Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza jazyku Dart

The Analysis of Dart Language

2015 **Kamil Niemczyk**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Kamil Niemczyk**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Analýza jazyka Dart
The Analysis of Dart Language

Zásady pro vypracování:

Cílem bakalářské práce je zhodnotit možnosti skriptovacího jazyka Dart, a porovnat jeho vlastnosti s již rozšířenými jazyky pro tvorbu webových aplikací. Student analyzuje programovací jazyk Dart a představí další webové programovací jazyky. Provede podrobné srovnání jazyka Dart s jazykem JavaScript. V rámci praktické části student vytvoří vzorovou funkční aplikaci pro reprezentaci možností jazyka a provede jejich výkonnostní testování vzhledem k vybranému jazyku.

1. Představení skriptovacích jazyků pro tvorbu webu.
2. Podrobná analýza vlastností a možností jazyka Dart.
3. Syntaktické srovnání s jazykem JavaScript.
4. Přehled dostupných framevorků pro jazyk Dart a jejich funkční srovnání.
5. Srovnání výkonu vzorové aplikace napsané v jazyce Dart a JavaScript.

Seznam doporučené odborné literatury:

[1] Kathy Walrath, Seth Ladd, Dart: Up and Running, O'Reilly Media, 2012, ISBN: 978-1-4493-3089-7
[2] Glenford J. Myers, The Art of Software Testing, 2. vydání, John Wiley & Sons, New Jersey, 2004, ISBN: 0-471-46912-2
[3] Další literatura podle pokynů vedoucího práce

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Břetislav Paláček**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

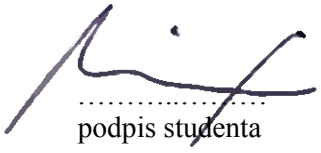
doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *29. července 2015*



……..……
podpis studenta

Poděkování

Rád bych poděkoval Ing. Břetislavu Paláčkovi za vedení mé práce, jeho připomínky a rady. Děkuji také svým přátelům a rodině za poskytnutí výborných podmínek pro studium.

Abstrakt

Práce porovnává programovací jazyky JavaScript a Dart a jejich vhodnost pro tvorbu webových aplikací. Podrobně rozebírá výhody a nevýhody obou jazyků.

V práci jsou ukázané některé obvyklé frameworky používané při tvorbě webových aplikací. Důraz je kladen na seznámení s frameworky používanými v jazyce Dart.

Na jednoduchém příkladu porovnává rychlost aplikace napsané v jazyce Darta a v jazyce JavaScript. Rychlost je porovnávána v různých webových prohlížečích.

Cílem práce je představit jazyk Dart, porovnat jej s jazykem JavaScript a poukázat na jeho výhody při tvorbě webových aplikací.

Klíčová slova

Dart; JavaScript; Transformovaná matice; Skriptovací jazyk

Abstract

My work compares JavaScript and Dart programming languages and their suitability for building some Web applications. It analyzes in detail the advantages and disadvantages of both languages.

The work shows some common frameworks used for creating web applications. An emphasis is placed on the introduction of frameworks used in the Dart language.

A simple example compares the speed of the application written in Dart and JavaScript langugages. Their speed is compared in different web browsers.

The aim of the work is to introduce the Dart language, compare it with JavaScript and point out its advantages in creating Web applications.

Key words

Dart, JavaScript, The transformed matrix, Scripting language

Seznam použitých zkratek

Zkratka	Význam
API	Application Programming Interface
CD	Compact Disc
CERN	Conseil Européen pour la recherche nucléaire
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
DDR	Double-Data-Rate
DOM	Document Object Model
ECMA	European Computer Manufacturers Association
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model-View-Controller
OOP	Object-oriented programming
PHP	PHP: Hypertext Preprocessor
VM	Virtual Machine

Obsah

Úvod

Je tomu již několik let, kdy se neodmyslitelnou součástí každodenního života většiny lidí stal internet. K jeho původnímu účelu - sdílení vědeckých informací jej však dnes již používá pouze nepatrná část uživatelů. Jeho zakladatelé jistě neměli ani tušení o tom, že se jednou bude internet používat například k nakupování prakticky veškerého zboží a služeb, provádění bankovních plateb, prodeji reklamy, prezentaci firem, sdílení s přáteli, seznamování, sledování aktuálního počasí, kontrolování letu všech civilních letadel na celém světě, elektronické komunikaci mezi úřady, ale třeba i k hraní her. Připadá nám jako naprostá samozřejmost, že když dnes řešíme téměř jakýkoliv problém, tak se nejprve jdeme "podívat" na internet.

Samozřejmě, že s téměř netušeným obrovským rozvojem možností, které internet nabízí, šel stejným tempem i vývoj nástrojů, pomocí kterých se internet vytváří. Pojďme si nyní prohlédnout historii a současnost nástrojů pro tvorbu webu. Těch nástrojů, které nám umožňují se ráno podívat na internetu na to, jaké bude venku počasí, kolik bude stupňů tepla a zda bude pršet, v poledne si pomocí nich objednáme oběd a večer zaplatíme bankovní účty.

Tato práce vysvětluje historický kontext tvorby webových aplikací a porovnává mezi sebou dva programovací jazyky: tradiční jazyk JavaScript a novější jazyk Dart.

V první kapitole je vysvětlené historické pozadí tvorby webových stránek a aplikací. Jsou ukázané používané tradiční postupy při vývoji aplikací a je zde zdůvodněná potřeba moderního jazyka pro tvorbu pokročilých aplikací.

Druhá kapitola podrobně rozebírá výhody a nevýhody jazyka JavaScript. Cílem kapitoly je ukázat na obtíže spojené s JavaScriptem při vytváření větších projektů a poukázat na potřebu přehlednějšího jazyka.

V třetí kapitole je představen jazyk Dart - jeho základní vlastnosti, rozdíl mezi tradičním jazykem JavaScript. Dále je představeno prostředí jazyka Dart - integrované vývojové prostředí a podpora jazyka Dart v různých prohlížečích.

Ve čtvré kapitole jsou podrobně porovnané jazyky Dart a Javascript z hlediska jejich syntaxe.

Pátá kapitola popisuje několik základních frameworků pro tvorbu pokročilejších aplikací v jazyce Dart a jejich porovnání s frameworkem jQuery a jQueryUi.

Šestá kapitola pak na konkrétní ukázkové aplikaci porovnává výkon jazyka Dart a jazyka JavaScript v prostředí různých webových prohlížečů.

1 Představení skriptovacích jazyků pro tvorbu webu

1.1 Historické pozadí vývoje skriptovacích jazyků

Kořeny webu, jak jej známe v dnešní podobě, sahají do jaderných laboratoří CERN do roku 1989. Jeho původní úlohou bylo sdílení informací ve vědecké obci mezi univerzitami a vědeckými institucemi [1]. Nejvýznamnějším pozůstatkem je jazyk HTML a myšlenka, že dokumenty mohou být mezi sebou volně provázané v rámci celé sítě.

1.2 Statické webové stránky

Původní představa webových stránek spočívala v prostém sdílení dokumentů. Dokumenty sice mezi sebou byly provázané - bylo možné interaktivně přímo v textu dokumentu pouhým kliknutím na odkaz načíst a zobrazit jinou stránku dokumentu či rovnou jiný dokument. Pro potřeby sdílení vědeckých informací to stačilo.

1.3 Dynamické webové stránky

Statické webové stránky přestaly brzy vyhovovat. Jedním z nejstarších serverů na českém internetu je Seznam.cz. Původně šlo skutečně o prostý seznam webových stránek v češtině - ale představme si obtížnost takového úkolu. Dokud bylo v seznamu několik stovek stránek, bylo možné editovat seznam ručně. Jakmile se však seznam rozrostl na několik tisíc či několik desítek tisíc záznamů, bylo nutné přestěhovat záznamy do specializovaného úložiště (databáze) a stránky vygenerovat automaticky.

Tvorba datových stránek se rozděluje - součástí serveru je typicky datový zdroj (databáze - MySQL, PostgreSQL, CouchDB a spousta dalších) a programovací jazyk, kterým se data z datového zdroje formátují do podoby webových stránek. Tyto webové stránky se pak servírují přes síť webovému prohlížeči, který má za úkol stránky zobrazit.

V prvopočátcích rozvoje webových stránek se pro formátování dat z datového zdroje používaly typicky CGI skripty.

Každý CGI skript byl samostatný spustitelný soubor. Problémem je, že takový CGI skript je závislý na platformě - nelze jednoduše přenést skript z jednoho počítače na jiný. Drtivá většina webových aplikací byla v minulosti navíc provozována na různých Unixových a Linuxových počítačích, kde byla obrovská roztříštěnost možností konkrétních operačních systémů.

Pro tvorbu CGI skriptů se používaly tehdy dostupné prostředky a jazyky - shell, Awk, C, C++ a jistě i mnohé další. Obrovskou nevýhodou těchto jazyků je, že žádný z těchto jazyků není od začátku koncipovaný přímo pro tvorbu webu. A právě z tohoto důvodu programování rozsáhlejších webových aplikací, jako je CGI skript musí být velmi pracné a neefektivní.

Proto se začínají objevovat zhruba od poloviny devadesátých let specializované jazyky určené přímo pro tvorbu webu - asi nejznámějším takovým jazykem je dnes PHP.

S vývojem sofistikovanějších skriptovacích jazyků bylo možné web rozšířit o dosud nevídané aplikace - internetové obchody, hry, firemní informační systémy, komunikační systémy, redakční systémy a mnoho dalších aplikací. Spolu s vývojem těchto aplikací a s jejich rostoucími požadavky se vyvíjejí i programovací jazyky. Například již zmiňované PHP se vyvíjí od roku 1994 až dodnes a přizpůsobuje se stavu poznání a zvyklostem v IT světě podle rostoucích požadavků programátorů a aplikací, které programátoři vyvíjejí.

Dynamické webové stránky v klasickém pojetí generují kompletní webovou stránku už na serveru a webový prohlížeč zde slouží jen jako pouhopouhý zobrazovač předpřipravených stránek. Od prohlížeče se v tomto modelu nepožadují žádné speciální schopnosti kromě schopnosti řádně zobrazit naservírovaný dokument.

Vývoj webu se v této etapě ubírá především cestou postupného rozšiřování a standardizaci jazyka HTML a CSS.

1.4 Zpracování dat na straně webového klienta

Jakmile se ve větší míře začaly vyvíjet složitější aplikace, začínalo být patrné, že představa webového prohlížeče jako hloupého zobrazovače nedostačuje.

Představme si modelovou situaci. Uživatel nakupuje v internetovém obchodě a do pole "telefonní číslo" vyplňuje svou mailovou adresu: "abcd1234xyz@gmail.com". Data formuláře putují zpět na server a mírně modifikovaný formulář s vyznačením chybně položky putuje zpět do prohlížeče uživatele. Takové řešení je neefektivní a především pomalé - je třeba si uvědomit, že ještě před pár lety byl internet pro většinu uživatelů přístupný pouze přes drahou a pomalou vytáčenou linku. Spolu s malou rychlostí internetového obchodu proto mizí i zákazníci.

Pro skriptování na straně prohlížeče tak existuje tlak jak z hlediska technologického, tak z hlediska ekonomického. Začala se hledat cesta, jak část zodpovědnosti za chování webové aplikace přesunout do webového prohlížeče - začíná se prosazovat jazyk JavaScript.

S postupem doby začaly být jasně zřejmé výhody zpracování dat na straně webového prohlížeče. Z původně dynamických stránek generovaných na straně serveru se postupně opět stávají částečně statické stránky - stránka se načte do prohlížeče pouze jednou a veškerá zobrazovaná data se načítají postupně, jak je potřeba.

Nároky na programovací jazyk jsou u takového přístupu mnohem vyšší, než u prosté kontroly údajů zadávaných ve formuláři. Začínají se objevovat omezení používaného jazyka JavaScript. Objevují se různé knihovny, které usnadňují práci s dokumentem zobrazeným v okně webového prohlížeče.

1.5 Webové aplikace

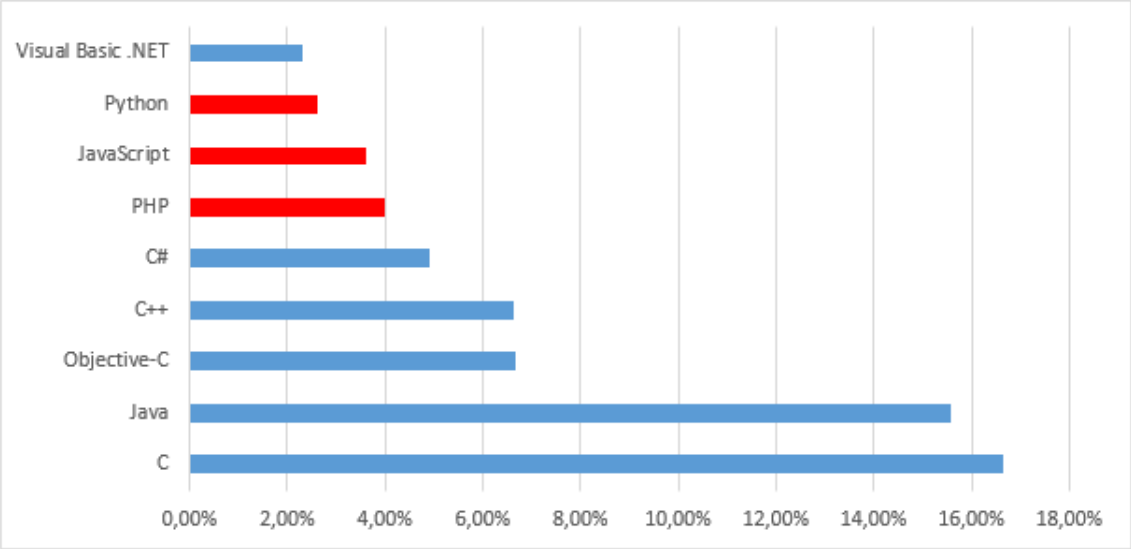
V posledních letech se ve velkém množství začínají objevovat aplikace, které byly dříve čistě doménou stolních počítačů - desktopů. Za typického představitele takové aplikace lze uvést tabulkový kalkulátor.

