Spuštění QuickStart aplikace „Hello Angular“

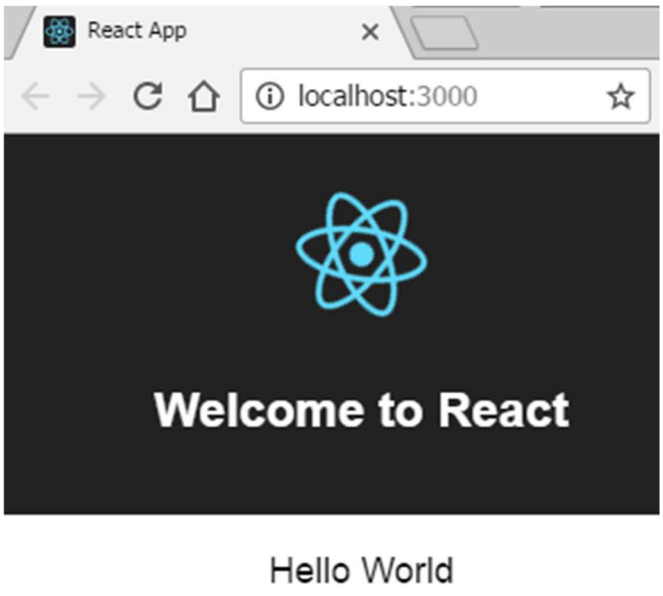


Obrázek 9: "Hello Angular" aplikace spuštěná po instalaci frameworku AngularJS 2 (zdroj: vlastní zpracování)

5.9.2 Instalace React

Pro instalaci React je opět zapotřebí balíčkovací nástroj npm. S použitím tohoto nástroje je instalace a spuštění „Hello World“ aplikace snadné, stačí postupovat dle kroků uvedených na domovské stránce frameworku React.
Instalace React netrvá opět déle než 30 minut a skládá z těchto kroků:

1. Instalace React za použití npm
2. Vytvoření „Hello World“ projektu
3. Spuštění aplikace „Hello World“



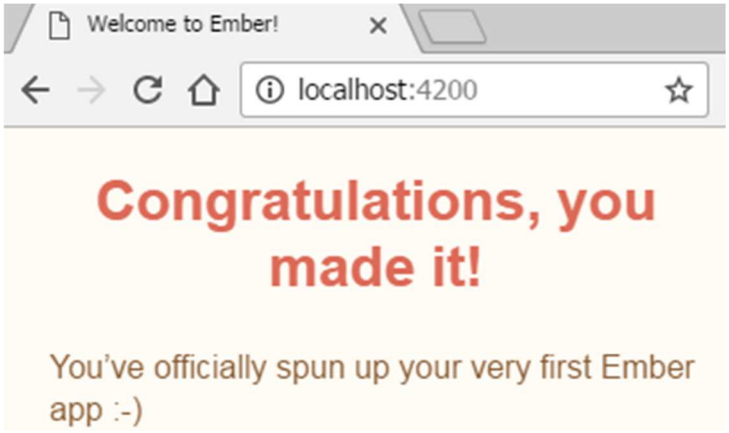
Obrázek 10: "Hello World" aplikace spuštěná po instalaci frameworku React (zdroj: vlastní zpracování)

5.9.3 Instalace Ember.js

I instalace Ember.js vyžaduje nástroj npm. S jeho pomocí je instalace dle quick-start příručky na domovské stránce snadná.

