UNIVERZITA HRADEC KRÁLOVÉ
UNIVERSITY OF HRADEC KRÁLOVÉ

FAKULTA INFORMATIKY A MANAGEMENTU
KATEDRA INFORMATIKY

FACULTY OF INFORMATICS AND MANAGEMENT
DEPARTMENT OF INFORMATICS

MODERNÍ JAVASCRIPTOVÉ FRAMEWORKY
MODERN JAVASCRIPT FRAMEWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL DOBROVOLNÝ
AUTHOR

VEDOUCÍ PRÁCE — doc. Mgr. TOMÁŠ KOZEL, Ph.D
SUPERVISOR

Místo Rok

PROHLÁŠENÍ

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

Místo ....

podpis autora(-ky)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské/diplomové práce vedouci za metodické vedení práce a ...
@TODO

ANOTACE

Abstrakt práce v originálním jazyce

ANNOTATION

Překlad anotaceu v angličtině (nebo češtině pokud je originální jazyk angličtina)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM UKÁZEK

SEZNAM TABULEK

ÚVOD

@TODO přepsat pak podle skutečného obsahu ...

Tato práce se věnuje oblasti moderního JavaScriptu, zejména pak srovnání client-side frameworků. Jejím cílem je zjisti a porovnat, která z frameworků je pro programování isomorfní aplikace nejrychlejší na implementaci s nejnižším počtem vstupních informací. Zároveň její dokumentace odpovídá nejlépe skutečnosti. V potaz je bráno také bráno vystupování autorů na konferencích nebo licence.

První část této práce se věnuje stručnému popisu historie JavaScriptu. Kdo ho vytvořil, jaký byl jeho účel a čeho se podařilo dosáhnout. Také pak jeho vývoj až do současné podoby dle specifikace ECMAScript... @TODO

Druhá část je věnována porovnání frameworků na základě zvolených kritérií. Konkrétně programátorské náročnosti, výkonnosti, možností, kvality dokumentace, aktivity autorů a licencí. Během některých porovnání jsou použity velice krátké ukázky implementace demonstrující použití na příkladu. To hlavně pak v

Třetí část obsahuje popis implementace aplikace ve vybraném frameworku. Aplikací je client relationship manager. Jejím úkolem bude správa projektů, práce s požadavky na úpravu a hlášení chyb. Evidování odpracovaného času na konkrétních požadavcích. Klienti dané společnosti budou mít možnost se přihlásit a nahlásit jeden z typů požadavků. Administrátoři pak budou míst možnost přiřadit požadavek pracovníkovi a ten na něj vykazovat svou činnost. Hlavním cílem aplikace bude demonstrovat důvody zvolení konkrétního frameworku na reálné aplikaci.

V poslední části se práce zabývá shrnutím dosažených závěrů. Obsahuje srovnání všech frameworků pro různé parametry a reprezentuje informace v přehledné podobě tabulek.

Při psaní této práce nebyl navržen ani nevyhotoven žádný další JavaScriptový framework.

... @TODO

1 JAVASCRIPT A JEHO HISTORIE

Javascript je interpretovaný jazyk s objektově orientovanými možnostmi. Jeho syntaxe je převzata z jazyka Java, který byl pro jeho vytvoření inspirací. JavaScript však není ani Java, ani skript, jak by mohl někdo z názvu odvodit.

Pro některé programátory mohl nebo může být JavaScript jazykem, o kterém nechtějí ani slyšet. Jeho rozdílná implementace v prohlížečích, divoký vývoj v posledních letech a vysoké požadavky na znalosti pro vstup do ekosystému, mohou být problém pro nejednoho programátora.

1.1 Vznik

JavaScript spatřil světlo světa poprvé v roce 1995. Jeho autorem byl Brendan Eich pracující pro Netscape communication. Pro návrh jazyka se inspiroval Javou, Scheme a Self. [6] Úkolem bylo autora bylo vytvořit jazyk podobný jazyku Java. To se mu také do určité míry povedlo.

1.2 JavaSript a Java

JavaScript může být často spojován s jazykem Java který byl jeho inspirací. Je možno se dopracovat k názoru, že je JavaScript zjednodušenou verzí Javy. To však není ani trochu pravda. Pokud nebude brána v potaz syntaktická podobnost a vlastnost přinést do webové stránky aktivní části (u Javy již nepoužívané). Nemají jazyky nic společného. [6]

1.3 ECMAScript

ECMAScript je normalizovaný skriptovací jazyk a to společností ECMA od roku 1997. [10] Tento jazyk má pak své interpretace jako JavaScript, JScript nebo ActionScript.

Specifikace jsou vydávány vždy v Červnu roku pro nějž jsou určeny. Tedy ECMAScript 2015 (ES2015) v Červnu roku 2015.

1.3.1 ES5

ECMAScript 5th edition (ES5) je pátou edicí standardizovaného EcmaScriptu vydanou v roce 2009. Jedná se o nejpodporovanější verzi ECMAScriptu. Je implementována téměř ve všech moderních prohlížečích a držela si pozici 6 let.

1.3.2 ES6 / ES2015

V roce 2015 byla zavedena nová konvence pojmenovávání. Z ECMAScript 6th edition (ES6) se stal z důvodu jednoduší identifikace ES2015. V komunitě se však používá název ES6 velice často.