Tabulkový kalkulátor ve webovém prohlížeči můžeme směle prohlásit za zcela nový jev. Těžko bychom mohli označit tabulkový kalkulátor za obyčejnou webovou stránku.

Nový typ aplikací potřebuje nové přístupy. Původní JavaScript vzniknul jako jednoduchý prostředek pro manipulaci dokumentu ve webovém prohlížeči. Pro usnadnění práce vznikají různé frameworky, ale všechny nevýhody a omezení samotného jazyka zůstávají a komplikují tvorbu rozsáhlých projektů. Proto se objevují jazyky jako je TypeScript nebo Dart.

1.6 Běžně používané programovací jazyky

V následující kapitole si představíme dnes běžně používané programovací a skriptovací jazyky. Výzkum provedla společnost Tiobe Software [8], která hodnotí především množství jednotlivých požadavků na Google a další vyhledávače. Údaje byly zaznamenávány v období od března 2014 do března 2015. Pro odlišení jsou skriptovací jazyky znázorněné červenou barvou.



Obrázek 1.1: *Celosvětová oblíbenost programovacích jazyků*

1.7 PHP

Dle výzkumu společnosti Tiobe Software je dnes nejpoužívanější jazyk z rodiny jazyků skriptovacích, právě PHP. Na velké oblíbenosti tohoto skriptovacího jazyka má určitě velkou zásluhu relativně snadné pochopení, díky spoustě volně dostupných tutoriálů a také dostupnost hostingových služeb. Tento oblíbený jazyk provádí skripty na straně serveru, to znamená, že k uživateli je přenášen až výsledek jejich činnosti a skripty nemohou fungovat bez serveru.

1.7.1 Historie PHP

Počátky PHP spadají do roku 1994, tehdy tato zkratka znamenala Personal Home Page v dnešní době Hypertext Preprocessor. První verze jazyku byla napsána v PERLu, později bylo přepsáno do jazyka C kvůli vysoké vytíženosti serverů. Jazyk PHP se nejvíce prosadil v roce

2000, kdy byla uvolněna verze 4.0. obsahovala již komponenty jako je HTTP session, buffering výstupu a především podporu pro mnoho WWW serverů. Prozatím poslední verzí je PHP 5.6, která byla oficiálně spuštěna 28. srpna 2014. Dle oficiálních stránek PHP [1] by měla v říjnu roku 2015 vyjít PHP verze 7.0, která je dvakrát rychlejší, než verze 5.6 a spotřebuje o 30% méně paměti. PHP 7.0 bude podporovat anonymní třídy, bude možné zapsat návratovou hodnotu přímo do definice funkce a bude obsahovat další vychytávky.

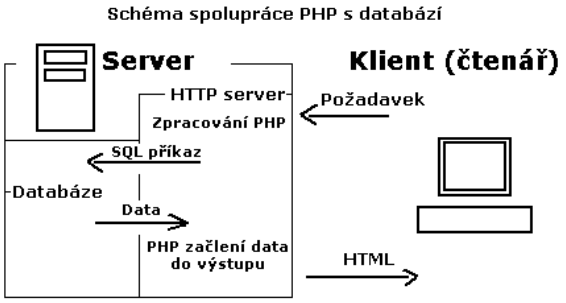
1.7.2 Výhody PHP

- Poměrně snadný na pochopení
- Syntaxe podobná jazyku C, tedy pro většinu vývojářů docela blízký
- Přes 5500 funkcí v základní knihovně PHP
- Multiplatformost
- Velké množství projektů a hotových kódů šířena pod některou ze svobodných licencí
- Velmi dobrá podpora na hostingových službách

1.7.3 Nevýhody PHP

- Jazyk je interpretovaný, při každém spuštění sebemenšího PHP skriptu musí server kód přeložit
- Nejednotné pořadí parametrů

1.7.4 Schéma komunikace jazyka PHP



Obrázek 1.2: *Schéma komunikace PHP*

1.7.5 Ukázka zdrojového kódu PHP

V následující ukázce můžete vidět syntaxi skriptovacího jazyka PHP. V ukázce se jedná o funkci pro výpočet faktoriálu.

```
function faktorial($cislo) {
   if ($cislo == 0)
      return 1;
   return $cislo * faktorial($cislo - 1);}
```

1.8 JavaScript

Dalším z rodiny skriptovacích jazyků je JavaScript. Tento jazyk je na rozdíl od jazyka PHP obvykle vykonávaný na straně klienta. JavaScript má některé nepříjemné vlastnosti, které komplikují tvorbu rozsáhlejších projektů. Proto vznikají jazyky, které se snaží odstranit nepříjemné vlastnosti jazyka JavaScript a vnést do programování prohlížečů řád potřebný pro tvorbu rozsáhlejších aplikací: TypeScript, Dart a další.

1.8.1 Historie jazyka JavaScript

V roce 1995 ve firmě Netscape vznikl jeden z dnes nejpopulárnějších skriptovacích jazyků, původně pojmenovaný Mocha, později LiveScript a nyní JavaScript, jehož název vznikl od programovacího jazyku Java, nicméně nesmíme tyto dva jazyky zaměňovat, jelikož kromě podobné syntaxe a části názvu nemají spolu nic společného. Dva roky po první zmínce o JavaScriptu byl tento jazyk standardizován asociací ECMA a našel své místo ve všech internetových prohlížečích, bohužel ne každý prohlížeč uznává všechny metody tohoto jazyka a díky tomu skript, který nám funguje v jednom prohlížeči, nemusí fungovat v jiném. V roce 2009 byl uveden serverový framework Node.js. Základem Node.js je JavaScriptový interpret V8 vyvíjený společností Google a několika standartních knihoven.

1.8.2 Výhody jazyka JavaScript

- běží na straně webového prohlížeče
- široká podpora v různých prohlížečích
- syntaxe podobná jazyku C
- je dostupné velké množství informací a zkušeností

1.8.3 Nevýhody jazyka JavaScript

- Nelze pracovat se soubory
- Nekompatibility mezi prohlížeči
- Roztříštěnost knihoven, neexistuje standardní knihovna
- Nízký výkon
- Není určený pro velké projekty

1.9 TypeScript

JavaScript získává čím dál více důležitosti na webu. Bohužel má dost nedostatků, na které vývojaří postupně narážejí. Z tohoto důvodu vznikají další programovací jazyky, které svůj kód do JavaScriptu překládají a tímto jsou jakýmsi rozšířením JavaScriptu. Jedním z těchto jazyků je programovací jazyk s otevřeným zdrojovým kódem Typescript, vytvořený a spravovaný společnosti Microsoft. Jedná se o nadstavbu jazyku JavaScript, tzn. skript vytvořený pomocí jazyku TypeScript je přeložen do jazyku JavaScriptu a tím pádem nemá žádný vlastní kompilátor. Tato chytrá nadstavba poskytuje typovou kontrolu, třídy, moduly rozhraní a podporu hlavičkových souborů.

1.9.1 Historie TypeScriptu

TypeScript se zrodil 1. října 2012 ve společnosti Microsoft. Byť je tento jazyk poměrně mladý, již nyní dosáhl velké popularity, především díky přidání možnosti statického typování a zanechání ostatních výhod JavaScriptu. TypeScript je stále ve fázi vývoje. Ve verzi 0.9 byly přidány generické datové typy. Ve verzi 1.3 byl přidán modifikátor přístupu protected. Verze 1.4 je obohacená o další funkce jako jsou například konstanty a přísnější generika. Poslední verzí je TypeScript verze 1.5.

1.9.2 Výhody TypeScriptu

- Podpora statických typů
- Syntaxe umožňuje pokročilé metody OOP
- Přehlednost i složitějších skriptů
- Každý JavaScript je validní s TypeScriptem

1.9.3 Nevýhody TypeScriptu

- Podpora není ve všech editorech
- Nutnost kompilace do JavaScriptu

2 Podrobná analýza vlastností a možností jazyka JavaScript

Abychom mohli srovnat jazyky JavaScript a Dart, je potřeba podrobně ozřejmit výhody a nevýhody současného stavu jazyka JavaScript.

2.1 Výhody jazyka JavaScript

Některé výhody jazyka JavaScript vycházejí z rozdílu zpracování na straně serveru a na straně webového prohlížeče - nezávisejí na použitém skriptovacím jazyku. Tyto výhody jsou proto zmíněny jen stručně. Podrobněji jsou pak rozebrané výhody proti jazyku Dart nebo ostatním jazykům používaným na straně prohlížeče.

2.1.1 Javascript je vestavěný v libovolném moderním prohlížeči

JavaScript je vestavěný prakticky v libovolném moderním prohlížeči, včetně kuriozit jako je třeba čistě textový webový prohlížeč Links (kvůli množství chyb byla podpora JavaScriptu z webového prohlížeče Links odstraněna, prohlížeč však lze nakonfigurovat a přeložit dodnes i s podporou jazyka JavaScript). I když se implementační detaily v jednotlivých prohlížečích liší, lze se spolehnout, že věci, které díky JavaScriptu fungují v jednom prohlížeči, bude možné prostředky JavaScriptu naprogramovat i v jiném prohlížeči.

2.1.2 JavaScript dovede snadno modifikovat stránku

Pomocí mechanismu DOM je dostupný pro programátora v JavaScriptu libovolný element na stránce. Elementy na stránce lze modifikovat, přidávat nové elementy či mazat existující elementy.

2.1.3 Jednoduchý a efektivní přenos dat: Json

JavaScript poskytuje jednoduchý a efektivní formát pro přenos dat mezi serverem a webovým prohlížečem: Json. Tento formát svými možnostmi připomíná jiný formát určené pro vyměnu a přenos dat: XML.

2.1.4 JavaScript má syntaxi podobnou jazyku C

Ze syntaxe jazyka C dnes vychází velké množství různých široce rozšířených a používaných programovacích jazyků (PHP, C#, C++, Java).

2.1.5 JavaScript je objektově orientovaný

Ačkoliv je syntaxe JavaScriptu při vytváření tříd poměrně neobvyklá, jde o plnokrevný objektově orientovaný jazyk.

2.1.6 JavaScript má dlouhou historii

JavaScript je součástí prostředí, ve kterém se weboví programátoři pohybují, už od roku 1995. Weboví programátoři jsou díky tomu zvyklí pracovat v JavaScriptu, existuje velké množství zkušeností, tutoriálů, příruček a dalších zdrojů informací o jazyce JavaScript.

2.1.7 JavaScript je velmi rozšířený

Jazyk JavaScript v nějaké jednodušší či složitější podobě používá pravděpodobně většina webových programátorů - jazyk je velmi rozšířený. Díky tomu je snazší na pracovním trhu získat programátora seznámeného s jazykem JavaScript.

2.1.8 JavaScript není omezený jen na webové prohlížeče

JavaScript je interpretovaný jazyk a jeho interpreter není omezený pouze na spolupráci s webovými prohlížeči. Interpreter jazyka JavaScript lze snadno připojit například k aplikacím vytvořeným v jazyce C++ a frameworku Qt - díky tomu lze i klasické desktopové či serverové aplikace vybavit možností vytvářet uživatelské skripty.

2.2 Nevýhody jazyka JavaScript

2.2.1 Nelze pracovat se soubory

JavaScript byl vytvářený pro potřeby webových prohlížečů - z bezpečnostních důvodů proto nedokáže pracovat se soubory. Jediným způsobem, jak uložit nějakou informaci pro pozdější použití, je využití mechanismů webového prohlížeče - konkrétně cookies.

2.2.2 JavaScript se liší prohlížeč od prohlížeče

Různé prohlížeče disponují různými verzemi interpretru jazyka JavaScript. Různé prohlížeče také jinak zpřístupňují dokument pomocí mechanismu DOM. Programátor je tak postavený před problém řešit jednu úlohu několika různými způsoby (typické je například jak zajistit, že JavaScript bude spuštěný až po načtení celého dokumentu).

2.2.3 JavaScript je dynamický typový jazyk

Dynamická typová kontrola probíhá až za běhu programu. JavaScript používá dynamické typování, tudíž nepožaduje specifikaci datového typu u proměnných a ty mohou odkazovat na hodnotu jakéhokoli typu. V případě programátorské chyby nás kompilátor neupozorní na chybu a tím se ladění programu stává náročnější.