Instalace Ember.js netrvá opět déle než 30 minut a skládá z těchto kroků:

1. Instalace Ember.js za pomocí npm
2. Vytvoření Ember.js quickstart aplikace
3. Spuštění Ember.js



Obrázek 11: "Hello World" aplikace spuštěná po instalaci frameworku Ember.js (zdroj: vlastní zpracování)

6 Shrnutí výsledků

Základní otázkou při hledání nejpoužívanějších javascriptových jazyků, frameworků a knihoven bylo, jakým způsobem tyto technologie vyhledat tak, aby vznikl ucelený a konečný seznam nejvíce používaných frameworků pro vytváření moderních webových a mobilních aplikací. Vypsat a vyhledat všechny frameworky je téměř nemožné, protože neustále vznikají nové a některé z nich nejsou ani veřejně dostupné. Počet webových frameworků, knihoven a dalších technologií je tak obrovský, že vytvořit ucelený seznam je nemožné. Nakonec při procházení zdrojů dostupných na internetu bylo zjištěno, že dané zdroje vždy popisují konečnou množinu nejlepších nebo nejpoužívanějších frameworků dle subjektivního názoru autora.

Řešení základní otázky tedy spočívalo ve vyhledání nejlepších a nejpoužívanějších javascriptových frameworků z dostupných zdrojů, vytvořených několika autory na sobě nezávislých. Po prohledání relevantních zdrojů na internetu vznikl seznam 39 javascriptových frameworků, mezi které jsou zařazeny také jQuery a Node.js. Je zcela zřejmé, že by tento seznam mohl obsahovat daleko větší počet technologií, ale pro přehlednost této práce, která je zaměřená především na použivatelnost a oblíbenost těchto technologií, bylo vybráno jen základních 39 technologií. Tento základní seznam byl dále podroben několika zkoumáním, které se zaměřily vždy na 10 nejlepších v dané kategorii. Výsledkem tohoto zkoumání byl seznam 16 nejoblíbenějších frameworků. Důkazem této oblíbenosti jsou také reference na celosvětově známé společnosti, které tyto frameworky využívají.

Na základě vytvořeného seznamu 16 frameworků byly vybrány tři nejpoužívanější frameworky a to: AngularJS, React a Ember.js. Tyto tři frameworky byly prozkoumány z hlediska výkonu tak, aby si vývojář mohl udělat rychlý obrázek o tom, jak si v této kategorii daný framework stojí. Popis instalace demonstruje snadnost a rychlost, s jakou lze s danými frameworky začít pracovat.

AngularJS, React a Ember.js jsou v dnešní době určitě nejrozšířenější a nejpoužívanější javascriptové frameworky, které mají velkou podporu open-source komunity a tím je jim garantována udržitelnost rozvoje v budoucnu, široká

podpora na diskuzních fórech a dostupnost mnoha zdrojů, které se pomohou vypořádat s případnými výzvami, které použití těchto frameworků může přinést.

Otázka, jakou technologii nakonec použít je vždy na architektovi, který je zodpovědný za výsledný softwarový produkt. Pokud architekt ještě nepřišel do styku s moderním javascriptovým frameworkem, může vybrat jeden či několik z uvedených frameworků a může si být jist, že se mu dostane dostatečné podpory open-source komunity v případě, že se dostane do úzkých.

7 Závěry a doporučení

Téma moderních webových technologií zažívá v současné době obrovský boom ruku v ruce s nástupem chytrých zařízení. Množství technologií, které lze zvolit pro dodání výsledné aplikace je nepřeberné. Tato práce odpověděla na jednu ze základních otázek, které webové frameworky jsou v této době nejpoužívanější.

Zorientovat se v této problematice může býti tvrdým oříškem, jak jsem si sama ověřila. Počet technologií užívaný napříč webovou komunitou se zdá býti nekonečným. Najít objektivní zdroj s informací o tom, která technologie je nejoblíbenější je nemožné. Každý, kdo se o to pokusí vždy poskytne pouze vlastní subjektivní názor. Jedinou možností, jak se dobrat alespoň trochu objektivního žebříčku se ukázalo shrnutí různých zdrojů s různými úhly pohledu. V práci se podařilo několika nezávislými zdroji potvrdit, že v současné době patří mezi tři nejoblíbenější webové technologie AngularJS, React a Ember.js.