Do jazyka byly přidány konstanty, block-scope proměnné, block-scope funkce a další. Nejzajímavější novinkou jsou však tzv. arrow funkce a třídy. Array funkce umožňují velice krátký zápis funkce. Příkladem může být procházení pole pomocí `[1, 2, 3].map(item => item*10)`.

@TODO dopsat další novinky? celkem hodně zajímavých přínosů.

1.3.3 ES2016

ECMAScript 2016 (ES2016) je specifikací pro rok 2016 jak už nová zkratka napovídá. Nejde zdaleka o tak rozsáhlý update jako v podobě ES2015, ale jde o update zaručující ECMAScriptu posun po malých krocích.

Mezi novinky patří rozšíření integrovaného pole o funkci `[].includes()` a možnost exponenciálního výpočtu pomocí krátkého zápisu dvou hvězdiček (základ ** exponent).

1.3.4 ES2017

Nejnovější specifikace ECMAScript 2017 (ES2017) zatím nebyla oficiálně vydána. Její podoba se stále utváří. Dosavadně nejzajímavější novinkou je pak `async/await`, která slouží jednodušší obsluze asynchronních částí aplikace.

1.4 Možnosti využití

JavaScript již není pouze nástrojem k rozblikání stránky. Umožňuje programovat komplexní aplikace.

V dnešní době je možno využít jej pro téměř všechny platformy. Jako izomorfní, single-page, desktopovou nebo mobilní aplikaci.

1.4.1 JavaScript v prohlížeči

JavaScript je jedním z prostředků umožňující tvorbu interaktivních webových stránek. Jeho hlavním účelem je přidání aktivních prvků, které nefungují pouze na bázi dotaz/odpověď jako např. u PHP.

S využitím moderních technologií internetu a frameworků poskytuje nástroj umožňující tvorbu vysoce interaktivních webových stránek.

Velkou výhodou využití JavaScriptu v prohlížeči je vlastnost být serverově nezávislý. Každý návštěvních webu je výpočetní jednotkou, která potřebuje od serveru pouze data. Sama už vyřeší interpretaci (např. vykreslení, směrování atd..).

1.4.2 Server

Node.js byl přestastaven na konferenci JSConf 2009 mladým programátorem jménem Rayn Dahl. Jednalo se o projekt který byl platformou kombinující V8 engine (Google) a událostní smyčky (event loop). Node.js umožňuje použití JavaScriptové aplikace jako serveru. [7]

Kvůli nespokojenosti komunity se složitostí prosazováním změn a rychlosti implementace nových specifikací dle ECMAScript se část komunity rozhodla o odtržení a založila na projektu Node.JS další server-side nástroj se jménem io.js.

Io.js je komunitou řízený nástroj pro tvorbu server-side aplikací v JavaScriptu. Io.js dokázalo v několika prvních měsících přilákat více aktivních vývojářů než Node.js za celou dobu života předtím. [5]

1.4.3 Desktop

Výhodou využití JavaScriptu pro dektopovou aplikaci je jeho podpora všemi platformami. Ve spojení s HTML a CSS pak tvoří mocný stroj pro aplikace všeho druhu.

Pro tvorbu desktopové aplikace lze JavaScriptu využít stejně jako například Javy. To díky frameworkům jako Electron nebo Appjs. Electron je v současnosti nejpopulárnějším frameworkem pro tvorbu destktopové aplikace. Jeho hlavní přednosti jsou jednoduchost, podpora nativních API systémů a možnost rychlé tvorby windows instalátoru. [9]

1.4.4 Mobilní zařízení

Aplikace mobilních telefonů se těší v posledních letech veliké popularitě. JavaScript se tedy také snaží ukousnou si svoji část koláče na této platformě. Hlavní motivací pro použití JavaScriptu je fakt, že se jedná o jazyk znám velkému množství vývojářů. Dalším pozitivem je možnost sdílení zdrojového kódu mezi platformami.

Jednou z možností by bylo využít responzivní webové aplikace. To však není příliš populární řešení z důvodu user experience (UX), výkonu a potřeby být neustále připojen k internetu.

Veliké popularitě se tak těší React Native od Facebooku. Ten využívá nativních komponent prostředí do kterého je přenesen a je možno jej využít i s kombinací nativních programovacích jazyků. Facebook ho sám aktivně využívá u aplikací jako

Instagram, Facebook, Facebook ADS manager. Další velice populární aplikací naprogramovanou v React Native je Airbnb. [8]

1.5 Základní nástroje vývoje

Vývoj aplikace je v JavaScriptu obehnán velkým množstvím podpůrných nástrojů pro komfort vývojáře.

1.5.1 Package manager

Package manager je základním nástrojem pro přidávání závislostí v každé moderní aplikaci. Je znám a dostupný snad ve všech jazycích. Například v PHP je to Composer, v Javě Maven nebo v Ruby RubyGems.

NPM

NPM je nejrozšířenějším v jazyce JavaScript. Ještě nedávno byl NPM využívám hlavně k instalaci balíčků podporujících funkčnost aplikace. Jeho působnost se rozšířila i na front-end. V současné době jsou však z NPM dostupné všechny balíčky, frameworky a utility. Nahradil tedy nástroje jako je Bower.

