Vysoká škola ekonomická v Praze

Fakulty informatiky a statistiky

Katedra informačního a znalostního inženýrství

Studijní program: Aplikovaná informatika

Obor: Informatika

Praktické využití jazyka HTML5, CSS3, JS a PHP

BAKALÁŘSKÁ PRÁCE

Student : Karel Kvítek

Vedoucí : Ing. Jiří Zumr

Oponent : Ing. Daniel Vodňanský

2016

Prohlášení

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

V Praze dne 4. května 2016 . . . . . .

Karel Kvítek

Abstrakt

V teoretické části tato bakalářská práce popisuje historii a základní syntaxi jazyků HTML5, CSS3, JavaScript a PHP. V části věnované HTML nabízí přehled nových sémantických elementů a mapuje jejich podporu v mobilních i desktopových prohlížečích. Dále v teoretické části stručně představuje knihovnu JavaScriptu jQuery a její současné využití na webových stránkách v porovnání s dalšími knihovnami tohoto jazyku. V PHP části poskytne přehled vybraných klíčových vlastností, které nově přinesla verze PHP 7.

V praktické části jsou vytvořeny nové webové stránky pro společnost KL-Accounting s.r.o. s ohledem na nejnovější trendy a moderní přístup k tvorbě webových stránek. Každý krok tvorby je řádně vysvětlen a demonstrován na příslušném kódu.

Klíčová slova

HTML, CSS, JavaScript, jQuery, PHP.

Abstract

The theoretical part of this bachelor thesis describes the history and basic syntax of HTML5, CSS3, JavaScript and PHP. In the section devoted to HTML it provides an overview of the new semantic elements and maps their support in mobile and desktop browsers. Furthermore, the theoretical part is briefly introducing jQuery, the JavaScript library and its current usage on the websites compared with other libraries of this language. The PHP section provides an overview of selected key features that was newly introduced in PHP 7.

In the practical part new website is created for the company KL-Accounting Ltd. according to the latest trends and modern approach of creating web pages. Each step of the development is properly explained and demonstrated by the relevant code.

Keywords

HTML, CSS, JavaScript, jQuery, PHP.

Poděkování

Velice rád bych zde poděkoval vedoucímu práce panu Ing. Jiřímu Zumrovi za pomoc při kompletaci práce.

Dále bych chtěl poděkovat své rodině, která mě podporovala po celou dobu mého studia.

OBSAH

1. ÚVOD

Již od počátku devadesátých let je značkovací jazyk HTML základem webu a právě on do velké míry přispěl k vytvoření webu takového, jak ho známe dnes. K úspěchu HTML vedla především jeho jednoduchost a podpora napříč různými zařízeními. Rychlý rozvoj webových technologií a s nimi spojené zvyšující se požadavky uživatelů znamenaly vznik nových technologií, které měly základ v podobě HTML rozšířit.

Jako první vznikla potřeba oddělit vzhled od struktury, a tak přichází na řadu kaskádové styly označované jako CSS. Ty umožnily nejen oddělení webové struktury od jejího obsahu, ale přinesly taky onen koncept „kaskádování“.

K přechodu od jednoduchých statických webů ke komplexním dynamickým vedlo především rozšíření jazyka JavaScript, který však původně vůbec pro web navrhnut nebyl. Tento skriptovací jazyk, jenž běží na straně klienta, umožňuje vytvářet dynamický obsah a různé efekty na stránce.

Mezi serverové jazyky spojené s webovými technologiemi patří především jazyk PHP, který je v dnešní době vůbec tím nejpoužívanějším.

Potřeba přizpůsobit webový obsah pro různé zařízení, jako jsou mobilní telefony, tablety a televizní obrazovky, znamenala nutnost rozšíření možností veškerých webových technologií. Ty jsou tak pravidelně aktualizovány, aby vyhovovaly potřebám dnešní doby. HTML5 je nejnovější specifikací jazyka HTML, která byla finalizována 28. října 2014. V současné době se pracuje na specifikaci HTML 5.1, jejíž dokončení se odhaduje v průběhu tohoto roku. CSS se nachází ve verzi CSS3. Tato verze je ve vývoji již několik let a postupně se rozšiřuje. PHP je aktuálně ve verzi 7, používanější je však stále ještě verze 5, verze 6 nikdy nebyla vydána.

Všechny tyto webové technologie umožní vytvořit komplexní, dynamický a responzivní web, který splňuje požadavky dnešní doby a zároveň bere v úvahu moderní trendy v této oblasti.

Cílem této bakalářské práce je v teoretické části popsat základní syntaxi zmíněných jazyků, přiblížit jejich historii a představit vybrané klíčové prvky nejnovějších specifikací.
V praktické části bude tato bakalářská práce ukazovat postup při tvorbě nových webových

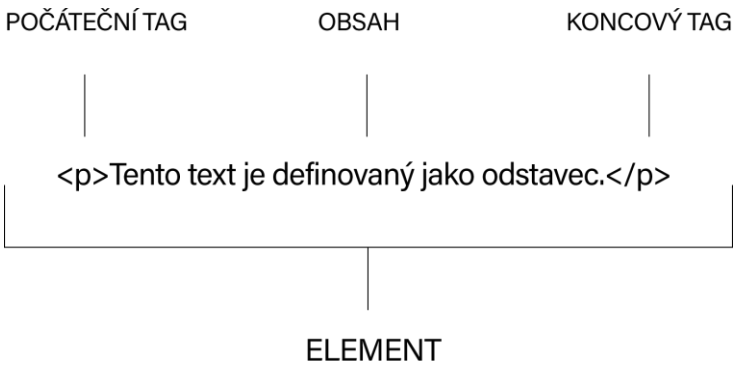
stránek pro společnost KL-Accounting s.r.o.. Tyto stránky budou představovat moderní postupy tvorby s vhodným využití zmíněných jazyků.

2. HTML

2.1 SYNTAXE JAZYKA HTML

Jazyk HTML (HyperText Markup Language) je značkovací jazyk, který definuje strukturu webových stránek a tvoří tak základ při jejich tvorbě. Označuje se jako značkovací, protože obsahuje množinu značek (neboli tagů), kde většina z nich je párová, tzn., že obsahují počáteční a koncový tag. Párový tag definuje význam obsahu, který se v něm nachází. Párový tag se svým počátečním a koncovým tagem, tak může například říkat „Toto je nadpis“, „Toto je odstavec“ atp.

Příklad:



Obrázek 1:

Znázornění HTML syntaxe (Zdroj: Autor)

Počáteční tag, obsah a koncový tag se souhrnně označuje jako element. Element *p* definuje na webové stránce odstavec - <*p*> je počáteční tag, </*p*> je koncový tag a textový obsah, který se nachází mezi nimi je poté definován jako odstavec.

Množina tagů HTML však obsahuje i tagy nepárové. Takové tagy neobsahují koncový tag a nelze tak mezi ně vložit textový obsah. Jako příklad lze uvést jedny z nejpoužívanějších nepárových tagů - tag <*br*>, který značí konec řádku a tag <*img*>, pomocí kterého se vkládají do stránky obrázky.

Počáteční tagy v sobě mohou nést další informace o elementu. Tyto informace se definují pomocí tzv. atributů. Atributy se zapisují jako dvojice *NázevAtributu*="*HodnotaAtributu*".

Příklad:

```
<img src="img/obrazek.jpg">
```

Zde hodnota atributu *src* určuje zdroj obrázku. [1]

2.2 HISTORIE JAZYKA HTML

Zcela první definici značkovacího jazyka HTML vytvořil Tim Berners-Lee již v roce 1991. Tato definice byla součástí projektu WWW, jehož účelem bylo umožnit vědcům sdílet výsledky svých výzkumů po celém světě. Prvním veřejným popisem jazyka se stal dokument HTML Tags, který obsahoval základní HTML značky. Ty umožňovaly rozdělení textu do několika logických částí, různé druhy zvýraznění textu a přidání odkazů a obrázků na stránku.

Se zvyšováním požadavků uživatelů na webovou stránku se musela specifikace HTML rozšiřovat o nové prvky. Tim Berners-Lee tak ve spolupráci s IETF (Internet Engineering Task Force) přišel s verzí HTML 2.0, jenž především definovala práci s formuláři.

Následovala verze označovaná jako HTML+, která rozšiřovala HTML o matematické vzorce, prvky zajišťující lepší vzhled dokumentů a možnosti vytváření tabulek.

V roce 1995 vznikl návrh standardu HTML 3.0, za kterým stál Dave Ragget z laboratoří Hawlett-Packard. Tato verze se však nikdy implementace v prohlížečích nedočkala.[2]

Jako náhrada za HTML 2.0 vznikla až verze s označením HTML 3.2, která vyřadila z HTML 3.0 většinu prvků a některé nové přidala. HTML 3.2 přinesla například prvky pro pokročilejší stylování a zarovnání textu. [3] Od ledna 1997 se tato verze stala doporučením konsorcia W3C. [2]

Ještě tentýž rok vychází verze HTML 4.0, která přidala plnou podporu kaskádových stylů, skriptů a rámů.[4]

V roce 1999 přišla verze HTML 4.01, která opravovala chyby v předchozí verzi a ještě důrazněji doporučovala oddělení vzhledu od obsahu za pomocí kaskádových stylů. [5]

Po verzi HTML 4.0 vznikaly snahy převést další verze do jazyka XML (Extensible Markup Language). Tyto snahy vyústily v roce 2000 ve vznik specifikace jazyka s označením

XHTML 1.0.[6] Postupem času se však ukázalo, že krok touto cestou nebyl správný a jazyk XHTML oproti HTML nepřinesl žádnou převratnou funkčnost.

Nespokojenost výrobců prohlížečů s vývojem jazyka (X)HTML vedla k tomu, že roku 2004 vytvořili pracovní skupinu WHATWG (Web Hypertext Application Technology Working Group), jejímž cílem bylo vytvořit novou verzi HTML, která by stávající obohatila o nové prvky, jenž by sloužily potřebám moderního webu.[2] Jejich snaha přispěla k tomu, že W3C upustila od tvorby XHTML a připojila se k WHATWG při práci na nové specifikaci HTML, označované jako HTML 5.0.

V říjnu roku 2014 se, po dlouhých 15 letech, zrodila nová verze jazyka HTML, HTML 5.0. Tato specifikace obohatila HTML o řadu nových moderních prvků, mezi něž patří například elementy *video* a *audio* nebo třeba kreslící plátno *canvas*.[7]

V současné době již existuje i verze HTML 5.1, která se však zatím nachází pouze ve stavu „working draft“. Její finální specifikace se očekává koncem roku 2016. Podle tvůrců je cílem této specifikace přidání věcí, které jsou interoperabilní a odstranění těch, které nejsou. Úmyslem je také zapojení většího počtu zainteresovaných lidí a organizací za účelem ujištění se, že vývoj bude odrážet požadavky a potřeby komunity. [8]

2.3 NOVÉ SÉMANTICKÉ ELEMENTY V HTML5

Do současné verze jazyka HTML bylo přidáno několik nových elementů, které zdokonalují sémantiku webových stránek. V následujícím textu si tyto elementy blíže popíšeme.

Před uvedením HTML 5 byly stránky povětšinou z velké části tvořeny elementy *div* s atributy *class* nebo *id*, kterým byl typicky přiřazen název daného obsahu. Zdrojový kód takové stránky (v HTML 4) by tak zjednodušeně mohl vypadat například takto:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
    <head>
    </head>
    <body>
        <div id="header">Záhlaví stránky</div>
        <div id="navigation">Navigace stránky(menu)</div>
        <div id="main">Hlavní obsah stránky</div>
        <div id="footer">Patička stránky</div>
    </body>
</html>
```

V HTML 5 by pak stránka mohla vypadat následovně:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <header>Záhlaví stránky</header>
        <nav>Navigace stránky(menu)</nav>
        <main>Hlavní obsah stránky</main>
        <footer>Patička stránky</footer>
    </body>
</html>
```

Z těchto dvou ukázek je vidět, že element *div* může být v HTML 5 v některých případech nahrazen novými sémantickými elementy.

2.3.1 HEADER

Element *header* představuje hlavičku stránky. Typicky uvnitř něj najdeme nadpis, navigaci, logo či vyhledávání. Tento element se v některých případech používá na stránce i vícekrát, například jako hlavička dané sekce.

Příklad:

```
<header>
    <h1>Nadpis</h1>
    <nav>
        <ul>
            <li>Položka1</li>
            <li>Položka2</li>
            <li>Položka3</li>
        </ul>
    </nav>
</header>
```

2.3.2 FOOTER

Element *footer* představuje patičku stránky. Tradičně uvnitř něj najdeme autorské údaje a odkazy na související stránky. Stejně jako u elementu *header* platí, že může být použit na stránce vícekrát.

Příklad:

```
<footer>
    <p>&copy; Karel Kvítek 2016</p>
    <a href="https://www.w3.org/">Odkaz</a>
</footer>
```

2.3.3 SECTION

Element *section* reprezentuje určitý úsek stránky – tematická seskupení obsahu (typicky s vlastním nadpisem).

Příklad:

```
<section>
    <h2>Nadpis</h2>
    <p>Odstavec</p>
</section>
```

2.3.4 ARTICLE

Element *article* tvoří tu část stránky, která je chápána jako nezávislý a samostatný celek. Jeho obsah by tak měl dávat smysl i sám o sobě.

Příklad:

```
<article>
    <h2>Nadpis</h2>
    <p>Odstavec</p>
</article>
```

2.3.5 ASIDE

Element *aside* představuje tu část stránky, která je logicky spjata s okolním obsahem. Často se používá uvnitř elementu *article* a pro tvoření tzv. sidebarů.

Příklad:

```
<article>
    <h2>Nadpis</h2>
    <p>Odstavec</p>
    <aside>Obsah elementu aside související s okolním obsahem</aside>
    <p>Odstavec 2</p>
</article>
```

2.3.6 NAV

Element *nav* představuje tu část stránky, ve které se odkazuje na další stránky, anebo na určitou část aktuální stránky.

Příklad:

```
<nav>
    <ul>
```

```
        <li><a href="polozka1.html">Položka1</a></li>
        <li><a href="polozka2.html">Položka2</a></li>
        <li><a href="polozka3.html">Položka3</a></li>
    </ul>
</nav>
```

2.3.7 HGROUP

Element *hgroup* slouží k seskupování nadpisů, tedy elementů *h1 - h6*. Může se tak například v sekci vytvořit nadpis a s ním související podnadpis.

