Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Tvorba webových aplikací v jazyce DART

Creation of web applications by DART language

Bakalářská práce

Vypracovala: Michaela Čapková

Vedoucí práce: PaedDr. Petr Pexa, Ph.D.

České Budějovice 2016

Zadání bakalářské práce

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michaela ČAPKOVÁ**
Osobní číslo:	**P13095**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**Tvorba webových aplikací v jazyce Dart**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Cílem bakalářské práce je zpracování problematiky nového objektově orientovaného programovacího jazyka DART, vyvíjeného společností Google. Tato open-source platforma je určena k tvorbě webových aplikací ve značkovacím jazyce HTML5. Dart obsahuje i vlastní vývojové prostředí, knihovny a prohlížeč, který dokáže Dart spustit a také překladač do jazyka JavaScript, jehož nedostatky se snaží odstranit a v budoucnu možná zcela nahradit. Teoretická část bude zaměřena na samotný Dart, bude popsána syntaxe, funkce, obohacení a další možnosti, které spolu s Dartem přichází, ale i problémy, které zatím nejsou vyřešeny. Bude provedeno porovnání s jazykem JavaScript a zjištěny výhody či nevýhody jazyka Dart. V praktické části bude vytvořena sada webových aplikací, které budou podrobně popsány a použité způsoby řešení vysvětleny, bude také provedeno otestování podpory této nové technologie v aktuálních verzích desktopových a mobilních prohlížečů.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **BALBAERT, Ivo. Dart Cookbook. USA: Packt publishing, 2014. ISBN 9781783989621.**
2. **KOPEC, David. Dart for Absolute Beginners. USA: Apress, 2014. ISBN 978-1-4302-6482-8.**
3. **BRACHA, Gilad. The Dart Programming Language. USA: Addison-Wesley Professional, 2015. ISBN 978-0321927705.**
4. **Dartlang [online]. 2015 [cit. 2015-03-16]. Dostupné z: https://www.dartlang.org/**
5. **Zdroják [online]. 2014 [cit. 2015-03-22]. Dostupné z: http://www.zdrojak.cz/**

Vedoucí bakalářské práce: **PaedDr. Petr Pexa, Ph.D.**
Katedra informatiky

Datum zadání bakalářské práce: **27. dubna 2015**

Termín odevzdání bakalářské práce: **29. dubna 2016**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 27. dubna 2015

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracovala samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 12. dubna 2016

Michaela Čapková

Abstrakt

Cílem této bakalářské práce je zpracování problematiky nového objektově orientovaného programovacího jazyka DART vyvíjeným společností Google. Tato open-source platforma je určena k tvorbě webových aplikací ve značkovacím jazyce HTML5. DART obsahuje ale i vlastní vývojové prostředí, knihovny, prohlížeč, který dokáže DART spustit a také překladač do jazyka JavaScript, právě jehož nedostatky se snaží odstranit a v budoucnu možná zcela nahradit.

Teoretická část bude zaměřena na samotný DART. Bude popsána syntaxe, funkce, obohacení a další možnosti, které spolu s DARTem přichází, ale i problémy, které zatím nejsou vyřešeny. Bude porovnán s podobným, již zmiňovaným jazykem JavaScript, a zjištěno jaké výhody či nevýhody oproti němu přináší.

V praktické části bude vytvořena sada webových aplikací naprogramovaných v tomto jazyce, které se umístí pomocí webové stránky na internet. Zhotovené aplikace budou podrobně popsány a použité způsoby řešení vysvětleny.

Klíčová slova

Dart, HTML5, CSS3, Google, JavaScript

Abstract

The aim of this thesis is to describe new object-oriented programming DART language which is created by company Google. This open-source platform is determined for creation of web applications in mark language HTML5. DART includes its own development setting, libraries, browser which can start DART and also translator to JavaScript language and it tries to remove its inadequacies and maybe even replace it in the future.

Theoretical part will be aimed to DART itself. The syntax, functions and other possibilities, which come together with DART and also the problems which have not been solved yet will be described. DART will be compared with similar JavaScript and its advantages and disadvantages will be find.

In the practical part a set of web applications programmed by DART will be created. The applications will be placed by web page on the internet. The created applications will be described and the code will be explain.

Keywords

Dart, HTML5, CSS3, Google, JavaScript

Poděkování

Děkuji panu PaedDr. Petru Pexovi, Ph.D. za veškerou pomoc při realizaci této bakalářské práce. Velice si vážím všech odborných rad, myšlenek a nápadů, které mi poskytl a skrze nichž moji práci obohatil. Rovněž si cením podpory, trpělivosti a vstřícnosti, kterou po celou dobu ochotně projevoval.

Velké poděkování patří také mým rodičům, kteří jsou mi dennodenně oporou ve studijních i životních situacích.

Obsah

1 Úvod

Bakalářská práce se zabývá problematikou nově vyvíjeného objektově orientovaného jazyka Dart a jeho využití při tvorbě webových aplikací za pomoci HTML5 a CSS3. Teoretická část je zaměřena na seznámení se s tímto jazykem. Kromě informací o jazyce samotném, vytvořených knihovnách, vývojovém prostředí nebo prohlížeči Dartium práce zahrnuje i srovnání s v současné době velice používaným jazykem při tvorbě webových stránek JavaScriptem.

Praktická část zahrnuje sadu aplikací, na kterých je názorně předvedeno, co lze v jazyce Dart vytvořit. Postup je podrobně popsán pomocí komentářů umístěných přímo v naprogramovaných souborech, které jsou přiloženy prostřednictvím CD. Tištěná verze obsahuje informace o aplikacích, grafice, ovládání a samotných funkcích. V první řadě se jedná o aplikaci Hodiny, kde si uživatel může nastavit svůj vlastní čas či se hodiny učit ve výukovém režimu. Práce dále obsahuje hru Slova. Principem této hry je zjištění hledaného slova pomocí postupně dosazovaných písmen. V tomto programu je vytvořená i animace využívající vektorovou grafiku. Další hrou je aplikace pojmenovaná Pipe s cílem propojit jednotlivé body, co nejkratší možnou čarou ve správném pořadí. Další ukázkou je využití Dartu k načítání a ukládání dat s využitím formuláře.

1.1 Východiska práce

Dnes se na webu obvykle vytváří logické funkce, náhledy obrázků, hry, aplikace, kalendáře, hodiny, animace a další dynamické prvky pomocí jazyka JavaScript. Čím častěji ale vývojáři tento jazyk využívají, tím více v něm nacházejí nedostatky nebo věci, které tam prostě chybí. Proto vznikl nový jazyk DART vyvíjený společností Google od roku 2011, který je určený na vývoj HTML5 webových aplikací.

V angličtině vyšlo už několik knih zabývající se dopodrobna tímto poměrně novým jazykem. V českém jazyce ale doposud nebyla sepsána obsáhlejší práce, která by popisovala jak samotný jazyk, tak další možnosti, které lze v souvislosti s DARTem využít.

1.2 Cíle práce

Cílem bakalářské práce je rozebrání objektově-orientovaného jazyka DART a dalších eventualit, které jsou v této souvislosti nabízeny, to znamená vývojové prostředí, knihovny, prohlížeč Dartium a překladač do jazyka JavaScript. Samozřejmě se popíší i kritická místa či problémy, které DART musí v budoucnu odstranit.

Část práce bude zaměřena na porovnání DARTu s jazykem JavaScript. Blíže se popíší výhody a nevýhody, které tyto jazyky v závislosti na sobě nabízejí neboli respektive schopnosti DARTu a důvody, proč by měl být používán namísto poměrně oblíbeného a prohlížeči podporovaného JavaScriptu.

V praktické části se naprogramuje sada aplikací v jazyce DART za použití HTML5 a CSS3. Aplikace budou vybrány tak, aby byly ukázány různé postupy, funkce a rozmanitost použití tohoto jazyka. Webové aplikace budou podrobně zdokumentovány a vysvětleny postupy, které na jejich vytvoření byly použity. Hotové aplikace budou umístěné na webovou stránku, která bude sloužit i jako malý průvodce jazykem DART. Stránka bude obsahovat, jak informace o tomto jazyce a jeho použití, tak i návod na vytvoření obsažených aplikací, jenž mohou být dobrou inspirací pro tvorbu vlastních aplikací obsahující podobné funkce, které dokáží webové stránky obohatit a oživit.

1.3 Metoda práce

V úvodu práce bude rozebrán jazyk DART, jeho specifikace a naopak podobnosti s jinými jazyky.

Bude popsáno vývojové prostředí DART Editor, knihovny a překladač do jazyka JavaScript, který je zatím nutno použít pokud chceme aplikaci vytvořenou v DARTu umístit na internet, jelikož běžné prohlížeče zatím tento jazyk nepodporují. Další část bude zaměřena na zjištění rozdílů Dartu oproti podobnému programovacímu jazyku JavaScript.

Dále bude vytvořena sada aplikací zaměřující se na různé principy řešení, kterých můžeme díky Dartu využít. To znamená, aplikace využívající změnu elementů v HTML stránce či vybírání objektů pomocí myši nebo vykreslování křivek. Poznatky se budou sepisovat do teoretické části. Podrobně bude vysvětlen způsob naprogramování jednotlivých aplikací. Hotové aplikace budou umístěny na webové

stránce, která bude rovněž obsahovat návod k vytvoření programu.

V různých mobilních a desktopových prohlížečích se otestuje funkčnost hotových aplikací.

2 Seznámení s jazykem DART

Dart je nový, inovativní, open-source programovací jazyk, který začala vyvíjet společnost Google v roce 2011 pod křídly dánského programátora Larse Baka, který je mimo jiné známý svou prací na virtuálních strojích, prohlížeči Google Chrome a projektech programovaných v JavaScriptu. Dart slouží k moderní tvorbě webových aplikací. V případě jednodušších aplikací s ním lze pracovat strukturovaně. Při tvorbě rozsáhlejších aplikací je výhodou možnost programovat objektově orientovaně s využitím tříd, dědičnosti či rozhraní. V některých ohledech se Dart podobá JavaScriptu, který se ale chystá v budoucnu nahradit. Proto se jeho vývojáři snaží, aby byl Dart více univerzální a flexibilní. Dart podporuje jak serverovou, tak i klientskou část. To znamená, že není potřeba spojení dvou odlišných jazyků (jako například v případě JavaScriptu s PHP). Současné verze prohlížečů Dart zatím nepodporují, k dispozici je ale kompilátor dart2js, který kód vytvořený v jazyce Dart přeloží do JavaScriptu a umožní ho spustit na libovolném prohlížeči [1, 7, 8].

Dart patří do skupiny interpretovaných programovacích jazyků, pro jejichž spuštění je nutná interpretace zdrojového kódu pomocí speciálního programu zvaný interpret [19].

Oficiálním logem jazyka se stala, jak už napovídá jeho samotný název, modrá šipka.

Obrázek 1: Oficiální logo programovacího jazyka Dart

Jak už bylo řečeno, Dart je vyvíjen jako otevřený software pod jednou z nejsvobodnějších licencí pojmenované BSD (Barkeley Software Distribution), tudíž

lze zdrojový kód prohlížet, upravovat a šířit pouze s omezením povinnosti uvedení informace o zmiňované licenci a jménu autora.