2.2.4 Uživatel může JavaScript zakázat

Ačkoliv to není příliš obvyklé - většina uživatelů na nastavení webového prohlížeče nijak nesahá, uživatel může ve webovém prohlížeči jazyk JavaScript vypnout, případně omezit některé funkce (typicky otevírání nových oken a záložek). Programátor tak musí řešit každou situaci dvakrát - jednou pro prostředí s JavaScriptem, podruhé bez JavaScriptu.

2.2.5 Neexistuje žádná standardní knihovna

V jazyce JavaScript neexistuje žádná standardní knihovna, která by za programátora řešila obvyklé problémy, například: práce s datem, práce s DOMem, šifrování, kontejnery, datové struktury, různá kódování a podobně. Místo toho jsou zde tisíce knihoven řešících malou část problematiky. Pro řešení kteréhokoliv problému pak existuje několik různých různě kvalitních a různě komptatibilních knihoven.

2.2.6 Nízký výkon

I když například engine V8 společnosti Google přinesl o něco vyšší výkon, JavaScript je stále považovaný spíše za pomalejší jazyk a je obtížné v něm vytvořit výkonnou aplikaci.

2.2.7 JavaScript není určený pro velké projekty

V době vzniku jazyka JavaScript si nikdo nepředstavoval, že by v něm někdo mohl vyvinout například tabulkový kalkulátor. Jazyk JavaScript není na projekty takového rozsahu koncipovaný. Na jednu stranu je obtížné v něm rozdělit jednu úlohu mezi více vývojových týmů a zajistit datovou kompatibilitu mezi jednotlivými funkčními celky. Na druhou stranu je pak složité oddělit tyto celky (knihovny) tak, aby se vzájemně neovlivňovaly.

3 Podrobná analýza vlastností a možností jazyka Dart

3.1 Představení jazyka Dart

Dart není pouze programovacím jazykem, je to celá open-source platforma pro vývoj strukturovaných webových aplikací vyvinutá společností Google. Pro uživatelův komfort je tato platforma koncipována jako tzv. „batteries-included“, což znamená, že stažením Dart SDK uživatel získá všechny potřebné nástroje pro vývoj.

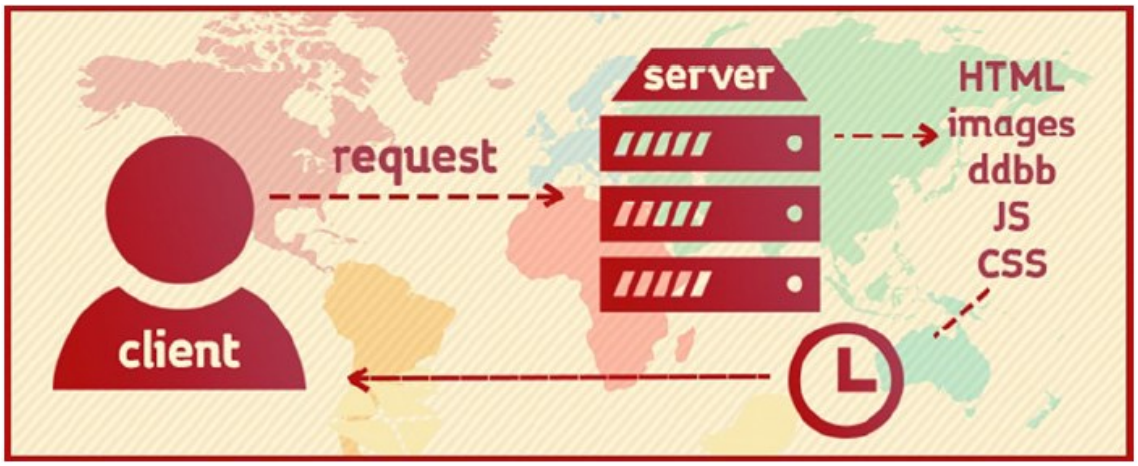
Za vznikem Dartu stojí Lars Bak a Kasper Lund. Cílem tohoto projektu bylo vytvořit platformu pro snadný vývoj komplexních webových aplikací, které se budou moci chlubit vysokým výkonem. Díky možnosti použití toho jazyka jak na straně klienta, tak na straně serveru, může vývojář získat homogenní prostředí pro vývoj aplikace. Jazyk Dart vznikl jako open-source projekt. Zaměřuje se především na tvorbu webových aplikací, ale v jazyce Dart můžeme také snadno vyvíjet konzolové aplikace.

Jazyk Dart by mohli ocenit především vývojáři v JavaScriptu, v kterém bývá psaní rozsáhlých webových aplikací velice vyčerpávající a vede k nepřehlednému kódu. Tím, jak je tento jazyk navržen, se snaží Dart překlenout propast mezi staticky typovanými jazyky jako je C# nebo Java a mezi dynamicky typovaným JavaScriptem. Jeho syntaxe patří do rodiny jazyků C/C++/Java/C# a z těchto jazyků si Dart také osvojil některé vlastnosti, díky čemuž je usnadněn přechod programátorů píšících své aplikace v těchto jazycích k jazyku Dart.

Nicméně, Dart na trh nepřichází s účelem nahradit, velmi rozšířený jazyk JavaScript, ale pouze jako další moderní volba pro vývoj webových aplikací s lepším výkonem, především pro opravdu velké projekty, kde je údržba kódu komplikovaná.

3.2 Dart pracuje na straně klienta

Za posledních pár let se programovací paradigma hodně změnilo. Jak jsem již v předchozí kapitole zmiňoval webové aplikace vyvinuté v programovacích jazycích, jako je PHP, Python nebo Ruby, měli na starost vše servery. Když uživatel navštívil webovou stránku, vždy server musel sestavit HTML dokument a načíst externí zdroje jako CSS, JavaScripty, obrázky a videa. Při každém databázovém dotazu musely tyto všechny procesy proběhnout znovu a to bylo pro servery velmi vyčerpávající. Typickým příkladem může být sociální síť, na které jsou v jeden okamžik tisíce uživatelů. Ať už si uživatel prohlíží fotky, prohlíží profily ostatních uživatelů nebo přidává status, pokaždé by musel server sestavit nový HTML dokument s externími zdroji. Tento způsob je jak náročný na servery, tak na síť. Princip, jak fungují webové aplikace vytvořeny tímto způsobem, můžete vidět na obrázku 3.1.



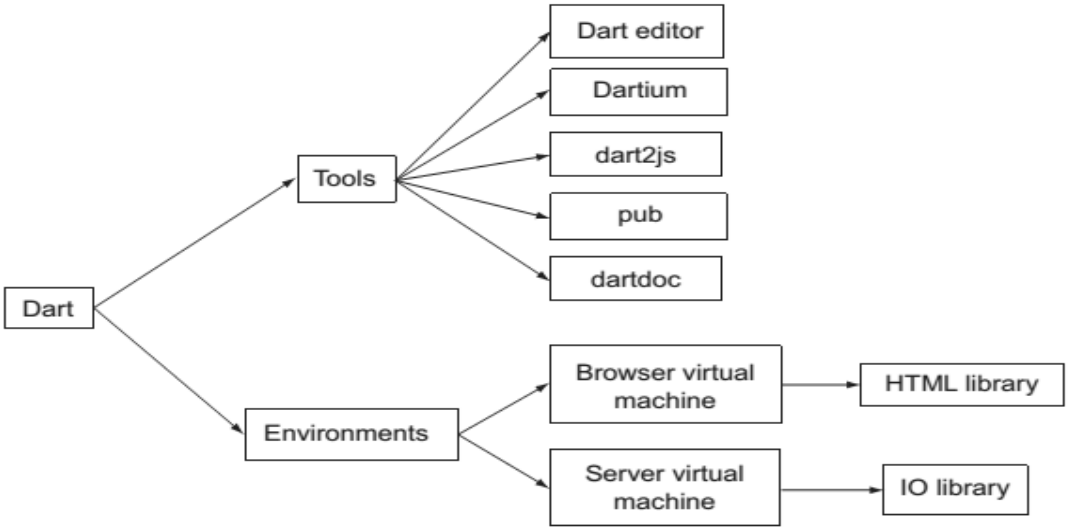
Obrázek 3.1: *Schéma standartního klient - server požadavku [4]*

Proto v dnešní době se vývojáři moderních programovacích jazyků jako je Dart, snaží využít procesor a paměť od klienta a tím převádějí procesy, kterými se zabýval server, na webového klienta. Při procházení webových stránek nebo přihlášení do svého profilu na sociální sítí server odešle do vašeho prohlížeče několik skriptů, které využívají váš webový prohlížeč, kde si vytváří místo pro uložení dat. Tímto způsobem jsou servery méně vytížené a zvládnou obsloužit mnohem více dotazů. Schéma komunikace je znázorněno na obrázku 3.2.

Obrázek 3.2: *Nové asynchronní schéma klient - server požadavku [4]*

3.3 Vývoj v Dartu

Jak již bylo zmíněno, Dart není pouze programovací jazyk, ale je to celá platforma (ekosystém) pro vývoj webových aplikací. Vývojář má k dispozici množství command-line nástrojů a knihoven obsažených v Dart SDK, které je volně ke stažení na oficiálních stránkách projektu. Dart SDK je dostupný pro platformy Windows, Linux a Mac OS X a nevyžaduje instalaci.



Obrázek 3.3: *Ekosystém Dartu*

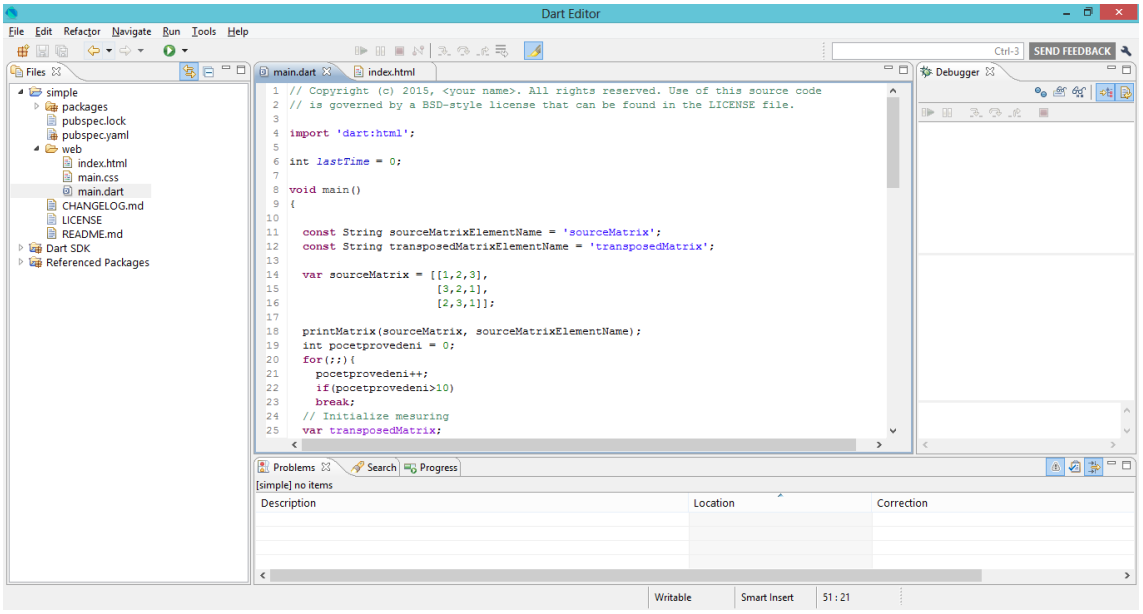
Výše uvedený obrázek 3.3, rozděluje ekosystém Dartu do dvou částí. První části jsou nástroje pro vývoj aplikací a druhou částí jsou běhová prostředí pro aplikace napsané v jazyce Dart a standardní knihovny.

V následujících několika odstavcích budou představeny nástroje pro vývoj aplikací v jazyce Dart. Tyto nástroje nalezneme v Dart SDK ve složce „bin“.

3.3.1 Vývojová prostředí

Pro psaní zdrojových kódů v jazyce Dart můžeme využít prostých textových editorů, avšak tyto editory nenabízejí uživateli komfort vývojových prostředí (IDE).