Příklad:

```
<hgroup>
    <h1>Nadpis</h1>
    <h2>Podnadpis</h2>
</hgroup>
```
[6]

2.4 PODPORA NOVÝCH SÉMANTICKÝCH ELEMENTŮ V HTML5

V současné době je podpora nových sémantických elementů v moderních prohlížečích již samozřejmostí, přesto jsou však stále takové, které nabízejí pouze částečnou podporu. Takové jsem v tabulkách vyznačil žlutou barvou, ty s plnou podporou pak zelenou.

2.4.1 DESKTOPOVÉ PROHLÍŽEČE

Tabulka 1: *Podpora sémantických elementů HTML 5 v desktopových prohlížečích*

IE	Edge	Firefox	Chrome	Safari	Opera
9 +	12 +	21 +	26 +	6.1 +	15+

V případě prohlížeče Internet Explorer podporu HTML5 sématických elementů lze označit pouze za částečnou z důvodu, že element *main* je pro tento prohlížeč stále „neznámý“, avšak je možné s ním pracovat a stylovat ho.

2.4.2 MOBILNÍ PROHLÍŽEČE

Tabulka 2: *Podpora sémantických elementů HTML 5 v mobilních prohlížečích*

iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile
7.1 +	8	4.4 +	7	36	49	45	10+

Podpora Opery Mini je označena jako částečná, protože zde chybí standardní stylování elementů. To se však dá jednoduše vyřešit přidáním vlastnosti *display* s hodnotou *default* každému takové elementu. V případě prohlížečů Blackberry Browser a IE Mobile se jedná o stejný problém jako u desktopové verze prohlížeče Internet Explorer. [9]

3. CSS

3.1 SYNTAXE CSS

CSS by se dalo označit jako sbírka metod pro grafickou úpravu webových stránek. Je to zkratka z Cascading Style Sheets, což se česky překládá jako kaskádové styly. Kaskáda je vlastnost, která říká, že se definice mohou vrstvit jedna na druhou, platí vždy ta poslední.

Její syntaxe se skládá ze dvou částí – selektor a blok deklarací. Selektor ukazuje na HTML element, který má být formátován. Každý blok deklarací je ohraničen složenými závorkami a skládá se z vlastnosti a její hodnoty (popř. i více hodnot), mezi nimiž je dvojtečka. Každá dvojice vlastnost-hodnota je od sebe oddělena středníkem a příslušný blok deklarací takovýchto dvojic může obsahovat libovolné množství.[10]

Obrázek 2:

Znázornění CSS syntaxe (Zdroj: Autor)

3.2 HISTORIE KASKÁDOVÝCH STYLŮ

Vznik CSS podmínila potřeba oddělení obsahu od vzhledu, která reagovala na zvyšující se nepřehlednost HTML. K dnešnímu dni existují čtyři verze kaskádových stylů.

Jako první spatřila světlo světa verze CSS1, která byla vydána v roce 1996.[11] Internet Explorer ve verzi 3 byl prvním rozšířeným prohlížečem, který CSS podporoval. Tato verze však podporovala pouze písma a barvy, rozsáhlejší podporu nabídly až čtvrté verze prohlížečů

Internet Explorer a Netscape Navigator. Pořád však chyběla podpora některých vlastností a u jiných se prohlížeče občas nechovaly podle očekávání. Díky těmto faktorům se CSS nedostávalo zpočátku velké obliby. Částečně tyto problémy řešila pátá verze Internet Exploreru.

Jako takový přídavek k CSS1 sloužila verze s označením CSS-P, která řešila pozicování.[12]

Druhá verze kaskádových stylů je z roku 1998 a nese označení CSS2.[13] Její cíle však míří výš, než je tomu u CSS1. CSS2 má ambice stát se jazykem pro formátování vzhledu dokumentu i pro jiné jazyky než jen HTML. S CSS2 nepřišlo jen rozšíření vlastností pro definování vzhledu písma a rozdělení formátování pro různá média, ale také mnohem širší podpora mezi prohlížeči.[12]

Aktuální verzí je CSS3. Tato verze je plně kompatibilní s předchozími verzemi a přináší novinky v podobě možností tvorby animací, 2D/3D transformací, vrženého stínu, zakulacení rohů či rozšíření selektorů.[14]

Verze CSS4 se neplánuje. Současná verze CSS je totiž rozdělena do několika na sobě nezávislých modulů, které mají určitou úroveň a do budoucna se tak očekává pouze postupné přidávání nových modulů a jejich úrovní. [15]

4. JAVASCRIPT

4.1 ZÁKLADNÍ SYNTAXE JAVASCRIPTU

Scriptovací jazyk JavaScript přidává webovým stránkám dynamickou funkcionalitu a interaktivitu. Běží na straně klienta v prohlížeči a v dnešní době se s ním setkáme v nějaké formě téměř na každé větší webové stránce. JavaScript pohání jedny z největších a nejpoužívanějších webových aplikací jako jsou Google Maps, Gmail nebo třeba Facebook.

Výsledný program je tvořen sekvencí příkazů a každý příkaz se v JavaScriptu odděluje středníkem. Všechny identifikátory v JavaScriptu jsou case sensitive, to znamená, že záleží na velikosti písmen. Tak například proměnná *krestniJmeno* a *krestnijmeno* jsou v daném skriptu považovány jako dvě různé proměnné. Při deklaraci proměnných se používá klíčové slovo *var*. Jednoduchá deklarace proměnné pak může vypadat třeba takto:

```
var a;
```

Dobrou praxí je psát program do samostatného souboru, který je oddělen od HTML a má koncovku *.js*. Pro spojení tohoto souboru s HTML souborem se používá HTML tag <*script*> a jeho atribut *src*. Hodnota tohoto atributu je pak cestou k souboru s JavaScript kódem. Příklad:

```
<script src="script.js"></script>
```

Druhou možností je vložit samotný kód přímo do HTML souboru, a to mezi počáteční a koncový tag elementu *script*. Počáteční tag však již nesmí obsahovat atribut *src*.

Příklad:

```
<script>
alert("Ahoj!");
</script>
```

Je potřeba zmínit, že každý HTML soubor může obsahovat libovolný počet <*script*> tagů, které je možno umístit kamkoliv v kódu. Obvykle tento tag však najdeme před koncovým tagem </*head*> nebo před koncovým tagem </*body*>. Výhodou umístění před koncovým tagem </*body*> je to, že prohlížeč nejdříve načte a zobrazí veškeré HTML a až poté spustí JavaScript příkazy.[16][17]

4.2 HISTORIE JAZYKA JAVASCRIPT

Jazyk JavaScript vytvořil Brendan Eich v květnu roku 1995. Zpočátku byl tento jazyk nazýván Mocha, v září téhož roku byl přejmenován na LiveScript a po získání obchodní známky od firmy Sun byl naposled přejmenován na konečný název JavaScript. Tento název je čistě obchodní tah – v té době začínal být velmi populární jazyk Java. JavaScript má však k jazyku Java velmi daleko, přebírá od něj pouze část syntaxe a názvu, avšak filozofie JavaScriptu je zcela odlišná.

Od té doby se JavaScript postupně rozšiřoval a v roce 1997 byl standardizován jako ECMAScript. O rok později přišla nová verze známá jako ECMAScript 2 a v roce 1999 vychází ECMAScript 3, jenž se stal dobrým základem pro moderní JavaScript.

Další významný moment v historii JavaScriptu přišel s příchodem skupiny technologií AJAX (Asynchronous JavaScript and XML), jehož jádro tvoří právě JavaScript. Pojem AJAX se poprvé objevil v roce 2005 v článku Jesse James Garreta a využití najde při tvoření webových aplikací, kde se data načítají na pozadí a není tak potřeba znovu načítat celou stránku, což přispívá k tvorbě dynamického webového obsahu.[18]

Velký podíl na masivním rozšířením jazyka JavaScript má firma Google, a to právě díky použití skupiny technologií AJAX ve svém Gmailu.[19]

Verze ECMAScript 4 nebyla nikdy standardizována, avšak práce na ní se stala výchozím bodem pro verzi, která je nyní tou současnou – ECMAScript 5.[20]

4.3 KNIHOVNA JQUERY

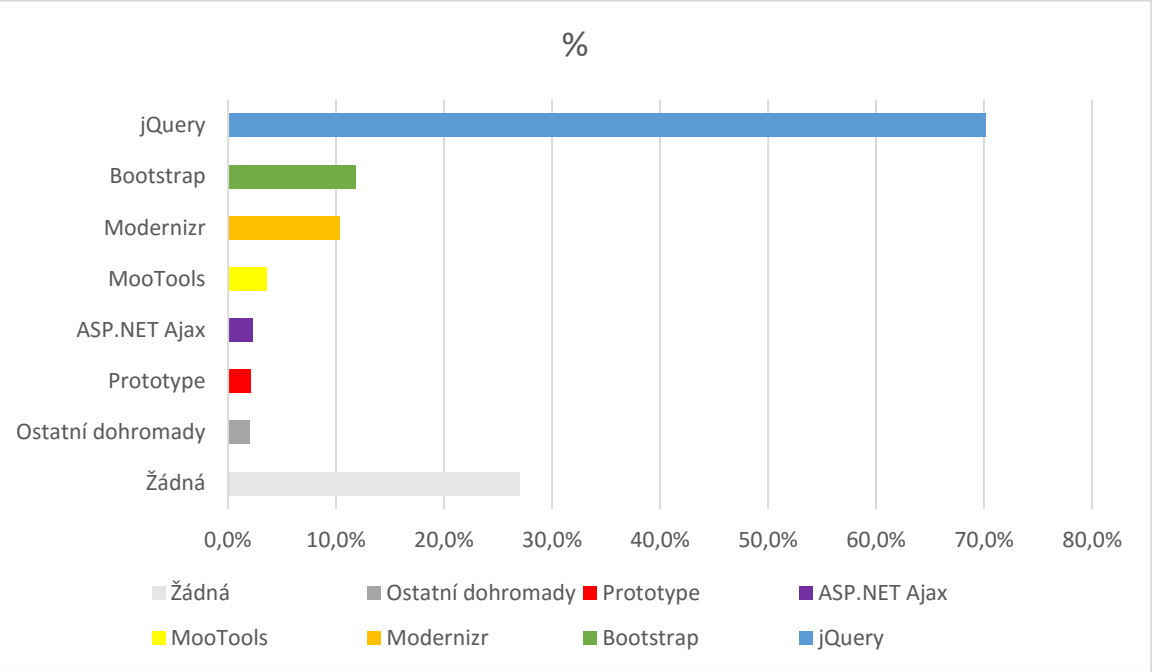
jQuery je JavaScriptová knihovna, která má za cíl zjednodušit kódování v JavaScriptu, čemuž také odpovídá jeho hlavní heslo: „Write Less, Do More“ (česky piš méně, dělej více). Vytvořena byla v roce 2006 a od té doby jeho popularita masivně vzrostla.[21]

jQuery nabízí velmi dobrou podporu v prohlížečích, jednoduchý přístup k HTML a manipulaci s DOM (Document Object Model), speciální funkce pro přístup k CSS, nástroje pro vytváření profesionálních animací a efektů, schopnost efektivně řídit události a v neposlední řadě také funkce pro řízení AJAX komunikace s webovým serverem.[22]

V současné době je jQuery vůbec nejoblíbenější a nejpoužívanější JavaScriptový Framework, což dokazují následující statistiky z webu *w3techs.com*.

Využití JavaScriptových knihoven na webových stránkách

Tabulka 3: *Graf využití JavaScriptových knihoven na webových stránkách*



Ze všech webových stránek jich 27 % nepoužívá žádnou JavaScriptovou knihovnu. jQuery využívá 70,1 % webových stránek, což je v porovnání s ostatními JavaScriptovými knihovnami suverénně nejvíce. [23]

5. PHP

5.1 ZÁKLADNÍ SYNTAXE JAZYKA PHP

PHP je flexibilní jazyk, jenž pracuje na straně serveru. Základy má v jazycích C a Perl, i když často připomíná spíše Javu.

PHP skript začíná *<?php* a končí *?>*. Výchozí koncovka pro PHP soubory je *.php*. Všechny klíčová slova, třídy, funkce a uživatelem definované funkce nejsou case sensitive, avšak všechny jména proměnných case sensitive jsou.[24] Klíčovou roli v syntaxi PHP hrají ještě další dva znaky, a to znak dolaru a středník.

Znak dolaru

Znak dolaru (*$*) je v PHP syntaxi velmi důležitý. Je ho totiž potřeba použít vždy, když deklarujeme proměnnou. Každá proměnná v PHP musí začínat právě znakem dolaru. Díky tomu může PHP parser pracovat rychleji, protože ihned ví, že se jedná o proměnnou.

Příklad:

```
<?php
$cislo = 1;
$retetec = “Ahoj”;
$pole = [];
?>
```

Středník

Jak je vidět v předchozím příkladu, každý PHP příkaz musí končit znakem středníku. Asi největší počet syntaktických chyb v PHP vzniká kvůli opomenutí středníku za nějakým PHP příkazem.[22]

5.2 HISTORIE JAZYKA PHP

Roku 1994 položil Rasmus Lerdorf základy PHP, když v jazyce C naprogramoval jednoduchý systém pro evidování přístupu k jeho životopisu, který měl na svých stránkách. Po čase se požadavky na tento systém zvyšovaly a Rasmus tak celý systém vylepšil, rozšířil, doplnil o dokumentaci a vydal pod názvem Personal Home Page Tools, zkráceně PHP Tools.

V roce 1995 Rasmus uvolnil zdrojový kód PHP Tools veřejnosti, kterou navíc vyzval, aby se snažila opravit chyby a přidala další vylepšení. V září téhož roku funkcionalitu PHP rozšířil a na krátkou chvíli upustil od názvu PHP a začal systém označovat jako Forms Interpreter neboli zkráceně FI. Jen o měsíc později Rasmus vydává kompletní přepracování zdrojového kódu, který nazval jako Personal Home Page Construction Kit.