Na tuto práci by se v budoucnu dalo navázat nejen její aktualizací vzhledem k tomu, že téměř s každým dnem je dostupná nová verze jednotlivých technologií nebo technologie samotná, ale také například konkrétnějším rozdělením frameworků dle jejich užití.

8 Seznam použité literatury

1. FLANAGAN David. *JavaScript: kompletní průvodce: referenční příručka pro programátory Webu*. První vydání Brno: Computer Press, 1998, 710 s. ISBN 80-7226-093-6.

2. SUEHRING Steve. *JavaScript: krok za krokem*. První vydání Brno: Computer Press, 2008, 336 s. ISBN 978-80-251-2241-9.

3. SCHUTTA Nathaniel T.; ASLESON, Ryan. *Ajax: Vytváříme vysoce interaktivní webové aplikace*. První vydání Brno: Computer Press, 2006, 272 s. ISBN 80-251-1285-3.

4. HOLZNER Steven. *Mistrovství v AJAXu*. První vydání Brno: Computer Press, 2007, 591s. ISBN 978-80-251-1850-4.

5. RESIG John. *JavaScript a Ajax*. První vydání Brno: Computer Press, 2007, 360 s. ISBN 978-80-251-1824-5.

6. POWELL Thomas, SCHNEIDER Fritz. *JavaScript 2.0-The Complete Reference, Second Edition* [online]. 2nd ed. Emeryville, California: McGraw-Hill Osborne Media, 2004 [citováno 14.11.2016], 976 s. s. 159–161. Dostupné z webové stránky: http://gamerfancreations.weebly.com/uploads/3/9/6/6/39665638/mcgraw-hill-javascript-the-complete-reference.pdf. ISBN 0072253576.

7. NOETICSUNIL. *Compile to JavaScript: 10 Best JavaScript Alternative Languages* [webová stránka], 2016. Noeticforce.com. Dostupné z webové stránky: http://noeticforce.com/alternative-programming-languages-that-compile-to-javascript (9.11.2016).;

8. BERTI Alberto (GitHub). *List of languages that compile to JS* [webová stránka], 2016. GitHub.com. Dostupné z webové stránky: https://github.com/jashkenas/coffeescript/wiki/list-of-languages-that-compile-to-js (9.11.2016).

9. NANCE Christopher. *TypeScript Essentials* [webová stránka]; Packt Publishing: Birmingham, 2014. Dostupné z webové stránky: https://books.google.cz/books/about/TypeScript_Essentials.html?id=Uob1BAAAQBAJ&source=kp_cover&redir_esc=y (11.11.2016).

10. MICROSOFT Corporation. *TypeScript* [webová stránka], 2016. TypeScriptLang.org. Dostupné z webové stránky: http://www.typescriptlang.org/index.html (10.11.2016).

11. YOUNG Ian. *CoffeeScript Application Development* [online]. První vydání Birmingham, UK: Packt Publishing Ltd., 2013 [citováno 14.11.2016]. Dostupné z webové stránky: http://pdf.th7.cn/down/files/1411/CoffeeScript%20Application%20Development.pdf. ISBN ISBN 978-1-78216-266-7.

12. GOOGLE Inc. *Dart* [webová stránka], 2016. DartLang.org. Dostupné z webové stránky: https://www.dartlang.org/ (14.11.2016).

13. MEZZETTI Gianluca, MØLLER Anders, STROCCO Fabio. Type Unsoundness in Practice: An Empirical Study of Dart. In *Proceedings of the 12th Symposium on Dynamic Languages*. 2016, s. 13–24.

14. BRACHA Gilad. *The Dart Programming Language* [online]. První vydání Addison-Wesley Professional, 2015 [citováno 14.11.2016]. Dostupné z webové stránky: https://books.google.cz/books?id=UHAlCwAAQBAJ&printsec=frontcover&hl=cs&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false. ISBN 9780133429954.