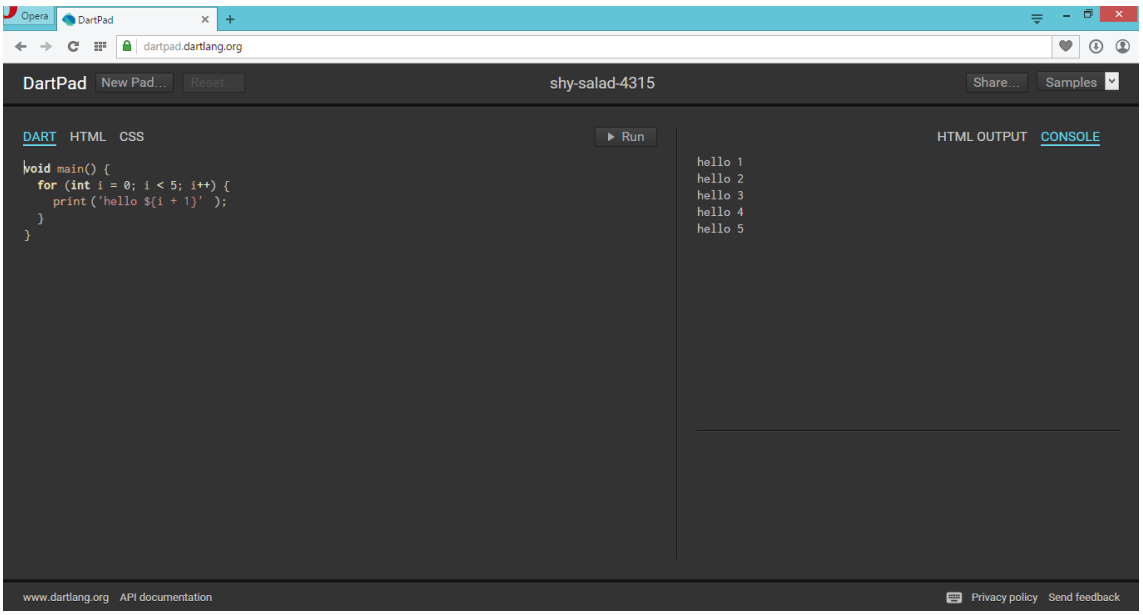
Prvním zmíněným vývojovým prostředím je Dart Editor, který je dostupný ke stažení na oficiálních stránkách projektu. V současnosti je však vývoj tohoto prostředí pozastaven a v novějších verzích Dart SDK jej nenajdeme. Poslední verzí Dart SDK v níž je Dart Editor dostupný je verze 1.10.1. Dart Editor je odlehčenou verzí vývojového prostředí Eclipse, a tudíž pro svůj běh vyžaduje Java Runtime Environment.



Obrázek 3.4: *Prostředí Dart Editoru*

Alternativou k Dart Editoru je IDE s názvem WebStorm od JetBrains, které již má v sobě předinstalovaný plugin pro vývoj aplikací v jazyce Dart. Uživatel také může pomocí pluginů rozšířit své IDE o možnosti podpory jazyka Dart. Takto můžeme rozšířit IDE jako např. IntelliJ IDEA nebo Eclipse.

Pro pouhé vyzkoušení jazyka Dart není třeba stahovat Dart SDK a instalovat některé z výše uvedených IDE. K tomuto účelu nám poslouží webová aplikace s názvem DartPad, kterou nalezneme na stránkách https://dartpad.dartlang.org/.



Obrázek 3.5: *Prostředí DartPad*

V levé části okna můžeme editovat Dart, HTML a CSS kód a po kliknutí na tlačítko Run se v pravé části okna objeví výstup aplikace. Uživatel zde může přepínat mezi HTML výstupem, nebo výstupem do konzole.

3.3.2 Dart2js

Jak již název napovídá, tento command-line nástroj obsažený v Dart SDK slouží k překladu kódu v jazyce Dart do jazyka JavaScript. Vzhledem k nepodpoře Dartu ve většině webových prohlížečů je nutné kompilovat zdrojový kód v jazyce Dart do jazyka JavaScript, aby bylo zajištěno, že uživatelem napsanou aplikaci bude možné spustit ve většině moderních webových prohlížečů.

Spuštění nástroje Dart2js se může lišit v závislosti na platformě (Windows/Linux). V operačním systému Windows se jedná o dávkový soubor s příponou .bat, který si můžeme jednoduše prohlédnout v textovém editoru. Na platformě Windows jej můžeme spustit následujícím způsobem:

```
dart2js.bat -o nameOfOutputJSFile.js dartSourceFile.dart
```

Přepínačem -o se specifikuje název výstupního souboru v jazyce JavaScript. V případě neurčení názvu výstupního souboru bude mít výsledný soubor s JavaScript kódem defaultní název out.js. Kromě tohoto souboru budou vygenerovány ještě další dva soubory. První s příponou js.map, který je použit při debuggování JavaScript kódu v prohlížeči a obsahuje mapování Dart kódu na kód v jazyce JavaScript, a druhý soubor s příponou .js.deps obsahující seznam referencí na soubory použité při kompilaci.

3.3.3 Dartium

Dartium je webový prohlížeč určený pro běh aplikací v jazyce Dart. V podstatě se jedná o open-source prohlížeč Chromium obsahující Dart Virtual Machine. Díky Dart VM odpadá nutnost kompilace Dart kódu do jazyka JavaScript a umožňuje nám plně využít schopností a výkonnosti jazyka Dart.

3.3.4 Knihovny

V Dartu knihovny importujeme pomocí klíčového slova import následovně:

```
import 'dart:html';
```

V tabulce 3.1 uvádím seznam knihoven a jejich použití. Knihovna dart:core je využívána všemi aplikacemi napsanými v jazyce Dart, tudíž ji není nutné explicitně importovat.

Tabulka 3.1: *Seznam knihoven v Dartu*

Název knihovny	Použití
dart:core	Základní funkcionalita (vestavěné datové typy, kolekce, atd.)
dart:html	HTML elementy, manipulace s DOM
dart:io	Manipulace se soubory, procesy, sockety, HTTP
dart:math	Matematické konstanty a funkce, generování náhodných čísel
dart:async	Podpora asynchronního programování
dart:isolate	Konkurenční programování pomocí tzv. Isolates
dart:mirrors	Reflexe
dart:convert	Kódování, UTF-8, JSON, atd.

3.4 Vlastnosti jazyka Dart

V následující části práce si představíme základní vlastnosti jazyku Dart.

3.4.1 Volitelné datové typy

Dart je dynamický typový jazyk. Můžeme v něm napsat programy, které nemají definované datové typy var a spustit je stejně jako v JavaScriptu. Anebo můžeme typ přidat například integer, String, double, boolean, number. Přídáním datového typu získáváme hned několik výhod. Programátoři se lépe orientují v kódu a mohou využít automatického dokončování názvu proměnné. Dart poskytuje statickou kontrolu, která nás může včas varovat před potenciálními problémy a to nám pomůže v ladění programu. V neposlední řadě nám používaní datových typů, pomůže zlepšit výkon naších aplikací a snížit nároky na paměť.

```
var x = 10;
num y = 5;
```

3.4.2 Vstupní funkce main()

Říká se, že Dart má známou syntaxi, takže podobně jako v Javě nebo C, když spustíme náš program, začne se nejdříve provádět kód, který je ve funkci *main()*. Což je výhodné, protože nemusíme hledat počátek kódu. Následující kód nám do konzole vytiskne text Hello world!!!.

```
void main() {
  print("Hello world!!!");
}
```

3.4.3 Operátor kaskády

Podobně jako v programovacích jazycích jako je C++ nebo C# můžeme i v Dartu využívat kaskádové operátory. Výhody operátorů kaskády v Dartu je to, že se kód lépe čte, je kratší a srozumitelnější. V následujícím kódu si ukážeme, jak z DOMu vytáhneme tlačítko s identifikátorem *my - button* a pomocí funkce *main* mu změníme vlastnosti, náš text v tlačítku bude

Open Window napsaný modrým písmem Arial. Dále jsme si v kódu zaregistrovali listener, který při kliknutí na tlačítko otevře nové okno.

V další ukázce kódu si ukážeme, jak stejný kód můžeme díky kaskádovým operátorům zkrátit.

```
void main() {
  var myButton = querySelector("#my-button");
  myButton.text = "Open Window";
  myButton.style.background = "blue";
  myButton.style.fontFamily = "Arial";
  myButton.onClick.listen(openWindow);


void main() {
  querySelector("#my-button")
    ..text = "Open Window"
    ..style.background = "blue"
    ..style.fontFamily = "Arial"
    ..onClick.listen(openWindow);
}
```

3.4.4 Zjednodušený zápis syntaxe

Dart nám dává spoustu možností, jak si usnadnit zapisování kódu. V následující ukázce můžete vidět, jak lze zkrátit zápis konstruktoru a přiřadit hodnoty do třídních proměnných.

```
class Vehicle {
  int numberOfWheels;
  int numberOfEngine;

  Vehicle(int numberOfWheels, int numberOfEngine) {
    this.numberOfWheels = numberOfWheels;
    this.numberOfEngine = numberOfEngine;
  }
}
class Vehicle {
  int numberOfWheels;
  int numberOfEngine;
  Vehicle(this.numberOfWheels, this.numberOfEngine);
}
```

3.5 Pojmenované konstruktory

V Dartu si můžeme zvolit, zda budeme psát datové typy či nikoli, tak nemá smysl nic jako přetěžování funkcí. Dart nemá možnost přetížit metody, což může být problém zejména u konstruktorů. Inženýři z Googlu si s tímto problémem poradili pomocí pojmenovaných konstruktorů.

V následující ukázce si ukážeme, jak takovéto pojmenované konstruktory vypadají.

```
class Vehicle {
  int numberOfWheels;
  int numberOfEngine;
  String color;

  Vehicle(this.numberOfWheels, this.numberOfEngine);
  Vehicle.withColor(this.color);
}
```

3.6 Getter a Setter

Dart na rozdíl od běžných programovacích jazyků má pouze dva modifikátory přístupu private a public. Většinu metod a proměnných necháváme veřejně dostupných. Pomocí modifikátoru *private* omezíme přístup pouze metodám například v knihovnách. Rovněž na rozdíl od programovacího jazyku Java zde nepoužíváme klíčové slovo *private* pro označení soukromé proměnné, ale zapíšeme jej pouze podtržítkem *_color*.

Jelikož stále hovoříme o moderním jazyku Dart, který je plný zjednodušování máme i v následujícím kódu možnost vidět, jak si kód můžeme zkrátit, například symbol znaménka rovná se a uzavírací špičaté závorky nám nahrazuje *{return}*. Dále v kódu vidíme, jak bychom měli správně vypisovat proměnné (pomocí znaku dolaru *$*).

```
String get color => "$_color is the best";
set color(String color) => _color = color;
```

4 Syntaktické srovnání s jazykem JavaScript

4.1 Výchozí funkce

V jazyce Dart je vždy prvně proveden kód, který je uveden ve funkci main a ta je automaticky volána při spuštění programu. V jazyce JavaScript se může výchozí funkce jmenovat jakkoliv nic méně ji musíme volat manuálně. Někdy se v JavaScriptu píše funkce bez názvu, která je automaticky volaná při spuštění programu. Výhodou Dartu je lepší orientace v kódu, jelikož programátor s jistotou ví, že prvně se vykoná funkce *main*.

Dále si v tabulce 4.1 můžeme všimnout syntaxi zapsání funkcí. V jazyce Dart nepíšeme klíčové slovo *function*, což nám opět může připomínat programovací jazyk Java, C, apod.

Tabulka 4.1: *Ukázka kódu výchozí funkce*

Dart	Javascript

`main() {`
`  // náš kód`
`}`
`main(List<String> args) {`
`  // funkce main s agumenty`
`}`
	`function main() {`
	`  // náš kód`
	`}`
	`main(); // manuálně volaná funkce`

4.2 Knihovny

Na rozdíl od JavaScriptu máme možnost v Dartu vytvářet vlastní knihovny. Použití můžete vidět v ukázce (tabulka 4.2).

Tabulka 4.2: *Ukázka kódu knihovny*

Dart	Javascript

`class Car {`
`  move() => 'GO!';`
`}`
`// použití knihovny`
`import 'vehicles.dart';`
`var civic = new Car();`
`import 'vehicles.dart' as cars;`
 	
`var civic = new cars.Car();`
	`// není podporováno`

4.3 Proměnné - vytvoření a přiřazení hodnoty

Dart je dynamický typový jazyk. Můžeme v něm napsat programy, které nemají definované datové typy (*var)* a spustit je stejně jako v JavaScriptu. Anebo můžeme typ přidat například integer, String, double, boolean, number (tabulka 4.3). Přidáním datového typu získáváme hned několik výhod. Programátoři se lépe orientují v kódu a mohou využít automatického dokončování názvu proměnné. Dart poskytuje statickou kontrolu, která nás může včas varovat před potenciálními problémy a to nám pomůže v ladění programu. V neposlední řadě nám používaní datových typů nám pomůže zlepšit výkon aplikací a snížit nároky na paměť.

Tabulka 4.3: *Ukázka kódu proměnné*

Dart	Javascript
`String myName = 'Kamil';` `var myOtherName = 'Kamil';`	`var myName = 'Kamil';`

4.4 Proměnné - výchozí hodnoty

V jazyce Dart je vše objektem, jak můžete vidět v tabulce 4.4, při vytvoření proměnné bez přiřazení hodnoty proměnné nabývají hodnotu "*null*".

Tabulka 4.4: *Ukázka kódu výchozí hodnoty proměnných*

Dart	Javascript

`var myName;`
`// == null`
 	
`int x;`
`// == null`
	`var myName;`
	`// == undefined`

4.5 Final proměnné

V Dartu máme možnost definovat proměnné, které nejdou dále modifikovat. Tyto proměnné uvozujeme klíčovým slovem *final*. Jak můžete vidět v tabulce 4.5, proměnné final jako každé jiné proměnné v Dartu můžeme určit datový typ. V případě pokusu přepsání final proměnné nám kompilátor vyhodí chybovou hlášku.