V dubnu 1996 byl kód opět přepracován a Rasmus poprvé představil systém pod názvem PHP/FI. V červnu téhož roku pak spatřila světlo světa jeho druhá velká verze PHP/FI 2.0.

Následovala verze PHP 3, která byla první verzí, jenž už velmi připomínala PHP tak, jak ho známe dnes. Proti verzi předchozí byla mnohem rychlejší a měla značně rozšířenou funkcionalitu. Do té doby největší skok však přišel až s verzí PHP 4.

Jen krátce po představení PHP 3 se začalo pracovat na přepsání jádra PHP. Hlavním cílem bylo dosáhnout zvýšení výkonu u komplexních aplikací a vylepšení celkové modularity PHP. Věcí, která k tomuto dopomohla, byl nový engine, zvaný Zend Engine, jenž byl představen poprvé v roce 1999. V květnu roku 2000 byla verze 4 oficiálně vydána a přinesla především rozšířenou podporu pro webové servery, HTTP sessions, buffering výstupu, bezpečnější způsoby jak zacházet se vstupem od uživatele a také nové jazykové konstrukce.

Pátá verze, PHP 5, se objevila až o čtyři roky později. Představila druhou verzi svého enginu, Zend Engine 2.0, a spoustu nové funkcionality. Asi nejzásadnější změna byla v objektovém modelu jazyka.

Vývojový tým PHP se rozhodl, že v příští hlavní verzi název PHP 6 vynechá. Tento název již v minulosti existoval jako experimentální projekt, který však do produkční fáze nikdy nedospěl. Aby se předešlo zmatkům, bylo stanoveno, že další velká verze se bude jmenovat rovnou PHP 7. Ta přišla v prosinci 2015 a opět přinesla obrovský počet nových vylepšení a novinek. Ty nejzásadnější jsou popsány v následující kapitole. [25][26][27]

5.3 STRUČNÝ PŘEHLED NOVÝCH KLÍČOVÝCH VLASTNOSTÍ PHP 7

5.3.1 NOVÁ VERZE ENGINU ZEND

Od roku 1999 je PHP poháněno enginem s názvem Zend. PHP 5.x používá jeho druhou verzi pojmenovanou Zend Engine II, která rozšiřuje funkcionalitu první verze a výrazně navyšuje výkonnost jazyka PHP. S PHP 7 však přichází na řadu další verze s kódovým označením PHPNG. Největší předností této verze je výrazně lepší optimalizace paměti a zavedení kompilace „právě včas“ (just-in-time). Výsledkem je až dvakrát vyšší výkon.

5.3.2 KONSISTENTNÍ PODPORA 64 BITOVÝCH WINDOWS

Pod pojmem konsistentní 64 bitová podpora si můžeme představit nativní podporu 64 bitových celých čísel a velkých souborů. To by mělo zajistit bezproblémový běh PHP 7 na 64 bitových systémech Windows.

5.3.2 NOVÉ OPERÁTORY

5.3.2.1 OPERÁTOR KOALESCENCE

Tento operátor se značí *??* a vrací levý operand, pokud se hodnota operandu nerovná *NULL*. V opačném případě vrací pravý operand.

5.3.2.2 OPERÁTOR SPACESHIP

Značí se <=> a slouží k porovnání dvou výrazů. V případě, že je první porovnávaný výraz menší než druhý, vrací -1. Pokud se porovnávané výrazy rovnají, jako návratovou hodnotu dostaneme číslo 0. A v případě, kdy je první porovnávaný výraz větší než druhý, výsledkem bude 1.

5.3.3 DEKLARACE TYPŮ

5.3.3.1 DEKLARACE NÁVRATOVÉHO TYPU

Podobně jako u deklarace typu argumentu, můžeme v PHP 7 nastavit datový typ pro návratovou hodnotu funkce. Příkaz return tak nyní může určit, jaký datový typ příslušná funkce vrátí. Na výběr jsou stejné datové typy jako v případě deklarace datového typu argumentů.

5.3.3.2 DEKLARACE SKALÁRNÍHO TYPU

Deklarace skalárního typu nabízí dva druhy – coercive a strict, kde první zmíněný je standardní. Datové typy pro parametry string, int, float a bool mohou být nyní vynuceny – buď nuceně (coercive) nebo striktně (strict).

5.3.4 ODSTRANĚNÍ ZASTARALÝCH FUNKCIONALIT

V PHP 7 bylo odstraněno mnoho zastaralých funkcionalit i starých a nepodporovaných Server API (Application Programming Interface) a dalších rozšíření.

5.3.5 DEFINOVÁNÍ KONSTANTNÍCH POLÍ POMOCÍ DEFINE()

PHP 7 nabízí možnost definovat konstantní pole pomocí funkce *define()*. V PHP 5.6 to bylo možné pouze za pomocí klíčového slova *const*.

5.3.6 ANONYMNÍ TŘÍDY

Do PHP 7 byla přidána podpora anonymních tříd. Tyto třídy najdou využití v momentě, kdy potřebujeme vytvářet jednoduché, jednorázové objekty. Je to taková třída, která není pojmenovaná. K vytvoření takovéto třídy slouží klíčová slova *new class*. [27][28]

6. VYTVOŘENÍ NOVÉ WEBOVÉ STRÁNKY PRO SPOLEČNOST KL-ACCOUNTING S.R.O.

6.1 ANALÝZA SOUČASNÝCH STRÁNEK

Současná verze webových stránek společnosti KL-Accounting s.r.o. je realizována za pomocí redakčního systému Wordpress a šablony s názvem *Accounting – WP Business theme for Accountants* zakoupené na webu *themeforest.net*. Tato šablona i redakční systém již v základu obsahují mnoho možností a doplňků, které jsou ve většině případů nad rámec potřeb a požadavků na webovou prezentaci společnosti KL-Accounting s.r.o. Domnívám se také, že použití redakčního systému se hodí spíše na weby, kde je často přidáván nový obsah, jako například blogy, již však ne tolik na jednoduché statické stránky.

Myslím si, že vytvoření webové prezentace na míru se bude schopno více přizpůsobit požadavkům tohoto webu. Stránka se navíc zbaví zbytečného kódu a funkcí redakčního systému a šablony, jež v současném řešení zůstávají zcela nevyužity.

Po konzultaci se společností KL-Accounting s.r.o. bylo domluveno vytvořit stránky, které budou vzhledově vycházet ze stránek původních, budou splňovat náležitosti moderního webu a nebudou obsahovat nadbytečný kód. Díky tomuto se webová stránka „odlehčí“ a zrychlí se tak i její načítání. Obsah zůstane zachován.

6.2 PŘÍPRAVA

Nové stránky budou postaveny na technologiích HTML 5 a CSS 3 s vhodným využitím JavaScriptové knihovny jQuery a jazyka PHP.

Aby bylo možno jazyk PHP plně využívat, je potřeba webový server, PHP a databázi. Všechny tyto technologie zajišťuje *webhosting*. Pro vývoj a testování je však vhodné využít lokální server. Je proto nutné si tyto technologie doinstalovat na svůj počítač, a to po jednoduchém stáhnutí instalačního balíčku s názvem XAMPP, který obsahuje webserver Apache, PHP i MySQL databázi. Celý tento balíček je plně zdarma a je rozdělen do tří hlavních variant – podle toho, který operační systém je na počítači použit. Tento balíček byl

stážen z oficiálního webu *apachefriends.org*. Na výběr je verze pro Linux, Windows a OS X, každá z nich pak obsahuje několik dalších verzí. Jelikož je na mém počítači nainstalován operační systém Microsoft Windows 10, byla vybrána verze pro Windows.

Obrázek 3:

Náhled stránky apachefriends.org (Zdroj:[29])

Po jednoduché instalaci je nutno v XAMPP zapnout moduly Apache a MySQL. Ovládací panel, kde je toto možné se spustí ze složky, kam se XAMPP nainstaloval (standardně *C:\xampp*) přes soubor *xampp-control*. Zde se jednoduše u příslušných modulů klikne na tlačítko *start*.

Jako přístup k vytvoření této webové stránky jsem zvolil tzv. *mobile first*. To znamená, že v první fázi bude vytvořena verze pro mobilní zařízení a až poté se pomocí *media queries* (co *media queries* jsou, bude vysvětleno v pozdější části) přizpůsobí pro tablety a desktop.

Jako vývojové prostředí jsem zvolil Netbeans, které bylo stáhnuto z jeho oficiálních stránek - *netbeans.org*.

6.3 PRŮBĚH VYTVOŘENÍ NOVÝCH STRÁNEK

6.3.1 VYTVOŘENÍ HLAVIČKY STRÁNKY

V Netbeans se založí nový PHP projekt, čímž se vytvoří soubor *index.php*, jenž základně obsahuje tento základní kód:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        // put your code here
        ?>
    </body>
</html>
```

Jak lze vidět z uvedeného kódu, deklarace dokumentu *<!DOCTYPE html>* se vytvořila automaticky a značí, že se jedná o soubor s HTML5 kódem. Tato deklarace je povinná a je nutné ji uvést vždy na začátku kódu (před tagem *<html>*). Netbeans vytvořil i základní elementy každého HTML souboru - *html, head* a *body*, kde v elementu *head* definoval znakovou sadu pro tento dokument jako UTF-8. Uvnitř *head* předpřipravil také tag *<title>*, který je povinný pro každou HTML stránku a obsahuje název stránky, který pak lze vidět například na kartě příslušné stránky v prohlížeči. Jelikož byl vytvořen nový PHP projekt, tak se uvnitř *body* nachází prázdná definici PHP bloku.

Jako úplně první věc byla v projektu vytvořena složka *css*, do které byl nahrán soubor *normalize.css*. Tento soubor byl stáhnut na adrese *necolas.github.io/normalize.css/*. Obsahuje velký počet řádků s kaskádovými styly, které sjednocují zobrazení prvků na stránce ve všech prohlížečích. Proto název normalize, tedy normalizace.

Pro napojení souboru s kaskádovými styly na soubor HTML je nezbytné použít nepárový tag *<link>*, který patří dovnitř elementu *head*. Tagu *<link>* je potřeba definovat alespoň dva atributy. První z nich se jmenuje *href* a jeho obsah určuje cestu k požadovanému souboru s kaskádovými styly, v mém případě je tato cesta definována relativně, jak je vidět na kódu níže. Druhým atributem, který je potřeba definovat, je atribut *rel*, ten určuje vztah mezi HTML a souborem, který k němu chceme připojit, v tomto případě je nutno zvolit hodnotu *stylesheet*, což znamená, že se jedná o kaskádové styly.

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

V elementu *head* je deklarován také tento kód:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Tag <*meta*> s těmito atributy slouží jako dobrý základ pro každý responsivní web. Jelikož prohlížeče v chytrých mobilních telefonech či tabletech zobrazují web ve „zmenšené podobě“, aby byl vidět celý a až uživatel si jej musí ručně zvětšit. Toto v případě responzivních webů není vhodné a tak na řadu přichází definice <*meta name=“viewport“*>, která je schopna prohlížeče přemluvit k jinému chování. V mém případě hodnota atributu *content* říká, že se má šířka nastavit podle zařízení a meřítko, ve kterém se web zobrazí, bude 1:1.

Uvnitř *head* je vhodné uvést několik dalších tagů <*meta*>, které definují autora, popis a klíčová slova stránky. V případě tohoto projektu tyto definice vypadají takto:

```
<meta name="author" content="Karel Kvítek">
<meta name="description" content="Společnost, která poskytuje účetní a daňové služby">
<meta name="keywords" contant="účetnictví, daně, poradenství, mzdy">
```

Přidá se textový obsah elementu *title* a ve složce *css* se vytvoří další soubor s kaskádovými styly (tentokrát prázdný) a pojmenuje se *styles.css*. Tento soubor bude připojen k HTML souboru opět stejným způsobem jako soubor *normalize.css*.

Nyní je potřeba se přesunout do části *body*. Zde se jako první vytvoří párový tag <*header*>, který definuje hlavičku HTML stránky. V tomto případě obsahuje horní lištu, logo a hlavní navigaci stránky.

Horní lišta stránky není jen grafickým prvkem, ale obsahuje také telefonní kontakt.

```
<div class="top-bar">
            <a href="tel:+420774977237">+420 774 977 237</a>
 </div>
```

Element *a* slouží k definování odkazů na stránce. V tomto případě bylo v atributu *href* sděleno prohlížeči, že se jedná o telefonní číslo, které je možno na mobilních telefonech vytočit.

K telefonnímu číslu byla přidána příslušná ikonka s pomocí řešení s názvem Font Awesome (dostupné z *fortawesome.github.io/Font-Awesome/*). Už z názvu je patrné, že se jedná vlastně o písmo a také s ním lze takto zacházet. Není to však písmo ledajaké, jedná se o sbírku ikonek, která se s každou novou verzí rozšiřuje o další. S Font Awesome se pracuje stejně jako s klasický písmem, to znamená, že k jeho stylování se používá CSS. Autoři však zabudovali několik tříd přímo do frameworku a těch je možno využívat. Existuje několik

způsobů, jak je možné integrovat Font Awesome do vlastních stránek. Mezi nejvyužívanější a nejjednodušší patří vložení pomocí CDN (Content Delivery Network) a vlastního CSS. Při použití první zmíněné metody stačí vložit jeden řádek kódu do své stránky a všechny ikonky jsou ihned k dispozici. Druhou zmíněnou možností je stáhnutí adresáře *font-awesome* a následného nahrání do projektu. Poté je potřeba nalinkovat z HTML stránky soubor *font-awesome.min.css*.

Podrobný popis všech metod lze nalézt přímo na oficiální stránce - *fortawesome.github.io/Font-Awesome/get-started/*.

V tomto projektu byla využita metoda s CDN, kde není potřeba nic instalovat, ani stahovat. Do stránky se tak vložil pouze následující řádek kódu:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.5.0/css/font-awesome.min.css">
```

V následujícím kroku se v projektu vytvoří další složka, tentokrát s názvem *img*, do které budou postupně nahrány veškeré obrázky, jenž se na stránce vyskytnou.