Yarn

Yarn je v současné době nejpopulárnějším balíčkovacím nástrojem. Na githubu se pyšní téměř 22 tisíci hvězdiček. Jeho hlavní přednosti jsou dle tvůrců (Facebook) bezpečnost, rychlost a spolehlivost. [1]

1.5.2 Webpack

Webpack je tzv. mobule bundler, tedy nástroj transformující aplikaci do podoby vhodnou pro prohlížeč koncového uživatele.

Jeho hlavním přínosem je možnost psát aplikace dle nejnovějších standardů a využívat nejnovějších vlastností JavaScriptu. Příkladem mohou být specifikace ES2015 a ES2016. Tyto vlastnosti nejsou totiž podporovány všemi prohlížeči a Webpack tak umožňuje vývojáři využívat jejich komfortu a přesto tvořit aplikaci kompatibilní i se staršími prohlížeči.

Hot-reload je jedním z dalších skvělých vlastností tohoto nástroje. Jedná se o možnost vidět změnu v aplikaci i bez přenačtení stránky, tedy ztráty současného stavu aplikace (state). Možnost je dostupná pouze u vývojového serveru.

1.5.3 Spouštěč úkolů

Spouštěč úkolů je nejčastěji využíván pro kompilace. Příkladem může být kompilace Syntactically Awesome Style Sheets (SASS) na Cascading Style Sheets (CSS). Avšak jeho využití je mnohem širší. Může se starat o automatické spouštění testů při vývoji, restart dev serveru, kompilaci TypeScriptu na JavaScript nebo kontrolu zdrojového kódu pomocí JSLint.

Grunt

Jednotlivé úkoly se v Gruntu konfigurují pomocí gruntfile. Ty se pak spouští z client line interface (CLI).

```
module.exports = function(grunt) {

  grunt.initConfig({
    jshint: {
      files: ['Gruntfile.js', 'src/**/*.js', 'test/**/*.js'],
      options: {
        globals: {
          jQuery: true
        }
      }
    },
    watch: {
      files: ['<%= jshint.files %>'],
      tasks: ['jshint']
    }
  });

  grunt.loadNpmTasks('grunt-contrib-jshint');
  grunt.loadNpmTasks('grunt-contrib-watch');

  grunt.registerTask('default', ['jshint']);

};
```

Ukázka 1.1: gruntfile [3]

Gulp

Zásadní odlišností Gulpu od Gruntu je preferování programování nad konfigurací.

Důležité je definovat gulpfile.js. Ten pak vyžaduje, načítá a spouští ostatní.

```
var gulp = require('gulp');
var pug = require('gulp-pug');
var less = require('gulp-less');
```

```
4  var minifyCSS = require('gulp-csso');
5
6  gulp.task('html', function(){
7    return gulp.src('client/templates/*.pug')
8      .pipe(pug())
9      .pipe(gulp.dest('build/html'))
10 });
11
12 gulp.task('css', function(){
13   return gulp.src('client/templates/*.less')
14     .pipe(less())
15     .pipe(minifyCSS())
16     .pipe(gulp.dest('build/css'))
17 });
18
19 gulp.task('default', [ 'html', 'css' ]);
```

Ukázka 1.2: gulpfile.js [4]

2 SROVNÁNÍ FRAMEWORKŮ

Výběr správného frameworku může být zásadní pro budoucí vývoj v aplikace. Obzvlášť v JavaScriptu, kde balíčky, frameworky a utility vznikají a zanikají velice rychle.

Frameworků pro programování jednostránkové či izomorfní aplikace je v JavaScriptu víc než jen několik. Ve skutečnosti jich je tolik, že přesný počat je těžko dohledatelný. K porovnání byly tedy vybrány pouze tři frameworky. Jmenovitě Angular 2, Aurelia, Ember a Backbone. Další možností by určitě mohl být React od Facebooku. Je ho role na jevišti JavaScriptu je jistě nezaměnitelná. Avšak porovnání je věnováno frameworkům. React je pouze šablonovací knihovna.

2.1 Seznámení s frameworky

Všechny frameworky které jsou níže porovnávány mají spoustu společného, mají otevřené zdrojové kódy, podléhají licenci umožňující komerční využití a snaží se vyřešit problematiku vytváření jednostránkové aplikace za pomocí Model View Controller (MVC) či příbuzných konceptů. Všechny mají podporu šablonovacích systému, událostí a směrování.

Angular 2 dostal označení stabilní verze dne 14.9.2016. Mezi jeho zajímavé vlastnosti patří data binding, dependency injection (DI), easy-to-test a rozšíření HTML dialektu pomocí direktiv.

Aurelia je zajímavá především z důvodů integrace web component, databindingu, testovatelnosti a svého šablonovacího systému. Na Aurelii je také zajímavé, že je z dílny jednoho z členů týmu Angular 2.0 a to Roba Eisenberga.

Backbone se stal populárním velice rychle a to hlavně díky tomu, že se jedná o lehkou architekturu MVC.

Ember se pyšní striktnějším přístupem k vývoji aplikací. To z něj dělá důležitého hráče. Výhodou je možnost rychlé adaptace na neznámém projektu, který je vždy psán podobnou konvencí.