Tabulka 4.5: *Ukázka kódu proměnné typu Final*

Dart	Javascript

`final name = 'Kamil Niemczyk';`
`name = 'Nikola';`
`// ERROR: cannot assign value to final variable`
	`// není podporováno`

4.6 Kontrola prázdného řetězce

V Dartu prázdný řetězec poznáme tak, že na daný řetězec zavoláme metodu *isEmpty*, který vrací true v případě prázdného řetězce, v opačném případě vrací false (tabulka 4.6).

Tabulka 4.6: *Ukázka kódu kontrola prázdného řetězce*

Dart	Javascript

`var emptyString = '';`
 	
`if (emptyString.isEmpty) {`
`print('use isEmpty');`
`}`
	`var emptyString = '';`
	
	`if (!emptyString) {`
	`console.log('empty strings are treated as false');`
	`}`

4.7 Další nejasnosti v JavaScriptu

Jak můžete vidět v tabulce 4.7, na rozdíl od JavaScriptu v Dartu můžeme kontrolovat, zda-li naše proměnná má hodnotu 0, null případně jestli je hodnota v proměnné konečná.

Tabulka 4.7: *Ukázka kódu nejasnosti v JavaScriptu*

Dart	Javascript

`var zero = 0;`
 	
`if (zero == 0) {`
`print('use == 0 to check zero');`
 	
`}`
 	
`var myNull = null;`
 	
`if (myNull == null) {`
`print('use == null to check null');`
`}`
 	
`var myNaN = 0 / 0;`
 	
`if (myNaN.isNaN) {`
`print('use isNaN to check if a number is NaN');`
`}`
	`var zero = 0;`
	
	`if (!zero) {`
	`console.log('0 is treated as false');`
	`}`
	
	`var myNull = null;`
	
	`if (!myNull) {`
	`console.log('null is treated as false');`
	`}`
	
	`var myNaN = NaN;`
	
	`if (!myNaN) {`
	`console.log('NaN is treated as false');`
	`}`

4.8 Zjištění datového typu proměnné

Díky tomu, že Dart již od svého zrození je připraven na používaní datových typů u proměnných, tak velice snadno můžeme kdykoliv zjistit datový typ. Příklad můžete vidět v tabulce 4.8.

Tabulka 4.8: *Ukázka kódu zjištění datového typu*

Dart	Javascript

`var name = 'Bob';`
`name is String // == true`
`name is! int // == true`
	`var name = 'Bob';`
	`name instanceof String || typeof name === 'string';`
	`// == true`
	`(!(name instanceof Number || typeof name === 'number'));`
	`// == true`

4.9 Definování a instance třídy

Jak můžeme vidět v ukázce kódu (tabulka 4.9), tak definování tříd a celkově práce s třídami je v Dartu přirozenější. Také můžeme vidět v ukázce zkrácený zápis funkce a doporučenou práci s řetězci pomocí znaku dolaru. Instanci třídy vytvoří jak v JavaScriptu tak v Dartu stejně.

Tabulka 4.9: *Ukázka kódu třídy*

Dart	Javascript

`class Vehicle {`
`  String color;`
`  String detectColor() => 'Vehicle color is $color';`
`}`
`var vehicle = new Vehicle();`
`class Car extends Vehicle {`
`  num countWheels;`
`  Car(String color, this.countWheels) : super(color);`
`}`
`class Vehicle {`
`  String color;`
`}`
	`function Vehicle() {`
	`  this.color = null;`
	`};`
	`Vehicle.prototype.detectColor = function() {`
	`  return 'Vehicle color is ' + this.color;`
	`}`
	`var vehicle = new Vehicle();`
	`function Car(color, countWheels) {`
	`  Vehicle.call(this, color);`
	`  this.countWheels = countWheels;`
	`}`

5 Přehled dostupných frameworků pro jazyk Dart a jejich funkční srovnání

Jedním z velmi často používaných frameworků pro práci s webem v JavaSciptu je framework jQuery. Tento framework samozřejmě nemá nic společného s jazykem Dart, ale na jQuery můžeme poukázat jako na typického představitele dnes běžně požívaných frameworků a použít tento framework při srovnávání různých frameworků:

Hned na úvodní stránce webu https://jquery.com/ najdeme miniaturní sekci Brief Look s ukázkami typického použití:

DOM Traversal and Manipulation

```
$("button.continue").html("Next Step...");
```

Nastaví element <button> s třídou continue a změní jeho obsah na řetězec "Next Step..."

Event Handling

```
var hiddenBox = $("#banner-message");
$("#button-container button").on("click", function(event) {
    hiddenBox.show();
});
```

Při stisku libovolného tlačítka v kontejneru s id button-container zobrazí element s id banner-message (předpokládá se, že element je skrytý pomocí display:none v CSS)

Ajax

```
$.ajax({
    url: "/api/getWeather",
    data: {
        zipcode: 97201
    },
    success: function(data) {
        $("#weather-temp").html("<strong>" + data + "</strong> degrees");
    }
});
```

Zavolá skript /api/getWeather umístěný na serveru s parametrem zipcode=97201 vráceným textem přepíše obsah elementu s id weather-temp.

Framework jQuery pro práci s JavaScriptem toho samozřejmě zvládá podstatně více, podrobnosti lze najít ve skvěle zpracované učebnici: http://www.w3schools.com/jquery/.

jQueryUi

Rozšířením základní knihovny jQuery je jQueryUi (jQuery user interface), která umožňuje vytvářet formulářové elementy obvyklé spíše v desktopových aplikacích, než na webu: různá menu, pole pro zadávání různých položek, pole pro práci s kalendářem, tool tip, autokompletaci, záložky (taby), progressbary, dialogy a další.

Dále knihovna jQuery poskytuje množství podpůrných utilit pro práci s těmito elementy: drag&drop operace, barevné manipulace a různé grafické animace: skrytí/zobrazení elementu, změnu velikosti, přizpůsobení elementu jeho obsahu a další.

5.1 Frameworky v jazyce Dart

JavaScript samotný nabizí práci s DOMem, ale práce je těžkopádná, nepřehledná a v každém prohlížeči se může lišit. Proto vznikly knihovny, jako je právě jQuery. Existuje podobná knihovna i pro jazyk Dart? Už úvodní stránka k dokumentaci jazyka Dart [13] nabízí několik přístupů pro tvorbu webových stránek. Doporučují se tři různé frameworky:

- dart:html
- Polymer:dart
- AngularDart

5.2 dart:html

Základním rozhraním pro práci s browserem je balík dart:html. Tvůrci popisují základní práci na stránce Connect Dart & HTML [13].

Srovnání možností jazyka Dart a modulu dart:html je k nalezení na stránce: Improving the DOM.

Základem pro jakoukoliv práci s webem v prohlížeči je vyhledávání elementů na stránce. Dart a jeho modul dart:html přináší zjednodušení především ve srovnání s čistým jazykem JavaScript.

Užitečným vylepšením je využití kontejnerů (kolekcí) při použití funkce querySelectorAll() - tato funkce vrací několik různých elementů. Funkce v Javascriptu vracejí standardní pole (které je součástí jazyka JavaScript), funkce v dart:html vrací různé typy kontejnerů.

5.2.1 Konstruktory

Jazyk Dart a jeho modul dart:html zavádějí konstruktory pro veškeré elementy HTML stránky zobrazitelné ve webovém prohlížeči. Každý element se tak stává samostatnou třídou a lze s ním manipulovat jako s kteroukoliv jinou třídou jazyka Dart.

Tabulka 5.1: *Ukázka kódu JavaScript a html:dart*

JavaScript	html:dart

`document.createElement('div')`	`new DivElement();`
	`new ButtonElement();`
	`new InputElement();`

Konstruktoru lze předat i část HTML kódu, například:

```
TableElement table = new Element.html(
    '<table><tr><td>Hello<em>Dart!</em></table>');
```

5.2.2 Události

Dart používá pro události objektový přístup spojený s obvyklou anonymní funkcí jako callbackem:

```
var subscription = elem.onClick.listen((event) => print('click'));
subscription.cancel();
```

Pokud chceme ošetřit pouze první výskyt nějaké události, lze použít modifikovaný metodu *onEvent.first.then()*, například:

```
element.onMouseUp.first.then((event) => print("Button up"));
```

5.2.3 Callbacky

Callbacky jsou v programování tradičně využívané jako prostředek pro vytváření asynchronně pracujících programů. V JavaScriptu se často využívají jako callbacky anonymní funkce, což v důsledku vede k vnořování callbacků do sebe - programy se stávají nepřehlednými.

Tradiční cesta v JavaScriptu vede přes nastavení vlastností "on__", například:

```
element.onclick = print("click!");
```

jQuery

```
element.click(function() { print ("click!"); });
```

Vytváření callbacků v Dartu vypadá podobně:

```
element.onClick.listen((event) => print("click!"));
```

5.2.4 Rozdělení knihoven

Podobně jako pro komunikaci s DOM strukturou slouží modul dart:html, je možné pro další funkční celky použít jiné moduly, například:

- dart:web_sql - komunikace s SQL databází
- dart:svg - manipulace s SVG grafikou

Rozdělení knihoven na menší celky je věc, která není v čistém jazyce JavaScript řešená žádným způsobem.

Knihovna jQuery tvoří jeden celek (pokud nepočítáme jQueryUi) a některé funkce přístupné v jazyce Dart neřeší vůbec a přenechává je jiným knihovnám (například komunikaci s SQL databázemi).

5.2.5 Manipulace s obsahem dokumentu

Pro manipulaci s daty webové stránky zobrazené v prohlížeči pomocí struktury DOM nepotřebuje Dart žádný externí framework. Knihovna srovnatelná s jQuery je pro Dart zbytečná, neboť samotný jazyk Dart s modulem dart:html poskytuje programátorovi stejný komfort, jako použití knihovny jQuery.

Nejdůležitější součástí všech frameworků pro manipulaci s daty webové stránky zobrazené v prohlížeči prostřednictvím struktury DOM je nalezení požadovaného elementu. Zde se čistý JavaScript ukazuje jako velmi nepohodlná a trnitá cesta, závislá na různých prohlížečích. Selektory použité v jQuery a dalších knihovnách používají podobné postupy jako jazyk CSS pro popis vzhledu dokumentů, například:

- *#xyz* - vyhledá element s id "xyz"
- *.abc* - vyhledá všechny elementy třídy "abc"
- *input:hidden* - vyhledá všechny elementy input typu hidden
- *textarea* - vyhledá všechny elementy textarea.

Toto dotazovací schéma vycházející z jazyka CSS je společné pro všechny knihovny manipulující se strukturou HTML dokumentu [11].

5.3 Knihovna Polymer

Jazyk JavaScript a obvykle ani webové prohlížeče nemají žádné možnosti pro vytváření pokročilejších webových formulářů.

Jazyk HTML disponuje několika málo prostředky pro tvorbu formulářů:

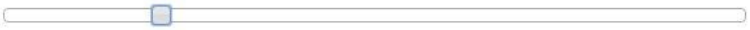
- input
- radio
- select
- textarea
- button

Pokročilejší prvky jazyky JavaScript a HTML postrádají, nenajdeme zde prvky jako je

- slider - nastavení hodnoty posuvníkem
- spinner - nastavení hodnoty zadáním textu nebo klikáním na tlačítko plus či minus
- progressbar - ukazatel stavu zpracování aplikace
- tab - záložky formuláře

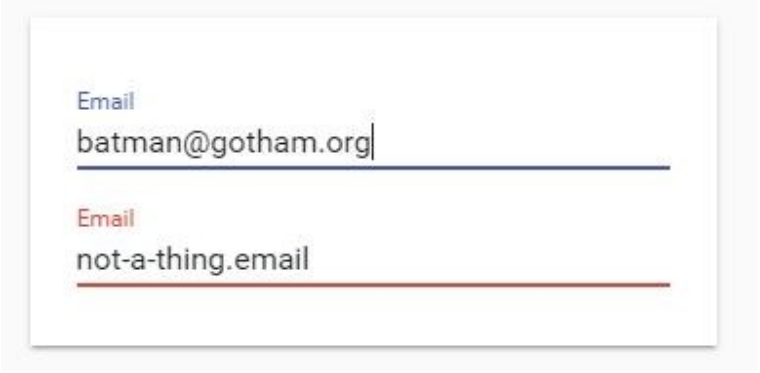
Pokročilejší formulářové prvky lze používat až s různými frameworky, jako je například jQueryUi.

Obrázek 5.1: *Datum jQueryUi*

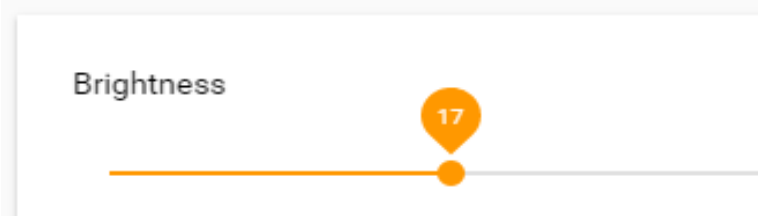


Obrázek 5.2: *Slider jQueryUi*

Jaké možnosti nabízí jazyk Dart pro podporu pokročilejších formulářových prvků máme v prostředí jazyka Dart k dispozici modul Polymer. Seznam možností modulu Polymer nabízí oficiální stránka projektu Polymer.