2.1 Využití jazyka

2.1.1 Dart jako klient

Možností vhodných pro použití Dartu při samotné tvorbě webových stránek, tak pro jejich zpestření, je mnoho. V Dartu lze naprogramovat aplikace zdokonalující webové stránky jako jsou například vizuální efekty, hlášky, kalendáře, galerie obrázků, zobrazování svátků a mnoho dalších. Použití je stejně vhodné pro programování her, které mohou být dokonale vyladěné pomocí kaskádových stylů, jelikož Dart skvěle komunikuje s elementy vytvořenými v HTML. Dobře se pracuje i s animacemi, protože jedním ze specifik je canvas určený k vykreslování rastrové grafiky a také svg zobrazující či tvořící obrázky pomocí křivek využívající vektorové grafiky. Zmiňované eventuality se týkají klientských skriptů, což znamená, že na počítač uživatele (tedy klienta) se odešle HTML stránka i skript. Ten se vykoná až při načtení stránky.

2.1.2 Dart jako server

Dart podporuje i serverové skripty, jejichž principem je odeslání požadavku klienta na server, který vykoná příkazy a uživateli posílá nazpět pouze zpracovaná data ve formátu HTML. Toho lze využít v případě spolupráce s databází či formuláři [9].

Jelikož Dart doposud není součástí webových serverů a webhostingů, nelze serverovou část běžně využívat. Variantou je využití vlastního webového fyzického nebo virtuálního serveru, kde má programátor možnost nahrání veškerých požadovaných souborů podle potřeby.

2.2 Vývojové prostředí

Dart nabízí několik možností, kde psát kód a organizovat vytvářené projekty.

2.2.1 DartPad

Pro první vyzkoušení Dartu není třeba instalovat editor. Stačí otevřít stránku https://dartpad.dartlang.org/, kde se zobrazí online editor DartPad. Orientace je velmi jednoduchá. Vlevo je možnost přepnutí mezi dártím kódem, indexem a kaskádovými styly. Po napsání kódu stačí stisknout prostřední tlačítko Run a běžící aplikace se objeví na pravé straně. V pravém horním rohu je záložka Samples, která poskytuje několik předem připravených ukázkových aplikací [1].

Obrázek 2: DartPad

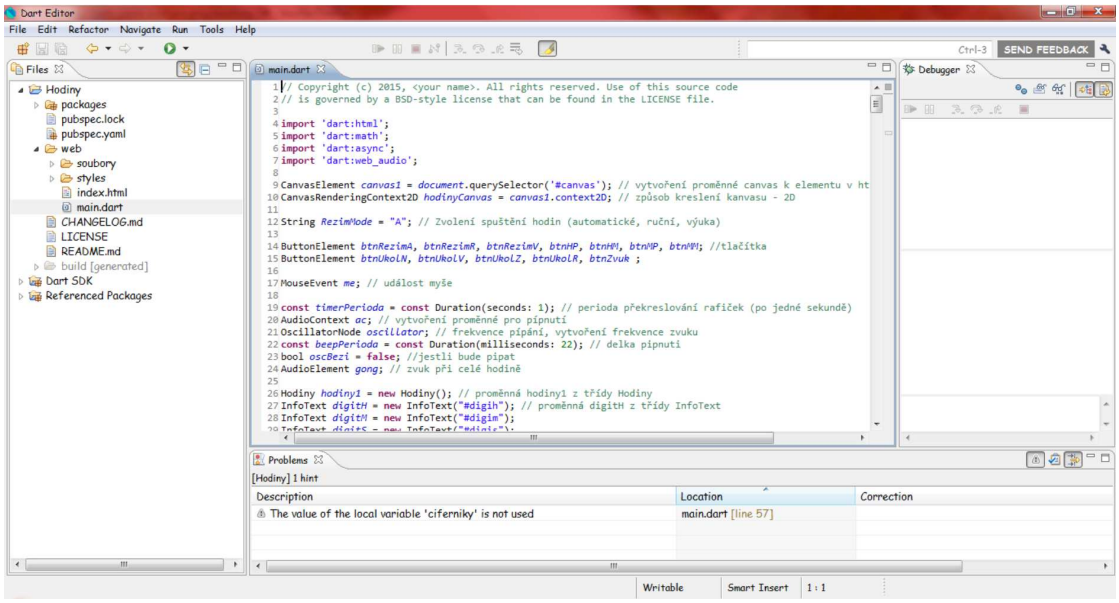
2.2.2 Dart editor

Další možností jak vytvářet programy pomocí jazyka Dart, je instalace editoru obsaženém v balíčku Dart SDK z jeho oficiální stránky, která je k nalezení pod adresou https://www.dartlang.org/downloads/. Editor jde spustit na operačních systémech Windows, Linux i MacOS. Poskytuje běžné operace jako kontrolu chyb, zvýrazňování proměnných, přehled funkcí projektu nebo napovídání při psaní kódu [2].

Dart editor obsahuje několik připravených programů, které uživatele navnadí na vlastní práci a také mu pomohou rozkoukat se v novém prostředí.

V horním menu se nachází lišta se všemi potřebnými nástroji. Pod ní jsou umístěny ikony podle nastavení uživatele. Klíčové je zelené tlačítko s bílou šipkou (Run), které spouští aplikaci v prohlížeči Dartium nebo v panelu vypisuje kód.

Z praktického hlediska je nejpřehlednější si vlevo od prostoru otevřených souborů umístit jejich seznam, vpravo pak přehled funkcí projektu. Dole je umístěn panel vypisující výsledek kódu (při použití příkazu print) nebo oznamující chyby [4].



Obrázek 3: Dart editor

Dart SDK zahrnuje i knihovny, prohlížeč a kompilátor, kterým se budou věnovat další kapitoly.

2.2.3 Další možnosti

Kromě Dart editoru a DartPadu lze využít vývojové prostředí WebStorm, kde je plugin podporující Dart předem nainstalován. Další variantou je doinstalování pluginu do některého dalšího vývojového prostředí, které to umožňuje. Jsou to vývojové platformy Eclipse a JetBrains.

2.3 Syntaxe

Syntaxe Dartu se podobá JavaScriptu, Javě i C#. Umožňuje práci v objektově orientovaném prostředí, což dovoluje poskládat program z více částí, které jsou

znovupoužitelné. V mnoha případech to tedy šetří čas a také zmírňuje chybovost, jelikož nalezenou chybu je potřeba opravit pouze jednou. Atributy a metody, které je určitá třída schopna vykonávat se postupně spojují do velkého celku a ten do sebe pomocí jednotlivých kousků kódu zapadá [11].

Celý projekt se skládá hned z několika souborů. Za prvé je potřeba index.html obsahující HTML elementy naformátované kaskádovými styly, které mohou být umístěny přímo v kódu pomocí stylopisu v hlavičce stránky a nebo externě v připojeném souboru. Dále soubor main.dart, který zahrnuje metody, které aplikace provádí. Při tvorbě rozsáhlého programu se funkce mohou rozdělit do několika souborů a vytvořit si tak vlastní knihovny.

Používá pouze dva typy viditelností, a to private (volají se jen z příslušné třídy) a public (možnost volat je odkudkoli). Označení public a private Dart ale nepoužívá. Proměnné třídy, které jsou private se před názvem doplňují podtržítkem. Viditelnost public je zapsána klasicky bez speciálního zvýraznění [12].

```
1 jmeno; // viditelnost public
2 _vek; // viditelnost private
```

Příklad 1: Znázornění viditelnosti proměnných

K proměnné public se dá přistupovat jednodušším způsobem. Ve třídě se proměnným typu public nemusejí vytvářet funkce get a set. K těmto proměnným lze přistupovat podle následujícího příkladu.

```
1 class Souradnice {int x, y;} // definice třídy
2
3 Souradnice sStart = new Souradnice(20 ,30); // vvytvoření proměnné
4 Souradnice sKonec = new Souradnice(20 ,80); // vytvoření proměnné
5 // použití proměnné
6 var cara = new Cara(sStart.x, sStart.y, sKonec.x, sKonec.y);
```

Příklad 2: Přístup k proměnné typu public

Specificky lze v Dartu přidělovat proměnným funkce a vlastnosti. Pokud má vytvořená proměnná více přiřazení, použijí se místo neustálého přepisování názvu proměnné dvě tečky.

```
1 tlac_kon = querySelector('#btnkon') // přiřazení elementu z HTML
2 ..onClick.listen(_Ukoncit) // přiřazení funkce
```

```
3 ..style.cursor = "pointer"; // změna stylů
```

Příklad 3: Přiřazení funkcí a vlastností proměnné

Dart se opravdu snaží zkracovat zápis kódu oproti jiným programovacím jazykům, kde jen to jde. Dalším příkladem je použití symbolu $ při vypsání hodnot v proměnných do textového řetězce. Místo střídání symbolů uvozovek při psaní textu a plusů při přidání hodnoty obsažené v proměnné ji postačí uzavřít do složených závorek a napsat před ni symbol dolaru.

```
1 print ("Jméno : ${jmeno} Počet bodů: ${body}");
```

Příklad 4: Výpis proměnných v textovém řetězci

Další šikovnou pomůckou může být řešení problému dlouhých řetězců. Pokud zabírá textový řetězec v editoru několik řádek, ale je požadováno, aby se zobrazoval za sebe, uzavře se textový řetězec na každé řádce do uvozovek. Po zobrazení na webové stránce (pomocí nějaké funkce) je řetězec jedním celkem.

```
1 var dlouhyText = 'Dlouhá zpráva,'
2         'která zabírá dvě řádky v editoru.';
3 // zobrazení: Dlouhá zpráva, která zabírá dvě řádky v editoru.
```

Příklad 5: Zobrazení dlouhého řetězce do jednoho celku

Při tvorbě šablon či souborů, které mají být přehledné se výborně hodí umístit řetězec do trojitých apostrofů na začátku a na konci textu. Snadno se vytvoří kratší řádky, které se po spuštění na stránce nebo výpisu v editoru zobrazí stejně jako v editoru [1].

```
1 var viceRadkoveZobrazeni ='''V Dartu můžeš vytvořit
2 pomocí trojitých apostrofů řetězec,
3 který se zobrazí přesně takto.''';
```

Příklad 6: Zobrazení řetězce na více řádcích

2.3.1 Třída

Třídou se rozumí množina objektů se stejnými vlastnostmi a schopnostmi. Nedefinuje jednotlivé objekty, pouze vytváří jednotné rozhraní pro všechny instance třídy.

Instance (neboli konkrétní objekt) na sebe už přebírá hodnoty atributů (vlastností). Odkazovat na tyto objekty můžeme v programech pomocí proměnných [10].

```
1  // Třída pes
2  class Pes {
3     String jmeno, rasa;
4     String getJmeno(){
5        return jmeno;
6     }
7  }
8  // proměnná v jiné třídě dědí vše ze třídy Pes
9  Pes hafan; // deklarace proměnné
10 void main(){
11    hafan = new Pes(); // vytvoření objektu
12 }
```

Příklad 7: Vytvoření objektu dědíci metody celé třídy

Dart používá i svůj vlastní způsob zápisu konstruktoru. Konstruktor se volá vždy při vytvoření instance třídy. Nastavuje její výchozí hodnoty nebo je může obdržet jako parametr (parametrický konstruktor) [13].

Proměnná uvedena v hranatých závorkách je volitelná, to znamená, že při vytvoření může, ale nemusí být uvedena.

```
1  class Zvire{
2  num nohy = 5;
3  num oci = 2;
4  // oci je volitelný parametr
5  Zvire (this.nohy , [this.oci]);
6  }
```

Příklad 8: Zápis konstruktoru

2.3.2 Funkce

Každá aplikace je vybavena hlavní funkcí nazvanou main(). Volají se zde funkce, které má program vykonat po spuštění.

