

# Učební pomůcka pro předmět Webové technologie

Michal Vrška

---

Bakalářská práce  
2016



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2015/2016

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Michal Vrška**  
Osobní číslo: **A13851**  
Studijní program: **B3902 Inženýrská informatika**  
Studijní obor: **Informační a řídicí technologie**  
Forma studia: **kombinovaná**

Téma práce: **Učební pomůcka pro předmět Webové technologie**  
Téma anglicky: **Lecture Notes for the Web-based Technologies Subject**

Zásady pro vypracování:

1. Uvedte základní principy práce se značovacími jazyky HTML5.
2. Vytvořte soubor laboratorních úloh od jednoduchých ke složitějším na práci s HTML5.
3. Ukažte principy práce s kaskádovými styly CSS.
4. Vytvořte soubor laboratorních úloh na demonstraci práce s CSS.
5. Analyzujte důvody pro používání scriptovacího jazyka javascript s důrazem na pojmy frontend a backend.
6. Přibližte studentům javascriptovou knihovnu jQuery a její použití v praxi.
7. Zkoumejte nyní nejčastěji používané JavaScript frameworky a demonstруйте jejich využití v praxi.
8. Vytvořte tuto učební pomůcku ve formě webové stránky.

Rozsah bakalářské práce: -  
Rozsah příloh: -  
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. JQuery [online]. Dostupné z: <https://jquery.com/>
2. Code Academy [online]. Dostupné z: <https://www.codecademy.com>
3. HTML Dog [online]. Dostupné z: [www.htmldog.com](http://www.htmldog.com)
4. Tutorials Point [online]. Dostupné z: [www.tutorialspoint.com](http://www.tutorialspoint.com)
5. W3 Schools [online]. Dostupné z: <http://www.w3schools.com/>
6. BAŠE, Ondřej. JQuery pro neprogramátory: průvodce využitím knihovny jQuery UI. 1. vyd. Brno: Computer Press, 2012, 368 s. ISBN 9788025137505
7. SCHAFER, Steven M. HTML, XHTML a CSS: bible. 1. vyd. Praha: Grada, 2009, 647 s. ISBN 9788024728506.
8. HOGAN, Brian P. HTML5 a CSS3: výukový kurz webového vývoje. Vyd. 1. Brno: Computer Press, 2011, 272 s. ISBN 9788025135761
9. CASTRO, Elizabeth a Bruce HYSLOP. HTML5 a CSS3: názorný průvodce tvorbou WWW stránek. 1. vyd. Brno: Computer Press, 2012, 439 s. ISBN 9788025137338

Vedoucí bakalářské práce: **Ing. Petr Šilhavý, Ph.D.**  
Ústav počítačových a komunikačních systémů  
Datum zadání bakalářské práce: **19. února 2016**  
Termín odevzdání bakalářské práce: **27. května 2016**

Ve Zlíně dne 19. února 2016



doc. Mgr. Milan Adámek, Ph.D.  
*děkan*



prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*

### **Prohlašuji, že**

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

### **Prohlašuji,**

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

.....  
podpis diplomanta

## **ABSTRAKT**

Tato práce má za úkol přiblížit čtenáři - nováčkovi problematiku vývoje webových stránek za použití základních technologií. Těmito technologiemi jsou hypertextový značkovací jazyk HTML, kaskádové styly CSS a skriptovací jazyk JavaScript. Toto trio představuje základ každé moderní webové stránky a je nedílnou součástí sady znalostí vývojáře webových stránek. Přečtením této práce by měl čtenář nabýt základní orientace na poli frontendového vývoje.

Klíčová slova: internet, webová stránka, webová aplikace, frontendový vývoj, HTML, CSS, JavaScript

## **ABSTRACT**

This paper serves the purpose of enlightening the reader – novice on the subject of webpage development using basic technologies. These are namely the HTML Hypertext Markup Language, CSS Cascading Style Sheets and the JavaScript scripting language. This trio resembles the core of every modern webpage and is a crucial part of the web developer's skillset. Reading this paper the reader should gain basic knowledge in the field of frontend web development.

Keywords: internet, webpage, web application, frontend development, HTML, CSS, JavaScript

Chci tímto poděkovat především mým rodičům za to, že nade mnou nikdy nezlomili hůl a byli mi vždy oporou. Bez jejich pomoci bych tuto práci nemohl psát.

Taktéž chci poděkovat vedoucímu své práce, panu Ing. Petru Šilhavému, Ph.D. za věcné rady a připomínky a za projevenou ochotu při vedení mé práce.

V neposlední řadě děkuji všem kantorům, se kterými jsem měl tu čest se setkat při mém studiu na UTB.

“Do what you can, with what you have, where you are.”

-Theodore Roosevelt

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

# OBSAH

<b>ÚVOD.....</b>	<b>8</b>
<b>I. TEORETICKÁ ČÁST .....</b>	<b>9</b>
<b>1 ZNAČKOVACÍ JAZYK HTML .....</b>	<b>10</b>
1.1 HTML5 .....	10
1.2 MARKUP – ZNAČKOVÁNÍ .....	12
1.2.1 ZÁKLADY ZNAČKOVÁNÍ .....	12
1.2.2 ZNAČKOVÁNÍ V HTML5 .....	13
1.3 STRUKTURA HTML DOKUMENTU .....	14
1.4 SPECIÁLNÍ ZNAKY .....	16
1.5 TAGY .....	16
1.5.1 ZÁKLADNÍ TAGY STRUKTUR .....	16
1.5.2 TAGY V HTML5 .....	17
1.6 FORMULÁŘE.....	19
1.6.1 TAG <i>INPUT</i> .....	20
1.6.2 TAGY <i>SELECT</i> A <i>OPTION</i> .....	20
1.6.3 TAG <i>LABEL</i> .....	20
1.7 HTML NAPŘÍČ PROHLÍŽEČI.....	21
1.7.1 KOMENTÁŘE.....	21
<b>2 KASKÁDOVÉ STYLY CSS.....</b>	<b>23</b>
2.1 STYLOVÁNÍ V HTML .....	23
2.2 TAG <i>STYLE</i> A STYLY .....	23
2.3 KASKÁDOVÉ STYLY .....	25
2.4 SYNTAX.....	26
2.5 SELEKTORY TAGŮ.....	26
2.6 SELEKTORY ID.....	26
2.7 RELATIVNÍ SELEKTORY.....	27
2.8 SELEKTORY TŘÍD.....	28
2.9 DIMENZOVÁNÍ A BOX MODEL .....	29
2.10 POZICOVÁNÍ V CSS .....	29
2.11 ZOBRAZENÍ V CSS.....	30
2.12 FLOAT V CSS .....	30
2.13 LADĚNÍ STYLŮ V PROHLÍŽEČI .....	30
<b>3 JAVASCRIPT.....</b>	<b>31</b>
3.1 VKLÁDÁNÍ JAVASCRIPTU DO HTML .....	31
3.2 PROMĚNNÉ A TYPY .....	33
3.3 ROZHODOVÁNÍ A PODMÍNKY .....	34
3.4 FUNKCE .....	36
3.5 RÁMCE (SCOPES).....	37

3.6	ZAPOUZDŘENÍ (CLOSURES) .....	38
3.7	OBJEKTY A POLE.....	38
3.8	SMYČKY .....	40
<b>4</b>	<b>JQUERY.....</b>	<b>41</b>
4.1	ZÁKLADY JQUERY.....	41
4.2	UDÁLOSTI.....	42
4.3	ZABALENÉ SADY .....	44
4.4	MODIFIKACE DOKUMENTU .....	44
4.5	SÍŤOVÉ POŽADAVKY SKRZ JQUERY.....	45
4.6	ZPRACOVÁNÍ FORMULÁŘŮ .....	47
<b>5</b>	<b>JAVASCRIPTOVÉ FRAMEWORKY A KNIHOVNY .....</b>	<b>48</b>
5.1	ANGULAR - GOOGLE .....	48
5.2	REACT - FACEBOOK .....	49
5.3	BOOTSTRAP - TWITTER .....	50
<b>II</b>	<b>PRAKTICKÁ ČÁST .....</b>	<b>51</b>
<b>6</b>	<b>ÚVOD .....</b>	<b>52</b>
<b>7</b>	<b>HTML.....</b>	<b>53</b>
7.1	ZÁKLADNÍ HTML DOKUMENT .....	53
7.2	FORMULÁŘ PRO VYHLEDÁVÁNÍ.....	55
<b>8</b>	<b>CSS.....</b>	<b>57</b>
<b>9</b>	<b>JAVASCRIPT A JQUERY .....</b>	<b>62</b>
<b>10</b>	<b>BOOTSTRAP .....</b>	<b>65</b>
10.1	NAVIGAČNÍ PANEL.....	66
10.2	JUMBOTRON.....	66
10.3	FORMULÁŘ.....	67
10.4	ZOBRAZENÍ VRACENÝCH VÝSLEDKŮ .....	68
10.5	PATIČKA .....	68
<b>ZÁVĚR .....</b>	<b>70</b>	
<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>71</b>	
<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>	<b>73</b>	
<b>SEZNAM OBRÁZKŮ .....</b>	<b>74</b>	
<b>SEZNAM PŘÍLOH.....</b>	<b>77</b>	

## ÚVOD

Není pochyb, že výměna informací a propojení obyvatel této planety přes globální síť je jak věci aktuální, tak neodmyslitelnou s výhledem do budoucna. Cestu jak přiblížit lidi, služby, idey lidem nabízí webové stránky a aplikace.

Internet a webové stránky jsou nedílnou součástí moderní doby. „Co není na internetu, jakoby neexistovalo“, těmito slovy lze popsat důvod, proč je o webové stránky a jejich tvorbu stále větší zájem. Vývoj webových stránek je dnes doménou desetitisíců profesionálů, vytvářejících působící prezentace, kterými dokáží v masách uživatelů svých stránek vzbuzovat pocity, či psychologickými principy ovlivňovat jejich rozhodování, například o nákupu nových bot či výběru dovolené.

Technologie značkovacího jazyka HTML, kaskádových stylů CSS a scriptovacího jazyka JavaScript jsou denním chlebem vývojáře webových stránek. Jejich úplné ovládnutí není jednoduché už jen proto, že se jako každá moderní technologie neustále vyvíjí a zdokonalují. Tyto technologie jsou základním stavebním kamenem každé moderní webové stránky a jsou používány také ke tvorbě nejrůznějších mobilních a webových aplikací, ke kterým uživatelé mohou přistupovat přes prohlížeče na svých zařízeních nejrůznějších rozměrů.

Jelikož existuje mnoho různých názorů, nelze s jistotou říct, která je nejlepší, nejvhodnější nebo nejuniverzálnější cesta k dobře fungující webové stránce. A stejně tolik jako názorů existuje i možných řešení webových stránek a webových aplikací. Najít si to nejsympatičtější z nich už je na každém vývojáři zvlášť.

## **I. TEORETICKÁ ČÁST**

# 1 ZNAČKOVACÍ JAZYK HTML

HTML je značkovací jazyk, struktura, jak by měla webová stránka vypadat. HTML je jazyk, který vznikl mnoha lety koncesí. Dá se říci, že je to standard vystavěný lídry trhu. Ze začátku bylo mnoho rozhodnutí o HTML děláno společnostmi internetových prohlížečů. Když úspěšné prohlížeče implementovaly určité funkce, bylo složité ustupovat, protože lidé tvořili kód proti těmto prohlížečům a měli aktivní webové stránky, které fungovaly. Ne vždy je patrné jak HTML funguje pod pokličkou ale pochopení procesu, jak tento jazyk vznikal, bude pro programátora při znalosti pravidel tohoto jazyka dostačující.

## 1.1 HTML5

Pod označením HTML5 se často ukrývá mnoho technologií. Technologie jako CSS3, funkce prohlížečů jako databáze indexů, souborové API a dotykové akce, někdy dokonce webová úložiště nebo webové SQL. Tyto technologie ale vyžadují porozumění základním principům HTML. V následujících kapitolách tedy bude pozornost věnována samotnému značkovacímu jazyku, jazyku ostrých závorek, který webové prohlížeče mohou interpretovat a vykreslit na obrazovku.

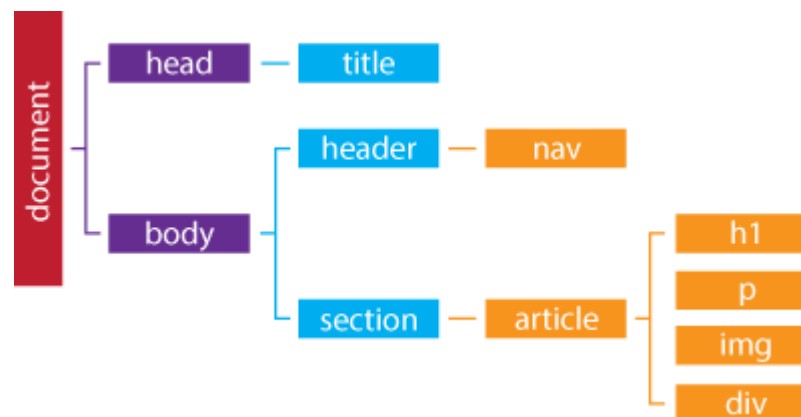
HTML je zkratka pro Hypertext Markup Language (hypertextový značkovací jazyk), jehož první vydání připadá na rok 1993. Je tedy vystaven společnosti už přes dvě dekády a je to jazyk, který dovoluje definovat různé části webové stránky, z nichž některé jsou pro uživatele viditelné, jiné ne. Je to derivát jazyka SGML (Standard Generalized Markup Language), který vymezil všeobecné značkovací jazyky. To je také důvod, proč HTML používá strukturu ostrých závorek.

```
<html>
  <head>
  </head>
  <body>
    <h1>Hello World</h1>
  </body>
</html>
```

Obr. 1-1 jednoduchý HTML dokument

Na Obr. 1-1 je zobrazen velmi jednoduchý HTML dokument. Začíná i končí značkami – tagy, obklopenými ostrými závorkami. Tyto tagy mají hierarchální strukturu, která by vždy měla začínat nejsvrchnějším *html* obalem. Ten obsahuje hlavičku *head*, která zahrnuje některá metadata o dokumentu, a tělo *body*, které reprezentuje, co uživatel uvidí.

HTML spoléhá na složku zvanou Document Object Model, zkráceně DOM. Ten reprezentuje hierarchii prvků HTML stránky. Zatímco tato hierarchie popisuje stránku, při pohledu na značkování HTML je vidět, že DOM je zároveň grafem prvků představujících objekty, které může uživatel vidět a na něž může působit. Zároveň je ale i sadou objektů, se kterými může pracovat programátor.



Obr. 1-2 Document Object Model

Dokument z Obr. 1-2 je složen ze sekcí *head* a *body*. *Head* může zahrnovat další objekty, jako například sekci *title*. *Body* může taktéž obsahovat další objekty a ty zase další, podle toho, jak jsou uspořádány v hierarchii kódu HTML.

## 1.2 Markup – Značkování

### 1.2.1 Základy značkování

Značkování v praxi znamená vytvoření hierarchie prvků.

```
<utb>Hello</utb>
```

Obr. 1-3 označovaný prvek

Každý prvek má své jméno, které je obklopeno dvojicí ostrých závorek. Toto značení znamená začátek struktury. Pro ukončení se použije stejné značení jako pro začátek, s tím rozdílem, že před jméno prvku přibude lomítko, které je indikátorem ukončující části prvku. Vše, co se nachází mezi začátkem a koncem elementu, je jeho obsahem. Existují také elementy nazývané sourozenci, tedy sestry nebo bratři daného elementu. Jedná se o prvky na stejné úrovni.

```
<utb>Hello</utb>  
<sister></sister>  
<brother />
```

Obr. 1-4 sourozenecké prvky

Prvek *sister* v Obr. 1-4 nemá žádný obsah, což je naprosto validní značkování. V případě, že prvek obsah nemá, nebo ho dokonce mít nemůže, lze použít lomítko, kterým je značen konec prvku, již u počátečního označení elementu, jak vidíme u prvku *brother*. Je to jednoduchý atomický prvek. Ve většině případů nebude mít svůj vlastní libovolný obsah, ale bude zahrnovat určitou hierarchii nebo vztah rodič–potomek, jako například na Obr. 1-5.

```
<parent>  
  <child />  
</parent>
```

Obr. 1-5 vztah rodič–dítě

V tomto případě je *parent* rodičem prvku *child*, který je dítětem, potomkem prvku *parent*. Tímto způsobem je možné vytvářet hierarchie prvků. Jeden rodič má potomka, ten má své potomky, ti zase své a tak dále.

### 1.2.2 Značkování v HTML5

Výše byly zmíněny základy značkování jako takového. Na ty je ale nutné nahlížet v kontextu HTML, tedy hypertextového značkovacího jazyka.

```
<div>
  <img />
  <p>Hello <b>World</b>, Goodbye</p>
</div>
```

Obr. 1-6 značkování HTML

Na Obr. 1-6 vidíme značku – tag nejvyšší úrovně typu *div*, který *div* je jedním ze základních v jazyce HTML a označuje část webové stránky. Jeho potomky mohou být například obrázek *img* nebo paragraf *p*. HTML tedy může mít atomické prvky (uvedený *img* je atomický vždy, nikdy nemá svůj vlastní obsah – je třeba podotknout, že ne všechny HTML tagy podporují samouzavíratelnost, například tagy *p*, *script* a *div* potřebují kompletní uzavírací tag) a také odstavce *p*, které mohou mít obsah libovolný. Obsah prvku *p* bude na webové stránce formátován jako odstavec. V případech s obsahem mohou být i další jeho části zabaleny do svého vlastního elementu. V Obr. 1-6 je takto označeno slovo *World*, zabalené do tagu *b*, čímž je indikováno formátování daného slova. V tomto případě to znamená, že slovo *World* bude napsané tučně (bold). I obsah HTML elementů je tedy dále označován, tagován, aby prohlížeč věděl, co má s danými prvky provést.

#### 1.2.2.1 Atributy

Elementy mohou také obsahovat parametry nazývané atributy. Jsou to informace, typicky pár jméno–hodnota, které jsou uvedeny uvnitř tagů elementů samotných. Tyto informace slouží k identifikaci prvků a určení jejich vlastností.

```
<div id="top">
  
  <p>Hello <b>World</b>, Goodbye</p>
</div>
```

Obr. 1-7 atributy elementů

Na Obr. 1-7 má tag *div* atribut typu identifikátor *id*, jehož filosofií je být pro každý prvek na stránce jednoznačný. Má hodnotu *top*. Element *img* má atribut zdroje (source) *src*, který odkazuje na zdrojový soubor daného obrázku. Jednotlivé elementy mohou obsahovat i více atributů, pokud je to nutné. Například tagy *img* by měly obsahovat atribut *alt*, což je

alternativní reprezentace obrázku pro uživatele s vypnutým zobrazováním obrázků. Toto je běžné například u osob slepých nebo částečně slepých. Některé z atributů mají tu vlastnost, že stačí jejich přítomnost k vyjádření určitého stavu, a nepotřebují tak uvedení hodnoty. Jsou to zejména atributy s hodnotami typu *true/false*.