Obrázek 5.3: *Validace emailu Polymer*



Obrázek 5.4: *Slider Polymer*

Polymer je svými možnostmi dále, než třeba již zmiňovaná knihovna jQueryUi. Jednotlivé elementy dělí do několika skupin [10]:

Železné elementy

Základní komponenty pro vývoj aplikace, z významnějších rozšíření lze uvést dva:

- kontejnery
- lokální storage (přístup na disk)

Papírové elementy

Rozšířené komponenty pro tvorbu webových formulářů:

- různé druhy tlačítek
- prvky pro tvorbu aplikací: menu, záložky, nástrojová lišta, dialogy
- rozšířené formulářové komponenty: posuvník, spinner

Google web elementy

Zprostředkuje přístup k API různých knihoven Google:

- kalendář, video, chromecast
- grafy, google-visualizations
- podpora pro různé Google aplikace: Google Feeds, Google Hangouts, Google Maps Google Maps streets Google Sheets Youtube

Další skupiny jsou zlaté, neonové, platinové elementy a molekuly

Tyto komponenty rozšiřují výše uvedené elementy o další funkce:

- různé vstupní formuláře (čísla kreditních karet, email, telefonní číslo)
- podpora pro Markdown - jednoduchý jazyk pro tvorbu dokumentace
- webové animace
- push služby, messaging, notifikace
- caching, offline obsah

Knihovna Polymer přístupná přímo z jazyka Dart nabízí programátorovi nepoměrně více, než knihovna jQueryUi určená pro jazyk JavaScript. Dá se říci, že knihovna jQueryUi, řeší pouze základní komponenty užitečné pro programování pokročilejších webových aplikací. Knihovna Polymer naproti tomu nabízí komponenty, které posunují programování v jazyce Dart na mnohem vyšší úroveň: řízení rozložení prvků, integrace s google službami, offline obsah a lokální storage, push služby, messaging notifikace a některé další méně významné prvky.

Knihovna Polymer je dostupná jak pro JavaScript, tak pro jazyk Dart. Vývojový tým jazyka Dart doporučuje knihovnu Polymer pro používání při programování v jazyce Dart, obvykle by proto neměl být důvod používat jiný framework, než doporučovaný.

5.4 Knihovna AngularDart

Dalším frameworkem doporučovaným vývojovým týmem jazyka Dart pro programování pokročilých webových aplikací v jazyce Dart je framework AngularDart. Framework Angular je podobně jako knihovna Polymer dostupná jak pro jazyk JavaScript, tak pro programovací jazyk Dart. Knihovna Angular je považovaná za pokročilejší a stabilnější, než knihovna Polymer.

V diskuzi jsou stručně shrnuty základní rozdíly mezi knihovnami Polymer a Angular [9]. Knihovna Angular je zde označena za mnohem robustnější a spolehlivější framework pro tvorbu produkčních aplikací, zatímco knihovna Polymer je v době diskuse ve stadiu alfa. Je nutné zdůraznit, že diskuse o rozdílech mezi knihovnami Polymer a Angular na serveru StackOverflow je ze srpna 2013. Během následujících dvou let prodělal vývoj knihovny Polymer velký skok kupředu, podobně se však vyvíjela i knihovna Angular.

Knihovna Angular se zdá být na internetu i podstatně lépe zdokumentovaná ve srovnání s knihovnou Polymer. I když vyhledávání na Google dává pro obě knihovny podobné počty nalezených stránek, výsledky pro knihovnu Angular jsou jednoznačnější, zatímco výsledky pro knihovnu polymer mohou být silně zkreslené. AngularJS či čistě Angular je vhodně zvolené unikátní jméno, zatímco Polymer je obecný výraz s nejméně dvěma používanými významy (Polymer - název frameworku, polymer - označení molekulární struktury).

Už jen fakt, že knihovna Angular je popisovaná na tak významném zdroji informací, jako jsou stránky W3schools naznačuje, že tato knihovna je lepší volbou pro vývoj pokročilých webových aplikací v jazyce Dart, než knihovna Polymer.

Příznivá je i existence několika různých článků a seriálů o framework Angular na českém webu [14].

5.4.1 Základní vlastnosti frameworku Angular

5.4.1.1 ***MVC***

Framework Angular zavádí do programování webu architekturu MVC (model - view - controller). Tento návrhový vzor rozděluje úlohu na tři nezávislé komponenty, které jsou nezávislé jedna na druhé a jejich změna má jen minimální vliv na ostatní dvě komponenty:

- model - uložení dat v datové struktuře. Struktura je specifická pro konkrétní implementaci.
- view - zobrazení datového modelu v prohlížeči
- controller - řízení návrhového vzoru, reaguje na různé události, například akce uživatele nebo změnu dat v datovém zdroji (databáze), updatuje data v modelu

Ačkoliv zde označuji MVC za návrhový vzor, často se lze setkat s názvem návrhová architektura. Návrhovým vzorem se v programování rozumí jednoduchá či složitější programová struktura určená k řešení nějakého konkrétního problému (například singleton - návrhový vzor, který zajišťuje v aplikaci, že objekt tohoto typu existuje pouze jednou). MVC se tomuto pojetí

návrhového vzoru poněkud vymyká a definuje spíše vzor pro tvorbu celých částí aplikace, než jen vzor pro řešení jednoho specifického problému.

Podstatnou vlastností pro architekturu MVC je nezávislost jednotlivých komponent, snadná rozšiřitelnost a několikanásobná použitelnost. Pro MVC programování je typické, že celá aplikace obsahuje jedinou instanci datového modelu (data jsou do paměti natažená jen jednou), ale různých view je v aplikaci hned několik - stejná data mohou být zobrazená v tabulce, ale i v grafu, případně v několika různých oknech najednou. Controller se pak stará o předávání signálů o změnách v datech jednotlivým view, případně o změny dat v modelu.

Architekturou MVC se framework zásadně liší od frameworku Polymer i od knihovny jQuery a jQueryUi.

5.4.1.2 Two Way Data-Binding

Two Way Data-Binding - česky dvoucestná synchronizace dat je jednou z velice užitečných vlastností frameworku Angular. Jak to funguje? Jednoduchá ukázka je přímo na stránkách dokumentace projektu Angular (https://docs.angularjs.org/api/ng/directive/ngBind).

V ukázce je vstupní pole pro zadávání textu a další pole, které zadaný text zobrazuje. Při změně textu ve vstupním poli se automaticky mění i text v textovém poli. V takto jednoduché podobě není složité naprogramovat takovou aplikaci v jQuery nebo přímo v JavaScriptu. Při použití knihovny Angular je však podstatná jiná skutečnost, využívá se zde návrhová architektura MVC, kde data (zadávaný text) jsou uložená v datovém modelu a vstupní pole i textové pole jsou dvě různá view nad tímto modelem.

Z vlastností architektury vyplývá, že controller dokáže propojit datový model i s jinými zdroji dat, než je jen jednoduché vstupní pole ve webovém formuláři. Ve složitějších případech tak lze jednoduše propojit třeba tabulku zobrazenou v okně webového prohlížeče a databázovou tabulku uloženou na serveru. Jednoduše lze i zajistit modifikace v databázové tabulce při změně dat uživatelem v okně webového prohlížeče, případně updatovat zobrazená data při změně dat v databázovém serveru.

5.4.2 Struktura programů napsaných v Angular

Struktura programů napsaných za použití webového frameworku Angular se svým pojetím liší od programů napsaných za použití jiných webových frameworků. Při vývoji aplikací za použití knihovny jQuery se vytváří statický HTML kód, se kterým pak webová knihovna jQuery dokáže manipulovat. Framework Angular naproti tomu vkládá do HTML kódu vlastní značky a oproti vývoji za použití například knihovny jQuery kód značně zpřehledňuje.

Tabulka 5.2: *Ukázka kódu srovnání jQuery a Angular*

jQuery	Angular

<!DOCTYPE html>
<html>
<script>
$(document).ready(function() {
$("#name-input").change(function() {
var text = $("#name-input").val();
$("#name-output").html(text);
});
});
</script>
<body>
 	
 Name: <input type="text" id="name-input">
 	
</body>
</html>
	<!DOCTYPE html>
	<html>
	<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"> </script>
	<body>
	
	 Name: <input type="text" ng-model="name">
	
	</body>
	</html>

Tatáž funkčnost je v zajištěna mnohem přehledněji, navíc použitím MVC lze distribuovat změnu vstupu stejně přehledně na libovolný počet dalších míst.

Angular rozšiřuje HTML kód o ng-direktivy, v příkladu jsou použité tři základní, zároveň je hezky ilustrované rozdělení podle MVC architektury:

- ng-app - definuje aplikaci Angular
- ng-model - direktiva spojuje hodnotu vstupního HTML elementu (input, select, textarea) s aplikačními daty
- ng-bind - direktiva spojuje aplikační data s HTML view

Kromě tří výše uvedených direktiv je celá řada:

- ng-init - inicializace aplikačních dat
- ng-repeat - klonuje HTML element pro každý záznam v datovém poli
- ng-controller - určuje kontroler pro část HTML kódu. Roli kontroleru zde zastupuje kód napsany v jazyce Dart.
- ng-disabled - nastaví vstupní prvek jako readonly - zakáže jeho změnu uživatelem

- ng-show - podle parametru (true nebo false) zobrazí nebo skryje prvek
- ng-hide - podle parametru (true nebo false) zobrazí nebo skryje prvek
- ng-click - určuje chování při kliknutí na prvek

5.4.3 Filtry

Angular umožňuje řadit za různé výrazy filtry propojené operátorem | (pípa), například takto:

```
<p>The name is {{ lastName | uppercase }}</p>
```

Dostupné filtry:

- currency - zformátuje číslo na měnový formát
- filter - vybere z předaného pole jen určité položky podle zadaných kriterií
- lowercase - změní string na malá písmena
- uppercase - změní string na velká písmena
- orderBy - seřadí pole podle uvedeného výrazu

5.4.4 HTTP klient v knihovně Angular

Podobně, jako webová knihovna jQuery disponuje funkcemi ajax a getJSON, i knihovna Angular dovede stáhnout data z externích zdrojů dostupných protokolem HTTP, například:

```
app.controller('customersCtrl', function($scope, $http) {
            $http.get("http://www.w3schools.com/angular/customers.php")
                .success(function(response) {
                    $scope.names = response.records;
                });
```

5.4.5 Podpora HTML tabulek

Pomocí direktivy ng-repeat se velmi snadno generují tabulky. S použitím různých filtrů lze u generované tabulky snadno zajistit například seřazení podle požadovaného sloupce nebo výrazu:

```
<table>
    <tr ng-repeat="x in names | orderBy : 'Country'">
        <td>{{ x.Name }}</td>
        <td>{{ x.Country }}</td>
    </tr>
</table>
```

Angular poněkud vybočuje z řady ostatních webových frameworků a velice těžko se porovnává. V jedné oblasti s sebou přináší mnohem pokročilejší způsob práce s HTML dokumentem (prakticky se eliminuje nutnost manipulace s DOM strukturou). Aplikaci maximálním způsobem zjednodušuje a zpřehledňuje.

Na druhou stranu takový přístup nemusí ctít jedno ze základních pravidel vývoje webových aplikací - může svádět k méně důslednému oddělení části obsahové (HTML), části vzhledové (CSS) a části aplikační (programová logika).

Ve srovnání s knihovnami jQueryUi a Polymer nepřináší žádné pokročilé formulářové prvky - spoléhá se pouze na standardní formulářové prvky jazyka HTML, případně na prvky z jiných knihoven.

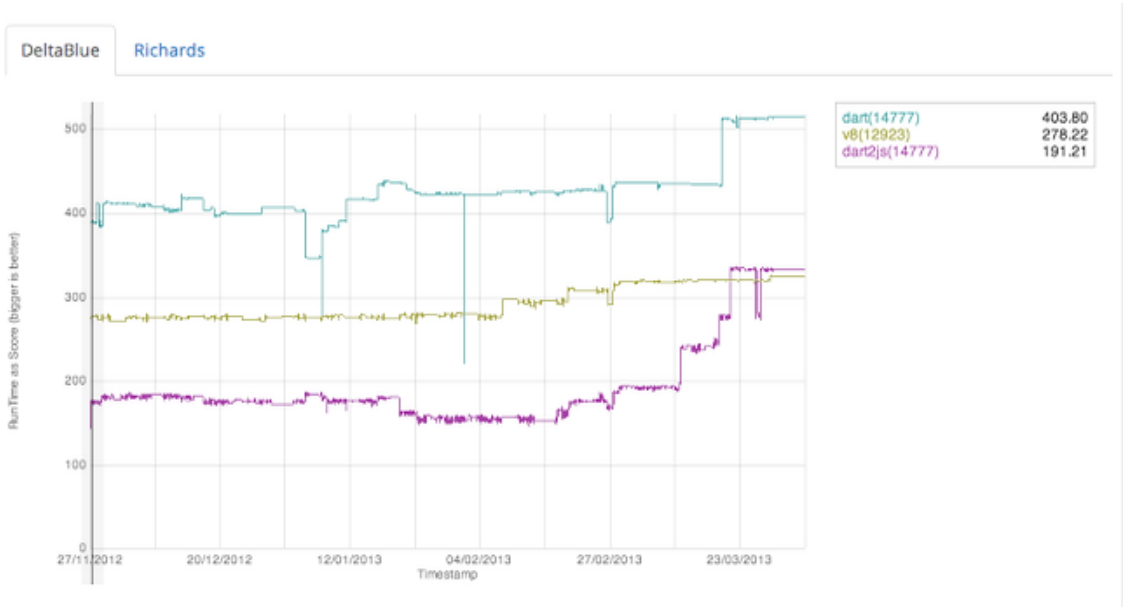
Svým pojetím připomíná spíš šablonový systém pro tvorbu HTML dokumentů, než framework pro tvorbu webových aplikací. Podobnými možnostmi při formátování dokumentů disponuje například šablonový systém Smarty (http://www.smarty.net/), ten se však používá ve spojení s jazykem PHP a běží na serveru.