15. ŠUBRTA Václav. J*avascriptové frameworky* [webová stránka], 2014. Grafická a multimediální laboratoř VŠE. Dostupné z webové stránky: http://gml.vse.cz/data/oppa-webdesign/javascriptove-frameworky.html (18.11.2016).

16. APPLE Inc. *Model-View-Controller* [webová stránka], 2015. Developer.apple.com. Dostupné z webové stránky: https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html (18.11.2016).

17. ECKSTEIN Robert. *Java SE Application Design With MVC* [webová stránka], 2007. Oracle.com. Dostupné z webové stránky: http://www.oracle.com/technetwork/articles/javase/index-142890.html (18.11.2016).

18. BONÉR Jonas, FARLEY Dave, KUHN Roland, THOMPSON Martin. *The Reactive Manifesto* [webová stránka], 2014. Reactivemanifesto.org. Dostupné z webové stránky: http://www.reactivemanifesto.org/ (18.11.2016).

19. WEBBER, K. *What is Reactive Programming*? [website], 2014. Medium.com. Dostupné z webové stránky: https://medium.com/reactive-programming/what-is-reactive-programming-bc9fa7f4a7fc#.7cl6crbft (18.11.2016).

20. ZIA Daniel. *6 Best JavaScript Frameworks to Learn In 2016*, 2016. DiscoverSDK Tools for Developers. Dostupné z webové stránky: http://www.discoversdk.com/blog/6-best-javascript-frameworks-to-learn-in-2016/ (21.11.2016).

21. NOETICSUNIL. *JavaScript Frameworks: The Best 10 for Modern Web Apps*, 2016. Noeticforce.com. Dostupné z webové stránky: http://noeticforce.com/best-Javascript-frameworks-for-single-page-modern-web-applications (21.11.2016).

22. KUZ Maria. *5 Best JavaScript Frameworks*, 2016. MLSDev: Web and Mobile App Development Company. Dostupné z webové stránky: http://mlsdev.com/en/blog/63-5-best-javascript-frameworks (21.11.2016).

23. IVANOVS Alex. *Top 23 Best Free JavaScript Frameworks for Web Developers 2016*, 2016. Colorlib.com. Dostupné z webové stránky: https://colorlib.com/wp/javascript-frameworks/ (21.11.2016).

24. O'BRIEN Niall. *JavaScript Frameworks in 2016*, 2016. Clock.co.uk. Dostupné z webové stránky: http://www.clock.co.uk/blog/javascript-frameworks-in-2016 (21.11.2016).

25. HOLLAND Burke. *What To Expect From JavaScript In 2016 – Frameworks*, 2016. Telerik Developer Network. Dostupné z webové stránky: http://developer.telerik.com/featured/what-to-expect-from-javascript-in-2016-frameworks/ (21.11.2016).

26. JSCRAMBLER. Top 8 Most Popular JavaScript Frameworks, 2016. Jscrambler.com. https://blog.jscrambler.com/top-8-most-popular-javascript-frameworks/ (21.11.2016).

27. BARRET Lucy. Top JavaScript Frameworks for Building Web Applications, 2015. MonsterPost. Dostupné z webové stránky: http://www.templatemonster.com/blog/top-javascript-frameworks-building-web-applications/ (21.11.2016).

28. PRAT. *15 Best JavaScript Frameworks for Developers*, 2015. Codecondo.com. Dostupné z webové stránky: http://codecondo.com/15-best-javascript-frameworks-for-developers/ (21.11.2016).

29. DO Van. *What Is The Best JavaScript Frameworks To Learn?*, 2016. Designveloper.com. Dostupné z webové stránky: https://blog.designveloper.com/2016/09/30/javascript-frameworks/ (21.11.2016).

30. CHANSUWATH Wutthichai, SENIVONGSE Twittie. *A Model-Driven Development of Web Applications Using AngularJS Framework*. Computer and Information Science (ICIS), 2016 IEEE/ACIS 15th International Conference

on. IEEE [online]. 2016, s. 1–6. Dostupné z webové stránky: http://ieeexplore.ieee.org/stamp/stams.jsp?arnumber=7550838.