Zápis funkcí lze zjednodušit pomocí symbolu =>. Syntaxe není podstatně kratší. Jedná se o záměnu složených závorek za zmiňovaný symbol. Tento způsob

jde použít pouze u funkcí s jedním výrazem a neobsahující podmínky (např. if). Příklad je uveden na metodách get (vrácení hodnoty) a set (nastavení hodnoty). Ačkoli funguje i typický zápis pro tyto metody běžně používaný v jiných jazycích, zaznamenávání getterů a setterů v rámci jedné řádky kódu tímto způsobem zvyšuje přehlednost.

```
1  // klasický zápis metody get
2  getPocet(){
3  return nohy;}
4
5  // možný zápis v jazyce Dart
6  getPocet() => return nohy;
7
8  // klasický zápis metody set
9  void setNohy(int nohy){
10 NoveNohy = nohy;}
11
12 // možný zápis v jazyce Dart
13 setNohy(int nohy) => (NoveNohy = nohy);
```

Příklad 9: Syntaxe metod get a set

2.3.3 Dědičnost

Klasicky v Dartu využíváme i dědičnost. Třída je schopna z nadtřídy (super třídy) používat její metody i atributy. Nově vytvořená třída tudíž nemusí znovu obsahovat již naprogramované funkce v rodičovské třídě.

2.3.4 Typová kontrola

V Dartu se mohou i nemusí uvádět datové typy. Proměnným nemusí být přiřazen konkrétní datový typ jako například v C# nebo Javě, postačí nadeklarovat var a datový typ zaměňovat v průběhu programování. Pro lepší přehlednost a porozumění kódu se i tak datové typy u většiny jasně zamýšlených proměnných uvádějí a fungují pak jako statické. V Dart editoru je automaticky zapnuta kontrola datových typů (tzv. checked mode), která se při chybném zjištění typu ohlásí [1, 2].

```
1  var i = 10; // neurčený datový typ
```

```
2 i = new Object();
3 i = "Ted' je to string";
4 print(i.length); // vypíše 18
```

Příklad 10: Dynamičnost jazyka Dart

2.3.5 Datové typy

Datové typy stanovují hodnoty, kterých může určitá proměnná nabývat a jaké operace s nimi lze následně provádět. Níže jsou uvedeny ty nejpoužívanější.

Dart používá dva datové typy pro čísla, a to:

- **integer** (int) pro celá čísla, který může nabývat libovolné velikosti v rámci Dartia (po kompilaci do JavaScriptu dlouhé integery nefungují),
- 64-bitový **double** (double) pro desetinná čísla.

S pomocí těchto datových typů lze provádět klasické matematické operace, např. sčítání, odčítání, násobení, zaokrouhlování či zjištění absolutní hodnoty.

```
1 21.2.ceil(); // 22.0 (zaokrouhlení nahoru)
2 21.6.round(); // 22.0 (zaokrouhlení podle hodnoty za des. čárkou)
3 21.2.floor(); // 21.0 (zaokrouhlení dolů)
4 (-22).abs(); // 22 (absolutní hodnota)
5 var vysledek = (8 * 3) / 4 + 2 - 1; // 7.0 (klasické operace)
```

Příklad 11: Matematické operace

Za zmínku stojí i rozdíl zápisu při desetinném a celočíselném dělení, kdy se před klasický znak / přidá tilda.

```
1 var desetinne = 5 / 2; // vysledek je 2.5
2 var celociselne = 5 ~/ 2; // vysledek je 2
```

Příklad 12: Rozdíl zápisu při desetinném a celočíselném dělení

Dalším datovým typem je textový řetězec **string** (String), který je možné uzavřít do dvojitých uvozovek nebo jednoduchých apostrofů. Ke konvertování čísla na řetězec se používá metoda toString(), v opačném případě při změně řetězce na číselný datový typ slouží metoda parse().

Další datový typ nazvaný **boolean** (bool) zaznamenává dvě hodnoty (true nebo false). Boolean v Dartu považuje za pravdivou pouze hodnotu true, vše ostatní pro něj znamená false. To je odlišnost vůči JavaScriptu, který považuje za true všechny neprázdné objekty (not-null) [1].

```
1 var zvire = 'pes';
2 if (zvire){
3 // V Dartu se hláška nevyvolá, protože pes!=true
4 // Kontrola dat. typů vyvolá výjimku - zvire není typu boolean
5 print ('Jsi zvíře!'); // V JS se hláška vyvolá - zvire != null
6 }
```

Příklad 13: Boolean v Dartu

Datový typ **pole** (List) má vyhrazené místo pro několik objektů se stejným datovým typem, počet objektů může být různý. Ke konkrétnímu objektu se přistupuje pomocí indexu, ten je číslován od nuly, to znamená, že index posledního objektu v poli je list.length - 1 [15].

Mapy (Map) jsou objekty, které se skládají z klíče a hodnoty. Klíč i hodnota může mít jakýkoli datový typ. Klíč je uveden vždy jen jednou, stejná hodnota může být použita vícekrát [1].

```
1  var list = [1, 2, 3, 4, 5]; // vytvoření pole
2  var zvire = { // vytvoření mapy
3  // Klíč : Hodnota
4     'prvni' : 'slon',
5     'druhy' : 'jelen',
6     'treti' : 'zebra'
7  };
8  // jiný způsob vytvoření mapy
9  var zvire = new Map();
10 zvire ['prvni'] = 'slon';
```

Příklad 14: Deklarace pole a mapy

Pro čas a datum je připraven datový typ **DateTime**.

2.3.6 Spolupráce s elementy HTML

Jelikož Dart slouží k tvorbě aplikací na webu, jeho základní dovedností je schopnost komunikovat s elementy vytvořenými v HTML dokumentu. Na začátku každého projektu je vytvořen index.html s obrázky, odstavci, odkazy, tlačítky, divy a dalšími klasickými elementy. Pokud program vyžaduje, aby některý z těchto prvků reagoval na nějakou funkci nebo ji spouštěl, musí být v souboru main.dart vytvořena proměnná k patřičnému elementu. Elementu v dokumentu HTML je přiřazen identifikátor (id) nebo třída (class), které jsou při deklaraci použity.

```
1 // část souboru index.html
2 <div id ="divukol"></div>
3 // deklarace v souboru main.dart
4 Element divKategorie = document.querySelector('#divukol');
```

Příklad 15: Přiřazení HTML elementu do funkcí Dartu

2.3.7 Asynchronizace a Future

Asynchronizace se používá, když aplikace vyžaduje, aby nějaká její část běžela „na pozadí“. To znamená, že se určité funkce programu vykonají, i když je uživatel nevidí. Této pomůcky se často využívá při načítání dat z databáze či formulářů.

Future (společně s await) se poté využije při čekání na doběhnutí asynchronních funkcí. Například při načítání většího množství dat future čeká až se načte vše, co je třeba a až pak povolí provedení dalších částí programu a umožní použít uložené proměnné někdy v budoucnu.

Při využití asynchronní funkce je třeba na začátku souboru naimportovat knihovnu pojmenovanou dart:async.

```
1 // Asynchronní funkce běžící na pozadí
2 // Funkce čeká na načtení všech dat
3 Future tiskDat() async {
4 String data = await getVsechnyData();
5 print(data); }
```

Příklad 16: Asynchronní funkce

Pro zachycení chyb je v souvislosti s futures využívána klauzule try-catch, která při nalezené chybě ohlásí výjimku.

```
1 // Asynchronní funkce s try - catch
2 Future tiskDat() async {
3 try {
4 String data = await getVsechnyData();
5 print(data); }
6 catch (e) { // chování při chybě } }
```

Příklad 17: Asynchronní funkce s try-catch

2.4 Prohlížeč

Jelikož Dart doposud nepodporují běžně používané prohlížeče, využívá prozatím speciálně vytvořenou verzi prohlížeče Google Chrome pojmenovanou Dartium. V tomto prohlížeči běží dartí aplikace bez problémů a potřeby kompilace na plný výkon. Prohlížeč Dartium je stažen společně v balíčku Dart SDK [1, 4].

Obrázek 4: Logo prohlížeče Dartium

2.5 Kompilátor

Jak už bylo řečeno v předešlé kapitole, jazyk Dart není doposud podporován aktuálně používanými prohlížeči. Aby mohl program běžet na webu, je nutné ho

kompilovat do jazyka JavaScript. Kompilátor dart2js je opět zakomponován při instalaci Dart SDK.

Kompilovaný soubor nemá podobu, kterou by použil programátor. JavaScriptí soubor se skládá z mnohem většího počtu řádků nepřehledného kódu.

Kompilace Dart souboru je pomocí Dart editoru velice snadná. Po otevření souboru pubspec.yaml umístěném pod hlavní složkou projektu se z nabídky Pub Actions zvolí možnost Run pub build - debug (další možností je z hlavní nabídky horního menu stisknout položku Tools/Run pub build - debug), který kód zkompiluje a umístí do složky projektu/build/web. Do této složky jsou vloženy všechny soubory projektu, které po umístění na web bez problému fungují na dostupných prohlížečích.

Velikost zkompilované složky do JavaScriptu je podstatně menší než původní velikost souborů v Dartu. Výkon zkompilované aplikace do jazyka JavaScript se ale snižuje.

2.6 Knihovny

Dart zahrnuje celou řadu knihoven. Obsahují řadu naprogramovaných tříd s funkcemi, které usnadní a urychlí tvorbu vlastní aplikace. V případě užívání knihoven je dobré stále deklarovat u proměnných konkrétní datový typ místo var. Knihovny typy totiž používají a očekávají konkrétní parametry a také konkrétní hodnoty vrací. Pokud programátor předá špatný parametr, objeví se výjimka v kontrolním panelu (checked mode) [1].

Knihovna se vloží na začátek souboru příkazem import a příslušnou knihovnou v apostrofech.

```
1 import 'dart:html';
2 import 'dart:math';
```

Příklad 18: Import knihovny

Mezi nejpoužívanější knihovny patří:

- dart:core (jádro funkcí, řetězce, čísla, kolekce, chyby, atd.),
- dart:html (HTML prvky, pro webové aplikace),

- dart:io (soubory, HTTP, serverové aplikace, I/O podpora),
- dart:async (pro asynchronní funkce),
- dart:math (matematické funkce, generátor náhodných čísel),
- dart:svg (dvojrozměrná vektorová grafika, animace),
- dart:web_audio (zvuk v prohlížeči).

Pro třídy a funkce, které programátor vytvoří a může využít i v jiných programech lze vytvořit vlastní knihovnu. Pod hlavní složkou projektu se nalézá složka lib, ve které se vytvoří nový soubor s koncovkou .dart. V úvodu tohoto souboru se uvede příkaz library a název knihovny. Poté stačí tuto knihovnu naimportovat jako klasickou knihovnu se změnou cesty, která ke knihovně vede. Před název knihovny se musí uvést package:nazevProjektu/nazevSouboru.dart, protože po spuštění aplikace pomocí tlačítka Run se knihovna překopíruje právě do složky package, ve které je uložena již hotová knihovna, která nejde přepisovat. Úpravy se provádí v souboru uloženém ve složce lib, kde se knihovna vytvořila.

```
1 library mojeKnihovna; // první řádka knihovny mojeKnihovna.dart
2
3 // naimportování vlastní knihovny do souboru main.dart
4 import 'package:NazevProjektu/mojeKnihovna.dart';
```

Příklad 19: Vytvoření vlastní knihovny

2.7 Nedostatky

Hlavním současným nedostatkem je již zmiňovaný fakt, že plný potenciál tohoto jazyka nelze zatím využít z důvodu nutné kompilace do jazyka JavaScript při využití na webu. Nejen, že tento proces programátora zdržuje, ale také dartím aplikacím snižuje původní výkonnost.