Jako první bylo do této složky nahráno logo společnosti, které bylo získáno na současných webových stránkách společnosti. Do nových stránek bylo poté vloženo pomocí nepárového tagu <*img*> s několika atributy. Atribut s názvem *src* určuje zdroj obrázku a atribut *alt* specifikuje alternativní text k obrázku pro případ, že obrázek nemůže být zobrazen. Atribut *alt* je jedním ze základních prvků při tvorbě přístupného webu. Přidán byl ještě atribut *id* pro snazší a přehlednější výběr elementu v CSS a atributy *width* a *height* pro nastavení šířky, respektive výšky.

```
<img src="img/logo-kl-acc.png" alt="logo" id="logo" width="250"
height="46">
```

Poslední část, která se bude v *header* nacházet, je navigace stránky. Typicky se dělá tak, že se vytvoří pomocí elementu *ul* neseřazený seznam s odpovídajícím počtem položek, jenž se definují pomocí elementů *li*. Aby se na stránce určilo, že jde o navigaci, celý seznam se „obalí“ elementem *nav*. HTML kód pro výslednou navigaci vypadá takto:

```
<nav>
                <ul>
                    <li><a href="#">Úvodní stránka</a></li>
                    <li><a href="#">O nás</a></li>
                    <li><a href="#">Ceník</a></li>
                    <li><a href="#">Služby</a></li>
                    <li><a href="#">Kontakt</a></li>
                </ul>
</nav>
```

Místo znaku # se později doplní název souboru, na který se bude odkazovat.

Dalším postupem je stylování dosud vytvořeného kódu. Využije se tak soubor *styles.css*, který je již založený.

Jako první se za pomocí selektorů vybere horní lišta stránky a její obsah. Samotné navigační liště se definuje barva pozadí a padding. Padding určuje velikost místa mezi obsahem elementu a jeho rámečkem. Pro lepší vysvětlení je vhodné si popsat jak tzv. box model v CSS funguje. Nejlépe k tomu poslouží následující obrázek.

Obrázek 4:

Box model v CSS (Zdroj: [30])

Vysvětlení jednotlivých části:

- **Content** – obsah, ve kterém se nachází text nebo obrázky
- **Padding** – plocha mezi obsahem a rámečkem (Border)
- **Border** – rámeček, který lemuje Padding a Content
- **Margin** – plocha za rámečkem [30]

Výsledná deklarace horní lišty v CSS pak vypadá takto:

```
.top-bar {
    background-color: #0fb3d0;
    padding: 0.5em 0;
    text-align: center;
}
```

Barva pozadí je definována pomocí hexadecimálního kódu a u *padding* je nastavena hodnota v relativní jednotce *em*. Tato jednotka mění svou velikost v závislosti na aktuální velikosti písma. 1 *em* odpovídá výšce jednoho řádku základního písma. Vlastnost *text-align* určuje zarovnání obsahu na stránce, v tomto případě je nastavena na hodnotu *center*, tedy na střed.

Pro lepší viditelnost a konzistenci v designu stránky bylo nutné definovat styly u obsahu horní lišty.

```
.top-bar a {
    color: #fff;
    text-decoration: none;
}
```

Pomocí vlastnosti *color* se nastavila barva písma na bílou a prostřednictvím vlastnosti *text-decoration* s hodnotou *none* se odstranilo podržení, které je u elementu *a* výchozí.

Aby bylo mezi jednotlivými prvky obsahu horní lišty více prostoru, bylo u ikonky telefonu nastaveno *padding-right* i *padding-left* na hodnotu *5px*.

```
.top-bar .fa-phone-square {
    padding-right: 5px;
    padding-left: 5px;
}
```

Jako další byly přiřazeny styly logu. Zde se pouze nastavil *margin-top* na hodnotu *15px*.

```
#logo {
    margin-top: 15px;
}
```

Tyto ukázky také demonstrují výběr třídy (*class*) a *id* v CSS. V případě horní lišty byla za použití znaku tečky vybrána třída s názvem *top-bar*. U loga byl použit znak #, jelikož se jedná o *id*.

Nyní se přejde ke stylování navigace. Pro zbavení se teček u každé položky v seznamu je třeba u něj použít vlastnost *list-style-type* s hodnotou *none*. Celá definice u neseřazeného seznamu, představující navigaci, vypadá následovně:

```
nav ul {
    list-style-type: none;
    padding: 0;
}
```

Padding na hodnotu *0* je zde nastaven z toho důvodu, že defaultní hodnota u tohoto elementu má u vlastnosti *padding-left* nastavenu hodnotu *40px*, což by bránilo přesnému vycentrování, čemuž je třeba předejít.

Další definice říká, že má být u každé položky v seznamu souvislý spodní rámeček o velikosti 1 px.

```
nav li {
    border-bottom: 1px solid #fff;
    background-color: #0fb3d0;
}
```

Jak je vidět z této ukázky, některým vlastnostem v CSS je možno přiřadit více hodnot.

Každému odkazu uvnitř navigace byly přiřazeny následující vlastnosti a hodnoty:

```
nav a {
    text-decoration: none;
    text-transform: uppercase;
    font-size: 1.2em;
    color: #fff;
    display: block;
    text-align: center;
    padding: 0.2em 0;
}
```

Vlastnost *text-transform* s hodnotou *uppercase* určuje, že všechen text, který je definován v navigaci jako odkaz, bude zapsán velkými písmeny. Aby bylo možno pracovat s elementem *a* jako s block elementem, je potřeba mu tuto vlastnost nastavit pomocí *display: block*. Element *a* je totiž standardně inline element.

Block element je takový element, který začíná vždy na novém řádku a je roztažen přes jeho celou možnou délku. Mezi nejpoužívanější block elementy patří například *div*, *form*, *h1 - h6* nebo třeba element pro odstavec - *p*.

Inline elementy nezačínají na novém řádku a jsou široké jen tolik, kolik je pro jejich zobrazení potřeba. Jako nejčastěji používané inline elementy lze uvést například již zmíněný element *a* nebo třeba elementy *span* a *img*.

Navigace na mobilním zařízení zabírá zbytečně moc místa a je vhodné ji ve výchozím stavu skrýt a zobrazit pouze po ťuknutí na příslušný prvek na stránce. Zde přichází na řadu jQuery. Nejprve se však musí onen prvek vytvořit v HTML a nastylovat v CSS.

Jako vzhled zmíněného prvku byl zvolen znak tří svislých čar nad sebou, jak je dnes dobrou zvyklostí „schovanou“ navigaci označovat.

Využit byl opět Font Awesome a do HTML kódu se přidala příslušná ikonka:

```
<div class="toggle"><a href="#"><span class="fa fa-bars"></span></a></div>
```

Tomuto prvku se přiřadila třída, aby bylo možné se na něj jednoduše odkazovat v CSS a také byl „obalen“ do elementu *a*, aby bylo vidět, že je možné na něj kliknout.

Do CSS byly přidány následující řádky kódu:

```
.toggle {
    font-size: 2em;
    float: right;
    margin-top: 20px;
}

.toggle a {
    text-decoration: none;
    color: #fff;
    background-color: #0fb3d0;
    display: block;
    padding: 0 0.4em;
}
```

Tímto kódem se upravil přidaný prvek tak, aby ladil se zbytkem stránky, a také mu bylo nastaveno vhodné zarovnání na stránce pomocí vlastnosti *float* s hodnotou *right*.

Prvek pro zobrazení hlavní navigace a logo byly „obaleny“ do elementu *div* s třídou *main-wrapper* a této třídě se v CSS určila, pomocí vlastností *max-width* a *width*, šířka na stránce a také zarovnání na střed, kde bylo využito metody s vlastností *margin* a její hodnoty *auto*. Tagu <*img*>, zde představující logo, se přidaly atributy pro výšku a šířku.

HTML:

```
<div class="main-wrapper">
                <img src="img/logo-kl-acc.png" alt="logo" id="logo"
width="250" height="46">
                <div class="toggle"><a href="#"><span class="fa fa-
bars"></span></a></div>
</div>
```

CSS:

```
.main-wrapper {
    max-width: 1200px;
    width: 95%;
    margin: 0 auto;
}
```

Nyní je potřeba zajistit, aby byla hlavní navigace na mobilních zařízeních ve výchozím stavu skrytá a zobrazila se pouze po ťuknutí na příslušný prvek, který zde představuje znak tří čar.

K zajištění této funkcionality se využije JavaScript, respektive jQuery. Aby bylo možné jQuery v projektu vůbec využívat, je potřeba ho nejdříve napojit na příslušný HTML soubor. Na oficiálních stránkách jQuery (*jquery.com*) se klikne na tlačítko download, které slouží k přesměrování na stránku, kde se nachází několik metod, jak je možné jQuery do svého projektu stáhnout. Jako v předchozím případě s Font Awesome, se využije metoda s CDN a daný kód se vloží do stránky (na dno obsahu elementu *head*).

```
<script src="https://code.jquery.com/jquery-1.12.0.min.js"></script>
```

Nyní je možné v projektu využívat jQuery. Vedle již vytvořených složek *img* a *css* se založí další a pojmenuje se *js*. V této složce bude následně vytvořen nový javascriptový soubor s názvem *script.js*.

Je potřeba napojit nově vytvořený soubor k souboru s HTML. Udělá se to tak, že na dně uvnitř elementu *head* se vytvoří tag <*script*> s atributem *src*, jehož hodnota bude relativní cesta k potřebnému souboru.

```
<script src="js/script.js"></script>
```

Ještě před psaním samotného jQuery kódu je potřeba v CSS nastavit, aby byla navigace ve výchozím stavu skryta a také chování pro třídu, která bude zobrazování navigace zajišťovat. Tato třída je pojmenována *show*.

```
nav {
    display: none;
}
.show {
    display: block;
}
```

Pokud nebude navigace obsahovat třídu *show*, hodnota vlastnosti *display* bude nastavena na *none* a navigace zůstane skryta. Pokud navigace třídu *show* obsahovat bude, hodnota *display* bude *block* a navigace se zobrazí.

V jQuery se zajistí, aby se elementu *nav* přidala třída *show* po kliknutí na prvek pro zobrazení navigace, pokud ji ještě neobsahuje a naopak.

```
$(document).ready(function () {
    $(".toggle").click(function () {
        $("nav").toggleClass("show");
    });
});
```

Část `$(document).ready(function () {});` zajišťuje, že se kód uvnitř bloku spustí až v momentě, kdy bude DOM kompletně načten.

```
$(".toggle").click(function () {
        $("nav").toggleClass("show");
    });
```

V této části se nejdřív vybere element se třídou *toggle* a stanoví se, co se stane po události (kliknutí). V tomto případě se vybere element *nav* a podle toho zda třídu *show* obsahuje nebo ne, se mu odebere nebo přidá.

Dále se v kořenovém adresáři vytvoří všechny stránky, které se na výsledné webové stránce budou vyskytovat. Jmenovitě se jedná o *about.php, pricelist.php, about.php* a *contact.php*. Všechny tyto stránky obsahují koncovku *.php*, protože v nich bude zahrnuta část PHP kódu.

Do již vytvořené navigace se doplní odkazy na tyto stránky:

```
<nav>
                <ul>
                    <li><a href="index.php">Úvodní stránka</a></li>
                    <li><a href="about.php">O nás</a></li>
                    <li><a href="pricelist.php">Ceník</a></li>
                    <li><a href="services.php">Služby</a></li>
                    <li><a href="contact.php">Kontakt</a></li>
                </ul>
</nav>
```

Hlavička je kompletně hotová, a jelikož vím, že všechny další stránky budou mít tuto hlavičku totožnou, využije se jazyka PHP a jeho funkce *include,* aby se nevytvářel duplicitní kód a v případě změny v hlavičce stačilo upravit kód pouze jednou.

Do projektu se přidá další složka a pojmenuje se *inc*. V této složce se vytvoří soubor s názvem *header.php*, jenž bude obsahovat HTML kód hlavičky stránky. Ten se tedy vyjme z *index.php* a vloží právě do *header.php*. Obsah *header.php* bude vypadat následovně:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1">
        <meta name="author" content="Karel Kvítek">
        <meta name="description" content="Společnost, která poskytuje
účetní a daňové služby">
        <meta name="keywords" contant="účetnictví, daně, poradenství,
mzdy">
        <title>KL-Accounting s.r.o.</title>
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.5.0/css/font-awesome.min.css">
        <link href="css/normalize.css" rel="stylesheet">
        <link href="css/styles.css" rel="stylesheet">
        <script src="https://code.jquery.com/jquery-
1.12.0.min.js"></script>
        <script src="js/script.js"></script>
    <body>
        <header>
            <div class="top-bar">
                <a href="tel:+420774977237"><span class="fa fa-phone-square
fa-lg"></span>+420 774 977 237</a>
            </div>
            <div class="main-wrapper">
                <img src="img/logo-kl-acc.png" alt="logo" id="logo"
width="250" height="46">
                <div class="toggle"><a href="#"><span class="fa fa-
bars"></span></a></div>
            </div>
            <nav>
                <ul>
                    <li><a href="index.php">Úvodní stránka</a></li>
                    <li><a href="about.php">O nás</a></li>
                    <li><a href="pricelist.php">Ceník</a></li>
                    <li><a href="services.php">Služby</a></li>
                    <li><a href="contact.php">Kontakt</a></li>
                </ul>
            </nav>
        </header>
```

Na začátek *index.php* i všech ostatních stránek se napíše jednoduchý PHP kód, který do nich zahrne obsah *header.php*.

```
<?php include("inc/header.php"); ?>
```

3.6.2 VYTVOŘENÍ PATIČKY STRÁNKY

Dále se v *index.php* vytvoří patička stránky, která bude obsahovat copyright informace. Použije se k tomu sémantický element HTML 5 *footer*.

```
        <footer>
            <p>&copy; <?php echo date("Y"); ?> KL-ACCOUNTING</p>
        </footer>
```

Aby bylo možné zapsat znak copyrightu, musí se použít HTML entita s příslušným kódem. V tomto případě *©*. Rezervované znaky a znaky, které na klávesnici nejsou, se vyjádří právě pomocí těchto entit. Kompletní přehled všech entit a k ním příslušných znaků lze nalézt například na - *dev.w3.org/html5/html-author/charref*.

Datum bylo vyjádřeno pomocí PHP funkce *date*. Zajistilo se tak, že se datum nebude muset měnit každý rok ručně a bude stále aktuální.

V souboru *styles.css* byly přidány elementu *footer* jednoduché styly, které jeho obsah vycentrují a barevně sladí s již vytvořenou hlavičkou.