### 1.3 Struktura HTML dokumentu

Všechny HTML dokumenty začínají tagem *html*, který je rodičem v celém dokumentu. Uvnitř tagu *html* je sekce *head*, která obsahuje prvky pro komunikaci s webovým prohlížečem, nastavení názvu stránky *title*, odkazy na stylovací soubory či nastavování metadat. Pro samotné tělo dokumentu se používá tag *body*. Struktura rodič *html* se dvěma potomky *head* a *body* je základní kostrou každé HTML stránky. HTML dokument by na svém začátku měl obsahovat informaci *DOCTYPE*, která indikuje typ dokumentu, a prohlížeč tak díky ní ví, jaké druhy elementů má v daném dokumentu očekávat.

```
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>

  </body>
</html>
```

Obr. 1-8 kostra HTML dokumentu

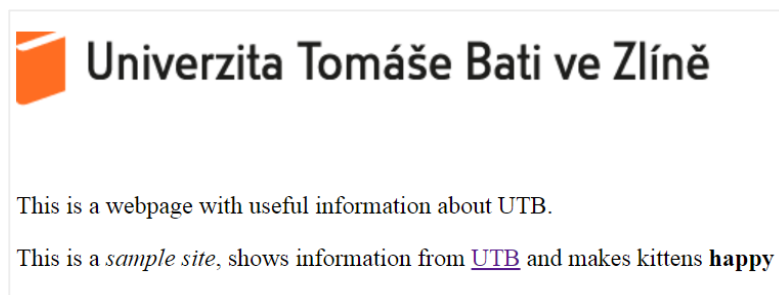
Na Obr. 1-8 je HTML dokument, který demonstruje základní strukturu. Do hlavy dokumentu *head* je vložen tag *title* označující její název v okně prohlížeče. V těle je použit tag *h1*, tedy nadpis (heading). V HTML je definováno 6 typů hlaviček, *h1–h6*, pro různé úrovně nadpisů. *H1* je z nich většinou největší a je používán pro vizuálně výrazné prvky stránky. Dále dokument obsahuje odstavce *p*. Tyto odstavce jsou prohlížečem automaticky separovány, protože jsou v HTML označeny jako paragrafy. Části těchto paragrafů mohou být samozřejmě dále značkovány, jak je vidět u tagů *i* a *b*. Tag *i* zajistí zobrazení textu kurzívou, zatímco tag *b* zapříčiní tlusté písmo. V HTML5 jsou ale tagy *i* a *b* doplněny o alternativní, které svým názvem více vystihují význam textu. Pro tag *i* je to tag *em* (emphasis, důraz), pro *b* je to *strong* (silně).

HTML, jak už bylo zmíněno, je hypertextový značkovací jazyk. Jeden ze základních principů funkce world wide webu je, že stránky mohou odkazovat na další stránky a uživatel tak může sledovat řetěz informací ze jednoho dokumentu na druhý. Text, který je schopný takového odkazování, se nazývá hypertext a v HTML se vytvoří pomocí tagu *a* (anchor, kotva). Tag *a* má atribut *href*, jehož hodnota určuje, kam má tento odkaz vést.

V dokumentu lze nalézt i tag *img* značící obrázek. Atribut *src* tagu *img* ukazuje cestu k obrázku, který má prohlížeč zobrazit. Adresa tohoto obrázku může být zadána jako relativní nebo absolutní. Relativní cesta je relativní k poloze dokumentu v souborovém systému, zatímco absolutní adresa se udává většinou pomocí URL adresy žádaného zdroje.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Github Search</title>
  </head>
  <body>
    <h1></h1>
    <p>This is a webpage allowing search in GitHub projects.</p>
    <p>This is a <em>sample site</em>, that shows information
      from <a href="http://github.com">GitHub</a>
      and <span>makes kittens</span> <b>happy</b>.
    </p>
  </body>
</html>
```

Obr. 1-9 kód jednoduchého dokumentu HTML



Obr. 1-10 vzhled stránky z Obr. 1-9

Jde vidět, že již s užitím jednoduchých tagů a nevelkého množství kódů lze vytvořit jednoduchý HTML dokument a začíná se rýsovat základní struktura dokumentu, který dokáže webový prohlížeč zobrazit a vyhodnotit právě na základě informací ze značek HTML.

## 1.4 Speciální znaky

HTML podporuje zobrazování speciálních znaků a také jejich escapování. Všechny speciální znaky začínají znakem `&` a končí středníkem. Ve vytvářeném dokumentu využijí například speciální znak `&copy;`, který se zobrazí jako symbol pro copyright. Escapované znaky jsou v HTML nazývány referencemi entit. Ty umožní vytvořit znaky, které by pro HTML nemusely být validní. Je jich poměrně mnoho, proto zmíním ty nejzajímavější. Reference entit `&lt;` (less than, menší než) a `&gt;` (greater than, větší než) jsou zajímavé, protože reprezentují právě symboly `<` a `>`, které v HTML nelze použít, neboť jsou oddělovači tagů, značí jejich začátek a konec. Pomocí referencí entit lze vyjádřit i znaky kódování Unicode, které také začínají ampersandem a pokračují křížkem. Za křížkem následuje buď decimální hodnota požadovaného Unicode znaku, nebo písmeno *x* a pak až jeho hodnota. Písmeno *x* zde instruuje prohlížeč při interpretaci kódu, že hodnota v systému Unicode bude vyjádřena hexadecimálním číslem. Například reference `&#160;` povede k zobrazení symbolu  $\Omega$ , kdežto `&#x160;` ke znaku Š. Tyto reference je možné uplatnit obzvláště v případech, kdy samo HTML referenci pro daný znak nemá, ale v Unicode existuje.

## 1.5 Tagy

### 1.5.1 Základní tagy struktur

#### 1.5.1.1 Tag *div*

Výše již byla uvedena základní myšlenka tagu *div*, a to že *div* je kusem HTML stránky. Je to pomůcka, jak sdružovat elementy do struktur. Pokud například *div* obaluje první dva elementy těla z Obr. 1-9, je tím míněno, že budou na webové stránce pohromadě. V případě dokumentu, se kterým pracuji, bude tento konkrétní *div* reprezentovat jakousi hlavičku. Stejně tak bude zabalena část s paragrafy *p*, která bude tvořit tělo obsahu této webové stránky. Vytvořím i třetí *div*, který bude sloužit jako patička.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Github Search</title>
  </head>
  <body>
    <div>
      <div></div>
      <h1>Github Search</h1>
      <div>This is a webpage allowing search in GitHub projects.</div>
    </div>
    <div>
      <p>This is a <em>sample site</em>, that shows information
        from <a href="http://github.com">GitHub</a>
        and <span>makes kittens</span> <b>happy</b>.
      </p>
    </div>
    <div>
      &copy; 2016 Kitten Kennel Antarctica
    </div>
  </body>
</html>
```

Obr. 1-11 tagy div obalující části stránky

Tagy *div* nemají své vlastní UI nebo vzhled, aniž by byly instruovány své vlastnosti změnit. Tedy přestože byly do dokumentu přidány strukturální prvky, nezměnil se jeho vzhled a stránka vypadá stejně jako na Obr. 1-10. Obecně v HTML *div* reprezentuje pravoúhelníkovou sekci stránky, proto se často používá na sestavení hrubého těla dokumentu. Tag *div* spadá mezi blokové elementy.

### 1.5.1.2 Tag *span*

Někdy je třeba označit řádek nebo jeho část a k tomuto účelu se používají řádkové elementy, obzvláště tag *span*. Označení kódu tagem *span* se užívá pro změnu stylu pouze označené části. Tyto změny mohou zahrnovat vše od změny fontu, dekorace textu a barvy, až po velikost nebo umístění.

## 1.5.2 Tagy v HTML5

Tagy *div* se stále hojně používají k označování sekcí nebo skupin elementů. HTML5 k nim nabízí tzv. sémantické tagy, což jsou tagy lépe pojmenované, aby bylo patrnější, k jakému účelu slouží. Některé vyhledávače dokonce dokážou tyto tagy vyhodnotit a rozlišit jejich obsah. Co se týče formátování, většina těchto sémantických tagů je formátována jako tag *div*. Pomocí sémantických tagů je tak při rozboru HTML kódu vyhledávačem, mobilním zařízením nebo prohlížečem umožněno hlubší porozumění struktuře dokumentu a případně ulehčeno rozhodování, co s kterou částí kódu provést nebo jak je důležitá.

### 1.5.2.1 Tagy *header*, *footer*, *section*

Tyto sémantické tagy se používají pro označení záhlaví, zápatí a sekce dokumentu. Pokud například bude stránku procházet robot vyhledávače, může odhadnout, že v záhlaví a v zápatí nejspíš nebudou zajímavé informace o tom, co stránka obsahuje, a může tak přiřkládat větší váhu jiným sekcím.

### 1.5.2.2 Tag *nav*

Tag *nav* je také sémantickým tagem a určuje navigační část stránky. Kdokoliv s touto znalostí se na kód podívá, bude vědět, že pomocí prvků v této sekci bude prováděna navigace po webové stránce. Proto se v sekci *nav* dá definovat v poměrně libovolné struktuře seznam odkazů na další stránky.

### 1.5.2.3 Seznamy

V HTML existují dva druhy seznamů – řazený, který je číslovaný, a neřazený, který tvoří sada odrážek. Řazený seznam začíná a končí značkou *ol* (ordered list, řazený seznam). Každý z jeho prvků je zabalen do tagu *li* (list item, prvek seznamu), a bude tak označen pořadovým číslem. Stejným způsobem lze vytvořit neřazený seznam, ve kterém bude místo pořadovým číslem označen každý prvek odrážkou. Tento typ má značku *ul* (unordered list, neřazený seznam) a často se používá v rámci tagu *nav* na organizaci seznamu odkazů. Standardně neřazený seznam používá jako odrážky černé tečky.

### 1.5.2.4 Tag *br*

Tento atomický tag zapříčiní pokračování obsahu, kterým je následován, na dalším řádku. Patří do speciální skupiny tagů zvaných *void elements* (prázdné prvky). Tyto prvky se vyznačují tím, že potřebují pouze otevírací tag a nemusí být uveden jejich zakončovací tag. Mezi běžně se vyskytující elementy této skupiny patří dále například *img*, *input* a *hr*.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Github Search</title>
  </head>
  <body>
    <header>
      <div></div>
      <h1>Github Search</h1>
      <div>This is a webpage allowing search in GitHub projects.</div>
      <nav>
        <ol>
          <li><a href="/index.html">Home</a></li>
          <li><a href="/contact.html">Contact</a></li>
          <li><a href="/about.html">About</a></li>
        </ol>
      </nav>
    </header>
    <section>
      <p>This is a <em>sample site</em>, that shows information
        from <a href="http://github.com">GitHub</a>
        and <span>makes kittens</span> <strong>happy</strong>.
      </p>
    </section>
    <footer>
      &copy; 2016 Kitten Kennel Antarctica
    </footer>
  </body>
</html>
```

Obr. 1-12 kód stránky se sémantickými tagy a seznamem

## 1.6 Formuláře

HTML formuláře mají značku *form* a dovolují stránce sbírat data a informace od uživatele a ty následně odesílat na server. Každý prvek musí mít svůj název, aby formulář při odesílání věděl, jak se která informace jmenuje. K tomu slouží atribut *name*. Tag *form* musí obsahovat atribut *action*, který udává koncový bod, kam odeslat data.

Protokol HTTP umožňuje data po internetu posílat různými způsoby, z nichž nejběžnější je požadavek GET. Při zadání URL do okna prohlížeče se provede právě požadavek GET vůči koncovému bodu na zadaném URL, který vrátí HTML kód stránky. Častěji se ale u formulářů používá požadavek POST. Který z těchto typů požadavků bude proveden, lze nastavit pomocí atributu *method*. Jeho výchozí hodnota je GET. Pokud je ale nastaven na POST, odešle se formulář na stejné URL udané atributem *action* s tím rozdílem, že se hodnoty z formuláře už neposílají jako část dotazového řetězce za tímto URL, ale jsou posílány v těle tohoto požadavku. Toho se využívá mimo jiné proto, že některé typy dat nemohou být součástí dotazového řetězce, a jak název napovídá, umí přenášet pouze data ve formě textu. Pokud by bylo třeba poslat například obrázek, užije se v těle požadavku POST formát zvaný *form encoded*, který takovou komunikaci umožní.

### 1.6.1 Tag *input*

Je základním prvkem každého formuláře. Tag *input* (vstup) má atribut *type*, který určuje, jaký typ informací bude sbírat.

#### 1.6.1.1 *type* = “text”

Ve výchozím stavu je *input* interpretován jako standardní textové pole, tedy jeho účelem je sběr textových informací, např. jména, hesla, emailu, vyhledávaného řetězce.

#### 1.6.1.2 *type* = “submit”

Typ *submit* (odeslat) zapříčiní zobrazení tagu *input* jako tlačítka. Zastává funkci odeslání formuláře na koncový bod, kde budou data zpracována.

#### 1.6.1.3 *type* = “checkbox”

Změnou typu na *checkbox* se *input* zobrazí jako zaškrťovací políčko. Nabývá hodnoty *true*, pokud zaškrtnuté je, *false* v případě opačném. Výchozí hodnotu lze nastavit pomocí atributu *checked*. Hodnota tohoto prvku se odešle pouze v případě, že je zaškrtnutý. Jeho hodnota pak bude řetězec *on*.

#### 1.6.1.4 *type* = “radio”

Typ *radio* umožňuje ve formuláři vytvořit sadu možností, z nichž lze vybrat pouze jedinou. Tato sada je realizována pomocí několika elementů *input*, které mají stejný atribut *name*. Každý z těchto prvků pak potřebuje atribut *value* (hodnota), který říká, jaká hodnota bude odeslána, pokud bude vybrán daný prvek *input*.

### 1.6.2 Tagy *select* a *option*

Tag *select* (výběr) označuje vyklápěcí výběrové menu, jehož každá položka je označena tagem *option* (možnost, volba). Pokud je požadováno, aby jedna z možností byla předoznačena při načtení stránky, je možné použít atribut *selected* (případně *selected =true*“).

### 1.6.3 Tag *label*

Při vytváření rozložení formulářů se může stát, že popisek nějakého prvku formuláře nebo nebude bezprostředně před nebo za ním. Pomocí tohoto tagu lze oštítkovat popisky jednotlivých prvků formuláře. Jednoznačný identifikátor *id* prvku formuláře spjatého

s daným popiskem se uvádí v atributu *for*, který nemusí být vždy shodný s atributem *name* daného prvku. Atribut *name* se používá v rámci formulářů pro označení toho, jaká data jsou posílána, zatímco *id* v rámci HTML slouží jako unikátní identifikátor jednotlivých elementů. Použití tagů *label* nezmění vzhled formuláře, ale přidá mu funkcionalitu. Prvek *label* nyní slouží jako ovládací prvek k danému prvku formuláře a reaguje například na kliknutí. Značky *label* také přidávají stránce sémantiku a označují, které části patří k sobě.

```
<form class="simple-form" id="gitHubSearchForm">
  <label for="searchPhrase">Search Phrase:</label>
  <input name="searchPhrase" id="searchPhrase" /><br />
  <input type="checkbox" name="useStars" id="useStars" />
  <label for="useStars">Use stars?</label><br />
  <label for="langChoice">Language:</label>
  <select name="langChoice" id="langChoice">
    <option>All</option>
    <option>JavaScript</option>
    <option selected>C#</option>
    <option>Java</option>
    <option>PHP</option>
  </select>
  <br>
  <input type="submit" value="search" /><br />
</form>
```

Obr. 1-13 příklad formuláře

## 1.7 HTML napříč prohlížeči

### 1.7.1 Komentáře

Jako většina jazyků používaných při vývoji webových stránek nabízí i HTML možnost přidání komentářů přímo do kódu stránky. Jejich začátek je značen kombinací znaků `<!--` a konec znaky `-->`. Cokoliv mezi těmito značkami – i napříč řádky – je až na pár výjimek prohlížečem ignorováno.

```
<!-- this is a simple comment -->
```

Obr. 1-14 jednoduchý komentář

Kromě ulehčení orientace v kódu se komentáře používají také k úpravám kódu pro správné zobrazení v různých prohlížečích.

```
<!-- [if lt IE 9]>
  <div>This div is only visible in Internet Explorer version lower than 9</div>
<![endif] -->
```

#### Obr. 1-15 komentář značící obsah pouze pro některé prohlížeče

Na Obr. 1-15 je v hranatých závorkách vidět podmínka pro jeden z prohlížečů. V tomto případě říká: „Pokud (*if*) je prohlížeč Internet Explorer (*IE*) nižší verze než (*lt*) 9, zobraz následující kód.“ Pouze některé prohlížeče jsou schopny tento kód z poznámek vyčíst. Ty, které to neumí, se budou chovat podle běžného značení komentářů a vypustí vše mezi začátečním a koncovým pseudotagem komentáře. Tímto způsobem se tedy dají použít kód, stylování, případně skript v různých místech dokumentu platné pouze pro určité prohlížeče.

## 2 KASKÁDOVÉ STYLY CSS

Každý element v HTML má atribut *style*, který umožňuje specifikovat pravidla pro vzhled. Ačkoliv je možné této vlastnosti využít, je pro práci jednodušší oddělit vzhled stránky od jejího uživatelského rozhraní. Jednou z možností, jak toho docílit, je využití tagu *style* v rámci bloku *head* dokumentu HTML, kam se vloží informace o stylu. Toto řešení má nevýhodu v tom, že je omezené na jeden HTML dokument. Proto se používá kaskádových stylů CSS. Ty obsahují všechna pravidla pro stylování stránky, nechávají v HTML pouze informace o struktuře a dovolují používat sadu stylů napříč několika dokumenty. Také je s jejich pomocí možné upravit styly pro konkrétní elementy v konkrétních dokumentech, a přepisovat tak pravidla platná pro celou stránku.

### 2.1 Stylování v HTML

```
<header style="background-color: lightblue;">
```

Obr. 2-1 stylování v HTML na řádku

Hodnota atributu *style* má svůj vlastní syntax a používá páry název–hodnota. Jméno stylové vlastnosti je odděleno dvojtečkou od své hodnoty. Následuje středník, který odděluje pár od dalšího v řadě. Ačkoliv je takovéto přidávání stylů možné, není vhodné, protože míchá informace o vzhledu s informacemi o struktuře. Na obr. Obr. 2-1 stylování v HTML na řádkuvlastnost *background-color* a její hodnota *lightblue*, říkají, že pozadí má být světle modré barvy.

### 2.2 Tag *style* a styly

Tag *style* se běžně vkládá do elementu *head*. Do tohoto tagu se potom přidává informace o stylu. Rozdílem oproti stylování v elementech samotných je, že je třeba určit, pro který element se má stylování použít. Na Obr. 2-2 stylovací pravidlo v hlavičce dokumentu je znázorněn zápis, který použije daný styl pro všechny elementy *header*. Skládá se ze jména tagu *header* a páru složených závorek. Do těchto závorek lze umístit stylové vlastnosti oddělené středníkem, stejně jako v případě Obr. 2-1 stylování v HTML na řádku

```
<head>
  <style>
    header {
      background-color: lightblue;
    }
  </style>
</head>
```

Obr. 2-2 stylovací pravidlo v hlavičce dokumentu

Sekce, která je obsažena v elementu *style* na Obr. 2-2 stylovací pravidlo v hlavičce dokumentu, se nazývá stylovací pravidlo. Jeho rolí je vybrat jeden nebo více prvků stránky a specifikovat, jak mají být naformátovány. Jedno pravidlo může obsahovat několik různých stylových vlastností prvku. Tyto vlastnosti jsou ne vždy jednohodnotové, a mohou tak udávat na jednom řádku více vlastností zároveň.