Šablonové systémy se používají v PHP pro oddělení obsahu dokumentů a programové logiky. Webový framework funguje podobně - přenáší však šablonový systém na stranu webového klienta.

6 Srovnání výkonu vzorové aplikace napsané v jazyce Dart a JavaScript

6.1 Srovnání výkonu JS, Dart, Dart2JS

Jak již jsem zmiňoval v předchozích kapitolách Dart je nejlepší volbou pro velké webové projekty na straně klienta. Dart produkuje velmi čitelný kód a po kompilaci do JavaScriptu běží v různých webových prohlížečích. Dart v Dartiu. 28. března 2013 byla v Dartu výrazně vylepšená kompilace do JavaScriptu. Kód kompilovaný z Dartu výkonnostně předčil ručně psaný JavaScript (obrázek 6.1).



Obrázek 6.1: *Srovnání výkonu Dartu, Javascriptu ručně psaného a kompilovaného z Dartu [7]*

6.1.1 Dart je výkonnější, než JavaScript

Dart je navržen tak, aby nebyl příliš tolerantní k vývojářům, a díky tomu se zvyšuje výkon aplikace. [6]

V Dartu má každý objekt statický tvar. V JavaScriptu mohou být atributy třídy přidány kdykoliv a tím ztrácíme jakoukoliv optimalizaci spojenou s instancí třídy.

V JavaScriptu kdykoliv jsou přidány prvky do pole, v tom okamžiku se pole převede na mapu, kde musíme ukládat dvojici klíč, hodnota. V tomto případě se to provede tak, že index je klíčem. Tento způsob zabírá obrovské místo paměti, což velmi ovlivní výkon aplikace.

Když víme dopředu, kolik budeme potřebovat prvku v poli, můžeme si v Dartu nastavit statické pole o pevné délce. Což je další způsob jak můžeme náš kód optimalizovat.

Dart má k dispozici jak datový typ celých čísel integer, tak desetinných double. Výpočty celých čísel jsou tři až čtyři krát rychlejší, než aritmetika čísel s desetinnou čárkou

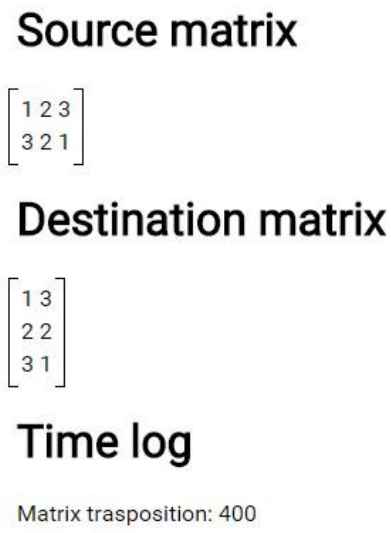
Dart je vytvořen ze stejné skupiny lidí, kteří vyvíjeli engine V8, takže veškeré jejich zkušenosti s JavaScriptem využili pří vývoji Dartu.

Od nuly je psán Dart Virtual Machine, který je určen speciálně pro Dart a je navržen tak aby aplikace v něm spuštěné byly co nejrychlejší.

6.2 Transpozice matice

Pro vlastní porovnání výkonu aplikace psané v jazyce Dart, a zkompilování z jazyka Dart do JavaScriptu jsem vytvořil triviální program. Program má za úkol z jakékoliv zadané matice vypočítat matici transformovanou a zaznamenat čas výpočtu. Transformovaná matice vznikne vzájemnou výměnou řádku a sloupců z matice původní. Pro jednotlivé prvky transformované matice tedy platí $a_{ij} = a_{ji}$.

Na obrázku 6.2 můžete vidět výsledek aplikace. Je zde naše původně zadaná matice, transformovaná matice a čas v milisekundách za jak dlouho byla transpozice vypočtena.



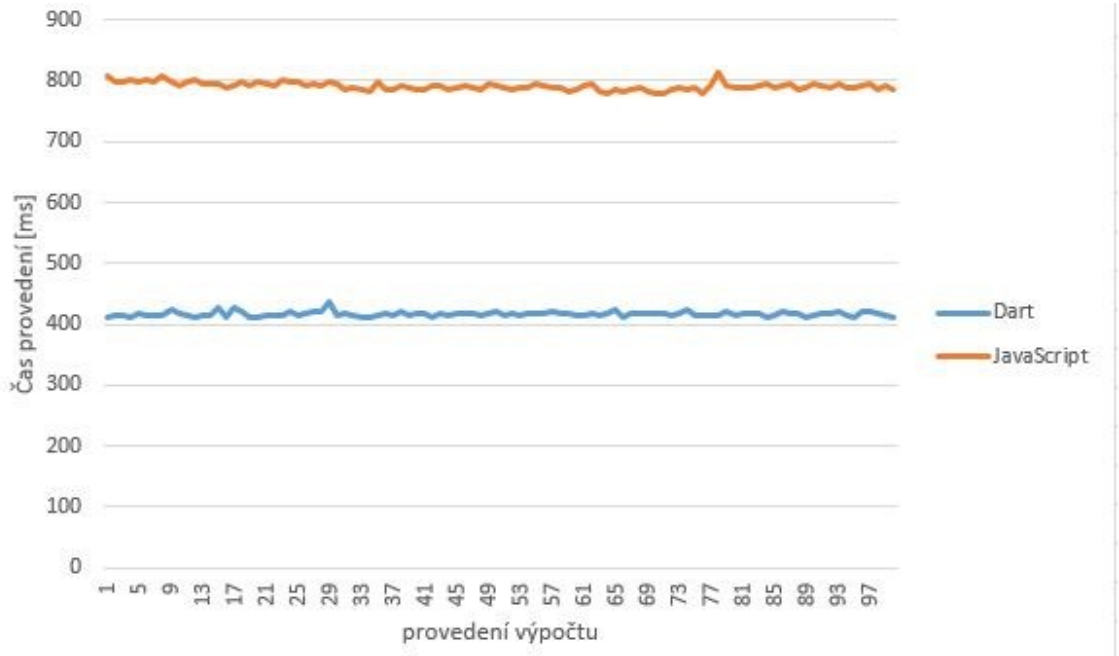
Obrázek 6.2: *Ukázka aplikace*

6.3 Vlastní srovnání výkon

Pro všechny měření jsem použil notebook HP s procesorem Intel Core i5 3210M a s operační pamětí 4GB DDR3. Aplikaci pro výpočet transponované matice vytvořené v jazyce Dart jsem spustil přímo v prohlížeči Dartium verze 39.0.2171.0, což je již zmiňovaný Chromium s Dart VM. Pro testování jsem zvolil čtvercovou matici s devíti prvky. Pro přesnější výsledky jsem spustil výpočet stokrát. Čas výpočtů byl vždy v rozmezí 410 ms - 436 ms (obrázek 6.3), tedy průměrný čas výpočtu byl 417,06 ms.

Tento výsledek jsem porovnával s kódem napsaným v Dartu a přeloženým do JavaScriptu pomocí kompilátor dart2js. Test jsem prováděl se stejnou matici a rovněž byl program cyklicky spuštěn stokrát za sebou. Tentokrát jsem použil prohlížeč Chromium verze 44.0.2403.89.

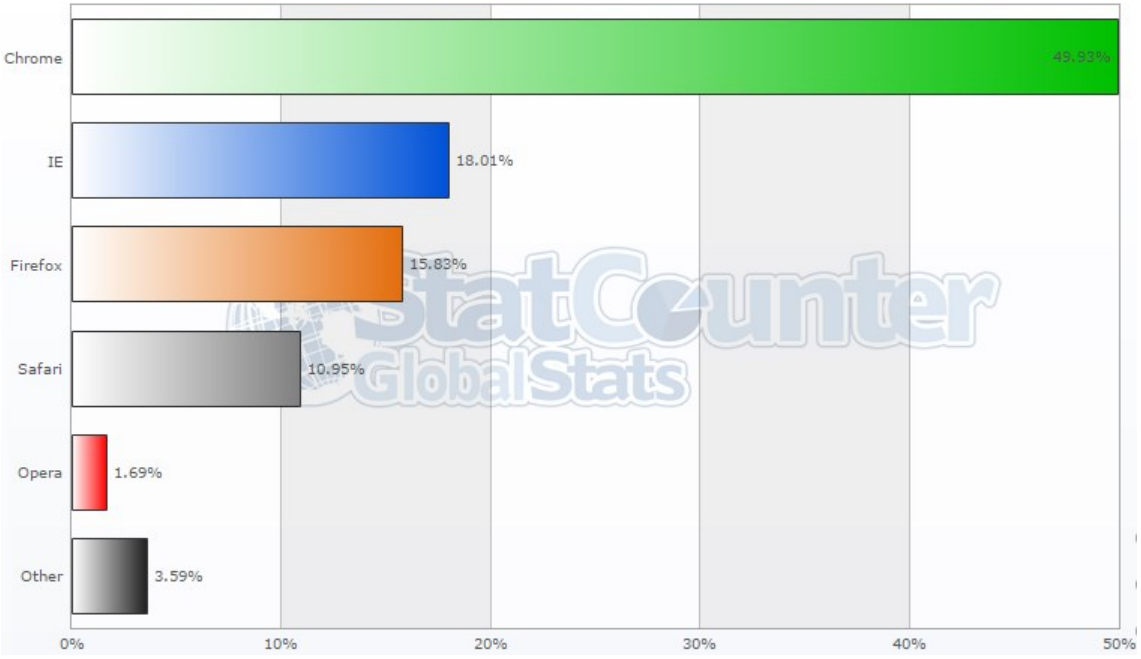
Naměřené hodnoty se pohybovaly od 779ms do 813ms, zde se dostáváme na průměrný čas jednoho výpočtu na 791,19 ms, což je o 47,29% horší výsledek, než program prováděný přímo v jazyce Dart.



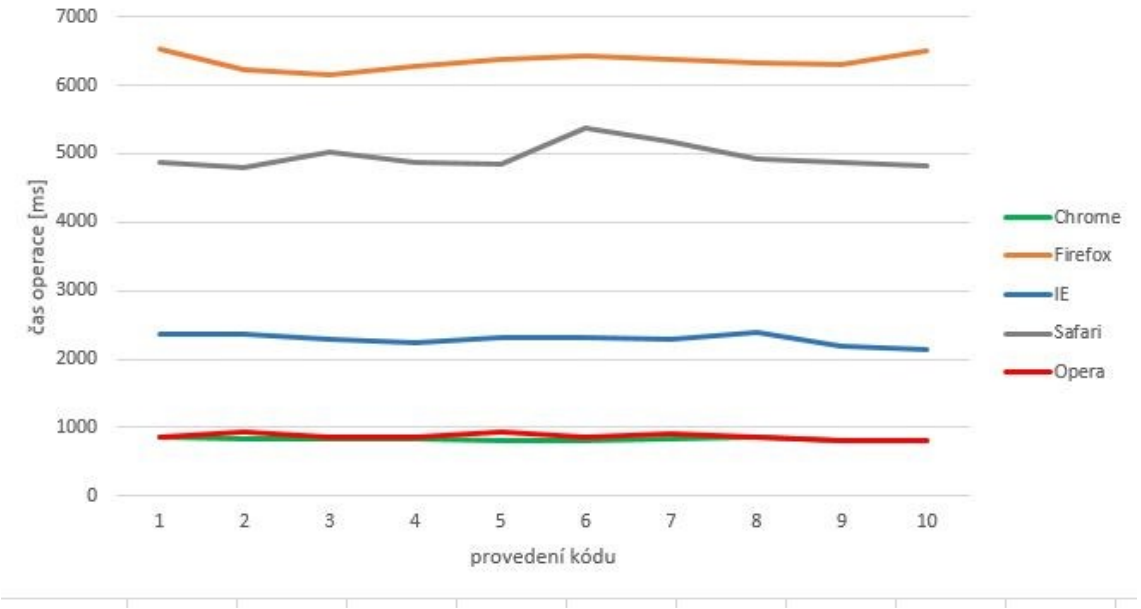
Obrázek 6.3: *Graf pro 100 výpočtů*

Na dalším grafu (obrázek 6.5) jsou vyobrazeny výsledky mého testování, stejné aplikace na výpočet transformované matice napsané v jazyce Dart a přeloženy do JavaScriptu, nicméně tentokrát jsem pro test použil jiné běžně používané webové prohlížeče, které webové prohlížeče jsou v dnešní době nejpoužívanější, jsem zjistil dle statistiky StatCounter (obrázek 6.4). Test jsem nyní provedl pouze desetkrát.