```
footer {
    text-align: center;
    background-color: #0fb3d0;
    color: #fff;
}

footer p {
    margin: 0;
    padding: 0;
    padding: 0.5em 0;
}
```

Nyní, podobně jako v případě hlavičky, se vytvoří ve složce *inc* PHP soubor, který bude obsahovat patičku a ten se pomocí funkce *include* zahrne na všechny stránky v projektu. Soubor bude pojmenován *footer.php* a jeho obsah se vytvoří vyjmutím veškerého zbylého HTML kódu, který se současně v *index.php* nachází. Obsah *footer.php* tak bude vypadat takto:

```
<footer>
    <p>&copy; <?php echo date("Y"); ?> KL-ACCOUNTING</p>
</footer>
</body>
</html>
```

Jednotlivé stránky se tak zredukovaly pouze na tento kód:

```
<?php include("inc/header.php"); ?>
<?php include("inc/footer.php"); ?>
```

Obě funkce byly ponechány v samostatném PHP bloku, protože se mezi ně bude vkládat HTML kód.

V případě, kdy je obsah stránky kratší než výška prohlížeče, patička zůstane hned za obsahem, někde uprostřed stránky. Lepším řešením by bylo, kdyby v takovýchto případech byla patička vždy „přilepená“ k dolnímu okraji stránky v závislosti na výšce prohlížeče.

K dosažení požadované vlastnosti bylo zvoleno využití CSS funkce *calc()* v kombinaci s jednotkou viewportu *vh*, což je zkratka pro *viewport height*. 1 vh pak označuje setinu výšky viewportu.

Do kódu se přidá nový *div*, který „obalí“ veškerý obsah v *body* kromě elementu *footer*. Tomuto *divu* se přidá třída a nazve se *wrap*. V *header.php* se tak přidá ihned za počáteční tag *<body>* počáteční tag požadovaného *div* elementu.

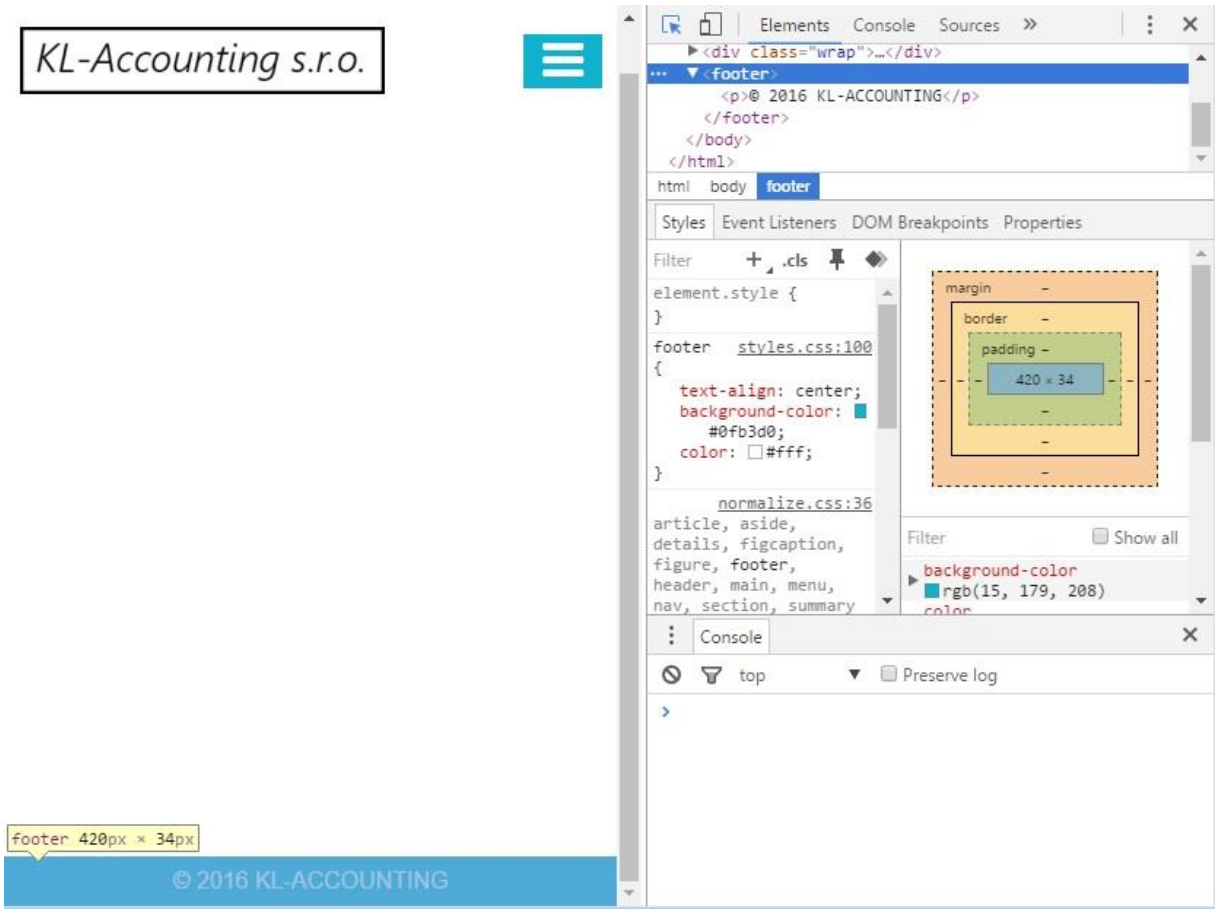
```
<div class="wrap">
```

Do *footer.php*, před element *footer* se přidá koncový tag tohoto *div* elementu.

```
</div>
```

Díky využití PHP funkce *include* nyní stačilo vložit kód na dvě místa a nebyla potřeba ho tak vkládat do každé stránky zvlášť.

Ve *styles.css* se vytvoří nové pravidlo, které vybere třídu *wrap* a přidá mu vlastnost *min-height*, jenž určuje minimální výšku vybraného elementu. Této třídě se nastaví hodnota *100vh*, což vlastně znamená, že obsah vybraného elementu bude přes celou výšku prohlížeče. Pouze nastavením této hodnoty by však byla patička skrytá a aby byla vidět, uživatel stránek by byl nucen scrollovat. Je tak potřeba od této hodnoty odečíst výšku elementu *footer*. Tato hodnota se v prohlížeči Google Chrome zjistí v nástrojích pro vývojáře, které se spustí vyvoláním klávesové zkratky *shift + ctrl + I*. V těchto nástrojích se pak v záložce *Elements* klikne na element, u kterého je potřeba zjistit požadované vlastnosti. Jak je vidět, přímo u elementu na stránce vyskočilo malé okénko, kde je napsaná velikost příslušného elementu.



Obrázek 5:

Nástroje pro vývojáře v prohlížeči Google Chrome (Zdroj: Autor)

Když je zjištěna výška elementu *footer*, je známo, jakou hodnotu je potřeba odečíst. Se samotným výpočtem pomůže funkce *calc()*. Jako její parametr se vloží příslušný výpočet. Této třídě se ještě definuje vlastnost *overflow*, která určuje, jak bude zacházeno s obsahem, jenž vyteče z rozměru prvku. Její hodnota se nastaví na *hidden*. Díky tomuto bude vyřešen problém, kdy se mezi tímto elementem a elementem *footer* vytvářelo horizontální bílé místo, které způsobilo, že patička nebyla přilepena na dolním okraji prohlížeče, ale byla posunuta z části pod něj a bylo nutné scrollovat.

```
.wrap {
    min-height: calc(100vh - 34px);
    overflow: hidden;
}
```

Patička je nyní „přilepena“ k dolnímu okraji stránky.

Dostávám se k části projektu, kdy je potřeba vytvořit obsah mezi hlavičkou a patičkou u jednotlivých stránek a začne se tedy lišit i obsah v kódu jednotlivých stránek. Jako zdroj

textového obsahu a obrázků budou využity současné stránky společnosti KL-Accounting s.r.o..

3.6.3 VYTVOŘENÍ SPECIFICKÉHO OBSAHU JEDNOTLIVÝCH STRÁNEK

3.6.3.1 HLAVNÍ STRÁNKA

Začne se u hlavní stránky, tedy *index.php*. Mezi hlavičku a patičku se vloží element *main*, do kterého se následně vloží fotka a text ze současných stránek ve formě odstavců (první s třídou *introduction*) a neseřazeného seznamu.

```
<main class="index">
    <img src="img/ucetni.jpg" width="2000" height="550" alt="účetní">
    <p class="introduction">
        Jsme společnost, která poskytuje účetní a daňové poradenství.
Nabízíme profesionální služby a specializujeme se na vedení daňové
evidence, účetnictví a veškeré poradenství v oblasti účetnictví, daní a
mezd.
    </p>
    <p>
        Účetní a daňová kancelář KL-Accounting nabízí:
    </p>
    <ul>
        <li>Zpracování účetnictví a daňové evidence</li>
        <li>Sestavení veškerých daňových přiznání a hlášení</li>
        <li>Komplexní vedení mzdové evidence</li>
        <li>Jednání s příslušnými úřady</li>
        <li>Poskytování poradenství v oblasti účetnictví a daní</li>
    </ul>
</main>
```

Pro lepší čitelnost se elementu *main* v CSS definují vlastnosti, které zajistí, že bude mezi okraji prohlížeče a obsahem stránky nějaké místo a obsah bude vycentrovaný.

```
main {
    max-width: 1200px;
    width: 95%;
    margin: 0 auto;
    text-align: center;
}
```

Tyto vlastnosti nebyly definovány jen pro konkrétní třídu tohoto elementu, ale obecně pro všechny elementy *main*, to zajistí, že stejného výsledku bude dosaženo na všech stránkách, kde bude element *main* použit.

Pro zobrazení jednotlivých položek již vytvořeného neseřazeného seznamu se použije režim rozvržení, který je novinkou v CSS3 - flexbox. Flexbox je vlastně množina CSS vlastností a jejich hodnot, které poskytují flexibilní způsob, jak rozložit obsah stránky. Díky jeho použití se zajistí, že se prvky seznamu budou chovat správně a předvídatelně při jakékoliv změně

velikosti obrazovky a na všech zařízeních. Flexbox je tak ideálním řešením pro responzivní stránky.

Flexbox se skládá z tzv. flex kontejneru a flex položek. Každý flex kontejner obsahuje jednu nebo více flex položek. Jako flex kontejner se může deklarovat jakýkoliv block nebo inline element.

Flexbox se deklaruje tak, že se elementu nastaví vlastnost *display* s hodnotou *flex* nebo *inline-flex*. Pomocí flexboxu se může měnit směr, velikost a pořadí elementů bez ohledu na jejich původní velikost a pořadí v HTML. [31]

V tomto projektu byl deklarován jako flex kontejner element *ul* a elementy *li*, které obsahuje, budou definovány jako flex položky.

Pomocí CSS selektorů se vybere neseřazený seznam a jeho položky uvnitř elementu *main* a přidá se jim základní styly, které vzhledově korespondují se zbytkem stránky.

```
main ul {
    list-style-type: none;
    padding: 0;
}
main ul li {
    text-align: center;
    font-size: 1.2em;
    color: #fff;
    background-color: #0fb3d0;
    margin: 0.5em 1.5em;
    padding: 0.5em;
    border-radius: 3px;
}
```

Vlastnost *border-radius* určuje poměr zakulacení rohů elementu. V tomto případě má hodnotu *3px* a je definována u elementu *li*.

Tomuto neseřazenému seznamu se přidá vlastnost *display* a nastaví se jí hodnota *flex*. Tímto se určilo, že element *ul* je flex kontejner a každý element v něm bude flex položkou.

```
main ul {
    list-style-type: none;
    padding: 0;
    display: flex;
}
```

Jelikož je výchozí hodnota u vlastnosti, která určuje směr prvků uvnitř flex kontejneru (*flex-direction*), nastavena na hodnotu *row*, což znamená umístnění prvků v řadě, a v případě mobilního zobrazení je potřeba, aby se prvky zobrazovaly pod sebou ve sloupci, je nutno tuto hodnotu přenastavit. U flex kontejneru se tak hodnota *flex-direction* nastaví na *column*.

```
main ul {
    list-style-type: none;
    padding: 0;
    display: flex;
    flex-direction: column;
}
```

V mobilní verzi v prohlížeči však zatím díky použití flexboxu není žádná změna zaznamenána. Vhodně se ho však využije při tvorbě desktopové verze.

3.6.3.2 O NÁS

Nyní se přesunu do souboru *about.php*, kde je potřeba přidat textový obsah a příslušnou fotku.

```
<?php include("inc/header.php"); ?>
<main>
    <img src="img/ucetnictvi.jpg" alt="účetnictví" id="logo" width="585"
height="404">
    <p>
        Jsme společnost, která poskytuje účetní a daňové poradenství.
Nabízíme profesionální služby a specializujeme se na vedení daňové
evidence, účetnictví a veškeré poradenství v oblasti účetnictví, daní a
mezd.
    </p>
    <p class="divider">&therefore;</p>
    <p>
        Služby poskytujeme právnickým osobám podnikatelského i
nepodnikatelského typu, fyzickým osobám podnikatelům i fyzickým osobám
nepodnikatelům, kteří potřebují v těchto oblastech poradenství, popřípadě
potřebují podat daňové přiznání.
    </p>
    …
</main>
<?php include("inc/footer.php"); ?>
```

Pro lepší čitelnost a oddělení jednotlivých odstavců byl mezi každou dvojici odstavců vložen element *p* s třídou, která se nazvala *divider*. Obsahem tohoto elementu je entita s označením *∴*.

```
.divider {
    color: #0fb3d0;
    font-size: 2.5em;
    text-align: center;
    padding: 0;
    margin: 0;
}
```

3.6.3.3 CENÍK

Jako další na řadě je potřeba upravit *pricelist.php*, který má obsahovat krátký text s odkazem na stažení ceníku ve formátu *xlsx* a několik tabulek, které představují samotný ceník. Element *table* deklaruje samotnou tabulku, *tr* je pak řádek v dané tabulce a *td* je buňka tabulky. Element *thead* představuje hlavičku dané tabulky.