2.2 Komunita

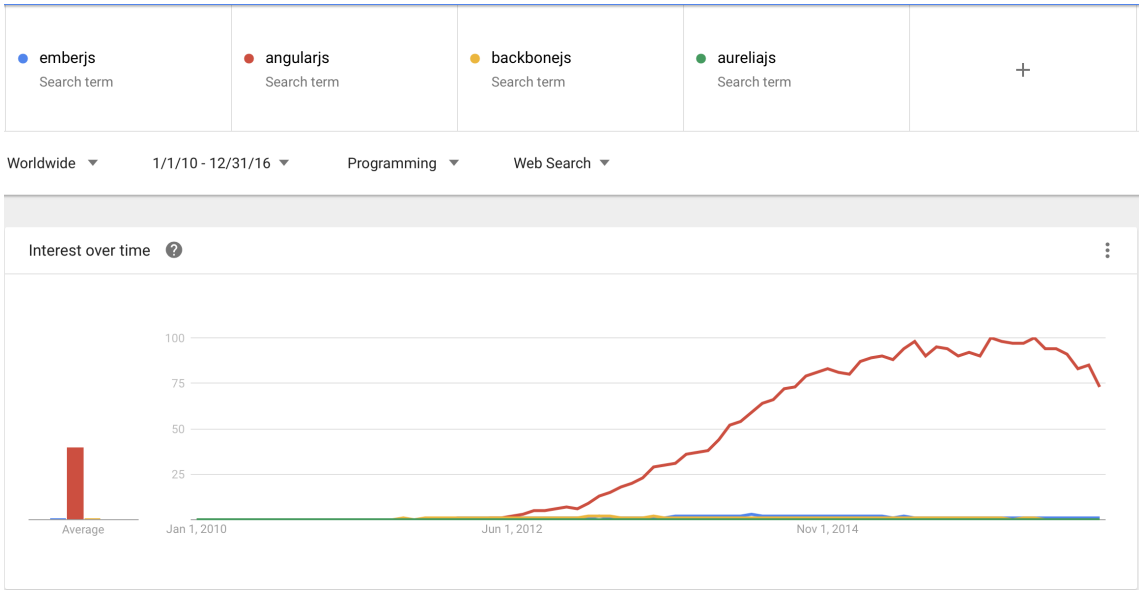
Velká komunita znamená více zodpovězených otázek, více modulů třetích stran, více Youtube tutoriálů, lepší otestovanost a další pozitiva. Proto je komunita je jedním z důležitých faktorů ke zvážení při výběru frameworku.

Metrika	Angular 2	Aurelia	Backbone	Ember
Hvězdy na Githubu	20 tisíc	9 tisíc	26 tisíc	17 tisíc
StackOverflow otázek	33 tisíc	2 tisíce	20 tisíc	20 tisíc
GitHub přistěvatelů	395	77	292	638
Otevřených issue	945	31	41	213
Vyřešených issue	7779	478	2264	4864
Pull requestů (celkem)	5250	391	1802	6190

Tab. 2.1: Číselné porovnání komunit

Tabulka 2.1 znázorňuje porovnání čísel z Ledna 2017. Data jsou získána z projektových stránek stránek publikovaných, Github.com, otázek položených na StackOverflow.com.

Zajímavým faktorem pro volbu můžou také být Google trends. Ty nabízí zobrazení hledanosti jednotlivých výrazů. Tedy nejrychleji rostoucí či umírající frameworky. Více v obrázku 2.1 níže.



Obr. 2.1: Google trends 2010-2016

2.3 Velikost frameworku

Čas načtení stránky je kritický pro úspěch aplikace. Koncoví uživatele nejsou nejtrpělivější publikum jakmile dojde na rychlost webové aplikace. Tedy je v zájmu všech věnovat pozornost rychlosti načtení.

Velikost frameworku a jeho čas inicializace, jsou dva faktory ke zvážení. V obou případech může aplikace ztratit cenné milisekundy. Framework o velikosti více než 500 kb už musí nabídnou něco extra aby vůbec stál za zvážení. V ideálním případě se mu pak projekt vyhne.

JavaScript bývá za tímto cílem vždy minifikovaný a komprimovaný pomocí gzip. Porovnány jsou tedy verze minifikované a gzipované frameworky. Velikost frameworku však může být zavádějící. K jeho chodu jsou totiž často vyžadovány přídavné moduly jako v případě Backbone kde je to Underscope(5 kb) a jQuery(32 kb) nebo Zepto (9,1 kb).

Metrika	Angular 2	Aurelia	Backbone	Ember
Velikost	766 kb	350 kb	7.6 kb	130 kb

Tab. 2.2: Velikost frameworků

2.4 Šablonovací systém

Jelikož se jedná o část kterou koncový uživatel uvidí, jde o důležitou součást. Vývojáři či kodérovi by měl usnadňovat reprezentaci dat v HTML.

Angular, Aurelia i Ember zahrnují některý z šablonovacích systémů. Vyjímkou je Backbone. Ten nenabízí žádný in-box systém. Nicméně jako výchozí nabízený je uveden Underscore. [2]

2.4.1 Angular 2

Jeho syntaxe je velice jednoduchá a rychle čitelná pro každého kdo zná HTML.

Proměnné

Pro vypsání hodnoty proměnných se využívá dvou složených závorek. Tedy `{{framework.name}}`.