31. RAMOS Miguel et al. *AngularJS in the wild: a survey with 460 developers*. arXiv preprint arXiv:1608.02012 [online]. 2016. Dostupné z webové stránky: https://arxiv.org/pdf/1608.02012.pdf

32. BRANAS, R. *AngularJS Essentials* [online]. UK: Packt Publishing Ltd., 2014 [citováno 22.11.2016]. Dostupné z webové stránky: http://www.ebooksbucket.com/uploads/itprogramming/javascript/AngularJS-Essentials-Branas-Rodrigo.pdf

33. VIPUL, A. M.; SONPATKI, Prathamesh. *ReactJS by Example-Building Modern Web Applications with React*. Packt Publishing Ltd, 2016 [citováno 22.11.2016]. Dostupné z webové stránky: https://books.google.cz/books?hl=cs&lr=&id=Ht3JDAAAQBAJ&oi=fnd&pg=PP1&dq=react+javascript&ots=EvEhENG9hQ&sig=HTi_sjKvXZ5E7FWu0O--WIcxLWo&redir_esc=y#v=onepage&q=react%20javascript&f=false. ISBN 978-1-78528-964-4

34. SKEIE Joachim Haagen. *Ember.js in Action* [online]. První vydání NY: Manning Publications Co., 2014 [citováno 22.11.2016]. Dostupné z webové stránky: https://www.manning.com/books/ember-js-in-action.

35. OCARIZA Frolin, et al. An empirical study of client-side JavaScript bugs. In: *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2013. s. 55-64. Dostupné z webové stránky: http://ece.ubc.ca/~frolino/projects/js-bugs-study/js_bugs_study_paper.pdf

36. NITZE André, SCHMIETENDORF Andreas. Modularity of JavaScript libraries and frameworks in modern web applications. In: Selected Topics to the User Conference on Software Quality, Test and Innovation (ASQT 2014), OCG, Klagenfurt, AT. 2014. Dostupné z webové stránky:

https://www.researchgate.net/profile/Andre_Nitze2/publication/265914279_Modularity_of_JavaScript_Libraries_and_Frameworks_in_Modern_Web_Applications/links/54bce1700cf29e0cb04c51ca.pdf

37. HASSMAN Martin. *Ember.JS – k čemu slouží a proč se o něj zajímat,* 2014. Zdroják.cz. https://www.zdrojak.cz/clanky/ember-js-cemu-slouzi-proc-se-nej-zajimat/ (22.11.2016).

38. KNEBEL Julien. *An In-Depth Introduction To Ember.js,* 2013. Smashing Magazine. Dostupné z webové stránky: https://www.smashingmagazine.com/2013/11/an-in-depth-introduction-to-ember-js/#main_concepts (22.11.2016).

39. ASHKENAS Jeremy. *BackboneJS Tutorial*, 2010. Tutorialspoint.com. Dostupné z webové stránky: https://www.tutorialspoint.com/backbonejs/backbonejs_overview.htm (22.11.2016).

40. BACKBONE.JS. Backbonejs.org. Dostupné z webové stránky: http://backbonejs.org/ (22.11.2016).

41. Meteor Development Group Inc. *The Fastest Way to Build JavaScript*, 2016. Meteor.com. Dostupné z webové stránky: https://www.meteor.com/ (22.11.2016).

42. SEVILLEJA Chris Learn. *Meteor.js From Scratch: Build a Polling App*, 2015. Scotch.io. Dostupné z webové stránky: https://scotch.io/tutorials/learn-meteor-js-from-scratch-build-a-polling-app (22.11.2016).

43. GitHub Inc. *Aurelia-framework*, 2016. GitHub. Dostupné z webové stránky: https://github.com/aurelia/framework (22.11.2016).