Některé funkce Dartu nefungují totožně ve všech webových prohlížečích. Proto je nutné aplikace důkladně otestovat a následně nesrovnalosti ošetřit.

Určité nedostatky se objevují i v samotném editoru. Při použití vlastní knihovny se někdy stává, že se knihovna správně neimportuje do projektu. Místo cesty pac-

kage:NazevProjektu/mojeKnihovna.dart je potřeba uvést relativní cestu k souboru knihovny na disku a projekt znovu zkompilovat.

3 Dart vs. JavaScript

Dart a JavaScript jsou dva jazyky určené k tvorbě webových aplikací, tudíž se sobě navzájem dosti podobají a programátor si mnohdy může vybrat, jaký z nich pro svou aplikaci použije. Stejné ale nejsou.

	Dart	JavaScript
Autor	Lars Bak	Brendan Eich
Vznik	2011	1995
Vývoj	Google	Netscape Communications Corporation Mozilla Foundation
Typová kontrola	volitelná	dynamická
Ovlivněn	JavaScript Smalltalk Erlang	Java C
Vlastnosti	objektově-orientovaný	objektově-skriptovací
	interpretovaný	interpretovaný
	serverový klientský	klientský

Tabulka 1: Základní charakteristika jazyků Dart a JavaScript

Úplně na počátku vzniku jazyka Dart se nadšeně předpokládalo, že JavaScript bude rychle a s přehledem nahrazen. Vývoj nového jazyka doprovázely ale i komplikace a nedostatky, které se programátorům příčily. Během několika let se Dart stále zlepšuje od možností efektivnějšího zápisu kódu po upozornění na chyby a jejich následné opravy či korekce editoru. Vývojáři Dartu zapracovali i na kompilaci do JavaScriptu, kde je v některých situacích kompilovaný kód výkonnější než samotný JavaScript [16].

Obrázek 5: Graf reprezentující výkonnost jazyka Dart, JS a kompilovaného kódu

Syntaxe obou jazyků je dosti podobná. Umí-li programátor JavaScript, naučí se i Dart, který se podobá i Javě. Dart si ale se zápisy kódu hraje. Snaží se je zkracovat, čímž šetří čas a také zpřehledňuje kód. Výborně také pracuje s třídami a dědičností, což je známé a hodně používané v mnoha dalších jazycích.

Jako příklad čistšího kódu lze uvést inicializaci pole, která je zapsána buď to výčtem konkrétních položek nebo počtem nedefinovaných [17].

```
1 // inicializace pole v JavaScriptu
2 var a1 = new Array(1, 2, 3, 4, 5);
3 var a2 = new Array(5);
4
5 // inicializace pole v Dartu
6 List a1 = [1, 2, 3, 4, 5];
7 List a2 = new List(5);
```

Příklad 20: Inicializace pole v Dartu a JavaScriptu

Při použití JavaScriptu je nutné rozeznávat hodnoty null a undefined. Druhá zmiňovaná se dá vysvětlit jako hodnota nedefinovaná. Vyskytuje se například při volání návratové hodnoty, která ale žádnou nevrací či v případě nedefinované proměnné nebo nedefinované vlastnosti objektu. Jedná se o situace, ve kterých by většina jiných programovacích jazyků ohlásila chybu. Je to tedy záležitost, kterou programátor záměrně neočekává. Naproti tomu hodnotou null sám programátor

označuje nezadanou hodnotu. Jelikož lze ale tyto dvě hodnoty zaměňovat ve valné většině scénářů a hodnota undefined je tak trochu nadbytečná, Dart jednoduše používá pouze null [17, 18].

Menší či větší odlišnosti jazyků spočívají i v samotné logice zápisu kódu, např. datový typ boolean, který v JavaScriptu považuje všechny neprázdné objekty za true se v Dartu chová opačně. Připisuje pravdivou hodnotu právě a pouze true, vše ostatní pro něj znamená false.

JavaScript pracuje na straně klienta. Proto musí například pro práci se soubory využívat nějakého serverového jazyka, nejčastěji se jedná o PHP. Tuhle spolupráci Dart zvládá také (viz aplikace Slova). K tomu však i on sám umí vystupovat jako jazyk serverový (viz aplikace Formulář) a pracovat tak s databázemi, soubory či formuláři.

4 Aplikace

Praktická část této práce zahrnuje sadu webových aplikací umístěných na webové stránce s adresou http://dartime.cloud. Z důvodu instalace serverové části a následného spuštění dartího skriptu na serveru pro aplikaci Formulář nebylo možné použít běžný hosting. Místo toho bylo nutné pronajmutí virtuálního serveru, který umožňuje nahrání všechno potřebného.

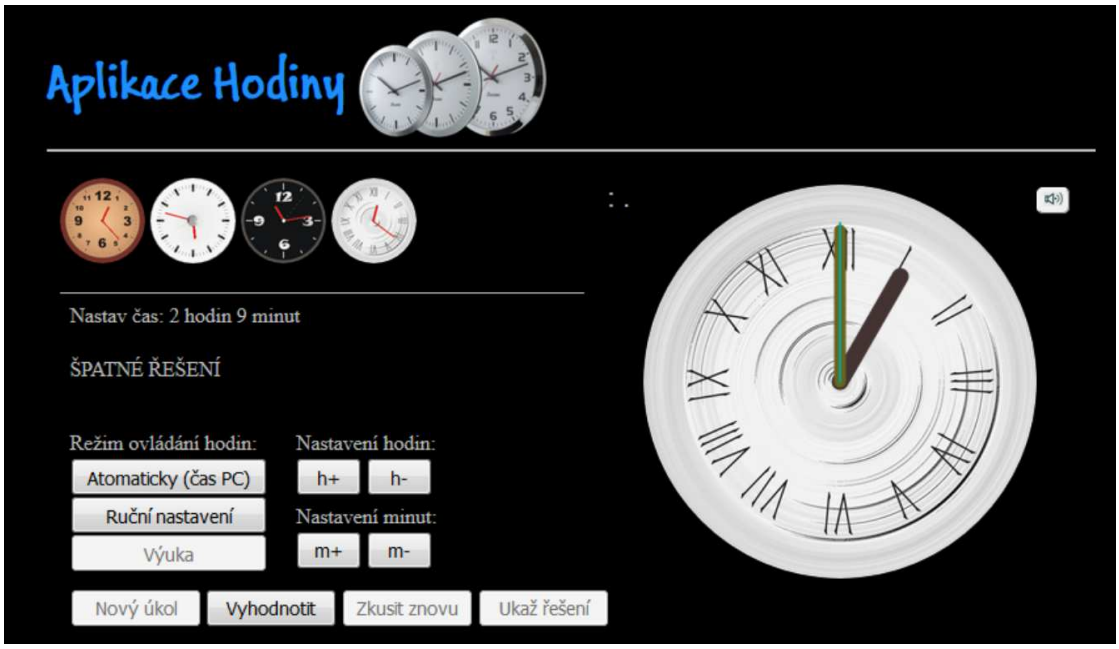
Stránka kromě samotných aplikací obsahuje i popis tvorby programu a informace o jazyce Dart. Typy naprogramovaných aplikací byly vybrány tak, aby na nich byly ukázány různé eventuality, které lze díky jazyku Dart naprogramovat.

V následujících kapitolách je uveden popis každé aplikace, její grafické rozvržení, způsob ovládání a popis programu obsahující nejzajímavější funkce a princip tvorby aplikace. Veškeré zdrojové kódy obohacené o podrobné komentáře jsou přiloženy na CD, které tato práce zahrnuje.

4.1 Aplikace Hodiny

Aplikace Hodiny umožňuje zobrazení aktuálního času na rafičkových hodinách. Dále si uživatel může pomocí tlačítek nastavovat vlastní čas, který je zobrazen digitálně i rafičkově. Třetí variantou je spuštění výukového režimu, pomocí kterého se na obrazovce zobrazí požadovaný čas, jenž má uživatel za úkol nastavit na hodinách. Pokud jej nastaví správně, může si nechat zobrazit další čas. Při špatné odpovědi se naskytují dvě varianty. Buď to uživatel zkusí nastavit hodiny znovu nebo si zobrazí výsledek pomocí příslušného tlačítka. Aplikace je obohacena i o zvuk, respektive tikot rafiček a oznámení celé hodiny.

Aplikace je určena menším dětem, které si díky ní mohou procvičit hodiny a všem ostatním pro zobrazení času.



Obrázek 6: Vzhled aplikace Hodiny

4.1.1 Grafika

Základ vzhledu aplikace je řešen pomocí HTML5 a CSS3. Přehledně jsou na stránce rozmístěna tlačítka a velký ciferník, na kterém se zobrazuje čas. Uživatel má možnost výběru ciferníku z několika možností, které jsou zobrazeny jako miniatury vlevo. Pod tímto výběrem je zobrazeno textové pole, které uvádí instrukce a zadává úkoly. Níže se zobrazují tlačítka s odpovídajícími popisky.

Celkový design aplikace byl zvolen tak, aby zapadl do webové stránky, na které jsou umístěny další aplikace.

Jednotlivé ciferníky byly vytvořeny vektorově pomocí programu CorelDraw a následně uloženy ve formátu png, který umožňuje průhlednost pozadí. Aplikace obsahuje ciferníky pro snadnější určení času zobrazující všechny číslice i hodiny bez uvedení číslic či s číslicemi římskými, u kterých může být správné nastavení času větším problémem.

4.1.2 Ovládání

Aplikace Hodiny se ovládá pomocí tlačítek:

- h-, h+ - pomocí těchto tlačítek se posouvá hodinová ručička (kliknutí na tlačítko h+ znamená přičtení jedné hodiny, h- pak vrácení se o jednu hodinu zpět),
- m-, m+ - pomocí těchto tlačítek se posouvá minutová ručička (kliknutí na tlačítko m+ znamená přičtení jedné minuty, m- pak vrácení se o jednu minutu zpět),
- automatický čas - toto tlačítko nastaví aktuální čas počítače v digitálním i rafičkovém zobrazení; čas dále běží a uvádí reálný čas (pokud např. uživatel nechá aplikaci běžet v jiném okně a vrátí se zpět, ihned ví, kolik je hodin),
- ruční nastavení - po stisknutí tohoto tlačítka se čas nastaví na 00:00, pomocí výše zmiňovaných tlačítek (h+, h-, m+, m-) lze posouvat rafičky na ciferníku, čas se zobrazuje i digitálně,
- výuka - po stisknutí tlačítka zmizí digitální čas a zobrazí další možnosti výběru:
  - ▷ nový úkol - tlačítko zobrazí v příslušném divu čas, který má uživatel nastavit,
  - ▷ vyhodnotit - aplikace vyhodnotí řešení a zobrazí hlášku o správnosti,
  - ▷ zkusit znovu - pokud je řešení špatné, uvolní se tlačítko zkusit znovu, po stisknutí je opět k dispozici tlačítko vyhodnotit,
  - ▷ ukaž řešení - při špatném řešení se po stisknutí tohoto tlačítka zobrazí na ciferníku správný čas.

Hlavní ciferník lze libovolně měnit stisknutím na miniatury hodin. Tlačítko s reproduktorem zapíná/vypíná zvuk tikotu vteřinových rafiček.