Cykly

Nezbytnou částí programátorova života jsou podmínky a cykly. Možností zápisu je hned několik. Nejzajímavější je však zápis s hvězdičkou. Ten z ukázky 2.1. Důvodem pro využít tohoto způsobu je lepšího rozeznání direktiv upravujících strukturu HTML.

```
1 <ul>
2   <li *ngFor="let framework of frameworks">
```

```
3       {{framework.name}}
4     </li>
5   </ul>
```

Ukázka 2.1: Cyklus nad jednoduchým polem v frameworku Angular 2

Databinding

Důležitou součástí šablonovacího systému Angularu je také databinding. Jeho úkolem je zajistit zobrazení hodnoty proměnné pokud se změní stav aplikace v závislosti na čase nebo akci. Obousměrný binding pak zajišťuje také možnost načítat hodnotu proměnné například z inputu do aplikace a pak ji zpětně reprezentovat, jak je uvedeno v ukázce 2.2. Ukázka neobsahuje pouze šablonu. Angular vyžaduje v takovém případě kompletnější kód obsahující i komponentu.

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'hello-angular',
5    template: `Fill name: <input type=text [(ngModel)]="name"><br>
6    <p>Hello {{name}}!</p>`
7  })
8  export class HelloComponent {
9    name = "angular 2";
10 }
```

Ukázka 2.2: Obousměrný databinding v frameworku Angular 2

Komponentu z ukázky 2.2 je pak možno vykreslit pomocí kódu z ukázky 2.3.

```
1  ...
2  <body>
3    <hello-angular></hello-angular>
4  </body>
5  ...
```

Ukázka 2.3: Vykreslení komponenty v frameworku Angular 2

Události

Další podstatnou částí jednostránkové aplikace jsou události. Pomocí událostí aplikace reaguje na uživatelovy podměty. Například kliknutí, změnu hodnoty nebo jiné. Angular umožňuje zápis dvěma způsoby. První je pomocí závorek `(click)=`. Další možností je pak využití prefixu `on-`. Ukázka 2.4 obsahuje oba případy. Kliknutím na tlačítko se pustí akce `onClick()`.

```
<button (click)="onClick()">
<button on-click="onClick()">
```

Ukázka 2.4: Události v frameworku Angular 2

2.4.2 Aurelia

Aureliin zápis šablon je člověku znalému HTML bezproblémový. Avšak oproti Angularu 2 je hůře zapamatovatelný.

U šablony je vyžadováno aby byla zabalena značkou `<template></template>` a používala příponu HTML.

Proměnné

Proměnné se Aurelia zapisuje stejně jako Template Literals v ES6, pomocí složených závorek a dolaru před nimi. Příkladem může být zobrazení vlastnosti name z proměnné framework `${framework.name}`.

Cykly

V Aurelii jsou dostupné 2 typy podmínek sufixované výrazem .bind. Příkladem je `if.bind="isAvailable"`. První možností podmínky je tedy `if`. Ta při nesplnění vynechá zcela prvek z HTML. Druhou možností je `show`. Ta při vykreslování rozhodne zda má být prvek doplněn třídou `aurelia-hide`.

Třída `aurelia-hide` může být modifikována pomocí vlastního CSS. Výchozí vlastnost je pak `display: none;`.

Cykly Aurelia nabízí typu iterace nad polem, rozsah, iteraci objektu Set, klíč a hodnota (map) či Objektu obecně. V ukázce 2.5 je znázorněna iterace nad jednoduchým polem.

K iteracím je přidáván prefix `repeat` s tečkou.

```
<template>
  <ul>
    <li repeat.for="framework of frameworks">
      ${framework.name}
    </li>
  </ul>
</template>
```

Ukázka 2.5: Cyklus v frameworku Aurelia

Databinding

Databinding se Aurelii specifikuje sufixem `bind` s tečkou. Navazovat pak lze veškeré proměnné přidružené komponenty. V ukázce 2.6 je znázorněn příklad zdravící komponenty reagující na vstup uživatele.

```
<template>
  Fill name: <input type=text value.bind="name"><br>
  <p>Hello ${name}!</p>
</template>
```

Ukázka 2.6: Šablona využívající databinding v frameworku Aurelia

Na rozdíl od Angularu 2 (ukázka 2.2), Aurelia nezapisuje komponentu (ukázka 2.7) a šablonu do jednoho souboru. Je tedy třeba ještě vytvořit jeden další JavaScriptový soubor obsahující proměnné využívané v šabloně.

```
export class Greeter {
  constructor() {
    this.name = 'Aurelia';
  }
}
```

Ukázka 2.7: Komponenta v frameworku Aurelia

Události

Aurelia zapisuje události podobným způsobem jako binduje data. K provázání je využito sufixu `bind` s tečkou. Například událost kliknutí je pak demonstrována ukázkou 2.8.

```
<button click.bind="onClick">
```

Ukázka 2.8: Událost kliknutí v frameworku Aurelia

2.4.3 Backbone

Jako jediný framework z výběru, Backbone nenabízí přiložený šablonovací systém v balíčku. Lze ho velice jednoduše integrovat s různými knihovnami třetích stran. Na oficiálních stránkách je však doporučeno použít Underscore.

V Undescore se vše zapisuje v podobě JavaScriptu zapsaného mezi `<% ... %>`. Konkrétní funkce se pak volají nad podtržítkem. Underscore je stejně jako Backbone minimalistický nástroj.

Undescore není přímou součástí Backbone. Nebude mu tedy věnována taková pozornost.

Proměnné

Undescore vypisuje proměnné automaticky uvedením mezi šipku, procento a rovnítko `<%= ... %>`. Pokud je žádoucí výpis HTML, použije se místo rovnítka (=) pomlčka (-).

Cyklus uvedený v příkladu 2.9 vypisuje frameworky do seznamu za využití Undescore. Je zjevné, že se nejedná o rozsáhlou knihovnu, ale o lehkou nadstavbu nad HTML.