44. Blue Spire Inc., 2016. Aurelia.io. Dostupné z webové stránky: http://aurelia.io/ (23.11.2016).

45. GitHub Inc. *Vuejs/ vue*, 2016. GitHub. Dostupné z webové stránky: https://github.com/vuejs/vue (23.11.2016).

46. YOU Evan. *The Progressive JavaScript Framework*, 2014-2016. Vue.js. Dostupné z webové stránky: https://vuejs.org/ (23.11.2016).

47. GitHub Inc. *Polymer*, 2016. GitHub. Dostupné z webové stránky: https://github.com/Polymer/polymer (23.11.2016).

48. POLYMER Authors. *Polymer library*, 2016. Polymer Project. Dostupné z webové stránky: https://www.polymer-project.org/1.0/docs/devguide/feature-overview (23.11.2016).

49. KNOCKOUTJS.COM. *Knockout*, 2016. Knockoutjs.com. Dostupné z webové stránky: http://knockoutjs.com/ (23.11.2016).

50. AMBLER, T.; CLOUD, N. Knockout. In: *JavaScript Frameworks for Modern Web Dev*. Apress, 2015. s. 121-153.

51. GitHub Inc. *Mercury*, 2016. GitHub Inc. Dostupné z webové stránky: https://github.com/Raynos/mercury (24.11.2016).

52. CHAUDHARY Mukund, KUMAR Ankur. *Practical JQuery*. Apress, 2015 [citováno 24.11.2016]. Dostupné z webové stránky: https://scholar.google.cz/scholar?start=10&q=jQuery+&hl=cs&as_sdt=0,5&as_ylo=2012. ISBN 978-1-4842-0788-8.

53. LERNER Benjamin S., et al. *Combining form and function: Static types for JQuery programs*. In: European Conference on Object-Oriented Programming. Springer Berlin Heidelberg, 2013. s. 79-103 [24.11.2016]. Dostupné z webové stránky: https://cs.brown.edu/~sk/Publications/Papers/Published/lelk-types-jquery-programs/paper.pdf

54. GitHub Inc. jquery/ sizzle, 2016. GitHub Inc.. https://github.com/jquery/sizzle/wiki (24.11.2016).

55. Node.js Foundation. *Node.js® Interactive*, 2016. Node.js.org. Dostupné z webové stránky: https://nodejs.org/en/ (24.11.2016).

56. GitHub Inc. *nodejs/ node*, 2016. GitHub Inc. Dostupné z webové stránky: https://github.com/nodejs/node (24.11.2016).

57. MADSEN Magnus, TIP Frank, LHOTÁK Ondřej. Static Analysis of Event-Driven Node.js JavaScript Applications. *ACM SIGPLAN Notices* [online]. 2015 [citováno 24.11.2016], s. 505–519. Dostupné z webové stránky: http://plg.uwaterloo.ca/~olhotak/pubs/oopsla15.pdf

58. ADDYOSMANI javascriptmvc.md, 2011. GitHub Inc. Dostupné z webové stránky: https://gist.github.com/addyosmani/1594678 (27.05.2017).

59. BERNARD Borek. Prezentační vzory z rodiny MVC, 2009. zdrojak.cz. Dostupné z webové stránky: https://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/ (27.05.2017).

60. POTEL Mike. MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java, 1996. Wildcrest Associates. Dostupné z webové stránky: http://www.wildcrest.com/Potel/Portfolio/mvs.pdf (27.05.2017).

61. DAJBYCH Václav. mvvm: model-view-viewmodel, 2009. dotnetportal.cz. Dostupné z webové stránky: http://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel (27.05.2017).

62. KRAUSE, Stefan. JS web frameworks benchmark – Round 2, 2016. STEFAN_KRAUSE.BLOG(). Dostupné z webové stránky: http://www.stefankrause.net/wp/?p=283 (27.05.2017).