```
<?php include("inc/header.php"); ?>
<main>
    <p>
        Cena je odvozována od množství dokladů a specifikace činnosti
jednotlivých zákazníků
    </p>
    <p>
        Kompletní ceník v Excelu si můžete stáhnout zde
    </p>
    <ul class="tables">
        <li>
            <table>
                <thead>
                    <tr>
                        <th colspan="2">Vedení daňové evidence</th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td>Položka evidence příjmů a výdajů</td>
                        <td>15 Kč</td>
                    </tr>
                    <tr>
                        <td>Minimální měsíční částka - neplátce DPH</td>
                        <td>500 Kč</td>
                    </tr>
                    <tr>
                        <td>Minimální měsíční částka - plátce DPH</td>
                        <td>1000 Kč</td>
                    </tr>
                    <tr>
                        <td>Zpracování závěrky</td>
                        <td>500 Kč</td>
                    </tr>
                </tbody>
            </table>
        </li>
        …
    </ul>
</main>
<?php include("inc/footer.php"); ?>
```

V kódu výše je vidět, že byla každá tabulka umístěna do položek neseřazeného seznamu, a to hlavně z důvodu lepší manipulace s jednotlivými položkami. Neseřazenému seznamu se přiřadila třída s názvem *tables*.

```
table {
    margin: auto;
}

td {
    min-width: 75px;
    border: 1px solid #0fb3d0;
}

.tables li {
    margin: 0.5em 0;
    color: #000;
    background-color: #fff;
}

.tables thead {
    color: #fff;
    background-color: #0fb3d0;
}
```

Vlastnost *min-width* s hodnotou *75px* je u elementu *td* nastavena z důvodu, že na mobilních zařízeních s nižším rozlišením se může cena a k ní příslušná měna odstavit na dva řádky. Při určení minimální šířky buňky tabulky však tomuto lze předejít.

Dále se vytvoří složka *files*, kam bude umístěn soubor s ceníkem - *cenik-kl-accounting.xlsx*.

```
<p>
        Kompletní ceník v Excelu si můžete stáhnout <a href="files/cenik-
kl-accounting.xlsx">zde</a>
 </p>
```

Tímto je stránka s ceníkem hotová a je možno přejít na *services.php*, tedy na stránku služby.

3.6.3.4 SLUŽBY

V HTML se zde vytvoří pět neseřazených seznamů.

```
<?php include("inc/header.php"); ?>
<main>
  <h1>V oblasti daňové evidence</h1>
  <ul>
    <li>Zpracování veškerých dokladů</li>
    <li>Evidence pohledávek a závazků</li>
    <li>Zpracování roční závěrky</li>
    <li>Vedení inventárních karet hmotného majetku</li>
  </ul>
  …
</main>
<?php include("inc/footer.php"); ?>
```

3.6.3.5 KONTAKT

Jako poslední zbývá stránka s kontaktem na společnost, *contact.php*. Zde bude obsah tvořit kontaktní informace společnosti, mapa s adresou společnosti a kontaktní formulář.

Kontaktní informace byly umístěny do elementu odstavce s třídou *contact-text*. Název společnosti bude třeba zobrazit tučně, takže se „obalil“ *span* elementem s třídou *bold*, které bude později v CSS tato vlastnost definovaná. Na konci každého řádku se umístil nepárový tag <*br*>, který zajišťuje ukončení řádku a další text tak pokračuje na nové řádce.

```
    <p class="contact-text">
        <span class="bold">KL-Accounting s.r.o.</span><br>
        IČO: 01823086<br>
        DIČO: CZ01823086<br>
        Luková 64<br>
        Brodek u Přerova<br>
        E-mail: <a href="mailto:kl-accounting@seznam.cz">kl-
accounting@seznam.cz</a><br>
        tel.: <a href="tel:+420774977237">+420 774 977 237</a><br>
    </p>
```

Opět se u telefonního čísla řeklo prohlížeči, že se jedná o telefonní číslo, které je možno vytočit a u e-mailu bylo definováno, že se jedná o e-mailovou adresu.

Mapa s adresou se nechá vygenerovat na Google mapách (*google.cz/maps*). Vygenerovaný HTML kód se pak vloží na příslušné místo na stránce.

Vygenerovaný kód z Google map vypadá takto:

```
<iframe
src="https://www.google.com/maps/embed?pb=!1m14!1m8!1m3!1d5184.975598412112
!2d17.369048!3d49.475292!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x4713aceed6be59
41%3A0x21f03237e0c1092d!2zTHVrb3bDoSA2NCwgNzUxIDAzIEJyb2RlayB1IFDFmWVyb3ZhL
CDEjGVza8OhIHJlcHVibGlrYQ!5e0!3m2!1scs!2scz!4v1461769551828" width="400"
height="300" frameborder="0" style="border:0" allowfullscreen></iframe>
```

Kontaktní formulář se v HTML vytváří pomocí elementu *form*, který v tomto případě obsahuje atribut s názvem *method* a hodnotou *POST*. To značí, že HTTP (HyperText Transer Protocol) metoda, za pomoci které se formulář odešle, bude právě *POST*.

Existuje ještě alternativní metoda, a to metoda *GET*. Hlavní rozdíl mezi *GET* a *POST* je v tom, že v případě *GET* se veškerá formulářová data předávají jako součást URL (Uniform Resource Locator), kdežto při použití metody POST se předají pouze v těle dotazu.

Protože je vhodné předejít tomu, aby se formulářová data zobrazovala v URL, bylo použito metody *POST*.

```
  <form method="POST">
        <h1>Kontaktní formulář</h1>
        <label for="name">Jméno:</label>
        <input type="text" id="name" name="name">
        <label for="email">E-mail:</label>
        <input type="email" id="email" name="email">
        <label for="message">Zpráva:</label>
        <textarea id="message" name="message"></textarea>
        <label for="year">Aktuální rok:</label>
        <input type="number" id="year" name="year">
        <input type="submit" value="Odeslat">
    </form>
```

Element *input* představuje vstupní pole, kam může uživatel vkládat datový obsah. Atribut *type* určuje, jaký typ bude dané pole mít (například text, číslo nebo email). V případě, že bude uvedena hodnota tohoto atributu jako *submit,* HTML definuje tlačítko, které bude sloužit k odeslání formuláře. Element *textarea* definuje vstupní pole, kam je možné vkládat větší množství textu, který může být na více řádcích. Pole pro aktuální rok se definovalo z důvodu ochrany proti spamu. Hodnotu tohoto pole bude kontrolovat PHP.

Pro kontaktní informace se definovaly následující styly:

```
.bold {
    font-weight: bold;
}
.contact-text a {
    color: #0fb3d0;
    text-decoration: none;
    font-weight: bold;
}
```

Styly pro kontaktní formulář:

```
form {
    max-width: 900px;
    margin: 10px auto;
    padding: 10px 20px;
    background: #f4f7f8;
    border-radius: 8px;
}

input, textarea {
    box-sizing: border-box;
    border: none;
    margin: 0;
    padding: 15px;
    width: 100%;
    background-color: #e8eeef;
    color: #008080;
    box-shadow: 0 1px 0 #0fb3d0 inset;
    margin-bottom: 30px;
}
```

Vlastnost *box-shadow* definuje stínování. Hodnota *inset* u vlastnosti *box-shadow* určuje, že se bude jednat o vnitřní stín.

Box-sizing s hodnotou *border-box* zajistí, že vlastnosti *width* a *height* zahrnují obsah, padding i rámeček.

Nyní přejdu k PHP části formuláře. Veškerý PHP kód, týkající se formuláře, bude psán do stejného souboru jako HTML část formuláře. Je potřeba ošetřit validaci formuláře, tedy ověření, zda je správně vyplněný. Musí se podchytit i možnost, kdy se formulář nemusel odeslat vůbec. Na úplný začátek *contact.php* se vloží následující kód:

```
<?php
$msg = '';
if ($_POST)
{
        if (isset($_POST['name']) && $_POST['name'] &&
                isset($_POST['email']) && $_POST['email'] &&
                isset($_POST['message']) && $_POST['message'] &&
                isset($_POST['year']) && $_POST['year'] == date('Y'))
        {
        }
        else
                $msg = "Formulář není vyplněný správně!";
}
?>
```

Uvedený kód je v podmínce, která kontroluje, zda se formulář odeslal a je něco v poli *$_POST*. Druhá podmínka kontroluje, zda byla odeslána jednotlivá pole formuláře a zda nejsou prázdná.

K odesílání e-mailu slouží funkce mb_internal_encoding("UTF-8");. Dále se ještě založí a naplní proměnné pro hlavičku (*$header*), adresu, kam se má e-mail odeslat (*$mail*) a předmět e-mailu (*$subject*). Pokud se odeslání podařilo, funkce mb_send_mail($mail, $subject, $_POST['message'], $header); vrátí *TRUE*, pokud odeslání selhalo, vrátí *FALSE*. Tato hodnota se pak uloží do proměnné *$success* a následně se podle ní nastaví proměnná *$msg*, která bude uchovávat hlášku o stavu formuláře.

```
<?php
mb_internal_encoding("UTF-8");
$msg = '';
if ($_POST) {
    if (isset($_POST['name']) && $_POST['name'] &&
            isset($_POST['email']) && $_POST['email'] &&
            isset($_POST['message']) && $_POST['message'] &&
            isset($_POST['year']) && $_POST['year'] == date('Y')) {
        $header = "From:" . $_POST['email'];
        $header .= "\nMIME-Version: 1.0\n";
        $header .= "Content-Type: text/html; charset=\"utf-8\"\n";
        $mail = "kl-accounting@seznam.cz";
        $subject = "Nová zpráva z kl-accounting.cz";
        $success = mb_send_mail($mail, $subject, $_POST['message'],
$header);
        if ($success) {
            $msg = "Email byl úspěšně odeslán!";
        } else
            $msg = "Email se nepodařilo odeslat. Zkontrolujte prosím
adresu.";
    } else
        $msg = "Formulář není vyplněný správně!";
}
?>
```

Nakonec se ještě deklaruje nad elementem *form* krátký blok PHP kódu, který vypíše obsah (pokud nějaký je) proměnné *$msg*.

```
<?php
    if ($msg) {
        echo('<p>' . $msg . '</p>');
    }
    ?>
```

3.6.3.6 PÍSMO A ODKAZY NA SOCIÁLNÍ SÍTĚ

Na stránce se změní ještě písmo a do patičky se přidají ikonky, které budou odkazovat na sociální sítě.

Pro přidání nového písma se přejde na stránku *google.com/fonts*, kde se vybere požadované písmo. Opět bude využito CDN a písmo se přes *link* element přidá na stránky.

```
<link
href='https://fonts.googleapis.com/css?family=Source+Sans+Pro:400,600,700'
rel='stylesheet' type='text/css'>
```

Protože je třeba písmo změnit na celé stránce, použije se v CSS hvězdičkový selektor, který vybere všechny elementy na stránce.

Písmo se nastaví přes vlastnost *font-family*, kde se jako první hodnota uvede požadované písmo a jako druhá hodnota písmo záložní, tzn. to, které se zobrazí v případě, když se první nepodaří z nějakého důvodu prohlížeči načíst. Jelikož je výchozí velikost nového písma jiná než písma původního, upraví se ještě hodnota vlastnosti *font-size*.

```
* {
    font-family: 'Source Sans Pro', sans-serif;
    font-size: 18px;
}
.fa-bars {
    font-size: 35px;
}
```

K nalezení potřebných ikonek se využije stránka *iconfinder.com*, kde se jednoduše ikonky podle názvu vyhledají, stáhnou a vloží do již vytvořené složky *img*.

Přejdu do souboru *footer.php* a pomocí elementu *img* se vloží ikonky na požadované místo na stránce - v tomto případě pod deklaraci copyrightu. Elementy *img* se „obalí“ elementem *a*, který bude odkazovat na příslušnou sociální síť.

```
<a href="https://www.facebook.com/klaccounting" target="_blank"><img
src="img/facebook-icon.png" alt="Facebook" width="64px" height="64px"></a>
<a href="https://twitter.com/kl_accounting" target="_blank"><img
src="img/twitter-icon.png" alt="Twitter" width="64px" height="64px"></a>
```

Elementu *a* byl ještě přidán atribut *target* s hodnotou *_blank*, který zajistí, že se odkaz otevře na nové stránce.

Ikonkám se pro lepší umístění na stránce nastaví v CSS ještě vlastnost *margin*.

```
footer img {
    margin: 5px 10px;
}
```

Pro lepší oddělení hlavičky od samotného obsahu stránku se přidají elementu *header* vlastnosti *padding-bottom* a *border-bottom*.

```
header {
    padding-bottom: 1.5em;
    border-bottom: 2px solid #f4f7f8;
}
```

3.6.4 MEDIA QUERIES

Media queries, neboli česky dotazy na média, jsou základem responzivního webdesignu. Vkládají se zpravidla na konec CSS souboru a jejich základní syntaxe vypadá následovně:

```
@media (podmínky) {
/* CSS kód, který se provede jen po splnění podmínek. */
}
```

Jelikož jsem si zvolil přístup *mobile first*, budu definovat podmínky typu – „*pokud bude šířka obrazovky dosahovat určité minimální hodnoty, proveď následující CSS příkazy*“.

Protože bude na stránce po zvětšení obrazovky mnohem více místa, je možno jednotlivým prvkům dát více „prostoru pro dýchání“ a také přidat nové.

Začne se s horní lištou, kde se v mobilní verzi nachází pouze telefonní číslo na společnost. Na lištu je vhodné přidat ještě e-mailovou adresu. Nejdříve se tak musí v HTML (*header.php*) vytvořit příslušný textový odkaz a v CSS se mu ve výchozím stavu nastaví hodnota *display* na *none*, aby na mobilních zařízeních zůstal skrytý.

```
<a class="mail" href="mailto:mailto:kl-accounting@seznam.cz"><span
class="fa fa-envelope fa-lg"></span>kl-accounting@seznam.cz</a>
```

Ikonka byla vložena opět pomocí Font Awesome a elementu *a* se přidala třída s názvem *mail,* aby bylo možné jednoduše tento element v CSS vybrat.

```
.mail {
    display: none;
}
```

Prvku je třeba přidat ještě padding a margin. Nejdříve se tak pouze rozšíří již stávající pravidlo o výběr požadované třídy.