```
1 <ul>
2   <% _.each(frameworks, function(framework) { %>
3     <li>
4       <%= framework.name %>
5     </li>
6   <% }); %>
7 </ul>
```

Ukázka 2.9: Cyklus v frameworku Underscore

2.4.4 Ember

Šablonovací systém frameworku Ember je Handlebars. Ten je vybudován nad populárním Mustache.

Jako přípona souboru je použita zkratka `.hbs`.

Proměnné

Pro zobrazení obsahu proměnných se využívá složených závorek `{{framework.name}}`.

Cykly

Iterování v Handlebars je zapisováno trochu výřečnějším způsobem. Před příkaz se zapisuje hashtag a proměnná, která je výstupem každého průchodu se uvozuje svislou čarou. Zápis je patrný z ukázky 2.10.

```
1 <ul>
2   {{#each frameworks as |framework|}}
3     <li>
4       {{framework.name}}
5     </li>
6   {{/each}}
7 </ul>
```

Ukázka 2.10: Cyklus v frameworku Handlebars

Handlebars podporuje velice zajímavou možnost zobrazení prázdného pole. Místo podmínky jako takové se uvede značka `{{else}}`.

Další zajímavou vlastností je automatické rozbalování proměnných. Pokud není v cyklu uveden název proměnné ve svislých čárách. Obsah je automaticky rozbalen na vnitřní proměnné (ukázka 2.11).

```
<ul>
  {{#each frameworks}}
    <li>
      {{name}}
    </li>
  {{else}}
    <li>Ember</li>
  {{/each}}
</ul>
```

Ukázka 2.11: Cyklus for s else větví v frameworku Handlebars

Komponenta

Ember stejně jako Aurelia (ukázky 2.6 a 2.7) rozděluje komponenty na dvě části kódu. První je funkčnost kontrolu (ukázka 2.13). Druhou částí je šablona samotná (ukázka 2.12). To je velice užitečné pro vývojáře. Ten má v takovém případě oddělenou logiku od šablony.

```
Fill name: <input type='text' value=name><br>
<p>Hello {{name}}!</p>
```

Ukázka 2.12: Šablona komponenty v frameworku Handlebars

```
import Ember from 'ember';

App = Ember.Application.create();
App.ApplicationController = Ember.Controller.extend({
  name:'Ember',
});
```

Ukázka 2.13: Databinding v frameworku Ember

Události

Ember události řeší pomocí helperu `action` s parametrem názvu akce. Dále nabízí možnost specifikovat při jaké události myši nebo klávesnici se událost spustí. Jednou z dalších možností je přímo v šabloně definovat zda se má nad prvkem spustit výchozí akce prohlížeče pomocí `preventDefault=true/false`.

```
<button {{action "click" on="click"}}>
```

Ukázka 2.14: Události v frameworku Ember

2.5 Angular 2

Angular obsahuje mnoho inovativních myšlenek. To je kritické pro framework ve světe vývoje internetových stránek.

Jelikož jednou z hlavních vlastností programátora je lenost, moderní frameworky či knihovny musí poskytovat co nejkratší a nejjednodušší zápisy. Pro porovnání například s jQuery (ukázka 2.15) kde textový vstup je propisován do těla HTML stránky.

```
$('#form input#name').on('value', function() {
    $('#form div').text('Hello ' + this.val() + '!');
});
```

Ukázka 2.15: Živě propisovaný textový vstup v jQuery

Stejnou funkčnost lze v Angularu 2 zapsat velice krátkým způsobem (ukázka 2.16).