63. PEYROTT Sebastian. More Benchmarks: Virtual DOM vs Angular 1 & 2 vs Others, 2016. auth0.com. Dostupné z webové stránky: https://auth0.com/blog/more-benchmarks-virtual-dom-vs-angular-12-vs-mithril-js-vs-the-rest/ (27.05.2017).

64. GRAZIOTIN, D.; ABRAHAMSSON, S. Making sense out of a jungle of JavaScript frameworks. In: *International Conference on Product Focused Software Process Improvement*. Springer Berlin Heidelberg, 2013. s. 334-337. Dostupné z webové stránky: https://arxiv.org/ftp/arxiv/papers/1306/1306.1773.pdf

65. Stapes.js, 2017. hay.github.io/stapes/. Dostupné z webové stránky: https://hay.github.io/stapes/ (30.05.2017).

66. KANAKIYA Jay. Stapes.js, 2013. jquer.in. Dostupné z webové stránky: http://jquer.in/javascript-frameworks-for-developing-rich-applications/stapes-js/ (30.05.2017).

67. VIRK Harminder. AdonisJs. adonisjs.com. Dostupné z webové stránky: http://www.adonisjs.com/ (30.05.2017).

68. BOSTOCK Mike. D3.js Data-Driven Documents, 2017. d3js.org. Dostupné z webové stránky: https://d3js.org/ (30.05.2017).

69. PROGRESS SITEFINITY. How Does Kendo UI for jQuery Cut Development Time?, 2017. Kendo UI. Dostupné z webové stránky: http://www.telerik.com/kendo-ui (30.05.2017).

70. ZENNER Roman. Framework vs. Platform, 2017. commercetools. Dostupné z webové stránky: https://commercetools.com/blog/2015/framework-vs-platform.html (30.05.2017).

71. Framework vs. Toolkit vs. Library, 2017. Stack Overflow. Dostupné z webové stránky: https://stackoverflow.com/questions/3057526/framework-vs-toolkit-vs-library (30.05.2017).

72. What is the difference between a framework and a library?, 2017. Stack Overflow. Dostupné z webové stránky: https://stackoverflow.com/questions/148747/what-is-the-difference-between-a-framework-and-a-library (30.05.2017).

73. SHAIKH Sabeer. Top 15 websites and apps built with AngularJS, 2015. Eduonix. Dostupné z webové stránky: https://www.eduonix.com/blog/web-programming-tutorials/top-15-websites-and-apps-built-with-angularjs/ (31.05.2017).

74. GitHub Inc Sites Using React, 2017. GitHub Inc. Dostupné z webové stránky: https://github.com/facebook/react/wiki/sites-using-react (31.05.2017).

75. Ember: A framework for creating ambitious web applications, 2017. Ember. Dostupné z webové stránky: https://www.emberjs.com/ (31.05.2017).

76. ROGERS A., BREWER G. Websites using Backbone.js, 2017. BuiltWith. Dostupné z webové stránky: https://trends.builtwith.com/websitelist/Backbone.js?var=b (31.05.2017).

77. CHARRINGTON Dwayne. Built With Aurelia, 2017. builtwithaurelia.com. Dostupné z webové stránky: http://builtwithaurelia.com/ (31.05.2017).

78. MASSINGER M., ULRICH A. Made with vue.js, 2017. madewithvuejs.com. Dostupné z webové stránky: https://madewithvuejs.com/app (31.05.2017).

79. PAPA John. Web Sites Using Knockout.js, 2013. johnpapa.net. Dostupné z webové stránky: https://johnpapa.net/web-sites-using-knockoutjs/ (31.05.2017).

80. Best Websites Examples Of Designs With jQuery, 2016. awwwards.com. Dostupné z webové stránky: https://www.awwwards.com/websites/jquery/ (31.05.2017).