```
<style>
  header {
    background-color: lightblue;
    border: solid 1px black;
    font-family: 'Times New Roman', serif;
  }

  body {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 14px;
  }
</style>
```

Obr. 2-3 pár stylovacích pravidel

Na Obr. 2-3 pár stylovacích pravidel je zobrazeno rozšíření Obr. 2-2 stylovací pravidlo v hlavičce dokumentu o další stylovací pravidlo. Toto pravidlo platí pro element *body* a udává základní informace o tom, jak má být formátován veškerý text, který obsahuje. Vlastnost *font-family* definuje typ fontu. Protože různá zobrazovací zařízení mohou disponovat různými fonty, definuje se pro *font-family* posloupnost fontů, která se potom při vykreslování prochází. Pokud není k dispozici první v řadě, zkontroluje se další. Pro případ, že není žádný z uvedených fontů k dispozici, uvádí se na konec posloupnosti heslo *serif* nebo *sans-serif* označující libovolné patkové nebo bezpatkové písmo. Vlastnost *font-size* definuje velikost písma. Pro prvek *header* byl definován rámeček *border* vzhledu *solid* (plný) o tloušťce 1px černé barvy. Jde vidět, že stejná vlastnost *font-family* je definována jak pro prvek *header*, tak pro prvek *body*.

Protože pro případ HTML kódu z přílohy P I jsou tato pravidla platná pro stejné prvky kódu, dochází ke kolizi definic a je třeba rozhodnout, která z nich bude mít větší prioritu.

Styly a prohlížeče využívají sadu pravidel kaskádování. To znamená, že čím je pravidlo specifičtější, tím je větší jeho priorita. V případě kódu z přílohy P I element *body* obaluje celé viditelné tělo HTML dokumentu a je rodičem specifičtější sekce dokumentu *header*. V tomto případě by proto byla pro *header* upřednostněna jeho vlastní specifičtější definice stylu.

Kaskádování má dvě pravidla:

1. Pokud se pravidlo v kódu vyskytuje později, je prioritnější než dřívější pravidlo.
2. Specifičtější pravidlo je prioritnější než pravidlo méně specifické.

Platí přitom, že pravidlo 2 je silnější než pravidlo 1.

Tato pravidla mohou být ignorována uvedením hesla *!important* v hodnotě stylové vlastnosti. Přestože se může zdát využití tohoto hesla výhodné, v praxi je používáno velmi málo a jeho použití většinou ukazuje na nesprávně navrženou kaskádu stylů.

### 2.3 Kaskádové styly

Stylování v hlavičce *head* HTML sice funguje, ale je stále dosti lokalizované. Všechna pravidla by musela být obsažena v každém dokumentu webové stránky, což by bylo v případě větších projektů těžce udržitelné a neflexibilní.

Proto se ve většině případů místo stylových informací přímo v HTML dokumentu používají separátní soubory kaskádových stylů s *.css* příponou. Tyto soubory se běžně uchovávají pospolu a je možné jich mít více – některé pro celou stránku, jiné pouze pro určité dokumenty.

CSS používá starý typ komentářů jazyka C, tedy znaky */\** pro začátek a *\*/* pro konec komentáře. Styly jsou definovány stejně jako v hlavičce *head* HTML dokumentu. Pro spojení souboru CSS s HTML dokumentem se do sekce *head* HTML dokumentu vloží tag *link*, u kterého je třeba nastavit dva atributy. Prvním z nich je atribut *rel*, který udává, o jaký typ odkazu se jedná. V případě CSS se jedná o typ *stylesheet*. Druhým je atribut *href*, který stejně jako u tagu *a* udává cestu k odkazovanému souboru.

```
<link rel="stylesheet" href="css/site.css">
```

Obr. 2-4 odkaz na soubor css v dokumentu HTML

## 2.4 Syntax

Stylové pravidlo má dvě části:

1. Selektor, určující, pro které prvky bude pravidlo platit. Jedná se o část pravidla před levou složenou závorkou.
2. Stylové vlastnosti ve složených závorkách oddělené středníkem.

Velmi běžná je situace, kdy je třeba označit více prvků v dokumentu stejnými styly. Aby bylo zamezeno duplicitě kódu, je možné pravidla pro jednotlivé prvky sdružovat. V takovém případě se jména pravidel oddělí čárkou, jak je tomu na Obr. 2-5.

```
header, footer {  
    background-color: lightgray;  
    border: solid 1px black;  
    font-family: 'Times New Roman', serif;  
}
```

Obr. 2-5 sdružená pravidla pro prvky header a footer

Stylové vlastnosti tohoto bloku budou aplikovány pro oba prvky v selektoru tohoto pravidla.

Syntax selektoru rozhoduje o tom, na které prvky budou pravidla aplikována.

## 2.5 Selektory tagů

Základní typ selektoru sestává z názvu tagu prvku, který bude stylován, jako u případů výše. Mimo to je možné využít pro specifičtější identifikaci i atributy daného prvku. V takovém případě bude selektor prvku následován hranatými závorkami obsahujícími pár atribut–hodnota rozdělený znakem =, tedy např. `input[type=text]`.

## 2.6 Selektory ID

Výše byla v selektoru pravidla uvedena vždy jména tagů. Ty jsou ale ve většině případů málo specifické. Aby bylo možné vybrat jednotlivý libovolný prvek, je třeba využít jeho atributu unikátního identifikátoru *id* v HTML.

Element s *id* se dá v CSS označit tak, že selektor bude mít na svém začátku znak # následovaný identifikátorem. V Obr. 2-6 stylové pravidlo uplatněné na jednoznačný prvek pomocí *id* se jedná o prvek s unikátním identifikátorem *main*.

```
#main {  
    border: solid 1px #ccc;  
    border-radius: 5px;  
    color: #202020;  
    margin: 20px 0;  
    padding: 5px;  
}
```

Obr. 2-6 stylové pravidlo uplatněné na jednoznačný prvek pomocí id

V Obr. 2-6 je zároveň vidět další způsob reprezentace barev v CSS. V tomto případě se jedná o reprezentaci v podobě hexadecimálních hodnot RGB, tedy červené, zelené a modré. Toto vyjádření začíná znakem # následovaným buď třemi dvouznakovými hexadecimálními čísly, nebo třemi jednoznakovými hexadecimálními čísly.

Obr. 2-6 také obsahuje dříve nezmíněné stylové vlastnosti *border-radius*, *color*, *margin* a *padding*. Vlastnost *border-radius* udává poloměr zakulacení rohu rámečku, zatímco *color* říká, jakou barvu bude mít samotný text. Vlastnosti *margin* a *padding* určují odsazení prvku. Vlastnost *margin* vyjadřuje odsazení krajů elementu od okolí, *padding* definuje odsazení obsahu elementu od jeho okrajů. U *margin* je použit dvouhodnotový zápis vzdáleností, kdy první číslo určuje odsazení shora a zdola a druhé zleva a zprava. U *padding* je jediná hodnota udávající odsazení ze všech stran.

## 2.7 Relativní selektory

V HTML kódu z přílohy P I se objevuje sekce, které byl přiřazen identifikátor *main*, v ní formulář *form* a v něm několik štítků *label*. Pro každý z těchto popisků by bylo možné přidat unikátní identifikátor. To by však bylo velmi pracné, obzvláště pokud mají nějaké podobné vlastnosti, jak je tomu například právě u formulářů. V takovém případě lze použít relativní selektory. Ty jsou vytvářeny jako posloupnost základních selektorů oddělených mezerou nebo znakem >. Vybíraný prvek je poslední v řadě.

```
#main > form > label {  
    font-weight: bold;  
}
```

Obr. 2-7 relativní selektor se znaky >

Na Obr. 2-7 lze vidět relativní selektor využívající znaky >. Tento znak udává, že selektor po jeho levé straně je přímým rodičem selektoru na straně pravé. V daném případě to

znamená, že relativní selektor vybere všechny elementy *label*, jejichž přímým rodičem je element *form*, jehož přímým rodičem je element s identifikátorem *main*.

Tyto přímé vztahy rodič–potomek jsou velmi křehké. Velice jednoduše se může narušit stylování, pokud by například mezi prvky *form* a *label* byla vložena další vrstva. Proto je robustnějším řešením používat oddělování jednotlivých selektorů pomocí mezer.

```
#main form label {  
    font-weight: bold;  
}
```

Obr. 2-8 relativní selektor s mezerami

Relativní selektor z Obr. 2-8 vybírá všechny prvky *label* nacházející se libovolně hluboko v prvku *form*, jehož libovolně vzdáleným předkem je prvek s identifikátorem *main*.

Ačkoliv je toto řešení robustnější, nemusí být vždy řešením lepším, a to například v případě, kdy vybíraný element explicitně musí být přímým potomkem jiného.

## 2.8 Selektory tříd

HTML elementy v těle dokumentu podporují atribut *class*. Idou tohoto atributu je vytvořit třídy elementů, které budou sdílet stylovací vlastnosti a pravidla. Atribut *class* může udávat několik tříd zároveň – v takovém případě se oddělují mezerou. Všechny tyto třídy budou brány v potaz při vykreslování dokumentu prohlížečem. Element *img* v sekci header v kódu HTML z přílohy P I má atribut *class* s hodnotou *bordered-image*. Hodnota tohoto atributu je stejně jako u *id* volitelná. V CSS je tato třída adresována selektorem začínajícím tečkou následovanou jménem třídy.

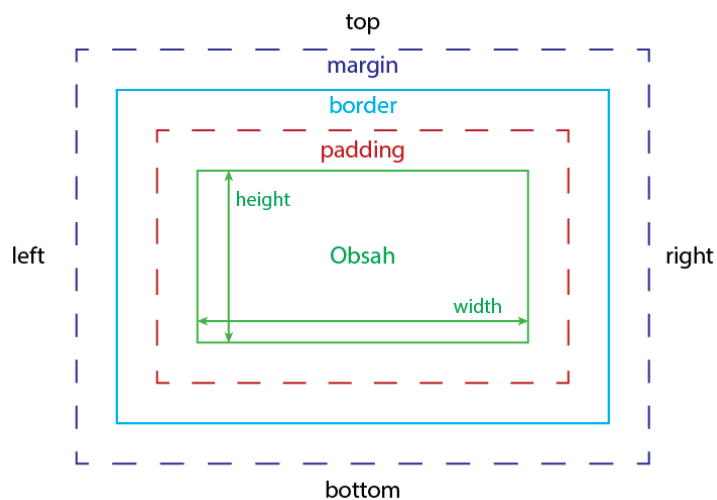
```
.bordered-image {  
    border: solid 1px #444;  
    border-radius: 2px;  
}
```

Obr. 2-9 selektor třídy s názvem bordered-image

CSS z Obr. 2-9 bude platit pro všechny libovolné elementy s atributem *class* o hodnotě *bordered-image* neboli pro všechny patřící do třídy *bordered-image*.

## 2.9 Dimenzování a box model

Vlastní velikost prvků v CSS je založena na krabicovém modelu neboli box modelu prvku. Jeho středem je obsah, který se nachází uvnitř tzv. vycpávky (stylovací vlastnost *padding*). Tu obklopuje rámeček (stylovací vlastnost *border*), který dále obaluje tzv. okraj (stylovací vlastnost *margin*). Z každé strany tohoto modelu je ještě hodnota vzdálenosti, pomocí které můžeme prvek posouvat různě po stránce. Jmenovitě *top*, *right*, *bottom* a *left*. Stylovací vlastnost *height* (výška) určuje výšku samotného obsahu. Nejsou v ní zahrnuty žádné z velikostí udané ostatními rozměry box modelu. Stejně tak vlastnost *width* určuje pouze šířku obsahu. Pokud má mít prvek na stránce určitou velikost, je tedy nutné přičíst k hodnotám *width* a *height* hodnoty *padding*, *border* a *margin* v obou směrech od obsahu.



Obr. 2-10 box model

## 2.10 Pozicování v CSS

Každému prvku lze přiřadit typ pozicování na stránce pomocí stylovací vlastnosti *position*, která je defaultně nastavena na hodnotu *static*. Hodnota *static* definuje, že prvek má být vykreslen v pořadí hierarchie HTML dokumentu na hierarchií přesně daném místě. Druhým typem hodnoty pro *position* je hodnota *absolute*, díky níž je možné zvolit přesnou polohu prvku v rámci stránky udáním vzdálenosti od okrajů stránky pomocí stylovacích vlastností *top*, *right*, *bottom* a *left*. Většinou je použita jedna vlastnost pro horizontální a jedna pro vertikální osu, např. *top* a *left*. Tato hodnota se využívá vesměs ve speciálních případech, například když je potřeba na stránce vykreslit prvek na stabilním místě i při skrolování.

## 2.11 Zobrazení v CSS

V kapitole 1.5.1 byly zmíněny základní strukturální prvky a byl uveden jeden blokový a jeden řádkový element. HTML elementy mají svou vlastnost *display* (zobrazení), podle níž je lze rozdělit na skupiny. Největší skupiny tvoří elementy typu *block* (blokové), *inline* (řádkové) a *inline-block* (řádkově blokové).

Blokové elementy jsou defaultně řazeny pod sebe, jsou jimi většinou strukturální prvky stránky a chovají se jako obdélníky. Patří mezi ně například *div* a *form*.

Řádkové elementy se používají v textuální reprezentaci a nerozbíjí strukturu textu. Jsou jimi třeba prvky *a* a *span*.

Prvky řádkově blokové se chovají jako blokové elementy s tím rozdílem, že se nerovnají pod sebe, ale vedle sebe, dokud nejsou omezeny šířkou svého rodiče. Pokud by šířka dalšího *inline-block* elementu v řadě měla přesáhnout šířku rodiče, byla by zalomena na další řádek.

V CSS je možno změnit typ *display* libovolného elementu na jiný z těchto typů použitím stejnojmenné stylovací vlastnosti a jedné z hodnot *block*, *inline* a *inline-block*;

## 2.12 Float v CSS

CSS vlastnost *float* (vznášet se) je jedna z náročnějších na správné použití a instruuje prohlížeč, aby prvky naskládal na stranu do volného prostoru. Nejpoužívanějšími jsou hodnoty *left* a *right*. V případě několika elementů se stejnou hodnotou vlastnosti *float* se tedy budou tyto elementy skládat ve směru této hodnoty od kraje rodičovského elementu.

## 2.13 Ladění stylů v prohlížeči

Moderní prohlížeče nabízí nástroje pro ladění CSS, které jsou si navzájem velice podobné. Další nástroje je možné získat často jako rozšiřující moduly samotných prohlížečů. Každý prohlížeč má svůj způsob zpřístupnění těchto nástrojů, nicméně běžně používanou klávesovou zkratkou pro jejich zobrazení je klávesa F12. Ladicí nástroje umožňují procházet HTML kód a vidět stylovací pravidla doopravdy použitá pro jednotlivé prvky. Při zvolení některého z prvků v HTML kódu zobrazeného ladicím nástrojem v liště lze vidět nejen pravidla definovaná pouze pro daný prvek, ale všechna pravidla, která kaskádovitě zdědil. Tímto způsobem lze poměrně jednoduše najít překlepy při psaní pravidel a odladit chyby v zobrazení v jednotlivých prohlížečích.

### 3 JAVASCRIPT

JavaScript je objektový skriptovací jazyk, který se v dnešní době využívá na prakticky všech webových stránkách, což bylo výhradní funkcí v jeho začátcích. S rostoucí popularitou se z něj stal poměrně robustní jazyk používaný nejen na webových stránkách neboli frontendu (to, s čím přichází do styku uživatel), ale i na serverech – backendu (to, s čím stránky komunikují pro získání dat a odpovědí). Je taktéž používán k automatizaci různých systémových úloh. JavaScript je nedílnou součástí arzenálu znalostí webového vývojáře.

Ačkoliv je JavaScript jazykem objektovým, nemá třídy objektů, jak jsou známy z jiných objektově orientovaných jazyků. Namísto dědičnosti tříd užívá takzvanou prototypickou dědičnost. To znamená, že dědičnost je založena na klonování objektů, které slouží jako prototypy.

JavaScript patří do skupiny dynamických jazyků. Tato vlastnost znamená, že je dynamicky typován a že se mohou měnit typy. Svůj typ mohou taktéž změnit i proměnné určitých typů.

JavaScript je považován za interpretovaný jazyk, ale ve většině případů je jeho kód kompilován těsně před jeho spuštěním. Jiné jazyky, jako například Java nebo C#, potřebují kompilační mezikrok, který jejich kód přeloží do mezijazyka. Až tento mezijazyk je těsně před spuštěním kompilován, stejně jako v případě JavaScriptu. I z důvodu vynechání tohoto mezikroku může JavaScript vykazovat podobně vysoké výkonnostní parametry jako jiné vysokovýkonnostní jazyky, a je tak vhodným kandidátem pro nasazení na servery.

#### 3.1 Vkládání JavaScriptu do HTML

Podobně jako u CSS je možné buď mít JavaScriptový kód jako součást HTML dokumentu, nebo jej uložit do externího souboru. Ke vložení do HTML se používá tag *skript*, a ačkoliv je běžně používán pro nahrání JavaScriptu, podporuje i další jazyky. Typ jazyku, který bude použit, je v tagu *skript* definován atributem *type*. Pro JavaScript je hodnotou tohoto atributu *text/javascript*. Do tohoto elementu je možné psát přímo JavaScriptový kód.

Kód JavaScriptu na rozdíl od jiných jazyků nevyžaduje ukončení řádku středníkem. Středník je na zakončení příkazu potřeba pouze tehdy, když za příkazem na stejném řádku následuje další příkaz.

Jedním ze základních klíčových slov je *var*. Klíč *var* dovoluje vytvořit novou proměnnou, která bude obsahovat nějaký typ dat. Může to být datum, číslo, textový řetězec nebo více

komplexní typy. Toto klíčové slovo je následováno mezerou, jménem proměnné a znakem následovaným reprezentací dat, které má tato proměnná obsahovat.

Velmi výhodné pro ladění JavaScriptu je použití objektu *console*. Je to výstup pro ladění kódu, kde jsou zobrazovány případné chyby. V prohlížeči lze tento výstup zobrazit použitím ladicích nástrojů, zmíněných v sekci 2.13.

JavaScript umožňuje s jednotlivými prvky na stránce manipulovat, a to skrz získání odkazu na daný prvek. Tuto referenci je možné zjistit tak, že bude použit kód *document* reprezentující DOM, na který bude zavolána metoda *getElementById* (získej element podle *id*), jejímž parametrem je identifikátor *id* požadovaného prvku. Existuje více podobných metod na získání elementů podle různých kritérií.

```
var resultsDiv = document.getElementById("results");
```

Obr. 3-1 získání odkazu na element pomocí objektu *document* a funkce *getElementById*

Každý prvek na stránce má své vlastnosti, které jsou pomocí JavaScriptu zobrazitelné a/nebo modifikovatelné. Pro přístup k těmto vlastnostem se používá zápis názvu proměnné následovaného tečkou a jménem vlastnosti. Na obr. 3-2 je kód modifikující vlastnost *innerHTML* prvku s identifikátorem *results*. Vlastnost *innerHTML* představuje HTML obsah daného prvku.

```
<script type="text/javascript">
  var msg = "hello JavaScript";
  console.log(msg);

  var resultsDiv = document.getElementById("results");
  resultsDiv.innerHTML = "This comes from JavaScript";
</script>
```

Obr. 3-2 *script* tag HTML dokumentu

Pokud by byl kód z Obr. 3-2 spuštěn v rámci hlavičky *head* dokumentu z přílohy P I, i přes správný JavaScriptový kód by nebyl přepsán obsah elementu s identifikátorem *results*, a to z toho důvodu, že se tento kód provádí ihned při načítání stránky. Jelikož je tento kód proveden v HTML dříve, než je vykreslen celý dokument (tag *head* předchází tag *body*), není ve chvíli provádění tohoto JavaScriptu k dispozici HTML element, který má být modifikován. Přestože existuje několik metod, jakými tento problém řešit, jedním z nejjednodušších způsobů je přesunout celý blok *script* z hlavičky na konec prvku *body*.

Takto bude zaručeno, že kód HTML bude prohlížečem vykreslen dřív, než přijde k tagu *script*.

Stejně jako CSS se JavaScript běžně neukládá v rámci HTML dokumentu, ale do separátních souborů, které jsou charakteristické příponou *.js*. Kód z těchto souborů se pak nahraje v rámci *script* bloku pomocí parametru *src*, odkazujícího na polohu daného souboru. Hodnotou může být buď relativní cesta od dokumentu, který tento prvek *script* obsahuje, nebo například plná URL adresa JavaScriptového souboru uloženého na internetu.

Tag *script* je jedním z HTML tagů, které potřebují uzavírací tag, a nelze ho proto ukončit v rámci otevíracího tagu, jako například *br*.