4.1.3 Popis funkcí aplikace

Všechny funkce aplikace jsou obsaženy v jediném souboru pojmenovaném main.dart. V úvodu jsou naimportovány knihovny, deklarovány proměnné a následně jim přiřazené funkce. Ciferníky lze měnit pomocí funkce **ZmenCifernik(me)**, která při

kliku nad zmenšeným obrázkem ciferníku hodiny změní načtením adresy příslušného obrázku. Prostor pro hlavní hodiny ukazující čas (případně ho na nich uživatel nastavuje) jsou vytvořeny pomocí canvasu. Obrázek hodin ve formátu png je vložen, rafičky se pak vykreslují podle instrukcí (viz funkce **KresliRafiku(String Ucel, int uhelx)** a **KresliHodiny()**). Úhly se vypočítají podle údaje hodin a minut.

Při spuštění aplikace se na hodinách zobrazuje aktuální čas z počítače vycházející z již přednastavené funkce **DateTime.now()**. Pomocí tlačítek lze pak čas na hodinách přenastavovat podle libosti či zadání (výukový režim). To zajišťují funkce **NastavRucne(me)** a **ZadatUkol(me)**. Porovnávání požadovaného času zadaného pomocí úkolu ve výukovém režimu a času, který uživatel nastavil na hodinách zajišťuje funkce **VyhodnotitUkol(me)** tím, že porovnávají nastavení počtu minut a počtu hodin.

Aplikace umožňuje i zobrazení digitálního času ve funkci **nastavDigiCas()**.

Aplikace obsahuje i další funkce nastavující vzhled aktivních (neaktivních) tlačítek, každému tlačítku jsou pak přiřazeny další funkce podle popisku (viz podkapitola Ovládání).

V tomto programu je i podrobněji vyzkoušeno přidávání zvuků, který lze pomocí ikonky zapnout či vypnout. Funkce **VytvoritOscilator()** a **Pipnuti()** zajišťuje vteřinové tikání hodin. V první funkci je vytvořeno samotné pípání, které zahrnuje např. frekvenci či typ zvuku, ve funkci druhé pak perioda, za kterou se tiknutí opakuje.

```
1  // Funkce, která vytvoří vteřinové pípání
2  void VytvoritOscilator(){
3  ac = new AudioContext();
4  oscillator = ac.createOscillator();
5  oscillator // přiřazení vlastností oscilátoru
6     ..type = "sine" // sínusový zvuk
7     ..frequency.value = 550 // frekvence v Hz
8     ..start(0);
9  oscillator.disconnect(0);
10 oscBezi = true;
11 }
12
13 // Funkce, která připojí zvuk pípnutí k přehrávání jen na určitý
```

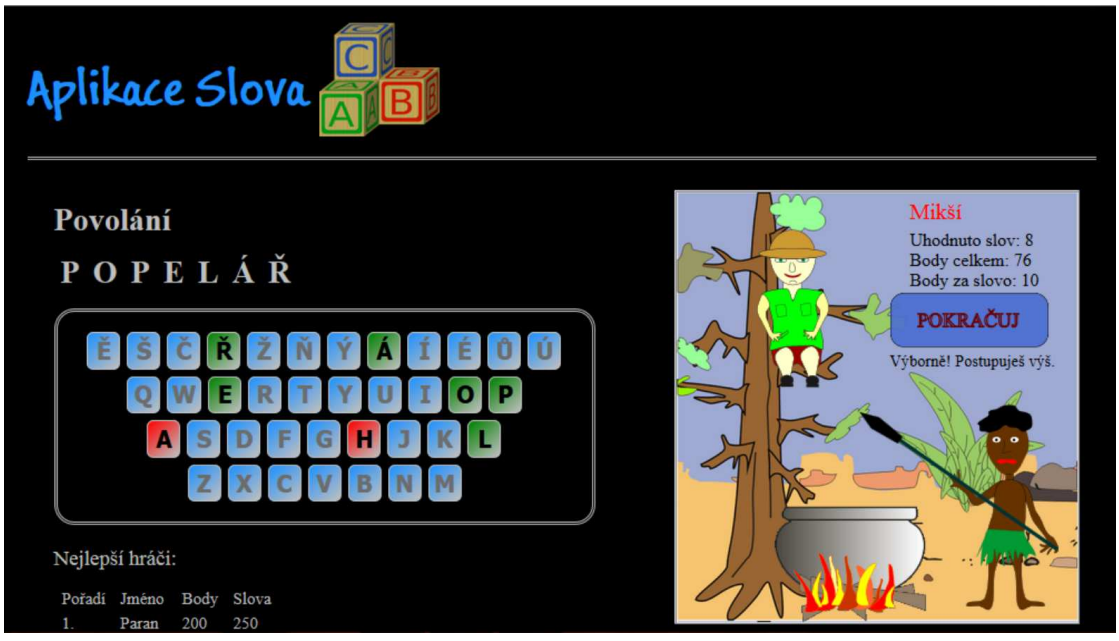
```
    okamžik
14  void Pipnuti() {
15  oscillator.connectNode(ac.destination, 0, 0); // připojení
        oscilátoru k přehrávání
16  new Timer (beepPerioda, () => oscillator. disconnect(0)); //
        odpojení oscilátoru po beepPerioda (22 milisekund)
17  }
```

Příklad 21: Vytvoření zvuku za pomoci oscilátoru přehrávající se v určité periodě

Dále byl vytvořen „gong“ nebo-li zvuk, který se opakuje každou celou hodinu. Ten už ale využívá zvuku nahraného ve dvou formátech, a to ogg a mp3 (funkce **VytvoritGong()**), jelikož některé přehrávače podporují jen jeden z nich.

4.2 Aplikace Slova

Aplikace Slova představuje hru založenou na principu známé a oblíbené „Šibenice“. Hráč má za úkol postupně doplňovat písmena hledaného slova. Aby určení nebylo tak složité, zobrazuje se nápověda, která přibližuje hledané slovo. Hra je určena pro pobavení dětem i dospělým.



Obrázek 7: Vzhled aplikace Slova

4.2.1 Grafika

Základní vzhled aplikace, který je tvořen pomocí HTML5 a CSS3 se podobá celkovému designu webové stránky, která na aplikace odkazuje.

Vlevo je umístěna klávesnice obsahující jednotlivé klávesy s písmeny. Klávesy jsou označovány zeleně při správném určení a červeně při chybném. Nad klávesnicí se zobrazuje hledané slovo nejprve pomocí podtržítek, později (při hře) se podtržítka zaměňují za správná písmena. Ještě výše se ukazuje nápověda, to znamená nadřazené slovo slovu skrytému (např. Komoda -> Nábytek). Pod klávesnicí nabíhá pořadí deseti nejúspěšnějších hráčů (podle počtu získaných bodů).

Vpravo je vytvořena animace zefektivňující celou hru. Obrázky byly vytvořeny vektorovou grafikou v programu CorelDraw. Při uhodnutí slova se postavička na stromě posune o větev výše. Pokud hráč slovo neuhodne, panáček bodnutý domorodcem se posune o jednu větev níže. Pokud hráč neuhodne vícekrát za sebou, panáček spadne do kotle, což zároveň znamená konec hry. Způsob naprogramování animace bude uveden v dalších kapitolách.

4.2.2 Ovládání

Po spuštění odkazu se hráči zobrazí formulář, který po něm žádá jméno. Po odsouhlasení se zobrazí hra. Ovládání je velice jednoduché. Hráč pomocí myši vybírá písmena patřící do hledaného slova. Po uhodnutí se vpravo na animaci objeví tlačítko Pokračovat, kterým se zobrazí nové slovo, také se přepočítají body. V opačném případě (při neuhodnutí) se ještě zobrazí neuhodnutá písmena.

Na začátku hry má hráč na kontě pět bodů za aktuálně hledané slovo. Při správném určení písmena mu bod přibývá, při špatném určení ubývá. Při uhodnutí všech písmen se počet bodů za slovo přesune do celkového počtu bodů. Pokud uživatel obdržel za aktuální slovo nula bodů, level nesplnil a panáček se posunul o větev níže. Po neuhodnutí několika slov za sebou se panáček dostává do kotle a hra končí (viz Grafika).

4.2.3 Popis funkcí aplikace

Jedná se o rozsáhlejší program, proto bylo vytvořeno více souborů s funkcemi. Do souboru main.dart jsou naimportovány mimo původní knihovny čtyři nově

vytvořené:

- funkce.dart,
- animace.dart,
- data.dart,
- zvuky.dart.

V úvodu souboru main.dart jsou deklarovány proměnné. Proměnným vytvořeným z HTML elementů jsou přiřazeny funkce, s kterými aplikace pracuje. Funkce **main()** zajišťuje spuštění celé aplikace. Dále soubor obsahuje funkce potřebné k vytvoření hráče, který zapíše své jméno do formuláře (funkce **zadejJmeno()**). Při uhodnutí (či neuhodnutí) hledaného slova se ve hře zobrazuje tlačítko s nápisem „Pokračuj“, pomocí kterého hráč může přistoupit k dalšímu slovu. Funkce **vytvorAnimBut()** díky stylům, jenž obsahuje, nastavuje vzhled tlačítka a jeho změnu rámečku při najetí myší. Další důležitá funkce zahrnutá ve zmiňovaném souboru je **VytvorKlavesnici()**, která pomocí cyklu for-each prochází jednotlivé klávesy (písmena), přiřazuje styly a indentifikátor, přičemž tedy vytváří celou klávesnici. Za zmínku stojí také funkce **KlavesniceVystup(MouseEvent me)**, která zaznamená klávesu, nad kterou se kliklo, zakáže u ní opakované stisknutí a mění její vzhled. Pokud se stisknuté písmeno rovná písmenu v hledaném slově, změní jeho styl a obarví ho zeleně. V opačném případě (při stisknutí písmena, které není obsaženo v aktuálně hledaném slově) zbarví klávesu červeně. Funkce **konecUkolu(int konec)** pak po skončení kola oznamuje hráči, zda slovo vyluštil, či nikoli, pomocí hlášky. Informace o jménu uživatele, stavu hry nebo počtu bodů poskytuje funkce **VypisHlaseni()**.

První vytvořenou knihovnou je soubor funkce.dart. Funkce **setPorovnejPismeno(String Pism)**, jak název napovídá, prochází písmena v hledaném slově a porovnává je s písmenem zadaným pomocí vytvořené klávesnice. Podle zjištění pak přidává body. Funkce **ukazReseni()** ukazuje správné řešení, pokud uživatel slovo neuhodl. Dále je v tomto souboru vytvořena třída **SlovnikC1**, kde jsou obsaženy funkce pro náhodné vybrání slova k uhodnutí. Toto slovo pak ze slovníku vymaže, aby se v jedné hře neopakovala. Třída **Hrac** pak zahrnuje informace

o hráči. Třída **Zebrik** zvýrazňuje aktuálního hráče a vypisuje jejich pořadí do tabulky. Práce s třídami se v Dartu osvědčila. Proměnné a funkce lze bez problému používat v jiných třídách a souborech.