81. PLUSZCZEWSKA Bianka. 9 Famous Apps Built with Node.js, 2016. brainhub.eu. Dostupné z webové stránky: https://brainhub.eu/blog/9-famous-apps-using-node-js/ (31.05.2017).

82. Which large websites use D3.js? 2016. quora.com. Dostupné z webové stránky: https://www.quora.com/Which-large-websites-use-D3-js (31.05.2017).

83. Websites using Kendo UI, 2017. wappalyzer.com. Dostupné z webové stránky: https://wappalyzer.com/applications/kendo-ui.

84. WANG Kevin. MIT License, 2014. tldrlegal.com. Dostupné z webové stránky: https://tldrlegal.com/license/mit-license (31.05.2017).

85. WANG Kevin. BSD 3-Clause License (Revised), 2014. tldrlegal.com. Dostupné z webové stránky: https://tldrlegal.com/license/bsd-3-clause-license-(revised) (31.05.2017).

86. WANG Kevin. Apache License 2.0, 2014. tldrlegal.com. Dostupné z webové stránky: https://tldrlegal.com/license/apache-license-2.0-(apache-2.0) (31.05.2017).

87. JIRAVA Jarda. Používáme Model-View-ViewModel – úvod, 2010. xaml.cz. Dostupné z webové stránky: http://xaml.cz/wpf/pouzivame-model-view-viewmodel-uvod/ (31.05.2017).

88. MVVM – Introduction, 2017. tutorialspoint.com. Dostupné z webové stránky: https://www.tutorialspoint.com/mvvm/mvvm_introduction.htm (31.05.2017).

89. What is the difference between MVC and MVVM?, 2017. stackoverflow.com. Dostupné z webové stránky: https://stackoverflow.com/questions/667781/what-is-the-difference-between-mvc-and-mvvm (31.05.2017).

90. EDELMAN Gil. How to Choose Your Tech Stack, 2017. SV/SG Silicon Valley Software Grous. https://svsg.co/how-to-choose-your-tech-stack/ (31.05.2017).

91. TechStacks, 2016. techstacks.io. Dostupné z webové stránky: http://techstacks.io/stacks (31.05.2017).

92. WALSH David. 6 Reasons To Use JavaScript Libraries & Frameworks, 2007. David Walsh Blog - JavaScript Consultant. Dostupné z webové stránky:

https://davidwalsh.name/6-reasons-to-use-javascript-libraries-frameworks (31.05.2017).

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2017/2018

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Pustowková Eva	Ropice 242, Ropice	11300618

TÉMA ČESKY:

Moderní javascriptové jazyky a frameworky v dynamickém webu

TÉMA ANGLICKY:

Modern Javacsript Languages and Frameworks in Dynamic Web

VEDOUCÍ PRÁCE:

Ing. Jiří Cabal - KIT

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl: Cílem této bakalářské práce je popis dnešních moderních javascriptových jazyků a frameworků, výběr nejvhodnějších a nejpoužívanějších pro dynamický web.

1. Úvod
2. Cíl práce
3. Javascript
4. Javascriptové frameworky
5. Nejpoužívanější javascriptové frameworky
6. Shrnutí výsledků
7. Závěry a doporučení
8. Seznam použité literatury

SEZNAM DOPORUČENÉ LITERATURY:

FLANAGAN David. JavaScript: kompletní průvodce: referenční příručka pro programátory Webu. 1st ed. Brno: Computer Press, 1998, 710 p. ISBN 80-7226-093-6.

SCHUTTA Nathaniel T.; ASLESON, Ryan. Ajax: Vytváříme vysoce interaktivní webové aplikace. 1st ed. Brno: Computer Press, 2006, 272 p. ISBN 80-251-1285-3.

RESIG John. JavaScript a Ajax. 1st ed. Brno: Computer Press, 2007, 360 p. ISBN 978-80-251-1824-5.

Podpis studenta: Pustowková **Datum:** 18.10.2017

Podpis vedoucího práce: **Datum:** 18.10.2017