```
<input [(ngModel)]="name" type="text" />
Hello {{name}}!
```

Ukázka 2.16: Živě propisovaný textový vstup v frameworku Angular 2

2.5.1 Pozitiva

Angular se může pyšnit jednou z největších komunit kolem webového frameworku. Je to znát i tím, že patronem Angularu je Google.

Výkon

Angular 2 oproti Angularu 1 velice zapracoval na výkonu. Oproti první verzi se jedná o nesrovnatelný skok.

CLI

Angular 2 nabízí vlastní CLI nástroj. Pomocí příkazu `ng` je pak vývojář schopný velice rychle zakládat nové aplikace, zakládat komponenty, spouštět vývojářský server atd..

Výhodou vývojářského serveru je to, že není třeba instalovat na každý stroj složité nástroje jako je Apache a nastavovat virtualhost. Vše je vyřešeno jednoduchým příkazem `ng serve`.

Aplikace je spuštěna a ihned dostupná na adrese místního pod nakonfigurovaným portem. Vývojářský server také zajišťuje automatickou obnovu prohlížeče při změně ve zdrojových kódech (hot-replace).

IDE

Důležitou součástí frameworku je podpora Integrated development environment (IDE). Pokud framework nabízí podporu IDE, vývojáři je nabídnuto velikého komfortu.

Základem podpory IDE jsou útržky kódu (snippet), napovídání (autocomplete) a statická analýza kódu.

Další výhodou Angularu 2 je využití TypeScriptu. Ten je široce podporován editory a nabízí tak i bez speciálního modulu vyšší komfort vývoje.

TypeScript

JavaScript a TypeScript jsou dvě nabízené možnosti zápisu zdrojových kódů v Angularu 2. Tím preferovanějším je TypeScript. Ten je také nabízen jako výchozí možnost v oficiální dokumentaci.

Základní vlastností TypeScriptu je transpilace do JavaScriptu. Výstupem je tedy vždy JavaScript. TypeScript však na rozdíl od JavaScriptu nabízí některé rozšířené možnosti a prvky objektového návrhu.

Dalším zajímavým prvkem Typescriptu pak je typová kontrola. Ta poskytuje vývojáři statickou kontrolu a případné predikce chyb ještě před samotným spuštěním. V kombinaci s kvalitním IDE se může jednat velice cennou úsporu času.

Testování

Angular 2 je od začátku navržen k jednoduchému testování. Pro jednotkové (unit) testy Angular preferuje nástroj Karma. Karma byla dříve známá pod jménem Testacular. Jedná se spouštěč testů s velice širokou podporou ve vývojářských prostředích.

Druhým nástrojem je Protractor. Protractor je využívám pro aplikační testy. Testy jež jsou psány a spouštěny nad aplikací jako by se jednalo o reálného uživatele používajícího aplikaci.

Animace

Jedním z integrovaných modulů jsou animace. Nabízeno je rozhraní pomocí kterého lze zapsat animace podporující UX změn stavů aplikace.

Příkladem může být tlačítko aktivní / neaktivní měnící stav uživatele. Při kliknutí je tlačítku změněna třída a Angular 2 se postará o zbytek. Uživatel je veden pomocí grafické animace změnou stavu.