```
<script type="text/javascript" src="js/index.js"></script>
```

Obr. 3-3 vložení odkazu na JavaScriptový soubor

## 3.2 Proměnné a typy

Všechny proměnné mají, pokud jsou deklarovány, svůj typ. Ten se dá jednoduše zjistit pomocí funkce *typem*, která přijímá jako svůj argument danou proměnnou. Její návratovou hodnotou je řetězec.

Typy proměnných jsou:

- *string* – textový řetězec, proměnné se přiřazuje jako *text* v uvozovkách, řetězce je možné spojovat pomocí operátoru *+*;
- *object* – složitější neprimitivní datový typ;
- *number* – číselná hodnota;
- *boolean* – booleanovská hodnota *true/false*;
- *undefined* – typ proměnné není jasný, díky tomuto typu se dá zjistit, jestli proměnné už byla hodnota přiřazena – tedy jestli je nějakého typu – v takovém případě by její typ již nabýval jednoho z výše uvedených.

Jednou ze základních vlastností JavaScriptu je, že ve výchozím nastavení nemusí být proměnné definovány, než jsou použity. To znamená, že použití klíčového slova *var* není podmínkou. Nevýhodou je, že v případě překlepu ve jménu proměnné, jejíž hodnota má být v kódu přepsána, bude v JavaScriptu vytvořena nová proměnná s danou hodnotou. Odladovat, proč se původní hodnota nezměnila, a najít tento překlep může být prací velmi zdoluhavou a frustrující. Proto je možné na začátku JavaScriptového kódu uvést jako řetězec

frázi “*use strict*;”. Tato fráze funguje ve verzích JavaScriptu ECMA 3 a vyšších, tudíž ve většině moderních prohlížečů, u kterých je standardem podpora verze ECMAScript 3. Instruuje kompilátor, aby použil striktnější pravidla při parsování tohoto kódu. Jedním z těchto striktnějších pravidel je právě nemožnost použití nedeklarovaných proměnných. Při jejich užití bude kompilátor hlásit chybu. Tato fráze se udává ve formě řetězce kvůli zpětné kompatibilitě u starších verzí JavaScriptu. V takovém případě bude tato fráze jednoduše interpretována jako řetězec bez hlubšího významu a ve výsledku ignorována.

### 3.3 Rozhodování a podmínky

Většina rozhodovacích procesů je v JavaScriptu prováděna konstrukcí *if*. Vyhodnocuje se, jestli obsah kulatých závorek za klíčovým slovem *if* (tzv. podmínka) je pravdivý, nebo nikoliv.

Konstrukce *if* má vždy syntax:

- Klíčové slovo *if* – začátek podmínky.
- Kulaté závorky s podmínkou (podmínkami) uvnitř – vyhodnocovaný výraz.
- Složené závorky obsahující kód – bude proveden při vyhodnocení podmínky jako pravdivé.

Volitelně může následovat:

- Klíčové slovo *else* a složené závorky obsahující kód – bude proveden při vyhodnocení podmínky jako nepravdivé – ta bývá uvedena na konci konstrukce *if*.
- Klíčové slovo *else* následované slovem *if* a složené závorky obsahující kód – bude proveden, pokud byly předchozí podmínky od **začátku celé konstrukce *if*** nepravdivé a podmínka **tohoto** bloku *if* je pravdivá, těchto bloků může konstrukce *if* obsahovat několik.

```
if (condition) {
    //will be executed if condition is true
};

if (condition) {
    //will be executed if condition is true
}else{
    //will be executed if condition is false
};

if (condition) {
    //will be executed if condition1 is true
}else if (condition) {
    //will be executed if condition1 is false and condition2 is true
}else{
    //will be executed if neither condition1 nor condition2 are true
};
```

Obr. 3-4 typy struktur podmínek *if*

V podmínkách se často používá porovnávání dvou hodnot, běžně proměnné a hodnoty nebo dvou proměnných mezi sebou. Toto porovnávání se provádí pomocí operátorů v podmínce se syntaxem *a [operátor] b*:

- $<$  – *a* je menší než *b*
- $>$  – *a* je větší než *b*
- $==$  – *a* se rovná *b*
- $<=$  – *a* je menší nebo rovno *b*
- $>=$  – *a* je větší nebo rovno *b*
- $!=$  – *a* se nerovná *b*

Jednou ze zvláštností JavaScriptu je, že hodnoty porovnávané operátory  $==$  (dva znaky rovno) a  $!=$  (vykřičník rovno, tedy nerovno) jsou interně převáděny tak, aby bylo možné je navzájem porovnat. Proto například v podmínce  $15=="15"$  číslo *15* bude řetězcem "15". Stejně tomu bude u libovolné nastavené hodnoty porovnávané s booleanovskou hodnotou *true*.

Porovnávání je tu prováděno na rovnost, ne na totožnost.

Porovnávání na totožnost jsou prováděna pomocí operátorů:

- $===$  (tři znaky rovno) – *a* je totožné s *b*
- $!==$  (vykřičník a dva znaky rovno) – *a* není totožné s *b*

Totožnost je také někdy označována jako exaktní nebo identická rovnost.

V podmínce je také možné použít znak *!* pro označení nepravdivosti následujícího výroku.

Podmínku  $a != true$  je tedy možné napsat též zkráceně jako  $!a$ .

Použití jediného znaku `=` v podmínce je validní, ale dělá podmínku nesmyslnou, protože takový operátor značí přiřazení hodnoty. Například v podmínce `a = true` by byla do proměnné `a` přiřazena hodnota `true`. Jelikož by toto přiřazení proběhlo v pořádku, podmínka by byla vyhodnocena jako pravdivá.

Podmínky je možné sdružovat pomocí operátorů `&&` (AND, obě podmínky musí být vyhodnoceny jako pravdivé) nebo `//` (OR, alespoň jedna z podmínek musí být pravdivá).

### 3.4 Funkce

JavaScript podporuje vytváření funkcí. Funkce je konstrukce, datový typ, jehož účelem je vytvořit znovu využitelný kód.

Skládá se z:

- klíčového slova `function`;
- názvu funkce – uveden buď mezi `function` a kulatými závorkami, nebo jako název proměnné, do které je funkce přiřazována;
- páru kulatých závorek – tyto mohou (ale nemusí) obsahovat parametry funkce, oddělené čárkou, parametry se v definici uvádějí jako názvy proměnných, kterými budou v rámci funkce reprezentovány;
- složených závorek obsahujících kód – tzv. tělo funkce, je prováděno při zavolání funkce.

Funkce se následně volá pomocí svého názvu, následovaného případnými parametry v kulatých závorkách. Volání může být provedeno kdekoliv v kódu následujícím za definicí funkce.

```
function showIt(msg){
    console.log(msg);
}

var showIt = function (msg){
    console.log(msg);
}
```

Obr. 3-5 možnosti definice funkce

```
showIt("some message");
```

Obr. 3-6 volání funkce z Obr. 3-5

V JavaScriptu je možné mít pouze jednu funkci jednoho jména, a tím je vyloučeno přetěžování, tak jak je používané v mnohých jiných jazycích. Případné odlišné chování při posílání různých počtů parametrů stejné funkci je nutné ošetřit v rámci těla funkce.

Často se používá tzv. callback. Tímto názvem je označována funkce *a*, posílaná další funkci *b* jako parametr, která má být v rámci funkce *b* provedena. Tento koncept se používá z toho důvodu, že mnoho technik v JavaScriptu je asynchroních. To v praxi znamená, že je možné nečekat na zpracování a výsledek nějaké operace a postarat se o zpracování výsledků až poté, co bude operace dokončena. Takovými operacemi jsou například reakce na chování uživatele a síťové požadavky.

```
function showItThen(msg, callback){
    showIt(msg);
    callback();
}

showItThen("showItThen called", function(){
    console.log("callback called");
});
```

Obr. 3-7 definice a volání funkce s callbackem

Někdy je žádoucí, aby při volání funkce byla nejen provedena řada operací, ale aby funkce vracela nějakou hodnotu, například v závislosti na výsledku prováděných operací. Takovou hodnotu (nebo celý objekt) lze propagovat do místa volání funkce pomocí klíčového slova *return*, následovaného požadovanou hodnotou, objektem nebo hodnotou ve formě proměnné. Pokud program dojde k výrazu *return*, je přerušeno vykonávání dané funkce a další kód této funkce už nebude proveden. Z tohoto důvodu je možné použít *return* bez definování návratové hodnoty pouze za účelem předčasného ukončení funkce.

### 3.5 Rámce (scopes)

Rámec udává, pro kterou část kódu platí jednotlivé definice proměnných a funkcí. Pokud kód není zapouzdřený nějakou funkcí, spadá do *globálního rámce*, který je sdílen veškerým kódem a každou JavaScriptovou knihovnou na stránce. Proto je třeba mít se na pozoru při jeho používání. Vše v globálním rámci je na webové stránce připojeno k objektu *window*, jehož pomocí se dá přistupovat ke všem globálním prvkům.

V JavaScriptu je pouze jediná možnost, jak vytvořit menší rámec a tou je vytvoření funkce. Ve funkcích je možné přistupovat jak k prvkům z globálního rámce, tak k prvkům

vytvořeným v rámci dané funkce. Nefunguje však naopak, že by bylo možné přistupovat k lokálním prvkům funkcí z globálního rámce.

### 3.6 Zapouzdření (closures)

Zapouzdření je v JavaScriptu koncept, který dovoluje proměnným z vnějších rámců, aby byly viděny ve vnitřních rámcích. Jedním z důležitých prvků této vlastnosti je nezávislost na časové ose.

```
var x = 1;

function someFunction(){
    //works as it wraps 'x' with a closure
    var y = x;
}

//much later
someFunction();
```

Obr. 3-8 příklad zapouzdření

Na Obr. 3-8 je ve vnějším rámci definována proměnná *x*. Funkce *someFunction* používá tuto vnější proměnnou a může být později zavolána v libovolném časovém okamžiku. I kdyby se z nějakého důvodu dostalo toto volání mimo původní rámec, kde je proměnná *x* definována, bude proměnná *x* uchována v paměti, protože ji funkce *someFunction* potřebuje. Dokud tedy existuje funkce *someFunction*, bude existovat i proměnná *x*. JavaScript se o tuto funkcionalitu postará zabalením, vytvořením zapouzdření této proměnné, protože díky této funkci ví, že tato proměnná je nezbytná.

### 3.7 Objekty a pole

Jak bylo zmíněno v kapitole 3.2, proměnné v JavaScriptu mohou být jedním z několika typů. Proměnné typu *object* nemají svůj vlastní definovaný typ v pravém slova smyslu a lze s jejich pomocí vytvářet konstrukce, které shlukují informace o dané proměnné a připomínají třídy tak, jak jsou známy z jiných programovacích jazyků. Objekt jako takový je jen sadou párů jméno–hodnota.

```
var result = {
  name: "jQuery",
  language: "JavaScript",
  score: 4.5,
  showLog: function () {

  },
  owner: {
    login: "utb_student",
    id: 123456
  }
}
```

Obr. 3-9 příklad jednoduchého objektu

K jednotlivým vlastnostem objektu se dá přistupovat pomocí spojení názvu objektu a požadované hodnoty tečkou. Kód z Obr. 3-9 Obr. 3-9 příklad jednoduchého objektu definuje pro objekt *result* sadu vlastností *name*, *language* a *score*. Taktéž je pro *result* definována funkce *showLog* a další vnořený objekt *owner*. Pro přístup například k vlastnosti *score* by byl použit zápis *result.score*.

Objekty v JavaScriptu podporují mutace, což znamená, že můžeme do objektu kdykoliv v jeho životním cyklu přidávat vlastnosti.

Dalším typem objektu je pole, tedy určitá sbírka hodnot nebo objektů. Pole jsou definována pomocí hranatých závorek. Mezi závorkami mohou být uvedeny jednotlivé prvky pole odděleny čárkou. Pokud žádné prvky uvedeny nejsou, je definováno prázdné pole, které se pravděpodobně bude hodnotami plnit. Prvky do pole lze přidávat zavoláním funkce *push* na dané pole s prvkem k přidání v argumentu.

Každé pole má své interní vlastnosti, které jsou definovány a nastavovány na pozadí. Jednou ze základních je vlastnost *length*, s jejíž pomocí je možné zjistit velikost pole, tedy počet jeho prvků.

K jednotlivým prvkům pole se přistupuje použitím hranatých závorek za jménem pole, v nichž je udán index prvku pole. JavaScriptová pole začínají nultým prvkem, první prvek pole tedy bude mít index 0.

### 3.8 Smyčky

Protože JavaScript vychází z jazyka C, používá stejný syntax také pro smyčky.

Základní smyčkou je smyčka *for*. Na začátku jejího zápisu se vytvoří iterační proměnná, ve druhé části se určí podmínka provádění a ve třetí části se iterátor inkrementuje. Při spuštění se vykonává tělo cyklu, dokud je naplněna podmínka v hlavičce.

Smyčka *for* se využívá například pro procházení polí, kdy stále se inkrementující iterátor ve smyčce představuje proměnlivý index, a dovoluje tedy přistupovat k jednomu prvku pole za druhým.

Ze smyčky lze předčasně vyskočit pomocí klíčového slova *break*. Použitím *break* je smyčka okamžitě ukončena a není dokončena ani aktuální iterace.

Klíčovým slovem *continue* jde ukončit aktuální iterace bez přerušení celého cyklu.

## 4 JQUERY

Jednoduše řečeno je jQuery JavaScriptová knihovna (ne framework) pro použití na straně klienta při vývoji webových stránek. Jejím účelem je umožnit vývojářům vytvořit interaktivní stránky, které fungují napříč prohlížeči. To zahrnuje mimo jiné zpracování interakce uživatele, manipulaci stránky a provádění síťových požadavků. Než přišlo jQuery, byl vývoj webových stránek pouze za použití JavaScriptu výrazně složitější, jelikož různé prohlížeče podporovaly různé funkce, různé verze JavaScriptu a různé verze specifikace HTML. Proto nebylo jednoduchým úkolem vytváření webových stránek, které by fungovaly dobře na všech platformách. Cílem jQuery bylo skrytí některých z těchto rozdílů, aby se vývojáři mohli soustředit na přidávání funkcionality a nemuseli se tolik trápit problémy s kompatibilitou.

### 4.1 Základy jQuery

Knihovnu jQuery lze získat ze stránek <http://jquery.com> ve dvou verzích. Největším rozdílem mezi verzemi *jQuery 1.x* a *jQuery 2.x* je, že verze *1.x* podporuje prohlížeč Internet Explorer ve verzích 6, 7 a 8, zatímco verze *2.x* nikoliv. Tento rozdíl je velmi důležitý. Důvodem udržování dvou verzí je, že ačkoliv je verze *jQuery 1.x* větší a v některých případech pomalejší než *jQuery 2.x*, funguje i na starších verzích Internet Exploreru. Protože je tento prohlížeč ve verzi 8 stále velmi rozšířen, je v případě, že jsou očekáváni uživatelé s tímto prohlížečem, nutné použít *jQuery 1.x*. Většina funkcionality je udržována v obou verzích.

Pro použití na webové stránce se soubor s knihovnou jQuery vloží do HTML pomocí tagu *script*, stejně jako u jiných JavaScriptových souborů. Tag *script* odkazující na tento soubor se musí nacházet v HTML předtím, než je jeho funkcionalita vyžadována.

```
<script type="text/javascript" src="js/jquery-2.1.1.min.js"></script>
```

Obr. 4-1 odkaz na soubor s knihovnou jQuery

Jednou ze základních funkcí knihovny jQuery je velmi snadné přístupu k jednotlivým prvkům Document Object Modelu. Toto přístupu a vybírání prvků na stránce se provádí za pomoci CSS selektorů, znalost CSS je tedy pro použití této knihovny nezbytná. Samotný syntax tohoto přístupu vychází z funkce *jQuery*, jejímž parametrem je právě CSS selektor.

```
var resultList = jQuery("#resultList");
```

Obr. 4-2 vyhledání a uložení prvku stránky s *id resultList* do proměnné

Následně lze použít různých funkcí na práci s vybranými prvky. Například *html*, což je ekvivalent parametru *innerHTML* v JavaScriptu, jehož pomocí se nastaví HTML obsah prvku. Zjednodušenou variantou *html* je funkce *text*, která jen nahradí textuální obsah.

```
resultList.text("This is from jQuery.");
```

Obr. 4-3 použití funkce *text* na obsah proměnné *resultList*

Množství funkcí v jQuery, většinou sloužících pro nastavování hodnot, je implementováno tak, že pokud není udán argument dané funkce, bude aktuální hodnota namísto přepsání vrácena.

Jelikož je funkcí velmi mnoho, bude několik z nich popsáno v průběhu této kapitoly.

Samotný objekt jQuery, skrze nějž hledáme prvky na stránce, má svůj alias. Tímto aliasem je znak *\$*. Většina kódů stránek používajících jQuery je vystavěna právě pomocí tohoto aliasu. Jediným důvodem je, že je tento zápis kratší. Tento alias je možný, protože JavaScript umožňuje použití určitých symbolů jako jména proměnných a funkcí.

```
var resultList = $("#resultList");
```

Obr. 4-4 ekvivalentní zápis pro Obr. 4-2 pomocí aliasu

## 4.2 Události

Pomocí jQuery lze reagovat na interakce uživatele. Tyto události (angl. event) je možné odchytávat a jejich důsledky řídit zavoláním funkce *on* na prvek stránky. Prvním argumentem funkce *on* je jméno dané události, druhým pak callback funkce, která bude provedena po odchycení dané události.

Jedním ze základních typů událostí je *click*. Jak název napovídá, reaguje na kliknutí na daný prvek.

```
var resultList = $("#resultList");
resultList.text("This is from jQuery.");

var toggleButton = $("#toggleButton");
toggleButton.on("click", function(){
    resultList.toggle(500);
})
```

Obr. 4-5 odchyení události kliknutí na prvek s *id toggleButton*

Funkce *toggle* přepíná mezi zviditelněním a skrytím prvku. Pokud je prvek viditelný, bude touto funkcí skryt. Pokud je skryt, bude zviditelněn. Jeho parametrem je číselná hodnota udávající délku animace. Všeobecně jsou v jQuery časové intervaly udávány v milisekundách a stejně je tomu i v případě funkce *toggle*.

Problémem, který nemusí být na první pohled zřejmý, je fakt, že ne vždy se v praxi načte celá stránka před načtením kódu JavaScriptu nebo jQuery. V takovém případě se může stát, že tento kód nebude mít na stránku požadovaný efekt. Tomuto chování se dá předejít odchyčením speciálního typu události v jQuery. Knihovna jQuery dovoluje zabalit libovolný vytvořený HTML element nebo DOM samotný jako objekt jQuery, a to pomocí syntaxu jQuery. Lze tedy takto zabalit celý *document* objekt a na ten poté používat funkce jQuery. Funkcí takto často používanou je funkce *ready*. Ta dovolí spustit kód, který obsahuje funkce v jejím parametru, až jakmile je obsah celého *document* objektu připraven. Druhotnou výhodou použití této funkce je zabalení celého skriptu, a tím pádem jeho vytržení z globálního rámce. Nebudou si tak s jinými JavaScriptovými soubory navzájem překážet svými proměnnými a funkcemi.

```
$(document).ready(function(){
    //some code to be ran after the document is ready
})
```

Obr. 4-6 zapouzdření kódu pomocí funkce *ready*

Velmi důležité je mít na paměti, že jQuery funkce jsou použitelné pouze na jQuery objekty. Proto je případné jiné objekty nutné zabalit stejně jako objekt *document* z Obr. 4-6.

### 4.3 Zabalené sady

Veškeré vyhledávání v objektu *document* má za následek vytvoření takzvané zabalené sady. Zabalená sada je množina všech prvků stránky, jimž odpovídá CSS selektor funkce *jQuery* (*\$*). Libovolná *jQuery* funkce zavolaná na tento výsledek potom bude, stejně jako v CSS, provedena vůči každému prvku z této sady.

Toto může být zvláště užitečné pro funkci *css*. Prvním parametrem této funkce je CSS vlastnost, která má být nastavena, a druhým její hodnota. Důvodem, proč použít funkci *css*, a ne přímo zadat tyto hodnoty do CSS, je, že takto lze měnit styly prvků právě v závislosti na interakci uživatele.