Další knihovna nese název animace.dart. Vytváří elementy a obsahuje obrázky pro animaci. Také jsou tu nastaveny pozice postav, které se v průběhu hry pohybují v závislosti na správném (či nesprávném) řešení. Funkce nastavuje další paramety u vložených obrázků jako například **vytvorKotel()**, **vytvorCe()**, **vytvorBe()**, které nastavují rozměry obrázků, pozice a styly. Jako poslední je zde obsažena série funkcí spouštějící animace postav a kotle.

```
1  // spuštění animace postavy černocha
2  void casovacCe(){
3  const animPeriodaC = const Duration(milliseconds: 80); // doba
       změny pěti obrázků
4  CasovacCe = new Timer.periodic(animPeriodaC, (Timer t) =>
       animaceCe()); // časovač, který spouští funkci po intervalu  }
5
6  // animace postavy černocha
7  void animaceCe(){
8  tikCe += 1; // počítání proběhnutí animace
9  if (tikCe > 5) tikCe2 -= 1;
10 else tikCe2 += 1; // kolikátý obrázek animace se zobrazí
11 postavaCe.setAttributeNS('http://www.w3.org/1999/xlink','href',
       obrCe[tikCe2]); // přiřazení správného obrázku do animace
12 if (tikCe == 9) {
13 CasovacCe.cancel(); CasovacCe = null; tikCe = 0; tikCe2 = 0;} //
       ukončení animace, vypunutí časovače
14 }
```

Příklad 22: Animace

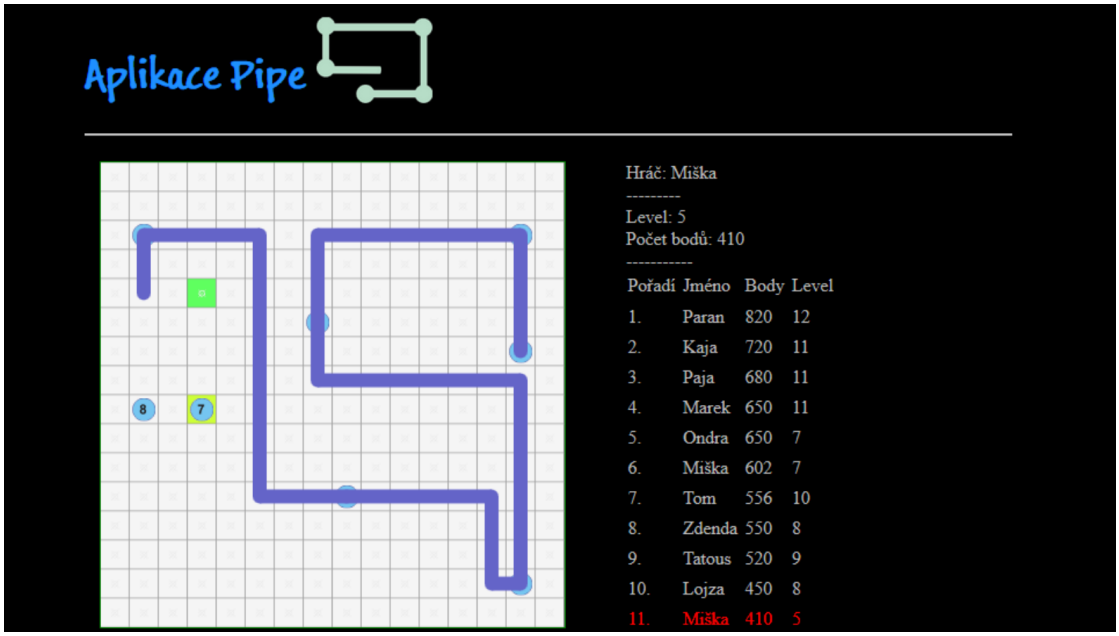
Knihovny zvuky.dart vytváří zvukové elementy a přiřazuje cestu k audio souboru. Aplikace vydává zvuk při uhodnutí slova, neuhodnutí slova a konci hry.

Poslední knihovnou je data.dart, kde je vytvořena kolekce kláves a mapa slov, které se v průběhu hry zjišťují. Mapa se skládá z kategorie (nadřazené slovo, které má při hře za úkol výběr zužit) a samotného slovo.

Pro ukládání a načítání dat (pořadí hráčů) ze souboru aplikace využívá funkcí vytvořených v PHP.

4.3 Aplikace Pipe

Aplikace Pipe představuje hru, jejímž principem je spojování vyznačených bodů čarou ve stanoveném pořadí.



Obrázek 8: Vzhled aplikace Pipe

4.3.1 Grafika

Základní vzhled aplikace, který je tvořen pomocí HTML5 a CSS3 se podobá celkovému designu webové stránky, která na aplikace odkazuje.

Hlavní část aplikace se nachází v levé části, jedná se o prostor, na kterém hráč spojuje jednotlivé očíslované body. Oblast je tvořena pomocí elementu svg, který slouží pro vektorovou grafiku. Celá oblast je rozdělena na 16x16 čtverečků. Na každém čtverečku je umístěn obrázek s očíslovaným bodem nebo s „prázdným obrázkem“.

Hra je interaktivní, to znamená, že hráč pomocí myši během hry vytváří „trasu“ vektorové čáry tvořené v rámci svg, pomocí které body spojuje.

V pravém bloku se nachází pořadí nejlepších hráčů společně se jménem, aktuálním levelem a počtem bodů právě hrajícího hráče.

4.3.2 Ovládání

Po spuštění je hráč nejprve požádán formulářem o zadání jména či přezdívky ve hře (slouží k následnému záznamu v pořadí). Po odsouhlasení tlačítkem se zobrazí samotná aplikace. Na herní ploše jsou umístěna kolečka s čísly, která má hráč za úkol pospojovat co nejkratší čarou ve správném pořadí. Pro lepší přehlednost se aktuální políčko k přejetí čarou vyznačuje žlutou barvou. Pokud se hráči podaří spojit všechny body, dostane se do dalšího levelu, ve kterém je vždy o „bod ke spoji“ více.

Čáru lze vykreslovat ve svislém a vodorovném směru.

Za každý splněný level se připočítávají body (aktuální počet bodů je vidět v pravé části hry - aktuální hráč označen červeně), při vykreslení čáry přes čtverečky se naopak body odečítají (viz cíl vytvořit co nejkratší spoj).

4.3.3 Popis funkcí aplikace

V hlavním souboru klasicky pojmenovaném main.dart se po deklaraci proměnných nachází spouštěcí funkce celé aplikace **main()**. Přihlašování do hry (zadání přezdívky) je řešeno obdobně pomocí formuláře jako v předešlé aplikaci Slova.

Aplikace opět pracuje s svg grafikou, nyní již s reakcí na konkrétní aktuální klik hráče. Funkce **posiceMysi(MouseEvent me)** zjišťuje při pohybu myši po hrací ploše, kde přesně se nachází kurzor, přičemž ukládá souřadnice konkrétního políčka do proměnné. Vykreslování čar je umožněno pouze vodorovně a svisle. Také nelze překřižovat již vytvořené linky. Další podmínkou hry je, že kontrolní body k přejetí se vyplňují postupně (podle číselného označení). Tohle vše kontroluje funkce **kontrolaPole()**. K lepší přehlednosti jsou doplněny styly, které vhodná políčka obarvují při najetí myši zeleně a nevhodná červeně. Pokud se nad políčkem klikne, spustí se funkce **zapisKlik()**, která počítá body, označí obsazená pole a spustí funkce pro další krok (obarvení dalšího kontrolního bodu žlutě, přepočítání pořadí hráčů). Hra je rozdělena na levely. S každým dalším kolem se vymaží vytvořené čáry, počet kontrolních bodů se zvyšuje a jejich pozice se náhodně mění (funkce **dalsiLevel()**).

Nejzajímavější funkcí aplikace je **kontrolujKonec(int sx, int sy)**. Tato funkce obsahuje vnořené funkce, které se spouští pomocí cyklu. Funkce vybere kolem ná-

sledujícího bodu všechna políčka, na která je možné kliknout, podle toho se následně určuje, jestli je možný další tah. Princip je ten, že se v prvním kroku kontrolují sousední pole kolem následujícího kontrolního bodu. Volná pole se ukládají do kolekce volneKeys. Vnořená funkce **volneKolem(int ix, int iy)** se následně pomocí cyklu spouští opakovaně s parametrem políček uložených v kolekci volneKeys. Když už se v průchodu nenajde žádné volné políčko, cyklus končí. Následuje funkce **jeliKonec(Souradnice poslKlik)**, která zjišťuje, zda je možné dostat se pomocí čar na další kontrolní bod. Pokud to nelze, hra končí.

Dalšími soubory programu jsou vytvořené knihovny func.dart a tridy.dart. Funkce **vytvoritPolicka(int radku, int sloupcu)** v prvně jmenovaném souboru vytváří jednotlivá políčka na hrací ploše a ukládá do nadeklarované mapy určené k dalšímu použití. Novou instanci kreslené čáry vytváří funkce **kresliCaru(Souradnice sStart, Souradnice sKonec)**.

```
1  /* Funkce kontroluje , jestli je možné projetí dalšího kontrolního
       bodu; vybere všechny možná volná políčka kolem následujícího
      kontrolního bodu; int sx , sy - souřadnice následujícího
      kontrolního bodu */
2  void kontrolujKonec(int sx , int sy) {
3  volneKeys.clear(); // vymazani kolekce z predchoziho spusteni
      funkce
4  String paklic;
5  bool konecCyklu = true;
6  List<String> volneKeys1 = new List(); // pomocna kolekce s klici
7  List<String> volneKeys2 = new List(); // pomocna kolekce s klici
8  /* vnorena funkce zjistí klíče polí kolem kontrolovaného pole; int
       ix , iy - souradnice kontrolovaneho pole uklada se do kolekce
      kontrolovaneKeys */
9  void volneKolem(int ix , int iy){
10 // kolem políčka se souřaradnicemi ix , iy se kontrolují čtyři
      sousední políčka (zda jsou volná)
11 List<String> kontrolovaneKeys = new List(); // kolekce s klici kde
       se jeste muze kliknout (kolem posledniho kliku)
12
13 int x;
14 int y;
15 int xx = ix;
16 int yy = iy;
```

```
17 // zjištění souřadnic sousedních políček (políčka okolo
       kontrolovaného bodu)
18 // souřadnice políčka se uloží do kolekce kontrolovaneKeys
19 1x = xx - 1;
20 if(x > 0 && x <= pocetSloupcu){
21 paklic = "$x:$yy";
22 kontrolovaneKeys.add(paklic);
23 }
24
25 x = xx + 1;
26 if(x > 0 && x <= pocetSloupcu){
27 paklic = "$x:$yy";
28 kontrolovaneKeys.add(paklic);
29 }
30
31 y = yy - 1;
32 if(y > 0 && y <= pocetRadku){
33 paklic = "$xx:$y";
34 kontrolovaneKeys.add(paklic);
35 }
36
37 y = yy + 1;
38 if(y > 0 && y <= pocetRadku){
39 paklic = "$xx:$y";
40 kontrolovaneKeys.add(paklic);
41 }
42
43 /* vnorena funkce ,kontroluje jestli je pole s danym klicem volne,
        kKlic je klic z kolekce kontrolovaneKeys, volne klice se
       ukladaji do kolekce volneKeys1 */
44 void kontroluj(String kKlic){
45 if (policka Svg[kKlic].volne){ // zjištění, zda je políčko volné
46 if (!volneKeys.contains(kKlic) && ! volneKeys1.contains(kKlic)) {
       // zda už kKlic není uložený v kolekci mezi volnými klíči
47 volneKeys1.add(kKlic); // uložení do kolekce volneKeys1
48 konecCyklu = false;
49 }
50 }
51 } // konec kontroluj
```

```
52 kontrolovaneKeys.forEach((s) => kontroluj(s)); // procházení
       kolekce
53 } // konec volneKolem
54 volneKolem(sx, sy); // první průchod cyklu - žluté kontrolní pole
55 volneKeys2.addAll(volneKeys1);
56 while(!konecCyklu){
57 volneKeys1.clear(); // vymazání obsahu kolekce
58 konecCyklu = true;
59 volneKeys2.forEach((s) => volneKolem(rozdelKlic(s, 'x'),
       rozdelKlic(s, 'y'))); // vybírání souřadnic x,y
60 volneKeys2.clear();
61 volneKeys2.addAll(volneKeys1);
62 volneKeys.addAll(volneKeys1);
63 }// konec while
64 }// konec kontrolujKonec
```