```
animations: [
  trigger('state', [
    state('inactive', style({
      backgroundColor: '#eee',
      transform: 'scale(1)'
    })),
    state('active',   style({
      backgroundColor: '#cfd8dc',
      transform: 'scale(1.1)'
    })),
    transition('inactive => active', animate('100ms ease-in')),
    transition('active => inactive', animate('100ms ease-out'))
  ])
]
```

Ukázka 2.17: Animace tlačítka v frameworku Angular 2

2.5.2 Negativa

Jedním z negativ je velikost frameworku. Velikost 766 kb (tabulka 2.2) je opravdu nezanedbatelné číslo.

Šablonovací systém

HTML Angularu 2 není HTML. HTML značky nejsou citlivé na velikosti písmen, ale Angular 2 značky jsou. Může se stát, že vývojář direktivu `ngIf=` napíše jako `ngif=` a pak stráví hledáním chyby cenný čas.

Další nevýhodou mohou být různé zápisy direktiv. Jednou je použita hvězdička, po druhé hranaté závorky a nakonec se může stát, že se vše zapíše do kulatých závorek a poté obalí hranatými. To definitivně všechny nováčky mate.

Také pokud je `*ngIf` použit u značky `<template>`. Jedná o direktivu ovlivňující výsledné HTML. Bude vygenerované HTML vypadat takto `<template [ngIf="true"]><templat`

Vývoj frameworku

Angular 2 si prošel bouřlivým vývojem. V době kdy se již mělo jednat o Release Candidate (RC) byly stále prováděny Breaking-change (BC).

2.6 Aurelia

...

2.6.1 Pozitiva

...

2.6.2 Negativa

...

2.7 Backbone

Backbone je tím nejtenčím řešením z vybraných frameworků. Jeho stopa ve výkonu aplikace je tím pádem mizivá.

2.7.1 Pozitiva

Křivka učení je přímá. Backbone díky své jednoduchosti nabízí rychlou implementaci aplikace pomocí konstrukcí které nabízí.

Dokumentace

Backbone má velice pěknou a jednoduchou dokumentaci. Díky tomu je čtení dokumentace rychlé a srozumitelné.

Dokumentace také disponuje spoustou příkladů a hotových řešení. V neposlední řadě pak ukázku projektů které Backbone používají.

Celá dokumentace je napsána jako jediná webová stránka.

Jednoduchost

Jelikož je Backbone tak malý a jednoduchý, nabízí velice mocný nástroj pro tvorbu aplikace. Výsledný nástroj totiž ve výsledku tvořen několika vývojářem vybraných knihoven a zásuvných modulů. Výsledný framework si vývojář vytvoří sám nad Backbone. Dostává velikou svobodu volby pro šablonovací systém, router a další. U Angularu 2, Aurelie nebo Emberu je třeba žít s tím co je již v frameworku zahrnuto.

2.7.2 Negativa

Backbone neposkytuje strukturu. Jedná se spíše o základní nástroj. Všechny složitější problémy jsou na vývojáři.

Jednoduchost

Stejně jako je jednoduchost výhodou Backbone je i jeho nevýhodou. Vývojář totiž musí projít spoustou balíčků a zásuvných modulů třetích stran aby si vybral. Rozhodování který balíček je nejvhodnější pro konkrétní případ může trvat dlouho.

Backbone podporuje databiding na velice základní úrovní. Pro změnu zobrazení v bude tedy třeba naprogramovat spoustu kódu.

Testování

Jednotkové testy je komplikované psát, protože s DOM dokumentem je v Backbone manipulováno přímo. Stejně tak je přímou manipulací ovlivněna znovupoužitelnost.

2.8 Ember

Emberu využívá přísné struktury aplikace. Nutí vývojáře držet se jeho konvencí a postupů. Pak je každá aplikace psaná v Emberu velice podobná a programátor se v ní rychle orientuje.

2.8.1 Pozitiva

V Emberu je zahrnut skvělý router stejně tak jako datová vrstva, zvaná ember data.

Konvence

Konvence jsou upřednostňovány nad konfigurací. To je užitečný koncept nabízející spoustu automaticky generovaného kódu, který by jinak vývojář muset napsat ručně (nevýhoda Backbone).

Příkladem může být Router. Ember je schopný automaticky generovat jméno routy pokud jej vývojář sám nedefinuje.

Datová vrstva

Ember obsahuje plně vyvinutou datovou vrstvu na rozdíl od Angularu a Backbone. Datová vrstva se integruje jednoduše oproti jakémukoliv JSON API. Požadavek na API je pouze aby následovalo několik zásadních konvencí. Ember už pak zajišťuje vše ostatní.

Datová vrstva také nabízí možnost testování. Je připravena na testování jednoduchých objektů i jejich vztahů.

Výkon

Životní cyklus Ember aplikace je nazýván "run loop". Jeho užitečnost spočívá v tom jak řídí běh aplikace. Postupně jednu po druhé spouští části životního cyklu. Jedním z jeho benefitů je pak možnost vůbec nepřekreslovat znova šablonu pokud to není nutné (nezměnila se data).

Ember také profituje z předkompilovaných šablon Handlebars. Ty jsou předem připraveny pro rychlé využití při vykreslování.

2.8.2 Negativa

I přestože Ember je velice povedený framework. Má několik nevýhod a jeho popularita není nejvyšší.

Komunita

Komunita Ember není tak rozsáhlá jako například u Angularu. To může vést ke komplikacím s řešený případných problémů vývojáře.

Framework je také velice otevřen příspěvkům od nezávislých vývojářů. To může vést k zavádění chyb nebo nekonzistencí do jádra.

Nestálost

API framewoku se často mění. Nejen ve velkých ale i v drobných verzích. Tím vzniká na StackOverflow velké množství nevalidního obsahu.

Příkladem může být vychvalovaný datový model. V jeho vývoji je tolik BC, že se StackOverflow může stát velice zavádějící pomocí.

Šablonovací systém

Hundlebars generuje mnoho značek `<script>` do HTML. Navíc šablonovací systém může díky této vlastnosti snadno rozbít kompatibilitu s ostatními frameworky nebo knihovnami. Příkladem může být jQuery.

3 ANALÝZA A NÁVRH APLIKACE

3.1 CRM systémy

crm! (**crm!**)

3.2 Databáze

3.3 Uživatelské rozhraní

3.3.1 Uživatelské role

dělat nadpisy ???

Klient

Administrátor

Vývojář

3.3.2 Projekty

3.3.3 Úkoly

3.3.4 Vývojářská perspektiva ?

...

4 POPIS IMPLEMENTACE

4.1 Prostředí pro vývoj

4.1.1 Webpack

4.1.2 Nodemon ??

4.1.3 Gulp

4.2 API

4.2.1 Framework

4.2.2 Knihovny

4.2.3 Struktura

4.3 Aplikace

4.3.1 Framework

4.3.2 Knihovny

4.3.3 Struktura

...

5 VÝSLEDKY A ZÁVĚR

Shrnutí studentské práce.

LITERATURA

[1] Yarn. Dostupné z URL: https://yarnpkg.com/, 18. Leden 2017.

[2] Backbone community. backbone.js [online]. Dostupné z URL: http://backbonejs.org, 15. Leden 2017.

[3] Grunt community. Sample gruntfile. http://gruntjs.com/sample-gruntfile, 2017.

[4] Gulp community. gulpjs.com. http://gulpjs.com, 2017.

[5] Io.js comunity. Roudmap io.js. http://roadmap.iojs.org, Leden 2017.

[6] David Flanagan. *JavaScript: The Definitive Guide*. O'Reilly Media, Inc., 2006.

[7] Node.js foundation. Node.js. https://nodejs.org/en/, Říjen 2016.

[8] Facebook Inc. React native. https://facebook.github.io/react-native/, 2017.

[9] Github Inc. Electron. http://electron.atom.io/, Leden 2017.

[10] Ecma International. Ecma-262. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf, Leden 2016.

[11] Pedro Teixeira. *Hands-on Node.js*. Leanpub, 2012.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

UX	user experience
CLI	client line interface
CSS	Cascading Style Sheets
SASS	Syntactically Awesome Style Sheets
ES5	ECMAScript 5th edition
ES6	ECMAScript 6th edition
ES2015	ECMAScript 2015
ES2016	ECMAScript 2016
ES2017	ECMAScript 2017
MVC	Model View Controller
MVP	Model View Presenter
DI	dependency injection
IDE	Integrated development environment
RC	Release Candidate
BC	Breaking-change

SEZNAM PŘÍLOH

A PŘÍLOHA

ZDE VLOŽIT LIST ZADÁNÍ