```
$("header nav li").css("font-weight", "bold");
```

Obr. 4-7 použití funkce *css* na sadu prvků

### 4.4 Modifikace dokumentu

Velmi často není potřeba nastavovat nebo upravovat všechny prvky zabalené sady najednou, ale jeden po druhém. K tomu lze v *jQuery* využít funkci *each*. Je přístupná jako funkce objektu *jQuery* v podobě jednoduché smyčky, která bude danou sadu procházet prvek po prvku. Jako první parametr přijímá sadu, kterou má iterovat, a jako druhý parametr callback funkci, která má být provedena pro každý prvek. Callback funkce běžně očekává dva parametry, z nichž první je číslo prvku a druhý představuje aktuální iterovaný prvek kolekce.

```
var resultList = $("#resultList");
resultList.empty();
$.each(results, function(i, item){
    var newResult = $("

Obr. 4-8 demonstrační příklad práce s prvky zabalené sady



Pro demonstraci bude použit kód z Obr. 4-8.



Funkce empty vyprazdňuje veškerý obsah elementu stránky, v tomto případě prvku s id resultList.



Funkce each bude iterovat nad objektem results. Parametr i představuje číslo iterovaného prvku a item odkazuje na prvek samotný. Pro každý z těchto prvků tak provede vytvoření HTML řetězce, který zabalí jako objekt jQuery. Pro každý takto vytvořený nový řetězec newResult nastaví funkci hover. Funkce hover určuje, co se bude dít, pokud myš uživatele vstoupí na daný prvek, nebo jej opustí. Tato funkce přijímá jako parametry dvě funkce. První funkce se provede při vstupu myši na prvek a druhá při jeho opuštění.



Jelikož se jedná o callback funkce, dovolí jQuery použít klíčové slovo this. To označuje daný prvek v DOMu, v tomto případě prvek, nad kterým se nachází myš.



Takto připravený prvek newResult je pomocí funkce append připojen na konec prvku resultList.



## 4.5 Síťové požadavky skrz jQuery



Získávání dat z internetu je další z oblastí, pro kterou má jQuery užitečné nástroje. Tato data jsou často nabízena servery ve formátu JSON, což je zkratka pro JavaScript Object Notation.


```

Je to unifikovaný způsob zápisu objektu posílaného po síti. Veškeré názvy vlastností a prvků jsou uvedeny v uvozovkách, ale celkový syntax je jinak velmi podobný zápisu objektů a polí z JavaScriptu.

Pro vytvoření síťového požadavku vůči serveru se používají metody *get* a *post* volané přímo na objektu *jQuery*.

Metoda *\$.get* přijímá jako první parametr URL adresu, vůči které má být požadavek vykonán. Druhým parametrem je callback funkce, jejímž parametrem je výsledek funkce *get*. V rámci síťových požadavků je nutné si uvědomit, že se jedná o asynchronní požadavky. To znamená, že se nečeká na jejich výsledek před provedením kódu, který následuje.

```
var gitHubSearch = "https://api.github.com/search/repositories";
$.get(gitHubSearch, function(){
    //console.log(r.items.length);
    displayResults(r.items);
});
```

Obr. 4-9 jednoduchý zápis funkce *get*

Při síťových požadavcích se ale může stát, že požadavek nebude z nějakého důvodu dokončen. To může nastat například při výpadku dotazovaného serveru nebo při chybě v komunikaci. Proto lze využít alternativní syntax, kdy URL je jediným parametrem funkce *get*. Na objekt, který tato funkce vrátí, je možno použít mnoho funkcí, z nichž některé slouží právě na řízení chování v případě chybných výsledků. Taková funkce *success* přijímá za parametr callback funkci, která bude provedena v případě řádného výsledku funkce *get*. Na funkci *success* lze potom řetězit další funkce, jako například *fail*. Ta naopak provede kód své callback funkce pouze v případě, že požadavek nebude řádně dokončen. Toto řetězení může pokračovat funkcí *done*, jejíž callback funkce bude provedena v případě úspěchu i neúspěchu.

```
var gitHubSearch = "https://api.github.com/search/repositories";
$.get(gitHubSearch)
  .success(function(r){
    //console.log(r.items.length);
    displayResults(r.items);
  })
  .fail(function(error){
    console.log("Failed to query GitHub");
  })
  .done(function(){
    //
  });
```

Obr. 4-10 zápis funkce *get* včetně zřetěžených funkcí *success*, *fail* a *done*

## 4.6 Zpracování formulářů

Pomocí jQuery je možné zpracovávat událost odeslání formuláře.

Při odeslání formuláře je možné provádět s údaji z jednotlivých polí různé operace, aniž by bylo provedeno odeslání požadavku na adresu z HTML parametru *action* formuláře. Toho může být dosaženo tak, že při odchycení události *submit* funkcí *on* bude v callback funkci uvedena návratová hodnota *false*. Toto je velmi běžně používaná technika, jelikož umožňuje místo přesměrování na URL z parametru *action* například zobrazení nových prvků na stránce na základě údajů z formuláře. Také je tímto způsobem možné provést validaci jednotlivých polí formuláře.

## 5 JAVASCRIPTOVÉ FRAMEWORKY A KNIHOVNY

JavaScript jako programovací jazyk dovoluje vytvářet i velmi složitou funkcionalitu. Vytvoření takové funkcionality ale mnohdy vyžaduje hluboké porozumění samotnému jazyku, je časově náročné a výsledný kód nemusí být dokonalý.

Ve vývojářské praxi jsou časová náročnost a správná funkčnost velmi důležitými parametry a proto jsou webovými vývojáři využívány JavaScriptové frameworky a knihovny. Ty nabízejí předpřipravenou funkcionalitu například ve formě funkcí a jsou většinou dobře dokumentovány. Další výhodou je, že díky jejich používání desetitisíce uživatelů a programátorů na obrovském množství různých zařízení jsou odkrývány chyby a nedostatky, které jsou často samotnou komunitou opravovány. Frameworky tak dělají práci s JavaScriptem rychlejší, zábavnější a z business perspektivy bezpečnější.

Již bylo zmíněno, že vývoj internetových stránek je historicky v rukou nevlivnějších hráčů na trhu. Ve dnešní době sociálních sítí jsou to právě společnosti tyto sítě provozující, kdo určuje laťku.

Následující frameworky a knihovny jsem vybral na základě poznatků z mé profesní praxe ve společnosti CN Group s.r.o.. Na vzorku cirká padesáti projektů na vývoj webových aplikací v rámci této společnosti jasně vyplouvají na povrch jako tři nejpoužívanější frameworky Angular JS od společnosti Google, React.js od společnosti Facebook a Bootstrap od společnosti Twitter.

Některé z výhod používání těchto frameworků jsou totožné, jelikož podstatou frameworků je úspora času a nákladů. Sdílenou výhodou těchto tří frameworků je například velikost zdrojových kódů, které jsou mnohonásobně menší v porovnání s kódem, který frameworků nevyužívá. Další takovou výhodou je popularita těchto frameworků. Od toho se odvíjí vysoká dostupnost praktických příkladů, řešení problémů a výukových kurzů.

### 5.1 Angular - Google

Angular je MVC framework velmi vhodný pro vytváření takzvaných Single Page Applications, tedy aplikací, které se tváří jako jediná webová stránka s proměnlivým obsahem. Je to jedno velké balení obsahující všechno, co by taková typická aplikace mohla potřebovat.

Jednou z nejpoužívanějších funkcí webových aplikací jsou formuláře a Angular nabízí možnosti, jak s těmito formuláři pracovat velmi jednoduše a zvýšit jejich interaktivitu. Jsou jimi takzvané vazby. Sem spadá typická jednosměrná vazba, která umožňuje specifikovat část stránky, kde bude zobrazena nějaká hodnota a když se tato hodnota změní, změní se i zobrazená hodnota. Angular podporuje i dvousměrnou vazbu, kdy je možné například při zadání hodnoty do pole *input*, které je Angularem hlídáno, tuto hodnotu zpracovat a na jejím základě upravit na webové stránce další prvky. Například by takto bylo možné spočítat a zobrazit uživateli při znalosti aktuálního stavu prostředků na bankovním účtu výši prostředků po provedení transakce pouze zadáním strhávané hodnoty do formulářového pole. A to bez odeslání jakéhokoliv síťového požadavku. Takovéto dvousměrné vazby jsou typickým prvkem stránek na bázi Angularu a tvoří jeho nedílnou součást a jednu z nejužitečnějších výhod.

V čem ale Angular vyniká nad konkurenčními frameworky je testovatelnost. Komponenta *ngMock* umožňuje izolovat Angular od backendového serveru. Nástroj *Protractor* dovoluje spustit Angular v prohlížeči a automatizovaně testovat funkčnost jednotlivých prvků. Dalším důležitým nástrojem pro testování je *Karma*. Jeho pomocí lze spouštět automatické testy v různých prohlížečích záraz a je tak populární, že je využíván i na projektech, které samotný Angular nevyužívají.

## 5.2 React - Facebook

React.js je JavaScriptová knihovna na vytváření uživatelských rozhraní, obzvláště excelující ve vytváření rozhraní responzivních, tedy přizpůsobujících se velikosti zařízení, na kterém jsou zobrazovány. Oproti frameworku Angular se stará pouze o to, co uživatel vidí. Používá zápis takzvaného JSX, což je mix JavaScriptu a HTML v jednom kódu. Takový kus kódu se nazývá komponenta a dělá kód vysoce přehledným, členitelným a umožňuje tvorbu kódu bez přepínání mezi okny s HTML a JavaScriptem. Vzhled celé stránky je možné renderovat na serveru, což umožňuje téměř okamžité zobrazení stránky.

Jelikož responzivní design je v Reactu na prvním místě, vznikl pro něj dceřinný projekt React Native, který umožňuje pomocí kódu psaného v JavaScriptu/JSX vytvářet nativní aplikace pro mobilní zařízení. Ve verzi 0.26 aktuálně podporuje pouze platformy Android a iOS. Pro případy, kdy použití knihovny React nestačí na vytvoření určité funkcionality, je možné tento kód rozšířit o nativní kód každé z platforem.

### 5.3 Bootstrap - Twitter

Stejně jako React je Bootstrap frameworkem zaměřeným především na tvorbu responzivních stránek s ještě silnějším důrazem na mobilní zařízení. Podstatou responzivity v Bootstrapu je použití dvanáctisloupcového návrhu stránky. Jednotlivým prvkům stránky se pak pomocí CSS tříd Bootstrapu určí, kolik sloupců z řádku budou zabírat při různých velikostech prohlížeče.

Svou koncepcí Bootstrap dovoluje pomocí kombinace svých vestavěných CSS tříd, JavaScriptových funkcí a komponent vytvářet responzivní design webových stránek velmi rychle. Tento design zároveň bude fungovat na všech moderních prohlížečích včetně mobilních zařízení.

## **II. PRAKTICKÁ ČÁST**

## 6 ÚVOD

V praktické části této práce bude vytvářena webová aplikace, umožňující zobrazování a vyhledávání projektů z portálu <https://github.com/> s názvem GitHub Search.

Postup bude principiálně kopírovat teoretickou část této práce. Bude tedy vytvořen základní HTML dokument, který následně obohatí CSS a JavaScript. Výsledná stránka bude v posledním kroku modernizována a přepsána do JavaScriptového frameworku Twitter Bootstrap.

Tyto stránky budou vytvářeny pomocí textového editoru Sublime Text 3 a bude k jejich zobrazení používán prohlížeč Mozilla Firefox ve verzi 45, nicméně lze použít libovolný textový editor a libovolný moderní prohlížeč.

## 7 HTML

V této části bude vytvořen základní HTML dokument stránky GitHub Search. HTML bude ze své koncepce obsahovat pouze strukturální prvky stránky a nebude obsluhovat vzhled a rozložení.

### 7.1 Základní HTML dokument

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Github Search</title>
5    </head>
6    <body>
7      <header>
8        <div></div>
9        <h1>Github Search</h1>
10       <div>This is a webpage allowing search in GitHub projects.</div>
11       <nav>
12         <ol>
13           <li><a href="/index.html">Home</a></li>
14           <li><a href="/contact.html">Contact</a></li>
15           <li><a href="/about.html">About</a></li>
16         </ol>
17       </nav>
18     </header>
19     <section>
20       <p>This is a <em>sample site</em>, that shows information
21         from <a href="http://github.com">GitHub</a>
22         and <span>makes kittens</span> <strong>happy</strong>.
23       </p>
24     </section>
25     <footer>
26       &copy; 2016 Kitten Kennel Antarctica
27     </footer>
28   </body>
29 </html>
```

Obr. 7-1 výchozí HTML dokument

Na Obr. 7-1 je zobrazen výchozí HTML dokument. Jeho začátek je označen speciálním tagem *!doctype html*, z nějž prohlížeč pozná, o jaký typ dokumentu se jedná. Celý zbytek dokumentu je obsažen v tagu *html*. Dále je dokument rozdělen na dvě části - *head* a *body*.

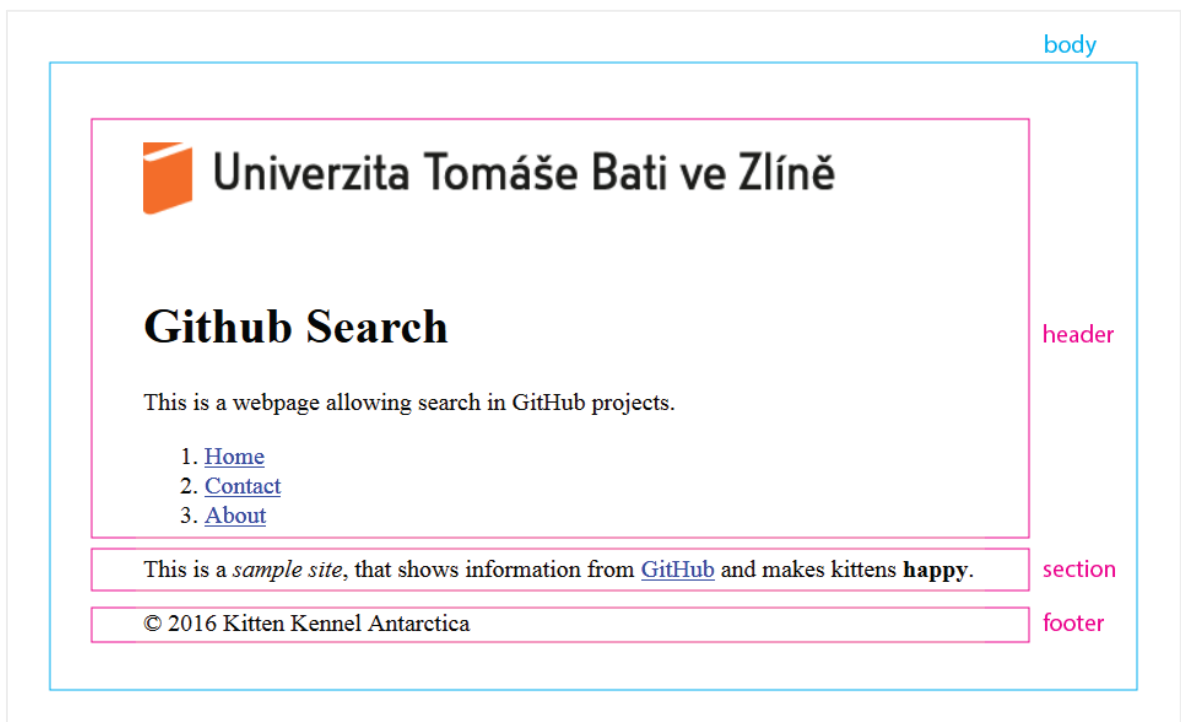
Část *head* obsahuje titulek stránky *title*, který bude zobrazen v popisu prohlížečového okna. Pod tímto jménem by také byla uložena případná záložka v prohlížeči. Sekce *head* není jinak v prohlížeči viditelná.

Část *body* je prezentace stránky, kterou uživatel za běžných okolností vidí. Je rozdělena na tři podčásti označené sémantickými tagy *header*, značící záhlaví stránky, *section*, označující sekci obsahu, a *footer*, který představuje patičku dokumentu. Tyto tagy se chovají jako běžné tagy *div*, jsou tedy blokovými elementy.

První *div* hlavičky obsahuje Obr. s logem UTB. Je uložen na adrese, kterou udává atribut *src* tagu *img*. Následuje nadpis *h1*, používaný pro označení účelu stránky, a za ním další *div* s krátkým popisem stránky. Následující prvek *nav* je dalším sémantickým tagem, označující navigační část stránky. Navigace bude zajištěna trojicí odkazů *a* jako prvků *li* číslovaného seznamu *ol*.

Tag *section* označuje sekci, tedy část stránky, kde se bude nacházet její vlastní obsah, a není tak hlavičkou ani patičkou dokumentu. Obsahuje odstavec *p* s textem. V rámci textu jsou použity tagy *em* a *strong* pro změnu důrazu daných slov, tag *a* pro vytvoření odkazu na stránku <http://github.com> a řádkový element *span*, který pro tuto chvíli nemá větší význam a slouží pouze pro demonstraci tohoto typu elementu (zůstane na řádku a nerozhodí strukturu odstavce).

Část *footer* obsahuje patičku tohoto dokumentu, která sestává z HTML entity *&copy;* představující znak copyright a textu.



Obr. 7-2 grafická reprezentace kódu z Obr. 7-1 s barevně označenými částmi

## 7.2 Formulář pro vyhledávání

```
<section>
  <p>This is a <em>sample site</em>, that shows information
    from <a href="http://github.com">GitHub</a>
    and <span>makes kittens</span> <strong>happy</strong>.</p>
  </p>
  <form>
    <label for="searchPhrase">Search Phrase:</label>
    <input name="searchPhrase" id="searchPhrase" /><br />
    <input type="checkbox" name="useStars" id="useStars" checked="true"/>
    <label for="useStars">Use stars?</label><br />
    <label for="langChoice">Language: </label><br />
    <select name="langChoice" id="langChoice">
      <option>All</option>
      <option>JavaScript</option>
      <option selected>C#</option>
      <option>Java</option>
      <option>PHP</option>
    </select>
    <br>
    <input type="submit" value="search" /><br />
  </form>
</section>
```

Obr. 7-3 sekce stránky obsahující formulář

V tomto kroku bude do stránky, přesněji do prvku *section*, přidán formulář, kterým bude později možné filtrovat výsledky – projekty z portálu GitHub.