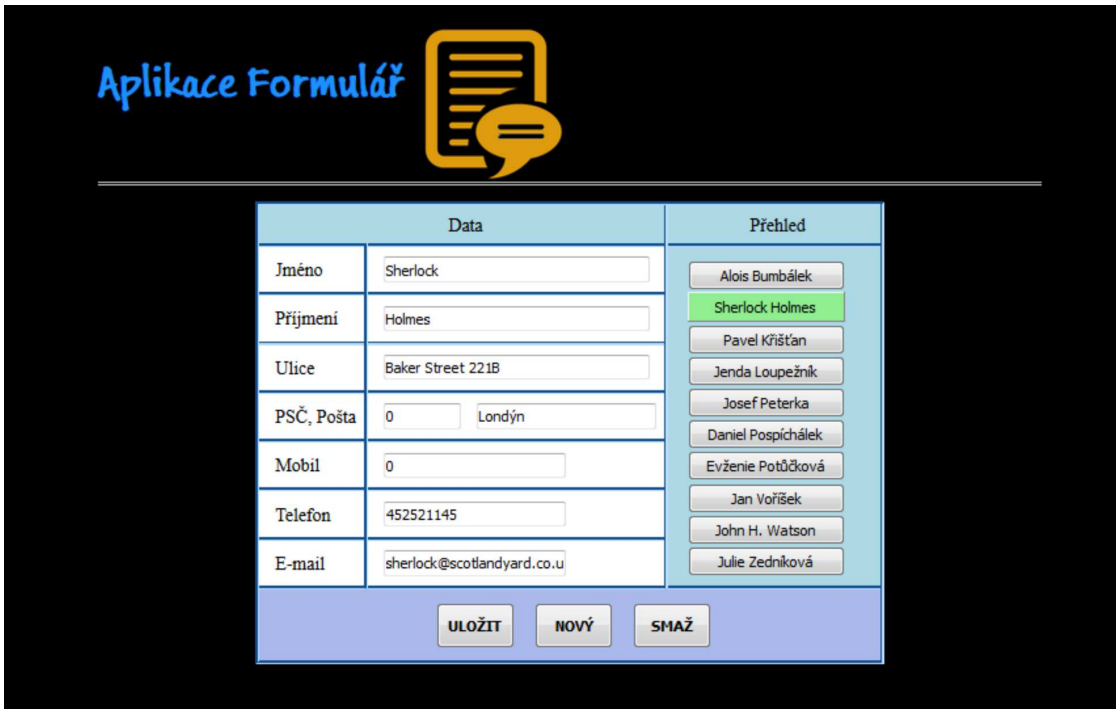
Příklad 23: Ukázka cyklu s vnořenou funkcí

Druhou knihovnou je tridy.dart, která obsahuje třídy. První je třída **Cara**, která představuje čáru podle souřadnic. Další je třída **Krouzek**, která vytváří zaoblené konce čar pro efektivnější vzhled. Třída **Policko** tvoří jednotlivé políčka hrací plochy. Třída **Souradnice** zajišťuje souřadnicové pozice jednotlivých políček. Třída **Hra** zahrnuje jméno hráče, počítání bodů ve hře a aktuální level s počtem kontrolních bodů k přejetí. Třída **Zebricek** obsahuje celkové pořadí hráčů a seřazuje je podle získaných bodů ve hře. Třída **Stupinek** představuje pořadí jednoho hráče a je využita ve třídě Zebricek.

Pro ukládání a načítání dat (pořadí hráčů) ze souboru aplikace využívá funkcí vytvořených v PHP.

4.4 Aplikace Formulář

Aplikace názorně ukazuje možnost využití Dartu jako serverového jazyka. Vytvořený formulář s širokou škálou možného využití slouží k zobrazování a ukládání dat.



Obrázek 9: Vzhled aplikace Formulář

4.4.1 Grafika

Formulář se skládá z textových políček umístěných vlevo, ve kterých se zobrazují/zapisují příslušná data. V tomto konkrétním případě se jedná o jména, adresy a další údaje o osobách. Vpravo je pak výpis všech zapsaných a uložených osob v databázi, jejichž údaje se zpětně zobrazí při výběru. Dole jsou umístěna tlačítka, která umožňují smazání záznamu či vytvoření nového.

Jednotlivé části formuláře jsou rozmístěny a ostylovány tak, aby zvyšovaly přehlednost a zlepšovaly celkový vzhled.

4.4.2 Ovládání

Uživatel využívá tlačítka s odpovídajícími popisky (Přidat, Smazat, jména osob, atd.). Data se zobrazují v textových políčkách formuláře.

4.4.3 Popis funkcí aplikace

Aplikace se skládá ze dvou samostatných částí, a to části serverové, která je vykonávána samostatně na serveru a klientské, která běží na počítači uživatele.

V klientské části jsou v úvodu souboru main.dart naimportovány knihovny:

- **dart:html** sloužící pro napojení funkcí na HTML elementy,
- **dart:js** využita pro zobrazení vyskakovacích oken alert a confirm,
- **dart:convert** pro konvertování dat při odeslání na server (formát JSON).

První důležitou funkcí po spouštěcí funkci **main()** je funkce **postServer()**, která odešle všechny data z formuláře metodou post.

```
1 void postServer(){
2 String jsonData = nactiData(); // načtení dat z formuláře
3 reqP = new HttpRequest(); // objekt zajišťující odeslání dat
4 reqP.onReadyStateChange.listen(navratPOST); // čeká na odpověď
     serveru, pak spustí navratPOST()
5 reqP.open('POST',serverUrl,async:false); // připravení dotazu
6 reqP.send(jsonData); // odeslání dat jsonData
7 }
```

Příklad 24: Odeslání dat z formuláře - klient

Funkce **navratPOST(_)** po přijetí dat ze serveru spouští funkci **getServer00()**, která ještě za pomoci funkce **navratGET00(_)** načte všechny řádky tabulky sloužící pro přehled osob (tlačítka v pravé části formuláře). Dalšími důležitými funkcemi jsou **getServer(String Idd)** a **navratGET2(_)**, které podle ID osoby načtou všechny položky formuláře konkrétní osoby při kliku na řádek se jménem v přehledu.

```
1 void navratGET2(_) {
2 if (reqG.readyState == HttpRequest.DONE && reqG.status == 200) {
     // když se regG (dotaz) zpracuje správně a správně proběhl
3 Map data = new Map(); // vytvoření mapy , data vrácená ze serveru
4 data['0'] = '0';
5 String jsonString = reqG.responseText; // vytvoření proměnné (
     vrácená data, která server poslal zpět)
6 if (jsonString.length > 0)
```

```
7  data = JSON.decode(jsonString); // když se něco vrátí ->
       dekodování dat - změna formátu a uložení do mapy
8  // vypsání políček formuláře
9  querySelector('#jmeno').value = data['Jmeno'];
10 querySelector('#prijmeni').value = data['Prijmeni'];
11 querySelector('#ulice').value = data['Ulice'];
12 querySelector('#psc').value = data['Psc'];
13 querySelector('#posta').value = data['Posta'];
14 querySelector('#mobil').value = data['Mobil'];
15 querySelector('#telefon').value = data['Telefon'];
16 querySelector('#email').value = data['Email'];
17 querySelector('#id').value = data['Id'];
18 }
19 }
```

Příklad 25: Načtení dat podle Id a vložení do formuláře - klient

Již zmiňovaný přehled osob je vytvořen pomocí tlačítek, která reagují na kliknutí uživatele podle funkcí uvedených výše. Samotná tlačítka se vytváří ve funkci **vytvorButtons(Map data)**, kde je procházena mapa dat a následně jsou vytvářeny nové elementy button, kterým jsou přiřazeny funkce a styly.

Poslední částí na straně klienta jsou funkce přiřazené tlačítkům v dolní části formuláře. Tlačítko Ulož odesílá data vepsaná do políček formuláře na server (funkce **ulozData(MouseEvent me)**).

```
1  void ulozData(MouseEvent me){
2  me.preventDefault(); // vymaže defaultní funkci u tlačítka
3  postServer(); // funkce, která odesílá data na server
4  if (butStisknuty != null) {
5  String butId = butStisknuty.getAttribute('id'); // zjištění id
       buttonu butStisknuty
6  getServer(butId); // znovunačtení dat ze serveru
7  }}
```

Příklad 26: Odeslání dat na server - klient

Tlačítko Smaž pak maže pomocí funkce **smazData(MouseEvent me)** právě zobrazené informace o osobě. Orientace probíhá podle id „akce“ v políčku formuláře, kterému je v tomto případě přiřazena hodnota DEL. Na straně serveru je

pak pomocí řídící struktury case akci Del přiřazena funkce s odpovídajícím sql dotazem.

Funkce **nactiData()** vytváří datový řetězec ve formátu JSON, který slouží pro odeslání dat na server.

Druhou částí aplikace jsou funkce zpracovávané serverem, který přistupuje k databázi. V úvodu souboru main.dart jsou naimportovány další důležité knihovny:

- serverová knihovna **dart:io**,
- **dart:async** pro vytváření asynchronních funkcí.

Prvním úkolem serveru je připojení se k databázi pomocí již vytvořené funkce ConnectionPool a informacemi o IP adrese klienta, portu, jménu uživatele (případně i hesla) a názvu databáze.

```
1 ConnectionPool pripojeniDB = new ConnectionPool(host:'localhost',
    port:3306 ,user:'root', db: 'test');
```

Příklad 27: Připojení k databázi - server

První asynchronní funkcí je spouštěcí funkce **main()**, která běží a zároveň čeká na přijetí celého dotazu od klienta. Následně přijaté dotazy třídí podle toho, zda jsou metody Post nebo Get.

- Pokud je dotaz metody POST, zpracuje se funkcí **dotazPost()**, která je rovněž asynchronní. Data dotazu jsou načtena a dekódována z formátu JSON určenému k posílání dat do String, se kterým pracuje server. Dále tato funkce rozlišuje switchem přijaté dotazy podle hodnoty atributu akce (ADD, UPD, DEL) a přiděluje odpovídající funkce.
  - ADD - asynchronní funkce **Future dbADD() async**, která přidává jeden řádek do tabulky databáze pomocí klasického sql dotazu pro vložení.
  - UPD - asynchronní funkce **Future dbUPD() async**, která aktualizuje záznam v jedné řádce tabulky databáze sql dotazem pro aktualizaci.
  - DEL - asynchronní funkce **Future dbDEL() async**, která smaže jeden řádek tabulky.

```
1  Future dbDEL() async{
2  var completer = new Completer();
3  // dotaz, který smaže záznam v databázi podle id
4  String dotazSQL = "DELETE from adresar WHERE Id =\'${
       mojeData['Id']}\';";
5  ftt1 = pripojeniDB.query(dotazSQL); // vykonání SQL příkazu
6  await ftt1.then ((vysledek) async{
7  completer.complete(vysledek);
8  });
9  odesliData("Zapis v tabulce smazan"); // klient si může
       zobrazit informaci o provedení příkazu
10 return completer.future;
11 }
```

Příklad 28: Smazání řádku tabulky databáze - server

- Pokud je dotaz metody GET, zpracuje se funkcí **dotazGet()**, která zjistí hodnotu q. Pokud je hodnota q rovna nule, znamená to načtení všech dat databáze. Každé jiné číslo je Id některé osoby, způsobuje načtení jedné řádky databáze.