Nejrychleji si s transformaci matice poradil Chrome s průměrným časem 828,7 ms, na druhém místě se umístil webový prohlížeč Opera verze 30.0.1835.125. Tento prohlížeč používá obdobné jádro jako Chrome a díky tomu jsou i výsledky měření jsou velmi podobné. Opera byla pomalejší pouze o 40,8 ms, tedy průměrný čas výpočtu byl 869,5 ms. Dalším testovaným webovým prohlížečem byl Internet Explorer verze 10.0.9200.17413 od firmy Microsoft, ten zvládl matici transformovat za 2286,1 ms. Webový prohlížeč Safari verze 5.1.7534.57.2 od společnosti Apple se umístil na čtvrtém místě s časem výpočtu 4955,7 ms. Jako poslední převedl naší matici webový prohlížeč Mozilla Firefox verze 35.0.1 za 6350,2 ms. Zde si můžeme všimnout obrovského rozdílu, zatímco Firefox vypočítá pouze jednu matici, Google Chrome stihne za stejný čas vypočíst téměř osm takovýchto matic.



Obrázek 6.4: *Celosvětová popularita internetových prohlížečů [12]*



Obrázek 6.5: *Graf pro 10 výpočtu v různých prohlížečích*

Závěr

Cílem práce bylo ukázat nový jazyk pro programování webových aplikací - Dart a porovnat jej s jazykem JavaScript. Porovnání probíhalo v několika oblastech.

Syntaxe jazyka Dart se více blíží obvyklým programovacím jazykům, především v oblasti spojené s definicí objektů a tříd. Zároveň zavádí nové prvky, které usnadňují a zpřehledňují zápis syntaxe.

Jazyk Dart přichází s kompletním vývojovým prostředím. Práce vývojáře je proto v prostředí jazyka Dart snazší v porovnání s JavaScriptem. Podstatným vylepšením prostředí jazyka Dart je existence standardních knihoven a jejich rozdělení na různé moduly.

Podstatnou částí práce při vývoji webových aplikací je manipulace se zobrazovaným dokumentem. JavaScript samotný nabízí množství různých nepřehledných funkcí, které se navíc mohou lišit v každém prohlížeči. Je proto zvykem používat různé knihovny - v porovnání figuruje jQuery. Dart naproti tomu disponuje standardním modulem dart:html a podobné knihovny nepotřebuje. Pro tvorbu rozšířených aplikací jsou představeny frameworky Polymer a Angular. Oba frameworky existují jak pro jazyk Dart, tak pro JavaScript. Možnosti obou jazyků jsou zde vyrovnané.

V poslední části práce jsem se zabýval porovnáním výkonu. Dospěl jsem k závěru, že Dart je dvakrát rychlejší, než Dart kompilovaný do JavaScriptu. Dokonce i ručně psaný program v JavaScriptu je mnohdy pomalejší než kód, který se z Dartu překompiluje do JavaScriptu.

Práce se věnuje porovnání obou jazyků především v oblasti syntaxe, vývojového prostředí, frameworků a výkonu. Z pohledu hodnocených kritérií se jazyk Dart jeví pro vývoj moderních webových aplikací jako vhodnější. V komerční praxi se však při posuzování vhodnosti jazyka pro daný projekt často berou v potaz i jiné aspekty jako je stabilita (výhodnější může být použití zavedených, odladěných a všeobecně známých technologií, u kterých se dá předpokládat další dlouhodobá podpora), cena (dá se předpokládat, že vývojář ovládající méně rozšířený jazyk bude dražší), vedení projektu (je možné ve zvoleném jazyce uřídit vývoj velkých projektů?).

Bylo by jistě přínosné porovnat obě technologie i z těchto hledisek, takové porovnání je však mimo rozsah této práce.

Použitá literatura

[1] Samizdatová skripta. MASOPUST, Lukáš. Historie webu [online]. 2009 [cit. 2015-07-24]. Dostupné z: http://skripta.lmsoft.cz/index.php?id=67

[2] Php [online]. 2015 [cit. 2015-07-24]. Dostupné z: http://cz2.php.net/supported-versions.php

[3] WALRATH, Kathy a Seth LADD. What is Dart? [online]. 2012 [cit. 2015-04-19]. ISBN 978-1-4493-3232-7. Dostupné z: http://it-ebooks.info/book/691/

[4] BELCHIN, Moises a Patricia JUBERIAS. Web Programming with Dart [online]. Apress, 2015 [cit. 2015-07-10]. ISBN 978-1-484205-57-0. Dostupné z: http://it-ebooks.info/

[5] WALRATH, Kathy a Seth LADD. Dart: Up and Running [online]. O'Reilly Media, 2012 [cit. 2015-06-17]. ISBN 978-1-4493-3089-7. Dostupné z: http://it-ebooks.info/book/1089/

[6] High Scalability. ,. High Scalability [online]. 2013 [cit. 2015-07-18]. Dostupné z: http://highscalability.com/blog/2013/3/20/dart-is-it-the-future-of-the-web.html

[7] Dart News & Updates. Dart News & Updates [online]. 2013 [cit. 2015-06-14]. Dostupné z: http://news.dartlang.org/2013/03/why-dart2js-produces-faster-javascript.html

[8] Tiobe. Tiobe [online]. 2015 [cit. 2015-07-19]. Dostupné z: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[9] Stack Overflow. Stackoverflow [online]. 2013 [cit. 2015-07-05]. Dostupné z: http://stackoverflow.com/questions/18089075/what-is-the-difference-between-polymer-elements-and-angularjs-directives

[10] Polymer. Polymer [online]. 2015 [cit. 2015-07-11]. Dostupné z: https://elements.polymer-project.org/

[11] W3schools. W3schools [online]. 2015 [cit. 2015-07-09]. Dostupné z: http://www.w3schools.com/cssref/css_selectors.asp

[12] StatCounter: Global Stats [online]. 2015 [cit. 2015-07-16]. Dostupné z: http://gs.statcounter.com/

[13] Dart [online]. 2015 [cit. 2015-07-24]. Dostupné z: https://www.dartlang.org/docs/

[14] Zdroják. MROZEK, Jakub. Začínáme s AngularJS [online]. 2012 [cit. 2015-07-24]. Dostupné z: http://www.zdrojak.cz/clanky/zaciname-s-angularjs/

Seznam příloh

Součástí BP je CD.

Adresářová struktura přiloženého CD:

1. BP_NIEMCZYK.pdf
2. Tabulky_prakticka_cast.xlsx
3. Transposed_Matrix
    - 3.1. gitignore
    - 3.2. CHANGELOG.md
    - 3.3. LICENSE
    - 3.4. pubspec.lock
    - 3.5. pubspec.yaml
    - 3.6. README.md
    - 3.7. packages
        - 3.7.1. browser
            - 3.7.1.1. dart.js
            - 3.7.1.2. interop.js
    - 3.8. web
        - 3.8.1. index.html
        - 3.8.2. main.css
        - 3.8.3. main.dart

Příloha A: *Naměřené hodnoty aplikace v Dartu*

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
412 ms	415 ms	414 ms	410 ms	418 ms	415 ms	416 ms	415 ms	424 ms	418 ms
[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]
416 ms	413 ms	416 ms	416 ms	428 ms	412 ms	429 ms	420 ms	411 ms	413 ms
[21]	[22]	[23]	[24]	[25]	[26]	[27]	[28]	[29]	[30]
416 ms	416 ms	414 ms	420 ms	414 ms	417 ms	420 ms	420 ms	436 ms	414 ms
[31]	[32]	[33]	[34]	[35]	[36]	[37]	[38]	[39]	[40]
419 ms	415 ms	412 ms	412 ms	414 ms	419 ms	414 ms	421 ms	415 ms	418 ms
[41]	[42]	[43]	[44]	[45]	[46]	[47]	[48]	[49]	[50]
419 ms	413 ms	417 ms	416 ms	419 ms	419 ms	418 ms	414 ms	417 ms	420 ms
[51]	[52]	[53]	[54]	[55]	[56]	[57]	[58]	[59]	[60]
416 ms	417 ms	416 ms	418 ms	417 ms	418 ms	420 ms	418 ms	418 ms	415 ms
[61]	[62]	[63]	[64]	[65]	[66]	[67]	[68]	[69]	[70]
416 ms	417 ms	416 ms	419 ms	425 ms	413 ms	418 ms	417 ms	417 ms	417 ms
[71]	[72]	[73]	[74]	[75]	[76]	[77]	[78]	[79]	[80]
419 ms	414 ms	418 ms	423 ms	416 ms	416 ms	415 ms	415 ms	420 ms	416 ms
[81]	[82]	[83]	[84]	[85]	[86]	[87]	[88]	[89]	[90]
419 ms	418 ms	417 ms	413 ms	415 ms	420 ms	417 ms	419 ms	413 ms	415 ms
[91]	[92]	[93]	[94]	[95]	[96]	[97]	[98]	[99]	[100]
417 ms	417 ms	420 ms	415 ms	413 ms	422 ms	421 ms	417 ms	416 ms	413 ms

* v hranatých závorkách je vždy uvedeno číslo měření a pod číslem měření je odpovídající čas vykonávaní programu

Průměrný čas vykonávání programu v Dartu je 417,06 ms. Naměřené hodnoty se pohybují od 410 ms do 436 ms. Směrodatná odchylka je 3,74.

Příloha B: *Naměřené hodnoty aplikace v JavaScriptu*

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
808 ms	799 ms	797 ms	801 ms	798 ms	801 ms	798 ms	807 ms	797 ms	791 ms
[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]
798 ms	802 ms	794 ms	796 ms	794 ms	790 ms	791 ms	797 ms	793 ms	799 ms
[21]	[22]	[23]	[24]	[25]	[26]	[27]	[28]	[29]	[30]
795 ms	793 ms	800 ms	798 ms	799 ms	792 ms	795 ms	793 ms	798 ms	796 ms
[31]	[32]	[33]	[34]	[35]	[36]	[37]	[38]	[39]	[40]
784 ms	788 ms	785 ms	783 ms	797 ms	787 ms	785 ms	792 ms	789 ms	785 ms
[41]	[42]	[43]	[44]	[45]	[46]	[47]	[48]	[49]	[50]
784 ms	791 ms	793 ms	787 ms	788 ms	791 ms	788 ms	784 ms	795 ms	793 ms
[51]	[52]	[53]	[54]	[55]	[56]	[57]	[58]	[59]	[60]
788 ms	784 ms	789 ms	788 ms	794 ms	791 ms	790 ms	788 ms	782 ms	786 ms
[61]	[62]	[63]	[64]	[65]	[66]	[67]	[68]	[69]	[70]
792 ms	795 ms	782 ms	780 ms	785 ms	782 ms	786 ms	789 ms	783 ms	779 ms
[71]	[72]	[73]	[74]	[75]	[76]	[77]	[78]	[79]	[80]
780 ms	784 ms	789 ms	784 ms	789 ms	780 ms	791 ms	813 ms	792 ms	788 ms
[81]	[82]	[83]	[84]	[85]	[86]	[87]	[88]	[89]	[90]
790 ms	789 ms	793 ms	795 ms	789 ms	792 ms	795 ms	787 ms	790 ms	794 ms
[91]	[92]	[93]	[94]	[95]	[96]	[97]	[98]	[99]	[100]
791 ms	788 ms	796 ms	790 ms	790 ms	793 ms	794 ms	784 ms	793 ms	784 ms

* v hranatých závorkách je vždy uvedeno číslo měření a pod číslem měření je odpovídající čas vykonávaní programu

Průměrný čas vykonávání programu v JS je 791,19 ms. Naměřené hodnoty se pohybují od 779 ms do 813 ms. Směrodatná odchylka je 6,22.

Příloha C: *Tabulka naměřených hodnot aplikace v JavaScriptu v různých prohlížečích*

Číslo měření	Chrome	Opera	IE	Safari	Firefox
1	852 ms	865 ms	2 374 ms	4 868 ms	6 535 ms
2	827 ms	928 ms	2 357 ms	4 790 ms	6 235 ms
3	829 ms	858 ms	2 279 ms	5 029 ms	6 152 ms
4	832 ms	857 ms	2 234 ms	4 881 ms	6 272 ms
5	815 ms	924 ms	2 313 ms	4 847 ms	6 369 ms
6	821 ms	862 ms	2 305 ms	5 372 ms	6 421 ms
7	834 ms	898 ms	2 284 ms	5 178 ms	6 368 ms
8	857 ms	860 ms	2 389 ms	4 915 ms	6 332 ms
9	810 ms	821 ms	2 189 ms	4 864 ms	6 314 ms
10	810 ms	822 ms	2 137 ms	4 813 ms	6 504 ms
Průměrný čas	828,70 ms	869,50 ms	2 286,10 ms	4 955,70 ms	6 350,20 ms
Směrodatná odchylka	15,27	35,03	76,71	176,37	111,27