```
.top-bar .fa-phone-square, .fa-envelope{
    padding-right: 5px;
    padding-left: 5px;
}
```

Tímto se jednoduše přidaly ikonce e-mailu stejné hodnoty *padding-right* a *padding-left*, jako má ikonka telefonu. Ikonce e-mailu se ještě samostatně přidá vlastnost *margin-left* s hodnotou *45px*.

```
.fa-envelope {
    margin-left: 45px;
}
```

Nyní už se dostávám k samotným media queries. Vytvoří se dotaz na média a jako podmínka se uvede minimální šířka s hodnotou *768px*, což je standartní šířka v rozlišení tabletů. Do vytvořeného bloku, který značí kód, jenž se má provést po splnění podmínek se napíše

vlastnost *display* s hodnotou *inline*, která zajistí, že prvek e-mailu v horní liště bude viditelný na zařízeních, které budou mít minimální šířku 768 px.

```
@media (min-width: 768px) {
    .mail {
        display: inline;
    }
}
```

Jako další bude potřeba, aby se navigace na zařízeních s touto minimální šířkou zobrazila na jednom řádku pod logem. Logo i navigace budou na stránce vycentrovány. Bude využit flexbox a tak se v souboru *header.php* vytvoří *div* element, který „obalí“ logo a navigaci.

```
<div class="header-wrapper">
                    <div class="main-wrapper">
                        <img src="img/logo-kl-acc.png" alt="logo" id="logo"
width="250" height="46">
                        <div class="toggle"><a href="#"><span class="fa fa-
bars"></span></a></div>
                    </div>
                    <nav>
                        <ul>
                            <li><a href="index.php">Úvodní stránka</a></li>
                            <li><a href="about.php">O nás</a></li>
                            <li><a href="pricelist.php">Ceník</a></li>
                            <li><a href="services.php">Služby</a></li>
                            <li><a href="contact.php">Kontakt</a></li>
                        </ul>
                    </nav>
                </div>
```

Nyní se přestoupí zpátky do CSS, do bloku pro podmínku, která již byla vytvořena, a vytvoří se zde několik řádků CSS kódu, které požadovanou funkcionalitu zajistí.

```
.toggle {
    display: none;
}

nav {
    display: block;
}

.main-wrapper {
    text-align: center;
}

.header-wrapper, nav ul {
    display: flex;
}

.header-wrapper {
    flex-direction: column;
    align-items: center;
}

nav ul li {
    margin: 0 1em;
    background-color: #fff;
}

nav a {
    color: #0fb3d0;
    font-weight: 600;
}

nav a:hover {
    color: #0a7b8f;
    border-bottom: 1px solid #0a7b8f;
}

.header-wrapper, main {
    width: 90%;
    margin: 0 auto;
}
```

Prvek pro zobrazení navigace na mobilních zařízeních (znak tří čar) byl skryt a naopak navigaci bylo určeno, že má být viditelná. Logo bylo vycentrováno pomocí výběru třídy *main-wrapper* a vlastnosti *text-align* s hodnotou *center*. U třídy *header-wrapper* a neseřazeného seznamu navigace se definovalo, že se jedná o flex kontejner. Dále bylo třídě *header-wrapper* určeno, že má elementy, jenž jsou jejími dětmi, zobrazit ve sloupci a vlastnost *align-items* s hodnotou center zajistí, že se tyto elementy vycentrují. Položkám neseřazeného seznamu byla nastavena velikost, barva a tučnost písma, barva pozadí a plocha za rámečkem pomocí vlastnosti *margin*. Byla také snaha dosáhnout toho, že když uživatel přejede myší přes položku v navigaci, tak se tato položka podtrhne a ztmaví. Toto řeší pseudo-třída *:hover*. Jako pseudo-třídy se označují takové třídy, které přidávají speciální vlastnosti na některé prvky. Jako poslední byla nastavena třídě *header-wrapper* a elementu

main šířka na stránce na hodnotu *90%* a pomocí vlastnosti *margin* byly tyto prvky vycentrovány.

U zařízení s šířkou obrazovky větší nebo rovnou 1085 px bude potřeba, aby byla navigace s logem na stejné úrovni (na jednom řádku), kde logo bude zarovnáno v levé části obrazovky a navigace v pravé.

Vytvoří se tak druhý dotaz na média, který bude mít, stejně jako první, jako podmínku vlastnost *min-width*, avšak tentokrát bude hodnota *1085px*. Třídě *header-wrapper* se změní zarovnání flex položek do řádku a vlastnost *justify-content* s hodnotou *space-between* zajistí požadované zarovnání prvků.

```
@media (min-width: 1085px) {
    .header-wrapper {
        flex-direction: row;
        justify-content: space-between;
    }

    .main-wrapper {
        text-align: left;
        max-width: 250px;
        margin: 0;
    }

    .header-wrapper, main {
        width: 85%;
        max-width: 1200px;
    }
}
```

Pro dotažení potřebného zarovnání se nastavilo ještě několik základních vlastností třídě *main-wrapper*. A u třídy *header-wrapper* a elementu *main* se opět o něco zúžila šířka zobrazené stránky a také byla nastavena její maximální hodnota na *1200px*.

Hlavička stránky je tak přizpůsobena pro všechna zařízení. Jako další se upraví patička, kde stačí určit, že na zařízeních s minimální šířkou 768 px se budou ikonky sociálních sítí zobrazovat menší. K dosažení této funkcionality se využije jQuery.

Vytvoří se funkce, která zajistí, že v případě, kdy bude hodnota vlastnosti *display* u třídy *mail* nastavena na *inline* budou mít ikonky velikost 36x36 px a v případě, kdy bude tato hodnota *none*, budou mít ikonky rozměry 64x64 px. Tato hodnota se nastaví pomocí media query, aby se měnila v závislosti na minimální šířce obrazovky.

```
function checkSize() {
    if ($(".mail").css("display") == "inline") {
        $("footer img").attr("width", "36px");
        $("footer img").attr("height", "36px");
    }
    else if ($(".mail").css("display") == "none") {
        $("footer img").attr("width", "64px");
        $("footer img").attr("height", "64px");
    }
}
```

Do bloku kódu `$(document).ready(function () { … });` se přidají další dva řádky.

```
    checkSize();
    $(window).resize(checkSize);
```

V prvním se zavolá již vytvořená funkce vždy, když se stránka načte, ve druhém se zavolá v případě, že se změní velikost okna.

Obě, hlavička i patička, jsou již responzivní a následně bude přizpůsoben obsah jednotlivých stránek.

Začne se u *index.php*, kde je potřeba, aby u dotazu na média s podmínkou *min-width: 1085px* bylo CSS pravidlo, které by změnilo zobrazení položek neseřazeného seznamu do jednoho řádku.

Již při vytváření tohoto seznamu se určilo, že se jedná o flex kontejner, takže nyní u něj stačí změnit hodnotu vlasnosti *flex-direction* na *row*.

```
main ul {
        flex-direction: row;
    }
```

Přejde se na stránku *about.php,* kde bude potřeba, aby se u stejného dotazu na média, jako v případě *index.php,* prvky zobrazovaly jinak. Při takovéto šířce je vhodnější umístit obrázek do levé části stránky a zbylý text do pravé.

Nejlepším způsobem zde bude opětovné využití flexbox. Bude tak potřeba flex kontejner a dvě položky v podobě obrázku a textu. Elementu *main* se přiřadí nová třída s názvem *about* a textové odstavce se „obalí“ do elementu *div* s třídou *about-text*.

Obsah souboru *about.php* tak vypadá následovně:

```
<?php include("inc/header.php"); ?>
<main class="about">
    <img src="img/ucetnictvi.jpg" alt="účetnictví" id="logo" width="585"
height="404">
    <div class="about-text">
    <p>
        Jsme společnost, která poskytuje účetní a daňové poradenství.
Nabízíme profesionální služby a specializujeme se na vedení daňové
evidence, účetnictví a veškeré poradenství v oblasti účetnictví, daní a
mezd.
    </p>
…
    </div>
</main>
<?php include("inc/footer.php"); ?>
```

Do CSS se k příslušnému dotazu na média přidají tyto dvě jednoduchá pravidla:

```
    .about {
        display: flex;
        flex-direction: row;
        margin-bottom: 18px;
    }

    .about img {
        margin-right: 20px;
    }
```

Ke stejnému dotazu na média se přidají i pravidla pro stránku *pricelist.php*. Zde je třeba, aby se tabulky zobrazily maximálně na dvou řádcích. Aby bylo možné požadované elementy v CSS zacílit, v *pricelist.php* se přiřadí elementu *main* třída a nazve se *pricelist*.

```
.pricelist ul {
        flex-wrap: wrap;
        justify-content: center;
    }

    .pricelist ul li {
        max-width: 390px;
    }
```

Vlastnost *flex-wrap* s hodnotou *wrap* určuje, že jsou flex položky zabalené dohromady. Aby některé tabulky nebyly na stránce příliš roztažené a byly hezky zarovnané, byla přidána flex položkám ještě vlastnost *max-width*.

Styly pro stránku *services.php* budou velmi podobné. Opět se vytvoří u elementu *main* třída, tentokrát s názvem *services*. Styly pro flex kontejner budou naprosto totožné jako v případě *pricelist.php*, takže stačí jen přidat k již vytvořenému pravidlu další selektor.

```
.pricelist ul, .services ul {
        flex-wrap: wrap;
        justify-content: center;
    }

.services ul li {
        width: 360px;
    }
```

Jako úplně poslední úprava na stránce bude u *contact.php*. Zde bude potřeba u dotazu na média s podmínkou *min-width: 1085px*, aby se kontaktní informace a mapa zobrazovaly v levé části a kontaktní formulář na stejném řádku v části pravé.

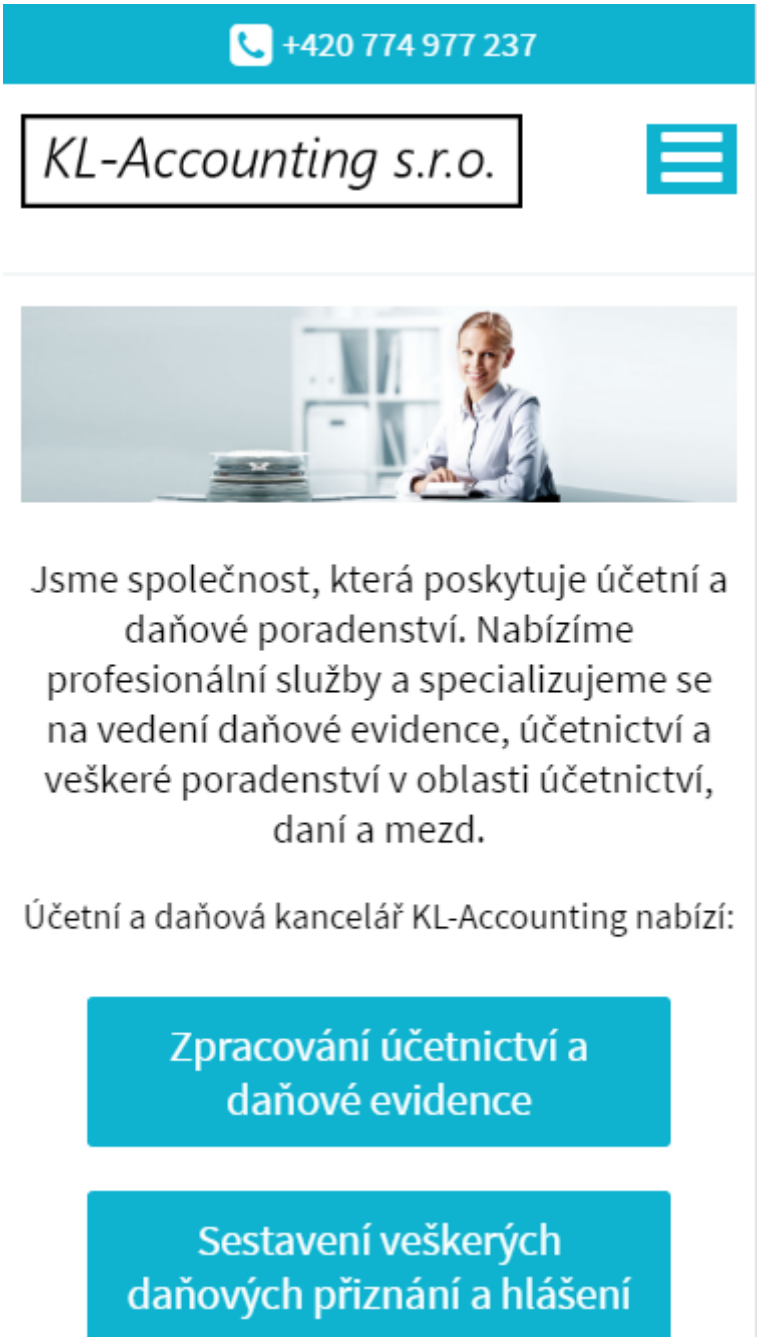
Asi nejlepším řešením je zde opět flexbox. V HTML se tak přidá třída *contact* k elementu *main* a „obalí“ se kontaktní informace a mapa dohromady pomocí elementu *div* s třídou *contact-text-map*.