Při požadavku na data v databázi je vytvořena asynchronní funkce pojmenovaná **nactiData(String dotazSQL)**, která čeká až se načtou všechny data z databáze a až pak je uloží do příslušné proměnné. Před odesláním dat klientovi je opět nutné dekódovat do formátu JSON funkcí **dekodujData(vysledek)**, která dekóduje všechny data pro výpis do formuláře nebo funkcí **dekodujData2(vysledek)**, která dekóduje jen jméno a příjmení použitelné do přehledu osob. Samotné odeslání dat zajišťuje funkce **odesliData(String dataPoslana)**, kde je důležité nastavení hlavičky obsahující nezbytné informace o datech (kódování, odblokování JavaScriptu, atd.).

4.5 Testování vytvořených aplikací

Funkčnost všech webových aplikací je zkoušena a testována v prohlížečích:

- Mozilla Firefox,
- Google Chrome,
- Internet Explorer,
- Edge,
- Opera.

Vzhled vytvořený pomocí kaskádových stylů a HTML5 funguje na všech prohlížečích. Komplikace nenastaly ani při ověření funkčnosti aplikací Hodiny a Slova. Problém se naskytl až u aplikace Pipe, kde uživatel vytváří čáru pomocí myši a souřadnic. Každý prohlížeč využívá k zjištění souřadnic jinou funkci a jiný počáteční bod.

Testování Dart funkcí offset, client, layer, page a screen určených k zjištění souřadnic kurzoru je znázorněno v následující tabulce. Hodnoty souřadnic jsou zjištěny pro levý horní a pravý dolní roh vektorového elementu v aplikaci Pipe.

		Edge		IE		Firefox		Chrome		Opera	
		LH roh	PD roh	LH roh	PD roh	LH roh	PD roh	LH roh	PD roh	LH roh	PD roh
offset:	načteno	**0, 0**	**800, 800**	**0, 0**	**800, 800**	**0, 0 do 50, 50**	**800, 800**	**0, 0**	**800, 800**	**0, 0**	**800, 800**
	posunutý	**0, 0**	**800, 800**	**0, 0**	**800, 800**	**0, 0**	**800, 800**	**0, 0**	**800, 800**	**0, 0**	**800, 800**
client:	načteno	100, 100	900, 900	100, 100	900, 900	100, 100	900, 900	100, 100	900, 900	100, 100	900, 900
	posunutý	100, 0	900, 800	100, 0	900, 800	100, 0	900, 800	100, 0	900, 800	100, 0	900, 800
layer:	načteno	100, 100	900, 900	100, 100	900, 900	**0, 0**	**800, 800**	0, 0	800, 800	0, 0	800, 800
	posunutý	100, 100	900, 900	100, 100	900, 900	**0, 0**	**800, 800**	0, 0	800, 800	0, 0	800, 800
page	načteno	**100, 100**	**900, 900**	**100, 100**	**900, 900**	**100, 100**	**900, 900**	**100, 100**	**900, 900**	**100, 100**	**900, 900**
	posunutý	**100, 100**	**900, 900**	**100, 100**	**900, 900**	**100, 100**	**900, 900**	**100, 100**	**900, 900**	**100, 100**	**900, 900**
screen:	načteno	100, 177	900, 980	105, 160	950,1000	100, 170	900, 970	100, 162	900, 963	100, 170	900, 973
	posunutý	100, 77	900, 880	106, 56	950, 900	100, 70	900, 870	100, 62	900, 864	100, 70	900, 870

Tabulka 2: Souřadnice v různých prohlížečích za použití odlišných funkcí

První problém při zjišťování souřadnic v Dartu může nastat při využití funkce *screen*, která počítá souřadnice od začátku obrazovky, a to i po posunutí stránky

nebo-li tzv. rolování. Vhodnou se nejeví ani funkce *page*, která sice funguje stejně na všech zmíněných prohlížečích, ale souřadnice se počítají od začátku stránky, tudíž je k elementu, který může být umístěn např. uprostřed stránky, připočtena příslušná hodnota.

Obrázek 10: Zjištění souřadnic - rozdíl mezi načtenou a rolovanou stránkou

Nejlepší variantou je v tomto případě funkce *offset* a *layer*, které reagují přímo na klíčový element. Jelikož ale funkce offset špatně spolupracuje s prohlížečem Mozilla Firefox a hodnoty jsou zde posunuté, což způsobuje posunutí vybraného políčka o řadu pixelů vůči kurzoru myši, je nutné nastavit obě funkce pro konkrétní prohlížeče.

Principem funkce, která rozlišuje prohlížeče je rozdělení obsahu hlavičky user-agent, kterou prohlížeče posílají jako svou identifikaci a následného zjištění klíčového slova, tedy názvu prohlížeče [20].

```
1  // Detekce prohlížeče
2  void BrowserDetect(){
3  String UserAg = window. navigator.userAgent.toString();
4  int ps = UserAg.lastIndexOf(')');
5  UserAg = UserAg.substring(ps);
6  if (UserAg.contains('Firefox ')) {Prohlizec = "Firefox"; }
7  else if (UserAg.contains('OPR')) {Prohlizec = "Opera";}
8  else if (UserAg.contains('Safari')) {Prohlizec = "Chrome"; }
9  else {Prohlizec = "IE"; }
10 }
```

Příklad 29: Detekce prohlížeče

Poté už stačí jen pomocí další funkce k proměnné, ve které je uložen název používaného prohlížeče, přiřadit odpovídající funkci Dartu k zjištění souřadnic.

```
1  // Funkce - pohyb myši nad políčkem
2  // zjišťuje, nad kterým políčkem je kurzor a ukládá do proměnné
3  void posiceMysi(MouseEvent me){
4  me.preventDefault();
5  int mx , my;
6  if(Prohlizec == "Firefox"){
7  mx = me.layer.x;
8  my = me.layer.y;
9  }
10 else {// ostatni prohlizece
11 mx = me.offset.x;
12 my = me.offset.y;
13 }
```

Příklad 30: Zjištění souřadnic políčka pod kurzorem myši

U aplikace Formulář se nenačítala aktuální data z databáze v prohlížeči Internet Explorer. Problém byl způsoben ukládáním a následným načítáním dat z cache. Aktualizované položky se tedy stále zobrazovaly ve stejné podobě jako před upravením. Do funkce odesílající data hlavičky byla přidána informace o tom, aby se odpověď neukládala do cache.

```
1  request.response
2  ..headers.add("cache-control", "no-cache");
```

Příklad 31: Nastavení hlavičky - neukládání do cache - server

5 Závěr

Bakalářská práce se zabývala programovacím jazykem Dart, který se využívá na webových stránkách. V teoretické části byly představeny informace o vzniku jazyka, syntaxi, knihovnách, vývojovém prostředí, prohlížeči apod. V praktické části byla vytvořena sada aplikací, na kterých byly předvedeny různé způsoby využití Dartu jako je použití canvasu, vektorové grafiky, zvuku, interaktivity či formulářů.

Dart je velice slibná, nová alternativa, která se dá využít k celé řadě vylepšení webu.

Jelikož při vývoji nového programovacího jazyka trvá vždy několik let než se rozšíří, úplně zpřístupní a začne být běžně podporován a používán bez omezení, ještě si asi na plné využití Dartu chvíli počkáme. Mezitím si ale získá více příznivců, čímž se rozšíří i množství knihoven a nápadů, kterých bude možné využít.

Literatura a zdroje

[1] Dartlang [online]. 2015 [cit. 2015-03-16]. Dostupné z: https://www.dartlang.org/

[2] Zdroják [online]. 2014 [cit. 2015-03-22]. Dostupné z: http://www.zdrojak.cz/

[3] THAU,. Velký průvodce JavaScriptem: tvorba interaktivních webových stránek v praxi. 1. vyd. Praha: Grada, 2009, 516 s. Profesionál. ISBN 978-80-247-2211-5.

[4] KOPEC, David. Dart for Absolute Beginners. USA: Apress, 2014. ISBN 978-1-4302-6482-8.

[5] CLARK, Richard. Beginning HTML5 and CSS3: the Web evolved : next generation Web standards. New York: Distributed to the book trade worldwide by Springer Science+Business Media, 2012, 600 p. Expert's voice in Web development. ISBN 1430228741.

[6] BALBAERT, Ivo a Dzenan RIDJANOVIC. Learning Dart. Birmingham: Packt Publishing, 2013, 388 s. ISBN 978-1-84969-743-9.

[7] SOUKUP, Tomáš. Dart 1.0. Jazyk pro web, který chce nahradit Javascript Více na: http://www.zive.cz/bleskovky/dart-10-jazyk- pro-web-ktery-chce-nahradit-javascript/sc-4-a-1713 24/de- fault.aspx#utm medium=selfpromo&utm source=zive&utm campaign= copylink. Zive.cz [online]. 2013 [cit. 2015-08-03]. Dostupné z: http://www.zive.Itebleskovky/dart-10-jazyk-pro-web-ktery-chce-nahradit- javascript/sc-4-a-171324/default.aspx

[8] Wikipedia. Lars Bak [online]. 2014 [cit. 2015-08-03]. Dostupné z: https://en.wikipedia.org/wiki/Lars Bak %28computer programmer%29

[9] Jak psát web [online]. [cit. 2015-08-03]. Dostupné z: http://www.jakpsatweb.cz

[10] Jak se naučit programovat [online]. 2004 [cit. 2015-08-04]. Dostupné z: http://jaksenaucitprogramovat.py.cz/

[11] IT Network [online]. 2015 [cit. 2015-08-04]. Dostupné z: http://www.itnetwork.cz/

[12] Dart: Lehký úvod pro programátory C# [online]. 2011 [cit. 2015-08-04]. Dostupné z: http://blog.kolman.cz/2011/12/dart-lehky-uvod-pro-programatory-c.html

[13] Programujte.com [online]. 2003 [cit. 2015-08-10]. Dostupné z: http://programujte.com/

[14] HTML5 [online]. 2011 [cit. 2015-08-11]. Dostupnné z: http://www.html5.cz/

[15] Sallyx.org [online]. 2015 [cit. 2015-08-12]. Dostupné z: http://www.sallyx.org/sally/c/c09.php

[16] VORÁČEK, Jan. Skriptovací jazyky pro tvorbu webových aplikací . Pardubice, 2013. Univerzita Pardubice.

[17] 10 reasons why Dart is cooler than JavaScript. Christian Grobmeier [online]. Německo, 2012 [cit. 2016-01-25]. Dostupné z: http://www.grobmeier.de/10-reasons-why-dart-is-cooler-than-javascript-03012012.html

[18] Rozdíl mezi null a undefined v JavaScriptu. David Majda [online]. 2009 [cit. 2016-01-25]. Dostupné z: http://majda.cz/blog/2624

[19] Dart vs JavaScript - Are they compiled or interpreted languages? Stackoverflow [online]. 2013 [cit. 2016-01-25]. Dostupné z: http://stackoverflow.com/questions/17601984/dart-vs-javascript-are-they-compiled-or-interpreted-languages

[20] Hlavička User-Agent. Je čas [online]. 2015 [cit. 2016-01-30]. Dostupné z: http://jecas.cz

Seznam obrázků

Seznam tabulek

Seznam příkladů

Přílohy

1. CD se zdrojovými kódy webové stránky a jednotlivých aplikací, dále plné znění bakalářské práce v PDF