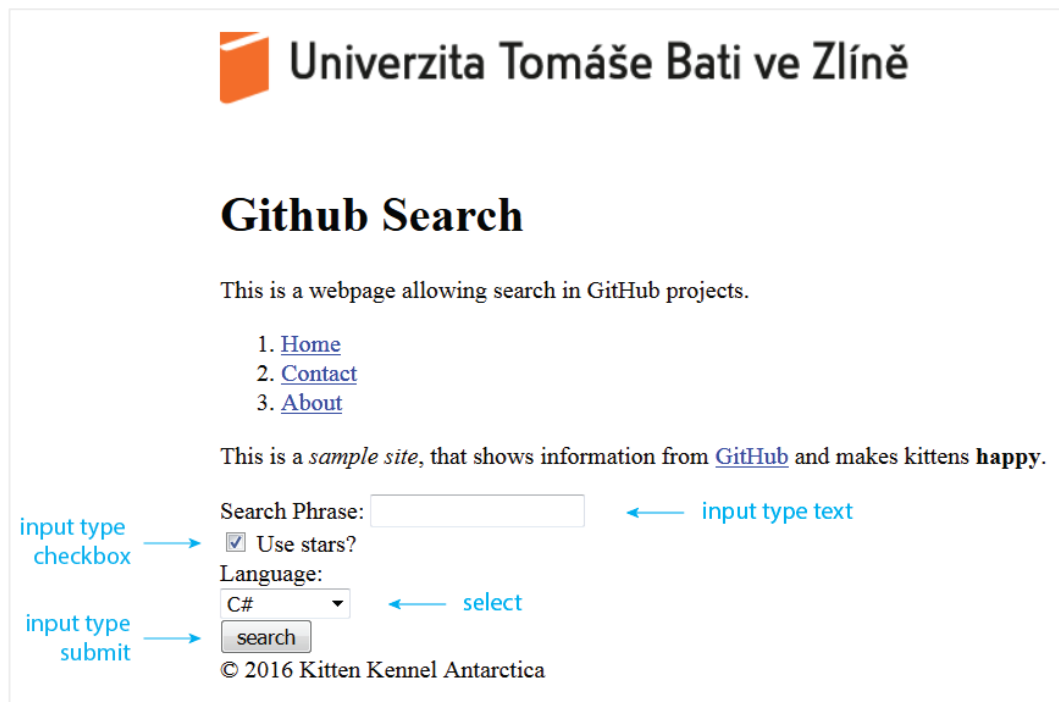
Tento formulář se skládá z prvků typu *input*, *select* a *button* s parametry *name* a *id* majícími stejnou hodnotu. Každý z těchto elementů má svůj popisek *label*, který je na svůj přiřazený prvek formuláře vázán přes argument *id* prvku pomocí argumentu *for*.

Prvky *input* jsou různých typů a budou plnit různé role:

- *input id="searchPhrase"* bude udávat hledanou frázi. Bez udání typu je toto nastavení výchozí, tedy *text*, a je tedy textovým polem.
- *input id="useStars"* je typu *checkbox* a je tak reprezentován zaškrtačacím políčkem. Jeho booleanovská hodnota zaškrtnuto-nezaškrtnuto bude instruovat vyhledávání, zda má řadit výsledky dle hodnocení. Parametr *checked* udává jeho výchozí stav, tedy zda je při načtení zaškrtnuto, či nikoliv.
- *input type="submit"* je odesílací tlačítko formuláře. Text, který bude obsahovat, je udán hodnotou parametru *value*.

Dalším prvkem formuláře je výběrací pole *select*. Toto pole obsahuje pět možností výběru *option*, každou pro jednu položku, ze kterých bude možné vybírat. Toto pole bude mít na

starosti filtraci výsledků podle zvoleného programovacího jazyka. Jeden z elementů *option* je označen parametrem *selected* a udává tak výchozí stav tohoto výběrového pole.



Obr. 7-4 vzhled stránky včetně kódu z Obr. 7-3

V průběhu dalších kapitol do něj budou přidávány další úpravy, tak aby právě řešené téma dávalo smysl.

Stav HTML kódu po kapitole 7. HTML praktické části je vidět na příloze P II.

## 8 CSS

Pro stylování stránky budou použity kaskádové styly `css`. Do HTML budou přidány pomocí tagu `link` v sekci `head` HTML dokumentu. Tento prvek má argument `rel`, který udává typ propojení, a argument `href` ukazující na soubor CSS styly obsahující.

```
* {  
    padding: 0;  
    margin: 0;  
}
```

Obr. 8-1 vynulování  
vlastností `padding` a  
`margin`

Jelikož ve výchozím nastavení nejsou pro některé prvky vlastnosti `padding` a `margin` nulové, je vhodné je předem nastavit na nulu jako na Obr. 8-1.

Dále bude obstaráno centrování obsahu doprostřed prohlížečového okna. Toho je dosaženo tak, že se kompletní obsah HTML prvku `body` zabalí do prvku `div` a ten se styluje. V našem případě bude toto stylování připadat na třídu `container`.

```
13 <body>  
14   <div class="container">  
56   </div>  
57 </body>
```

Obr. 8-2 zabalení obsahu `body` do `div` s třídou `container`

Tato třída se následně styluje jako na Obr. 8-3. Vycentrování zajistí vlastnost `margin`. Toto centrování bude fungovat pouze při udání šířky elementu.

```
.container {  
    /* běžná hodnota šířky kontejneru obsahujícího celou stránku,  
    vypadá dobře na většině monitorů */  
    width: 989px;  
    /* Hodnota auto u vlastnosti margin znamená,  
    že prohlížeč přidá na obě strany(nyní vlevo a vpravo)  
    stejnou hodnotu margin.  
    V praxi bude prvek vycentrován horizontálně. */  
    margin: 0 auto;  
    background-color: white;  
    padding: 10px;  
}
```

Obr. 8-3 styl třídy `container`

Aby byly obrysy stránky zřetelnější, přidáme styly pro prvek `body`. Ten je rodičem všem prvkům včetně `divu` třídy `container`, tedy je v pozadí celé stránky. Nastavíme vzhled písma celé stránky a barvu pozadí podle Obr. 8-4.

```
body {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 14px;
  background-color: #194C43;
}
```

Obr. 8-4 styl prvku `body`

Každou z částí `header`, `section` a `footer` ostylujeme podle Obr. 8-5.

```
header, section, footer {
  border: solid 1px lightgray;
  border-radius: 3px;
  padding: 10px;
}


section {
  margin: 10px 0;
}
```

Obr. 8-5 styl prvků `header`, `section`, `footer`

Těmto prvkům je tak přidán rámeček a jejich obsah je odsazen od okrajů. Prvek `section` je upraven, aby byl shora a zdola odsazen od ostatních prvků.

Další úpravou bude přesunutí prvku `nav` v HTML tak, aby bylo možné správně využít stylovací vlastnost `float`. Provedení naznačuje Obr. 8-6.

```
<header>
  <div><a href="/co
      <li><a href="/ab
    </ol>
  </nav>
</header>
```



```
<header>
  <nav>
    <ol>
      <li><a href="/inc
      <li><a href="/cor
      <li><a href="/abc
    </ol>
  </nav>
  <div> a zpracování výsledků tohoto požadavku.

Vložení JavaScriptového souboru je možné skrz HTML tag *script* odkazující na daný JavaScriptový soubor.

Vložíme takto dva odkazy do hlavičky HTML dokumentu viz. Obr. 9-1.

```
<script type="text/javascript" src="https://code.jquery.com/jquery-2.2.4.min.js"></script>
<script type="text/javascript" src="js/index.js"></script>
```

Obr. 9-1 vložení JavaScriptových souborů do HTML

Jako první je vložena knihovna jQuery, jako druhý script osbluhující tuto stránku. Knihovnu jQuery lze získat z oficiálních stránek <http://jquery.com/> nebo z internetové zdroje jako na Obr. 9-1.

Na začátek obslužného scriptu uvedeme funkci *ready* volanou na jQuery objekt *document*. Argumentem této funkce je callback, který se vykoná až po načtení celého HTML dokumentu. Tento callback tedy obsahuje vše, co chceme na stránce provádět.

Jelikož nevíme, kolik GitHub projektů bude mít za následek jejich hledání, implementujeme tlačítko dovolující je skrýt. Do prvku *#results* přidáme tlačítko *button* s id *#toggleButton* a *div* s id *#resultList*.

```
var resultList = $("#resultList");

var toggleButton = $("#toggleButton");
toggleButton.on("click", function(){
    resultList.toggle(500);

    if (toggleButton.text() == "Hide") toggleButton.text("Show");
    else toggleButton.text("Hide");
})
```

Obr. 9-2 obsluha výsledky skrývajícího tlačítka *#toggleButton*

V JavaScriptu provedeme navázání funkce na skrytí (a znovuzobrazení) *#resultList* a změnu textu tlačítka *#toggleButton* na událost kliknutí na toto tlačítko podle Obr. 9-2.

Implementujeme funkci *displayResults*, která se bude starat o zobrazení výsledků hledání. API služby GitHub vrací mimo jiných vlastností jednotlivých projektů jejich jméno *name*, programovací jazyk *language* a jméno majitele *owner.login*. Iterováním přes všechny výsledky vytvoříme pro každý z těchto projektů jeho vlastní *div*, který následně zobrazíme

na stránce. Než k tomu však dojde nastavíme chování prvku při události najetí myši pomocí funkce *hover*. Její callback funkce udávají, co se má stát při vstupu a odjetí myši z elementu.

Kód této funkce je na Obr. 9-3.

```
function displayResults(results){
    var resultList = $("#resultList");
    resultList.empty();
    $.each(results, function(i, item){
        var newResult = $("<div class='result'>" +
            "<div class='title'>" + item.name + "</div>" +
            "<div>Language: " + item.language + "</div>" +
            "<div>Owner: " + item.owner.login + "</div>" +
            "</div>");

        newResult.hover(function(){
            $(this).css("background-color", "lightgray");
        },function(){
            $(this).css("background-color", "transparent");
        })

        resultList.append(newResult);
    })
}
```

Obr. 9-3 funkce *displayResults*

Nyní můžeme přistoupit ke kódu samotného požadavku. Ten vyrobíme podle Obr. 9-4.

Pro jednodušší identifikaci, zavedeme *id="gitHubSearchForm"* pro náš formulář.

Celý požadavek bude odeslán po zachycení události odeslání formuláře *#gitHubSearchForm*. To bude provedeno při stisknutí tlačítka *input type="submit"* zavoláním callback funkce tohoto zachycení. Zároveň definicí návratové hodnoty callback funkce zajistíme, že formulář nebude odeslán sám o sobě, a bude nám umožněno jeho obsah zpracovat. V první řadě zjistíme hodnoty jednotlivých prvků formuláře, abychom je mohli přidat do požadavkového řetězce URI, reprezentovaného proměnnou *gitHubSearch*, na který budeme hodnoty odesílat. Dále pokračujeme pouze pokud má textové pole *#searchPhrase* nějakou hodnotu. Lehkou validací dalších polí přidáváme jednotlivé části dotazu k URI a následně pomocí funkce *.get* požadavek odesíláme. V případě úspěšného požadavku zobrazíme výsledky pomocí funkce z Obr. 9-3, jinak bude do *console* vypsána chyba.

Zbývá nastýlovat množinu výsledků.

Kompletní obsah JavaScriptového souboru po této kapitole, stejnětak jako úpravy HTML a CSS lze nalézt v přílohách P V a P VI.

```
$("#githubSearchForm").on("submit", function(){  
  
    var searchPhrase = $("#searchPhrase").val();  
    var useStars = $("#useStars").prop('checked');  
    var langChoice = $("#langChoice").val();  
  
    if (searchPhrase) {  
        resultList.text("Performing search...");  
  
        var gitHubSearch = "https://api.github.com/search/repositories?q=" +  
            encodeURIComponent(searchPhrase);  
  
        if (langChoice != "All"){  
            gitHubSearch += "+language:" + encodeURIComponent(langChoice);  
        }  
  
        if (useStars){  
            gitHubSearch += "&sort=stars";  
        }  
        $.get(gitHubSearch)  
            .success(function(r){  
                displayResults(r.items);  
            })  
            .fail(function(error){  
                console.log("Failed to query GitHub. Error: " + error);  
            })  
    }else{  
        resultList.text("No search phrase set.");  
    }  
  
    return false;  
})
```

Obr. 9-4 síťový požadavek na získání hledaných projektů

## 10 BOOTSTRAP

Nyní tuto stejnou stránku vytvoříme v responzivním provedení pomocí Twitter Bootstrap.

Předpokladem pro použití této knihovny je knihovna jQuery, kterou tato využívá.

Abychom mohli tuto knihovnu začít využívat, musíme ji vložit do HTML jako každou jinou JavaScriptovou knihovnu pomocí značky *script*. Jelikož budeme chtít její funkce použít, teprve až bude jisté, že dokument je načten, přidáme ji na konec elementu *body* dokumentu. Druhou částí nezbytnou pro použití Bootstrapu jsou jeho CSS styly. Ty přidáme do hlavičky klasicky pomocí tagu *link*. Většina použitých částí kódu je dostupná na stránkách Bootstrap [1] včetně dokumentace.

```
<!DOCTYPE html>
<html>
  <head>
    <title>GitHub Search</title>

    <!-- bootstrap css -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
crossorigin="anonymous">
  </head>
  <body>

    <!-- jquery -->
    <script type="text/javascript" src="js/jquery-2.2.3.min.js"></script>
    <!-- bootstrap js -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxs1yVq0tnepnHVP9aJ7xS"
crossorigin="anonymous"></script>

    <script type="text/javascript" src="js/index.js"></script>
  </body>
</html>
```

Obr. 10-1 instalace Bootstrapu

Jako první vytvoříme *div* třídy *container*. Tento *div* je základním prvkem při rozložení stránky pomocí Bootstrapu a bude obsahovat celý viditelný obsah stránky.

V tomto *divu* vytvoříme navigační panel podle Obr. 10-2.

## 10.1 Navigační panel

```

<nav class="navbar navbar-default" id="navbar">
  <div class="container-fluid">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
        data-target="#collapseNavRight" aria-expanded="false">
      <span class="sr-only">Toggle navigation</span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
    </button>
    <a class="navbar-brand" href="#">GitHub Search</a>
  </div>

  <!-- Collect the nav links, forms, and other content for toggling -->
  <div class="collapse navbar-collapse" id="collapseNavRight">
    <ul class="nav navbar-nav navbar-right">
      <li><a href="/index.html">Home</a></li>
      <li><a href="/contact.html">Contact</a></li>
      <li><a href="/about.html">About</a></li>
    </ul>
  </div><!-- /.navbar-collapse -->
</div><!-- /.container-fluid -->
</nav>

```

Obr. 10-2 navigační panel

Navigační panel je v Bootstrapu konstruován jako množina vnořených *div* prvků s třídami *navbar*, které popisují jeho jednotlivé části. Tlačítko *button* v *div.navbar-header* je viditelné pouze při malých rozlišeních a nahrazuje tehdy skupinu navigačních prvků v *div.navbar.navbar-collapse*. V případě Obr. 10-2 obsahuje navigační panel popis stránky *a.navbar-brand* a tři navigační odkazy v seznamu *ul.nav*.

## 10.2 Jumbotron

Za navigační panel přidáme komponentu Bootstrapu s názvem *jumbotron*, který je aplikován pomocí stejnojmenné třídy. Vytvoří na stránce velký poutací prvek.

```

<div class="jumbotron">
  <h1>Github Search</h1>
  <p>This is a webpage allowing search in GitHub projects.</p>
</div>

```

Obr. 10-3 prvek *jumbotron*

Dále na stránce potřebujeme vytvořit místo pro formulář a výsledky. K tomuto účelu využijeme mřížkový systém Bootstrapu. Ten spočívá v rozdělení obsahu na řádky pomocí třídy *row* a v rámci nich označení prvků třídami *col-[x]-[y]*. Pomocí těchto tříd *col* může být určena prvku pomocí hodnoty *[y]* jeho šířka v počtu sloupců až do čísla 12. Hodnota *[x]* může nabývat hodnot *xs*, *sm*, *md* a *lg*, což jsou zkratky pro extra-small, small, medium a

large. Tyto hodnoty udávají pro jakou velikost zobrazovacího zařízení bude hodnota  $[y]$  platit. Tomuto systému se říká dvanáctisloupcový design.

```
<div class="row">
  <div class="col-xs-12 col-md-5" id="formDiv">
  </div>
  <div class="col-xs-12 col-md-7" id="results">
  </div>
</div>
```

Obr. 10-4 použití tříd *row* a *col*

Prvek *div#formDiv* bude obsahovat formulář pro vyhledávání, prvek *div#results* výsledky hledání. Třídy *col* jim nastavené říkají, že na velmi malých až malých zařízeních bude prvek zobrazen přes celou šířku *row* (tedy 12 sloupců) a na středních a velkých zařízeních se tyto *divy* podělí o šířku v poměru 5:7.

### 10.3 Formulář

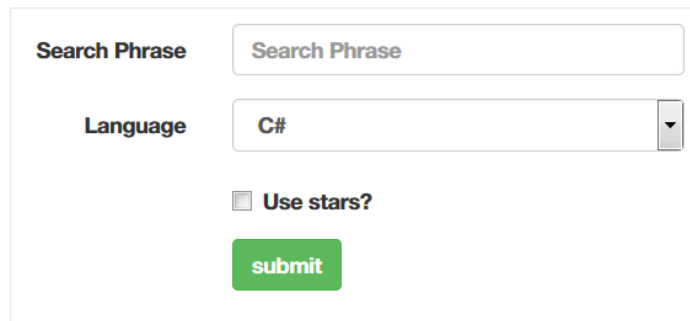
Vytvoříme formulář uvnitř *div#formDiv* podle Obr. 10-5.

```
<form class="form-horizontal" id="gitHubSearchForm">
  <div class="form-group">
    <label for="searchPhrase" class="col-sm-4 control-label">Search Phrase</label>
    <div class="col-sm-8">
      <input type="text" class="form-control" name="searchPhrase"
        id="searchPhrase" placeholder="Search Phrase">
    </div>
  </div>
  <div class="form-group">
    <label for="searchPhrase" class="col-sm-4 control-label">Language</label>
    <div class="col-sm-8">
      <select class="form-control">
        <option>All</option>
        <option>JavaScript</option>
        <option selected>C#</option>
        <option>Java</option>
        <option>PHP</option>
      </select>
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-4 col-sm-8">
      <div class="checkbox">
        <label>
          <input type="checkbox" name="useStars" id="useStars"> Use stars?
        </label>
      </div>
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-4 col-sm-8">
      <button type="submit" class="btn btn-success">submit</button>
    </div>
  </div>
</form>
```

Obr. 10-5 formulář vyhledávání

Prvek *form* je třídy *form-horizontal*, která říká, že má tento formulář zobrazovat popisky a jejich pole na řádku vedle sebe. Formuláři zachováme stejné *id* jako v předchozích

kapitolách. Ve formuláři jsou 4 skupiny třídy *form-group*, které značí shluk ovládacího prvku a jeho popisku. Obě třídy *form* a *form-group* plní roli třídy *row* a umožňují tak své přímé potomky stylovat pomocí tříd *col*. Ovládací prvky nesou třídu *form-control*, zatímco jejich popisky jsou třídy *control-label*. Ovládací prvky formuláře jsou stejných typů a mají stejné identifikátory, jako v předchozích kapitolách. Odlišné je odesílací tlačítko, které je třídy *btn*. Třída *btn-success* určuje zbarvení tlačítka na zeleno.



Obr. 10-6 vzhled formuláře

Doplníme vlastní pole pro výsledky *div#resultList* a skryvací tlačítko do *div#results*.

```
<div class="col-xs-12 col-md-7" id="results">  
  <button id="toggleButton" class="btn btn-default">Hide</button>  
  <div class="row" id="resultList">This is where results will appear eventually...</div>  
</div>
```

Obr. 10-7 modifikace *div#resultList* na třídu *row*

## 10.4 Zobrazení vrácených výsledků

V aktuální formě při použití skriptu z přílohy P V bude stále fungovat posílání síťového požadavku a přijímání výsledků. Ty by však bylo dobré řádně nastylovat pomocí Bootstrapu.

Modifikací *div#resultList* na třídu *row* získáme možnost použití tříd *col* na výsledky vyhledávání, mající třídu *result*. Těmto výsledkům nastavíme ve skriptu při jejich tvorbě třídu *col-xs-12* a *col-md-4*. Na zařízeních středních a větších se budou na řádku  $12/4 = 3$  výsledky. Pokud bude výsledků více než 3, budou další zalomeny na další řádek, protože je součtem sloupců, které zabírají, přesáhnut maximální počet 12 sloupců v řádku. Toto je velmi důležitá vlastnost dvanáctisloupcové mřížky Bootstrapu.

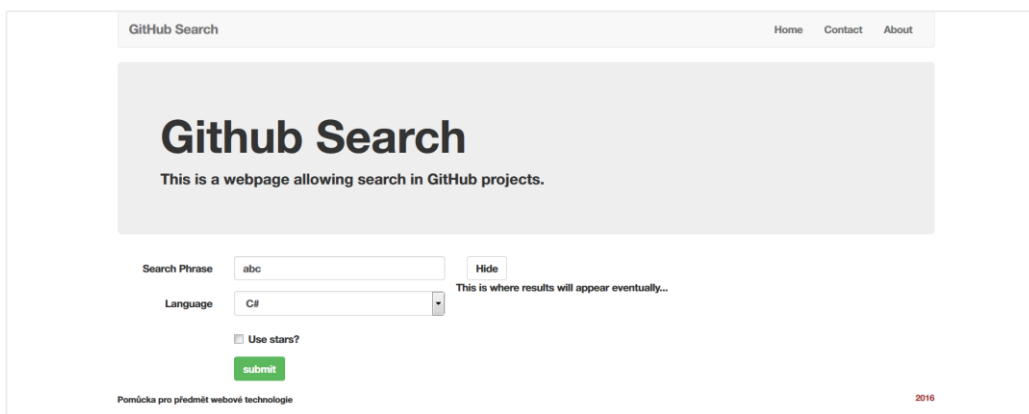
## 10.5 Patička

Na řadě je přidání patičky. Ta bude definována HTML z Obr. 10-8.