```
<?php include("inc/header.php"); ?>
<main class="contact">
    <div class="contact-text-map">
    <p class="contact-text">
        <span class="bold">KL-Accounting s.r.o.</span><br>
        IČO: 01823086<br>
        DIČO: CZ01823086<br>
        Luková 64<br>
        Brodek u Přerova<br>
        E-mail: <a href="mailto:kl-accounting@seznam.cz">kl-
accounting@seznam.cz</a><br>
        tel.: <a href="tel:+420774977237">+420 774 977 237</a><br>
    </p>
    <iframe
src="https://www.google.com/maps/embed?pb=!1m14!1m8!1m3!1d5184.975598412112
!2d17.369048!3d49.475292!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x4713aceed6be59
41%3A0x21f03237e0c1092d!2zTHVrb3bDoSA2NCwgNzUxIDAzIEJyb2RlayB1IFDFmWVyb3ZhL
CDEjGVza8OhIHJlcHVibGlrYQ!5e0!3m2!1scs!2scz!4v1461769551828" width="400"
height="300" frameborder="0" style="border:0" allowfullscreen></iframe>
    </div>
    <?php
    if ($msg)
        echo('<p>' . $msg . '</p>');
    ?>
    <form method="POST">
        <h1>Kontaktní formulář</h1>
        <label for="name">Jméno:</label>
        <input type="text" id="name" name="name">
        <label for="email">E-mail:</label>
        <input type="email" id="email" name="email">
        <label for="message">Zpráva:</label>
        <textarea id="message" name="message"></textarea>
        <label for="year">Aktuální rok:</label>
        <input type="number" id="year" name="year">
        <input type="submit" value="Odeslat">
    </form>
</main>
<?php include("inc/footer.php"); ?>
```

V CSS se stejným způsobem jako v předchozích případech nastaví požadované vlastnosti. Jelikož jsou u flex kontejneru vlastnosti a jejich hodnoty naprosto totožné jako v případě třídy *about* a elementu *img* této třídy, pouze se k již vytvořeným pravidlům přidaly potřebné selektory.

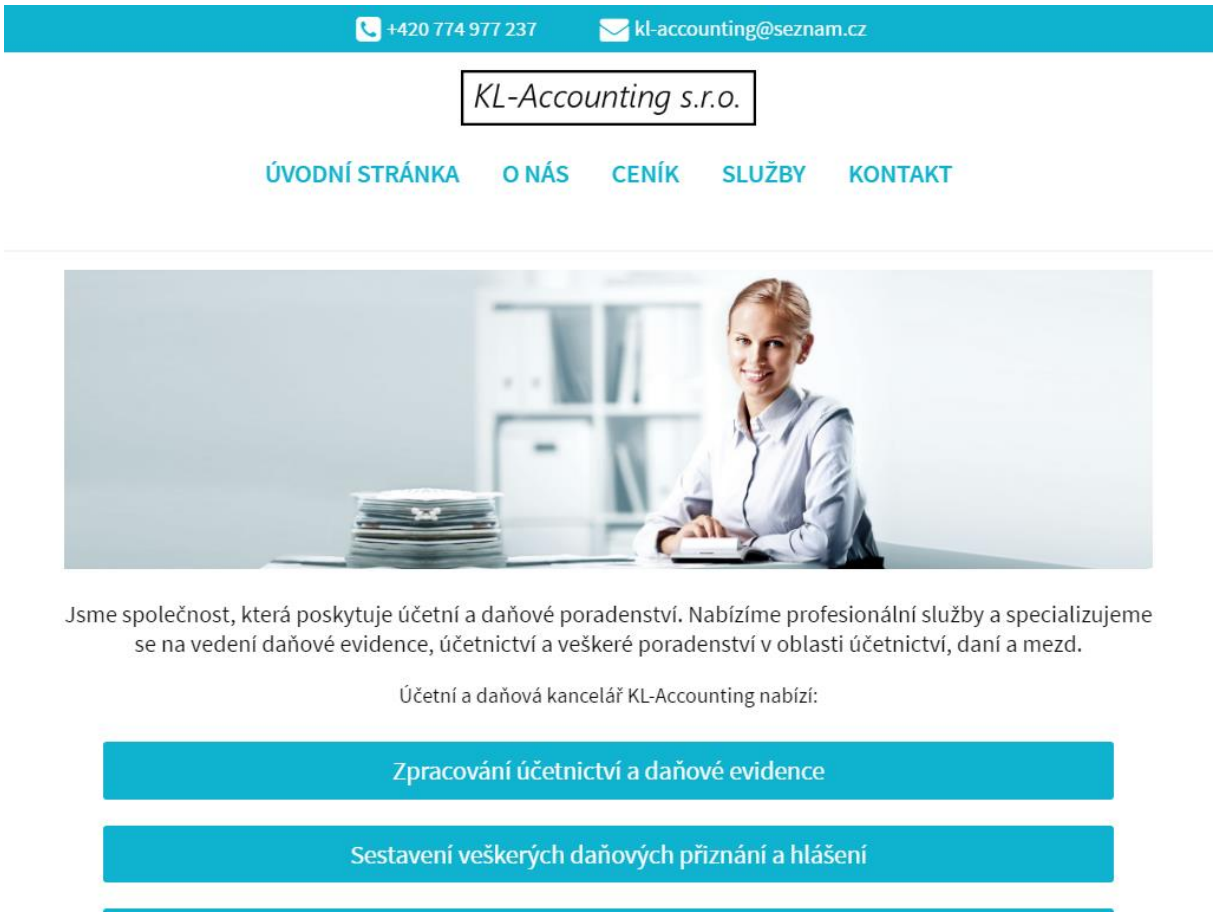
```
.about, .contact {
        display: flex;
        flex-direction: row;
        margin-bottom: 18px;
    }
.about img,.contact .contact-text-map  {
        margin-right: 20px;
    }
```

Porovnání vzhledu na různých zařízeních:



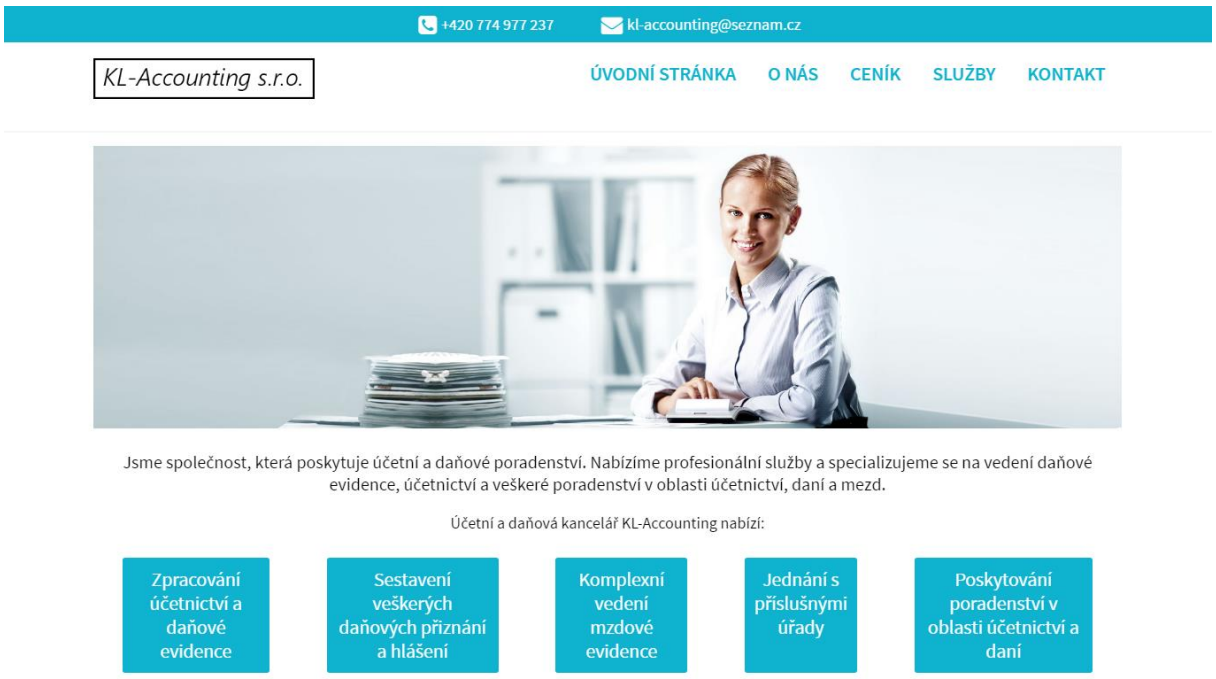
Obrázek 6:

Mobilní zařízení (Zdroj: Autor)



Obrázek 7:

Tablet (Zdroj: Autor)



Obrázek 8:

Desktop (Zdroj: Autor)

ZÁVĚR

Tato práce si v teoretické části stanovila cíl popsat základní syntaxi jazyků HTML5, CSS3, JavaScript a PHP, přiblížit jejich historii, poukázat na nové sémantické elementy specifikace HTML5 a představit nové klíčové prvky v PHP 7. V praktické části je zachycen postup tvorby webových stránek s využitím uvedených webových technologií.

Při tvorbě webových stránek pro společnost KL-Accounting s.r.o. jsem využil moderní postupy a poukázal jsem na vhodné možnosti využití vlastností CSS3 a prvků specifikace HTML5. Kde to bylo vhodné, tam jsem využil jazyků PHP a JavaScript pro docílení požadovaného výstupu. Výsledkem je vytvoření plně responzivních stránek s použitím současných postupů.

Trendem dnešní doby je vytvářet stránky, které jsou responzivní, přehledné, jednoduché a splňují zásady přístupnosti a použitelnosti tak, aby byly dostupné co nejširšímu okruhu uživatelů. Ukázat postup tvorby takovýchto stránek je hlavním cílem této práce.

SEZNAM LITERATURY

[1] MACDONALD, Matthew. *HTML5*. 2nd edition. Sebastopol, CA: O'Reilly & Associates, 2013. ISBN 9781449363260.

[2] KOSEK, Jiří. Historie a vývoj HTML. *Htmlguru.cz* [online], 2014 [cit. 2016-03-14]. Dostupné z: http://htmlguru.cz/uvod-historie.html

[3] HTML 3.2 Reference Specification. *W3C* [online]. c1997 [cit. 2016-03-14]. Dostupné z: https://www.w3.org/TR/REC-html32

[4] HTML 4.0 Specification. *W3C* [online]. 24. 04. 1998 [cit. 2016-03-14]. Dostupné z: https://www.w3.org/TR/1998/REC-html40-19980424/

[5] HTML 4.01 Specification. *W3C* [online]. 24. 12. 1999 [cit. 2016-03-14]. Dostupné z: https://www.w3.org/TR/html4/

[6] XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition). *W3C* [online]. 24. 01. 2000 [cit. 2016-03-14]. Dostupné z: https://www.w3.org/TR/xhtml1/

[7] HTML5. *W3C* [online]. 28. 10. 2014 [cit. 2016-03-14]. Dostupné z: https://www.w3.org/TR/html5/

[8] HTML: WHAT'S NEXT? *W3C* [online]. c2016 [cit. 2016-03-14]. Dostupné z: https://www.w3.org/blog/2016/03/html-whats-next//

[9] NEW SEMANTIC ELEMENTS. *Can I use.*. [online]. 12. 05. 2015 [cit. 2016-03-16]. Dostupné z: http://caniuse.com/#cats=HTML5

[10] CSS Syntax and Selectors. *W3schools* [online]. c1999-2016 [cit. 2016-03-16]. Dostupné z: http://www.w3schools.com/css/css_syntax.asp

[11] Cascading Style Sheets, level 1. *W3C* [online]. 17. 12. 1996 [cit. 2016-03-16]. Dostupné z: http://www.w3.org/TR/REC-CSS1/

[12] JANOVSKÝ, Dušan. Historie CSS. *Jakpsátweb.cz* [online]. [cit. 2016-03-16]. Dostupné z: http://www.jakpsatweb.cz/css/css-historie.html

[13] Cascading Style Sheets, level 2. *W3C* [online]. 1998 [cit. 2016-03-16]. Dostupné z: https://www.w3.org/TR/REC-CSS2/

[14] CSS CURRENT STATUS. *W3C* [online]. 2016 [cit. 2016-03-16]. Dostupné z: https://www.w3.org/standards/techs/css#w3c_all

[15] A Word About CSS4. *Xanthir.com* [online]. 19. 02. 2013 [cit. 2016-03-16]. Dostupné z: http://www.xanthir.com/b4Ko0

[16] JavaScript Syntax. *W3schools* [online]. c1996-2016 [cit. 2016-03-21]. Dostupné z: http://www.w3schools.com/js/js_syntax.asp

[17] JavaScript Basics. *Teamtreehouse.com* [online]. 2016 [cit. 2016-03-21]. Dostupné z: https://teamtreehouse.com/library/javascript-basics

[18] A Short History of JavaScript. *W3C* [online]. 2012 [cit. 2016-03-21]. Dostupné z: https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

[19] 1. díl - Úvod do JavaScriptu. *Itnetwork.cz* [online]. 2012 [cit. 2016-03-21]. Dostupné z: http://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk

[20] ECMAScript Documentation. *Ecmascript.org* [online]. 2016 [cit. 2016-03-21]. Dostupné z: http://www.ecmascript.org/docs.php

[21] History. *jQuery Foundation* [online]. c2016 [cit. 2016-03-26]. Dostupné z: https://jquery.org/history/

[22] NIXON, Robin. *Learning PHP, MySQL, JavaScript, CSS & HTML5*. Third edition. Sebastopol, CA: O'Reilly Media, Inc., 2014. ISBN 1491949465.

[23] Usage of JavaScript libraries for websites. *W3techs* [online]. c2009-2016 [cit. 2016-03-26]. Dostupné z: http://w3techs.com/technologies/overview/javascript_library/all

[24] PHP 5 Syntax. *W3schools* [online]. c1999-2016 [cit. 2016-03-28]. Dostupné z: http://www.w3schools.com/php/php_syntax.asp

[25] History of PHP. *Php.net* [online]. c2001-2016 [cit. 2016-03-30]. Dostupné z: http://php.net/manual/en/history.php.php

[26] News Archive - 2015. *Php.net* [online]. c2001-2016 [cit. 2016-04-03]. Dostupné z: http://php.net/archive/2015.php

[27] PHP 7: 10 Things You Need to Know. *HONGKIAT* [online]. c2009-2016 [cit. 2016-04-09]. Dostupné z: http://www.hongkiat.com/blog/php7/

[28] New features. *Php* [online]. c2001-2016 [cit. 2016-04-20]. Dostupné z: http://php.net/manual/en/migration70.new-features.php

[29] *Apache Friends* [online]. 2016 [cit. 2016-04-22]. Dostupné z: https://apachefriends.org/

[30] CSS Box Model. *W3schools* [online]. c1999-2016 [cit. 2016-04-27]. Dostupné z: http://www.w3schools.com/css/css_boxmodel.asp

[31] CSS3 Flexible Box. *W3schools* [online]. c1999-2016 [cit. 2016-04-27]. Dostupné z: http://www.w3schools.com/css/css3_flexbox.asp

SEZNAM OBRÁZKŮ A TABULEK

SEZNAM OBRÁZKŮ

SEZNAM TABULEK