```
<footer id="footer">
  <small>Pomůcka pro předmět webové technologie
  <span class="pull-right text-danger">2016</span></small>
</footer>
```

Obr. 10-8 patička stránky

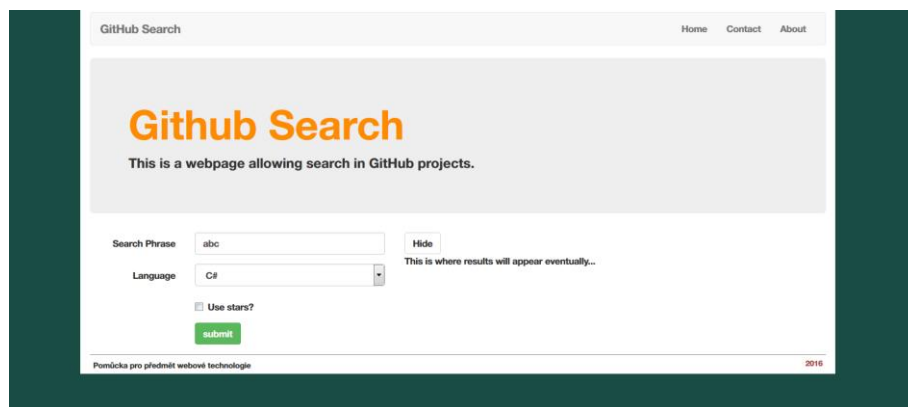
Bude mít id *footer*. Na zmenšení velikosti textu použijeme značku *small*. Třída *pull-right* je jednou z pomocných tříd Bootstrapu a má za úkol označený prvek posunout napravo. Třída *text-danger* obarví daný text červeně.



Obr. 10-9 vzhled stránky za použití Bootstrapu

Stránka nyní vypadá jako na Obr. 10-9, aniž by byl použit jakýkoliv vlastní CSS styl. Veškeré styly až do nyní pochází z Bootstrapu.

Aplikací stylů z přílohy P VII dosáhneme vzhledu jako na Obr. 10-10.



Obr. 10-10 dostylovaná stránka

Kompletní kód HTML je vidět v příloze P VIII.

## ZÁVĚR

V této práci byly popsány základní principy práce s opěrnými technologiemi webových aplikací a na příkladu vytvoření jednoduché webové stránky demonstrovány v praxi. Stejnětak jako použití správného vývojového prostředí je pro frontendový vývoj kritický správný výběr frameworků a knihoven v závislosti na potřebách vyvíjeného projektu. Ačkoliv je odvětví IT vývoje frontendových aplikací velmi širokosáhlou sférou, která vyžaduje mnohaleté zkušenosti, je vstup do tohoto světa a pochycení základů vývoje díky strmé učící křivce relativně jednoduché.

## SEZNAM POUŽITÉ LITERATURY

1. Bootstrap - The world's most popular mobile-first and responsive front-end framework. [online]. 2016. Dostupné také z: <http://getbootstrap.com/>
2. *jQuery* [online]. 2016. Dostupné také z: <http://jquery.com/>
3. *Code Academy* [online]. 2016. Dostupné také z: <http://www.codecademy.com>
4. *Tutorials Point* [online]. 2016 [cit. 2016]. Dostupné z: <http://www.tutorialspoint.com/>
5. *W3 Schools* [online]. Dostupné také z: <http://www.w3schools.com>
6. BAŠE, O. *jQuery pro neprogramátory: průvodce využitím knihovny jQuery UI.* Brno: Computer Press, 2012. ISBN 9788025137505.
7. SCHAFER, S. M. *HTML, XHTML a CSS: bible.* Praha: Grada, 2009. ISBN 9788024728506.
8. HOGAN, B. P. *HTML5 a CSS3: výukový kurz webového vývojáře.* Brno: Computer Press, 2011. ISBN 9788025135761.
9. CASTRO, E. a B. HYSLOP. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek.* Brno: Computer Press, 2012. ISBN 9788025137338.
10. HTML [online]. 2016. Dostupné také z: <http://w3c.github.io/html>
11. *HTML Dog* [online]. 2016. Dostupné také z: <http://www.htmldog.com>
12. React Native | A framework for building native apps using React [online]. 2016. Dostupné také z: <https://facebook.github.io/react-native/>
13. A JavaScript library for building user interfaces | React [online]. 2016. Dostupné také z: <https://facebook.github.io/react/index.html>
14. lynda.com software training & tutorials [online]. 2016. Dostupné také z: <http://www.lynda.com>
15. AngularJS — Superheroic JavaScript MVW Framework [online]. 2016. Dostupné také z: <https://angularjs.org/>

16. Andrew Ray's Blog. *React JS for Stupid People* [online]. 21. 09. 2014. Dostupné také z: <http://blog.andrewray.me/reactjs-for-stupid-people/>
17. PluralSight [online]. 2016. Dostupné také z: <http://pluralsight.com>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

HTML	HyperText Markup Language, hypertextový značkovací jazyk
CSS	Cascaded Style Sheets, kaskádové styly
JS	Skriptovací jazyk JavaScript
JSX	JavaScript syntax extension, syntaktické rozšíření jazyka JavaScript
DOM	Document Object Model, objektový model webového dokumentu
MVC	Model View Controller, architektonický vzor rozdělující aplikaci na tři části – model, pohled a ovladač.
API	Application Program Interface, rozhraní pro práci se softwarovou komponentou

**SEZNAM OBRÁZKŮ**

Obr. 1-1 jednoduchý HTML dokument.....	11
Obr. 1-2 Document Object Model .....	11
Obr. 1-3 označovaný prvek.....	12
Obr. 1-4 sourozenecké prvky.....	12
Obr. 1-5 vztah rodič-dítě.....	12
Obr. 1-6 značkování HTML .....	13
Obr. 1-7 atributy elementů.....	13
Obr. 1-8 kostra HTML dokumentu.....	14
Obr. 1-9 kód jednoduchého dokumentu HTML .....	15
Obr. 1-10 vzhled stránky z Obr. 1-9 .....	15
Obr. 1-11 tagy div obalující části stránky.....	17
Obr. 1-12 kód stránky se sémantickými tagy a seznamem.....	19
Obr. 1-13 příklad formuláře.....	21
Obr. 1-14 jednoduchý komentář .....	21
Obr. 1-15 komentář značící obsah pouze pro některé prohlížeče.....	22
Obr. 2-1 stylování v HTML na řádku .....	23
Obr. 2-2 stylovací pravidlo v hlavičce dokumentu.....	24
Obr. 2-3 pár stylovacích pravidel .....	24
Obr. 2-4 odkaz na soubor css v dokumentu HTML .....	25
Obr. 2-5 sdružená pravidla pro prvky header a footer .....	26
Obr. 2-6 stylové pravidlo uplatněné na jednoznačný prvek pomocí id .....	27
Obr. 2-7 relativní selektor se znaky >.....	27
Obr. 2-8 relativní selektor s mezerami.....	28
Obr. 2-9 selektor třídy s názvem bordered-image .....	28
Obr. 2-10 box model.....	29
Obr. 3-1 získání odkazu na element pomocí objektu <i>document</i> a funkce <i>getElementById</i> .....	32
Obr. 3-2 <i>script</i> tag HTML dokumentu.....	32
Obr. 3-3 vložení odkazu na JavaScriptový soubor .....	33
Obr. 3-4 typy struktur podmínek <i>if</i> .....	35
Obr. 3-5 možnosti definice funkce .....	36
Obr. 3-6 volání funkce z Obr. 3-5.....	36

Obr. 3-7 definice a volání funkce s callbackem.....	37
Obr. 3-8 příklad zapouzdření .....	38
Obr. 3-9 příklad jednoduchého objektu .....	39
Obr. 4-1 odkaz na soubor s knihovnou jQuery .....	41
Obr. 4-2 vyhledání a uložení prvku stránky s <i>id resultList</i> do proměnné.....	42
Obr. 4-3 použití funkce <i>text</i> na obsah proměnné <i>resultList</i> .....	42
Obr. 4-4 ekvivalentní zápis pro Obr. 4-2 pomocí aliasu.....	42
Obr. 4-5 odchycení události kliknutí na prvek s <i>id toggleButton</i> .....	43
Obr. 4-6 zapouzdření kódu pomocí funkce <i>ready</i> .....	43
Obr. 4-7 použití funkce <i>css</i> na sadu prvků.....	44
Obr. 4-8 demonstrační příklad práce s prvky zabalené sady .....	45
Obr. 4-9 jednoduchý zápis funkce <i>get</i> .....	46
Obr. 4-10 zápis funkce <i>get</i> včetně zřetěžených funkcí <i>success</i> , <i>fail</i> a <i>done</i> .....	47
Obr. 7-1 výchozí HTML dokument.....	53
Obr. 7-2 grafická reprezentace kódu z Obr. 7-1 s barevně označenými částmi .....	54
Obr. 7-3 sekce stránky obsahující formulář .....	55
Obr. 7-4 vzhled stránky včetně kódu z Obr. 7-3.....	56
Obr. 8-1 vynulování vlastností <i>padding</i> a <i>margin</i> .....	57
Obr. 8-2 zabalení obsahu <i>body</i> do <i>div</i> s třídou <i>container</i> .....	57
Obr. 8-3 styl třídy <i>container</i> .....	57
Obr. 8-4 styl prvku <i>body</i> .....	58
Obr. 8-5 styl prvků <i>header</i> , <i>section</i> , <i>footer</i> .....	58
Obr. 8-6 přesunutí HTML prvků navigace .....	58
Obr. 8-7 styly navigačního prvku .....	59
Obr. 8-8 styl nadpisu.....	59
Obr. 8-9 styl paragrafů.....	59
Obr. 8-10 styly formuláře .....	60
Obr. 8-11 styl pole výsledků.....	60
Obr. 8-12 vzhled stránky po přidání stylů CSS .....	61
Obr. 9-1 vložení JavaScriptových souborů do HTML .....	62
Obr. 9-2 obsluha výsledky skrývajícího tlačítka <i>#toggleButton</i> .....	62
Obr. 9-3 funkce <i>displayResults</i> .....	63
Obr. 9-4 síťový požadavek na získání hledaných projektů.....	64

---

Obr. 10-1 instalace Bootstrapu .....	65
Obr. 10-2 navigační panel.....	66
Obr. 10-3 prvek <i>jumbotron</i> .....	66
Obr. 10-4 použití tříd <i>row</i> a <i>col</i> .....	67
Obr. 10-5 formulář vyhledávání .....	67
Obr. 10-6 vzhled formuláře .....	68
Obr. 10-7 modifikace <i>div#resultList</i> na třídu <i>row</i> .....	68
Obr. 10-8 patička stránky.....	69
Obr. 10-9 vzhled stránky za použití Bootstrapu .....	69
Obr. 10-10 dostylovaná stránka .....	69

**SEZNAM PŘÍLOH**

- P I KÓD HTML DEMONSTRAČNÍ STRÁNKY
- P II HTML KÓD STRÁNKY PRAKTICKÉ ČÁSTI PO KAPITOLE 7. HTML
- P III HTML KÓD STRÁNKY PRAKTICKÉ ČÁSTI PO KAPITOLE 8. CSS
- P IV CSS KÓD STRÁNKY PRAKTICKÉ ČÁSTI PO KAPITOLE 8. CSS
- P V JAVASCRIPT + JQUERY KÓD Z KAPITOLY 9.
- P VI CSS A HTML ÚPRAVY Z KAPITOLY 9.
- P VII CSS NA DOSTYLOVÁNÍ STRÁNKY Z KAPITOLY 10. BOOTSTRAP
- P VIII KÓD HTML STRÁNKY Z KAPITOLY 10. BOOTSTRAP
- P IX HTML STRUČNÝ PŘEHLED
- P X CSS STRUČNÝ PŘEHLED

## PŘÍLOHA P I: HTML KÓD STRÁNKY

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>GitHub Project Lookup</title>
5     <!-- this is a simple comment -->
6
7     <!-- [if lt IE 9]>
8       <div>This div is only visible in Internet Explorer version lower than 9</div>
9     <![endif] -->
10    <link rel="stylesheet" href="css/site.css">
11  </head>
12
13  <body>
14    <header>
15      <div></div>
16      <div>This is a webpage with projects from GitHub.</div>
17      <nav>
18        <ol>
19          <li><a href="/index.html">Home</a></li>
20          <li><a href="/contact.html">Contact</a></li>
21          <li><a href="/about.html">About</a></li>
22        </ol>
23      </nav>
24    </header>
25
26    <section id="main" >
27      <p>This is a <em>sample site</em>, shows information about
28      projects on <a href="http://github.com">GitHub</a>
29      and <span>makes kittens</span> <b>happy</b>.
30    </p>
31    <form action="http://httpbin.org/post" method="POST">
32      <label for="searchPhrase">Search Phrase:</label>
33      <input name="searchPhrase" id="searchPhrase" /><br />
34      <input type="checkbox" name="useStars" id="useStars" />
35      <label for="useStars">Use stars?</label><br />
36      <label for="langChoice">Language:</label>
37      <select name="langChoice" id="langChoice">
38        <option>All</option>
39        <option>JavaScript</option>
40        <option selected>C#</option>
41        <option>Java</option>
42        <option>PHP</option>
43      </select>
44      <br>
45      <input type="submit" value="search" /><br />
46    </form>
47  </section>
48  <footer>
49    &copy; 2016 Kitten Kennel Antarctica
50  </footer>
51 </body>
52 </html>
53 <!-- comment -->
```

## PŘÍLOHA P II: HTML KÓD STRÁNKY PRAKTICKÉ ČÁSTI PO KAPITOLE 7. HTML

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Github Search</title>
5   </head>
6   <body>
7     <header>
8       <div></div>
9       <h1>Github Search</h1>
10      <div>This is a webpage allowing search in GitHub projects.</div>
11      <nav>
12        <ol>
13          <li><a href="/index.html">Home</a></li>
14          <li><a href="/contact.html">Contact</a></li>
15          <li><a href="/about.html">About</a></li>
16        </ol>
17      </nav>
18    </header>
19    <section>
20      <p>This is a <em>sample site</em>, that shows information
21      from <a href="http://github.com">GitHub</a>
22      and <span>makes kittens</span> <strong>happy</strong>.
23    </p>
24    <form>
25      <label for="searchPhrase">Search Phrase:</label>
26      <input name="searchPhrase" id="searchPhrase" /><br />
27      <input type="checkbox" name="useStars" id="useStars" checked="true"/>
28      <label for="useStars">Use stars?</label><br />
29      <label for="langChoice">Language: </label><br />
30      <select name="langChoice" id="langChoice">
31        <option>All</option>
32        <option>JavaScript</option>
33        <option selected>C#</option>
34        <option>Java</option>
35        <option>PHP</option>
36      </select>
37      <br>
38      <input type="submit" value="search" /><br />
39    </form>
40  </section>
41
42  <footer>
43    &copy; 2016 Kitten Kennel Antarctica
44  </footer>
45 </body>
46 </html>
```

## PŘÍLOHA P III: HTML KÓD STRÁNKY PRAKTICKÉ ČÁSTI PO KAPITOLE 8. CSS

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>GitHub Search</title>
5     <link rel="stylesheet" href="css/site.css">
6   </head>
7   <body>
8     <div class="container">
9       <header>
10        <nav>
11          <ol>
12            <li><a href="/index.html">Home</a></li>
13            <li><a href="/contact.html">Contact</a></li>
14            <li><a href="/about.html">About</a></li>
15          </ol>
16        </nav>
17        <div></div>
18        <h1>Github Search</h1>
19        <div>This is a webpage allowing search in GitHub projects.</div>
20      </header>
21
22      <section>
23        <p>This is a <em>sample site</em>, that shows information
24        from <a href="http://github.com">GitHub</a>
25        and <span>makes kittens</span> <strong>happy</strong>.</p>
26        <form class="simple-form">
27          <label for="searchPhrase">Search Phrase:</label>
28          <input name="searchPhrase" id="searchPhrase" /><br />
29          <input type="checkbox" name="useStars" id="useStars" />
30          <label for="useStars">Use stars?</label><br />
31          <label for="langChoice">Language:</label>
32          <select name="langChoice" id="langChoice">
33            <option>All</option>
34            <option>JavaScript</option>
35            <option selected>C#</option>
36            <option>Java</option>
37            <option>PHP</option>
38          </select>
39          <br>
40          <input type="submit" value="search" /><br />
41        </form>
42        <div id="results">
43          This is where results will appear eventually...
44        </div>
45      </section>
46      <footer>
47        &copy; 2016 Kitten Kennel Antarctica
48      </footer>
49    </div>
50  </body>
51 </html>
--
```

# PŘÍLOHA P IV: CSS KÓD STRÁNKY PRAKTICKÉ ČÁSTI PO KAPITOLE 8. CSS

```
1  /* site.css */
2
3  /* Tagy html a body nemají defaultně nulové hodnoty padding a margin.
4     Proto se často nulují a pak explicitně nastavují a následující pravidlo
5     běžně viditelné v css souborech. */
6  * {
7     padding: 0;
8     margin: 0;
9  }
10
11  .container {
12     /* běžná hodnota šířky kontejneru obsahujícího celou stránku,
13        vypadá dobře na většině monitorů */
14     width: 989px;
15     /* Hodnota auto u vlastnosti margin znamená,
16        že prohlížeč přidá na obě strany(nyní vlevo a vpravo)
17        stejnou hodnotu margin.
18        V praxi bude prvek vycentrován horizontálně. */
19     margin: 0 auto;
20     background-color: white;
21     padding: 10px;
22  }
23
24  body {
25     font-family: Arial, Helvetica, sans-serif;
26     font-size: 14px;
27     color: #202020;
28     background-color: #194C43;
29  }
30
31  header, section, footer {
32     border: solid 1px lightgray;
33     border-radius: 3px;
34     padding: 10px;
35  }
36
37  section {
38     margin: 10px 0;
39  }
40
41  header nav {
42     float: right;
43  }
44
45  header nav li {
46     display: inline;
47     font-size: 12px;
48     padding: 5px 10px;
49  }
50
51  header nav li a {
52     color: #888;
53     text-decoration: none;
54  }
55
56  header nav li a:hover {
57     color: darkorange;
58  }
59
60  header h1 {
61     margin: 10px 0 20px;
62  }
63
64  section p {
65     margin: 5px 0 10px;
66  }
67
68  .simple-form {
69     width: 300px;
70     display: inline-block;
71     vertical-align: top;
72  }
73
74  .simple-form label {
75     font-weight: bold;
76     width: 140px;
77     display: inline-block;
78     margin-bottom: 10px;
79  }
80
81  .simple-form input[type=text],
82  .simple-form input[type=checkbox],
83  .simple-form select {
84     width: 150px;
85  }
86
87  .simple-form input[type=submit] {
88     color: darkorange;
89     background-color: white;
90     border-radius: 25%;
91     padding: 10px;
92  }
93
94  #results {
95     width: 600px;
96     display: inline-block;
97     vertical-align: top;
98  }
```

## PŘÍLOHA P V: JAVASCRIPT+JQUERY KÓD Z KAPITOLY 9.

```
1 $(document).ready(function(){
2
3     "use strict";
4
5     var resultList = $("#resultList");
6
7     var toggleButton = $("#toggleButton");
8     toggleButton.on("click", function(){
9         resultList.toggle(500);
10
11         if (toggleButton.text() == "Hide") toggleButton.text("Show");
12         else toggleButton.text("Hide");
13     })
14
15     $("#githubSearchForm").on("submit", function(){
16
17         var searchPhrase = $("#searchPhrase").val();
18         var useStars = $("#useStars").prop('checked');
19         var langChoice = $("#langChoice").val();
20
21         if (searchPhrase) {
22             resultList.text("Performing search...");
23
24             var githubSearch = "https://api.github.com/search/repositories?q="
25                               + encodeURIComponent(searchPhrase);
26
27             if (langChoice != "All"){
28                 githubSearch += "+language:" + encodeURIComponent(langChoice);
29             }
30
31             if (useStars){
32                 githubSearch += "&sort=stars";
33             }
34             $.get(githubSearch)
35                 .success(function(r){
36                     displayResults(r.items);
37                 })
38                 .fail(function(error){
39                     console.log("Failed to query GitHub. Error: "+ error);
40                 })
41         }else{
42             resultList.text("No search phrase set.");
43         }
44
45         return false;
46     })
47
48     function displayResults(results){
49         var resultList = $("#resultList");
50         resultList.empty();
51         $.each(results, function(i, item){
52             var newResult = $("<div class='result'> " +
53                 "<div class='title'> " + item.name + "</div> " +
54                 "<div>Language: " + item.language + "</div> " +
55                 "<div>Owner: " + item.owner.login + "</div> " +
56                 "</div>");
57
58             newResult.hover(function(){
59                 $(this).css("background-color", "lightgray");
60             },function(){
61                 $(this).css("background-color", "transparent");
62             })
63
64             resultList.append(newResult);
65         })
66     }
67 }
68 })
```

## PŘÍLOHA P VI: CSS A HTML ÚPRAVY Z KAPITOLY 9.

css

```
.result {  
  margin: 5px 0;  
  padding: 5px;  
  border: 1px solid lightgray;  
}  
  
.result .title {  
  font-weight: bold;  
  margin-bottom: 5px;  
}
```

html

1.

```
<script type="text/javascript" src="js/jquery-2.2.3.min.js"></script>  
<script type="text/javascript" src="js/index.js"></script>
```

2.

```
<form class="simple-form" id="gitHubSearchForm">
```

3.

```
<div id="results">  
  <button id="toggleButton">Hide</button>  
  <div id="resultList">This is where results will appear eventually...</div>  
</div>
```

## PŘÍLOHA P VII: CSS NA DOSTYLOVÁNÍ STRÁNKY Z KAPITOLY 10. BOOTSTRAP

```
1  * {
2  |   padding: 0;
3  |   margin: 0;
4  | }
5
6  body {
7  |   background-color: #194C43;
8  | }
9
10 body > .container {
11 |   background-color: white;
12 |   padding-top: 5px;
13 | }
14
15 #navbar li a:hover {
16 |   color: darkorange;
17 | }
18
19 .jumbotron h1{
20 |   color: darkorange;
21 | }
22
23 .result {
24 |   padding: 5px;
25 |   border: 1px solid lightgray;
26 | }
27
28 .result > div {
29 |   text-overflow: ellipsis;
30 |   overflow: hidden;
31 |   white-space: nowrap;
32 | }
33
34 #resultList{
35 |   padding: 0 15px;
36 | }
37
38 .result .title {
39 |   font-weight: bold;
40 |   margin-bottom: 5px;
41 | }
42
43 #footer {
44 |   border-top: 1px solid gray;
45 |   padding: 5px;
46 | }
```

# PŘÍLOHA P VIII: KÓD HTML STRÁNKY Z KAPITOLY 10.

## BOOTSTRAP

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>GitHub Search</title>
6
7     <!-- bootstrap css -->
8     <link rel="stylesheet"
9       href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
10      integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
11      crossorigin="anonymous">
12     <link rel="stylesheet" href="css/site.css">
13   </head>
14   <body>
15     <div class="container">
16       <nav class="navbar navbar-default" id="navbar">
17         <div class="container-fluid">
18           <!-- Brand and toggle get grouped for better mobile display -->
19           <div class="navbar-header">
20             <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
21               data-target="#collapseNavRight" aria-expanded="false">
22               <span class="sr-only">Toggle navigation</span>
23               <span class="icon-bar"></span>
24               <span class="icon-bar"></span>
25               <span class="icon-bar"></span>
26             </button>
27             <a class="navbar-brand" href="#">GitHub Search</a>
28           </div>
29
30           <!-- Collect the nav links, forms, and other content for toggling -->
31           <div class="collapse navbar-collapse" id="collapseNavRight">
32             <ul class="nav navbar-nav navbar-right">
33               <li><a href="/index.html">Home</a></li>
34               <li><a href="/contact.html">Contact</a></li>
35               <li><a href="/about.html">About</a></li>
36             </ul>
37           </div><!-- /.navbar-collapse -->
38         </div><!-- /.container-fluid -->
39       </nav>
40
41       <div class="jumbotron">
42         <h1>Github Search</h1>
43         <p>This is a webpage allowing search in GitHub projects.</p>
44       </div>
```

```

45
46
47     <div class="row">
48         <div class="col-xs-12 col-md-5" id="formDiv">
49             <form class="form-horizontal" id="githubSearchForm">
50                 <div class="form-group">
51                     <label for="searchPhrase" class="col-sm-4 control-label">Search Phrase</label>
52                     <div class="col-sm-8">
53                         <input type="text" class="form-control" name="searchPhrase"
54                             id="searchPhrase" placeholder="Search Phrase">
55                     </div>
56                 </div>
57                 <div class="form-group">
58                     <label for="searchPhrase" class="col-sm-4 control-label">Language</label>
59                     <div class="col-sm-8">
60                         <select class="form-control">
61                             <option>All</option>
62                             <option>JavaScript</option>
63                             <option selected>C#</option>
64                             <option>Java</option>
65                             <option>PHP</option>
66                         </select>
67                     </div>
68                 </div>
69                 <div class="form-group">
70                     <div class="col-sm-offset-4 col-sm-8">
71                         <div class="checkbox">
72                             <label>
73                                 <input type="checkbox" name="useStars" id="useStars"> Use stars?
74                             </label>
75                         </div>
76                     </div>
77                 </div>
78                 <div class="form-group">
79                     <div class="col-sm-offset-4 col-sm-8">
80                         <button type="submit" class="btn btn-success">submit</button>
81                     </div>
82                 </div>
83             </form>
84         </div>
85         <div class="col-xs-12 col-md-7" id="results">
86             <button id="toggleButton" class="btn btn-default">Hide</button>
87             <div class="row" id="resultList">This is where results will appear eventually...</div>
88         </div>
89     </div>
90
91     <footer id="footer">
92         <small>Pomůcka pro předmět webové technologie
93         <span class="pull-right text-danger">2016</span></small>
94     </footer>
95 </div>
96
97 <!-- jquery -->
98 <script type="text/javascript" src="js/jquery-2.2.3.min.js"></script>
99 <!-- bootstrap js -->
100 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
101     integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxs5iyVq0tnepnHVP9aJ7xS"
102     crossorigin="anonymous"></script>
103
104 <script type="text/javascript" src="js/index.js"></script>
105 </body>
106 </html>

```

## PŘÍLOHA P IX: HTML STRUČNÝ PŘEHLED

### 1 ŠABLONA

```
<!doctype html>
<html>
  <head>
    <title></title>
    meta tagy
    CSS
    JavaScript
  </head>
  <body>
    Obsah
  </body>
</html>
```

### 2 Obecné

<code>&lt;body&gt;</code>	Viditelná část stránky
<code>&lt;head&gt;</code>	Nezobrazená část stránky
<code>&lt;html&gt;</code>	Vytváří HTML dokument
<code>&lt;title&gt;</code>	Název stránky v titulku prohlížeč
<code>&lt;!-- abc --&gt;</code>	Komentář, není interpretován prohlížečem

### 3 Head

TAG	POPIS	PARAMETRY · TYPY HODNOT
<code>&lt;style&gt;</code>	Definice stylů CSS ve stránce	type <ul style="list-style-type: none"><li>text/css – typ stylů je textové CSS</li></ul>
<code>&lt;link&gt;</code>	Odkaz na externí definice stylů CSS	rel <ul style="list-style-type: none"><li>stylesheet – typ odkazovaného souboru je stylesheet</li></ul> type <ul style="list-style-type: none"><li>text/css – typ obsahu je textové CSS</li></ul>
<code>&lt;script&gt;</code>	Skript	type <ul style="list-style-type: none"><li>text/javascript – typ obsahu je textový javascript</li></ul> language <ul style="list-style-type: none"><li>javascript – skript je psán jazyke javascript</li></ul>
<code>&lt;meta&gt;</code>	Metadata o dokumentu	charset <ul style="list-style-type: none"><li>UTF-8 – typ kódování textu na stránce je UTF-8</li></ul>

TAG	POPIS	PARAMETRY • TYPY HODNOT
		Pár name-content <ul style="list-style-type: none"> <li>Proměnná metadat <i>name</i> a její hodnota <i>content</i></li> </ul>

## 4 Odkazy

TAG	POPIS	PARAMETRY • TYPY HODNOT
<img>	Obrázek	src <ul style="list-style-type: none"> <li><i>URL</i> – adresa obrázku</li> </ul>
<a>	Odkaz	href <ul style="list-style-type: none"> <li>#? – odkaz na kotvu v rámci stejného dokumentu</li> <li><i>URL</i> – odkaz na jinou stránku</li> <li><i>URL#</i> – odkaz na kotvu v jiné stránce</li> <li>mailto:<i>EMAIL</i> – odkaz na <i>EMAIL</i></li> </ul>

## 5 Struktura

TAG	POPIS
 	Zalomení řádku
<code>	Označení zdrojového kódu
<div>	Strukturální prvek nebo blok textu
<em>	Text kurzívou, důraz
<h1>..<h6>	Nadpisy největší..nejmenší
<hr>	Horizontální oddělovač
<p>	Paragraf
<pre>	Předformátovaný text, zachovají se odřádkování a mezery v kódu
<span>	Formátování v řádku
<strong>	Tučný text
<sub>	Spodní index
<sup>	Horní index

## 6 Seznamy

TAG	POPIS
<code>&lt;dd&gt;</code>	Text definice
<code>&lt;dl&gt;</code>	Seznam definic
<code>&lt;dt&gt;</code>	Definovaný termín
<code>&lt;li&gt;</code>	Prvek seznamu
<code>&lt;ol&gt;</code>	Řazený (číslovaný) seznam
<code>&lt;ul&gt;</code>	Neřazený (nečíslovaný) seznam

## 7 HTML entity — speciální znaky

TAG	POPIS
<code>&amp;nbsp;</code>	Nezalamující se mezera
<code>&amp;quot;</code>	Úvozovka
<code>&amp;amp;</code>	Ampersand
<code>&amp;lt;</code>	Menší než; levá ostrá závorka
<code>&amp;gt;</code>	Větší než; pravá ostrá závorka

## 8 Tabulky

TAG	POPIS	PARAMETRY • TYPY HODNOT
<code>&lt;caption&gt;</code>	Popisek tabulky	
<code>&lt;table&gt;</code>	Tabulka	
<code>&lt;tbody&gt;</code>	Tělo tabulky	
<code>&lt;td&gt;</code>	Buňka tabulky	colspan – počet sloupců, které má přesahovat rowspan – počet řádků, které má přesahovat
<code>&lt;tfoot&gt;</code>	Patička tabulky	
<code>&lt;th&gt;</code>	Buňka hlavičky tabulky	colspan – počet sloupců, které má přesahovat
<code>&lt;thead&gt;</code>	Hlavička tabulky	
<code>&lt;tr&gt;</code>	Řádek tabulky	

## 9 Šablona tabulky

```
<table>
  <thead>
    <tr>
      <th></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td></td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td></td>
    </tr>
  </tfoot>
</table>
```

## 10 Formuláře

TAG	POPIS	PARAMETRY • TYPY HODNOT
<form>	Definuje formulář	action - kam má formulář při odeslání předat data method - typ síťového požadavku <ul style="list-style-type: none"><li>• GET, POST</li></ul>
<fieldset>	Skupina souvisejících prvků formuláře	
<input>	Prvek formuláře	type <ul style="list-style-type: none"><li>• button - tlačítko</li><li>• checkbox - zaškrtačací políčko</li><li>• file - soubor</li><li>• hidden - skryté</li><li>• image - obrázek</li><li>• password - heslo</li><li>• radio - přepínací tlačítko</li><li>• reset - resetovací tlačítko formuláře</li><li>• submit - odesílací tlačítko formuláře</li><li>• text - textové pole</li></ul>
<option>	Prvek rozbalovacího menu	selected - předzvolený prvek
<select>	Rozbalovací menu	
<textarea>	Víceřádkové textové pole	

# PŘÍLOHA X: CSS STRUČNÝ PŘEHLED

## 1 SYNTAX

<code>selektor {vlastnost: hodnota}</code>	CSS styl pro libovolný element
<code>&lt;link rel="stylesheet" type="text/css" href="style.css"&gt;</code>	Externí CSS
<code>&lt;style type="text/css"&gt; Selektor {vlastnost: hodnota} &lt;/style&gt;</code>	Interní CSS
<code>&lt;tag style="vlastnost: hodnota"&gt;</code>	CSS v řádku

## 2 Obecné

<code>/* */</code>	Komentář
<code>selektor třída</code>	Tečka (.) následovaná řetězcem
<code>selektor ID</code>	Křížek (#) následovaný řetězcem
<code>selektor div</code>	Formátování struktury nebo bloku textu
<code>selektor span</code>	Formátování řádkového element
<code>color</code>	Barva popředí – textu
<code>cursor</code>	Vzhled kurzoru myši
<code>display</code>	Typ zobrazení prvku <ul style="list-style-type: none"><li>• block, inline, inline-block, list-item, none</li></ul>
<code>overflow</code>	Přetékání elementu <ul style="list-style-type: none"><li>• visible, hidden, scroll, auto</li></ul>
<code>visibility</code>	Viditelnost <ul style="list-style-type: none"><li>• visible, hidden</li></ul>

## 3 Font

<code>font-style</code>	Styl fontu <ul style="list-style-type: none"><li>• normal, italic</li></ul>
<code>font-variant</code>	Varianta fontu <ul style="list-style-type: none"><li>• normal, small-caps</li></ul>
<code>font-weight</code>	Váha fontu <ul style="list-style-type: none"><li>• normal, light, lighter, bold, bolder</li><li>• číselná hodnota [100-900]</li></ul>
<code>font-size</code>	Velikost fontu

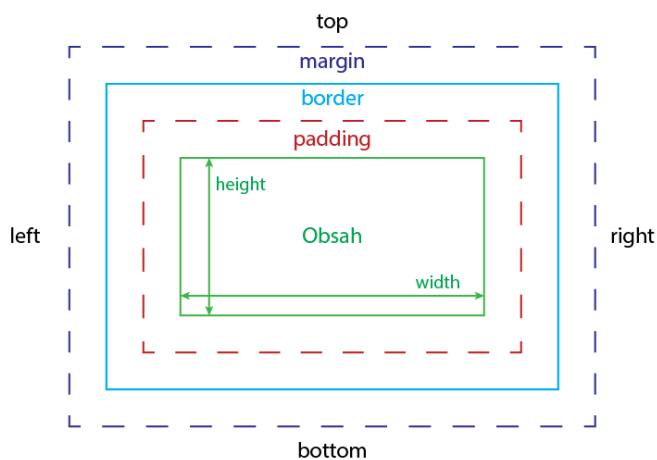
<code>font-family</code>	Konkrétní font(y)
--------------------------	-------------------

## 4 Text

<code>letter-spacing</code>	Velikost mezer mezi písmeny
<code>line-height</code>	Výška řádku
<code>text-align</code>	Horizontální zarovnání
<code>text-decoration</code>	Dekorace textu <ul style="list-style-type: none"> <li>• <code>blink</code>, <code>line-through</code>, <code>overline</code>, <code>underline</code>, <code>none</code></li> </ul>
<code>text-indent</code>	Odsazení prvního řádku
<code>text-transform</code>	Transformace textu <ul style="list-style-type: none"> <li>• <code>capitalise</code>, <code>lowercase</code>, <code>uppercase</code></li> </ul>
<code>vertical-align</code>	Vertikální zarovnání
<code>word-spacing</code>	Velikost mezer mezi slovy

## 5 Box model

<code>height, width</code>	
<code>margin-*</code>	
<ul style="list-style-type: none"> <li>- <code>top</code></li> <li>- <code>right</code></li> <li>- <code>bottom</code></li> <li>- <code>left</code></li> </ul>	
<code>padding-*</code>	
<ul style="list-style-type: none"> <li>- <code>top</code></li> <li>- <code>right</code></li> <li>- <code>bottom</code></li> <li>- <code>left</code></li> </ul>	
<code>border-*</code>	
<ul style="list-style-type: none"> <li>- <code>width</code></li> <li>- <code>style</code></li> <li>- <code>color</code></li> </ul>	<p>Tloušťka rámečku</p> <p>Styl rámečku</p> <ul style="list-style-type: none"> <li>• <code>dashed</code>, <code>dotted</code>, <code>double</code>, <code>groove</code>, <code>inset</code>, <code>outset</code>, <code>ridge</code>, <code>solid</code>, <code>none</code></li> </ul> <p>Barva rámečku</p>



## 6 Pozice

<code>clear</code>	Odsunout „plovoucí“ elementy kolem daného pod něj <ul style="list-style-type: none"> <li>• <code>both</code>, <code>left</code>, <code>right</code>, <code>none</code></li> </ul>
<code>float</code>	„Odplout“ element na stranu <ul style="list-style-type: none"> <li>• <code>left</code>, <code>right</code>, <code>none</code></li> </ul>
<code>top, right, bottom, left</code>	Poloha elementu od okraje

<b>position</b>	Typ polohování elementu <ul style="list-style-type: none"> <li>• static, relative, absolute</li> </ul>
<b>z-index</b>	Poloha překrývajících se objektů v z-souřadnici <ul style="list-style-type: none"> <li>• auto, číselná hodnota(vyšší čísla navrchu)</li> </ul>

## 7 Pozadí

<b>background-color</b>	Barva pozadí
<b>background-image</b>	Obrázek na pozadí <ul style="list-style-type: none"> <li>• url(' ')</li> </ul>
<b>background-repeat</b>	Opakování obrázku na pozadí <ul style="list-style-type: none"> <li>• repeat, no-repeat, repeat-x, repeat-y</li> </ul>
<b>background-attachment</b>	Pozadí skroluje s elementem <ul style="list-style-type: none"> <li>• scroll, fixed</li> </ul>
<b>background-position</b>	Poloha obrázku na pozadí <ul style="list-style-type: none"> <li>• (x y), top, center, bottom, left, right</li> </ul>

## 8 Seznamy

<b>list-style-type</b>	Typ odrážky nebo číslování seznamu <ul style="list-style-type: none"> <li>• Disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none</li> </ul>
<b>list-style-position</b>	Poloha odrážky nebo číslování seznamu <ul style="list-style-type: none"> <li>• inside, outside</li> </ul>
<b>list-style-image</b>	Obrázek, který bude představovat odrážku

## 9 Pseudoselektory

<b>:hover</b>	Vzhled prvku při přejetí myší
<b>:active</b>	Vzhled aktivního prvku
<b>:link, :visited</b>	Vzhled odkazu, Vzhled navštíveného odkazu
<b>:first-line, -letter</b>	Vzhled prvního řádku nebo písmene