Pedagogická fakulta
Faculty of Education

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

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta

Katedra informatiky

Bakalářská práce

Frontend responzivního webu s využitím HTML5 frameworků a buildovacích nástrojů

Frontend responsive website using HTML5 frameworks and task runners

Vypracoval: Jakub Tetík
Vedoucí práce: PaedDr. Petr Pexa, Ph.D.
České Budějovice 2016

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

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jakub TETÍK, DiS.**
Osobní číslo:	**P13113**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**Frontend responzivního webu s využitím HTML5 frameworků a buildovacích nástrojů**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Cílem bakalářské práce je zpracovat problematiku vývoje frontendu responzivních webových stránek s využitím HTML5 frameworků (Bootstrap, Foundation, HTML5 Boilerplate aj.) a buildovacích nástrojů (Grunt, Gulp aj). Zvolené nástroje časově usnadňují vývojářům práci, automatizují opakující se úlohy (kompilace CSS preprocesorů, spojování a minifikace souborů, generování dokumentace) a mají mnoho dalších výhod. HTML5 frameworky také obsahují šablony kódu, které vývojář již nemusí vymýšlet, šablony implementuje, případně modifikuje dle potřeby do projektu. V rámci práce bude provedena komparace s vývojem responzivních webových stránek bez pomoci těchto nástrojů a frameworků. Výstupem práce bude kompletní webová prezentace, na které budou vidět rozdíly jednotlivých frameworků a nástrojů, bude srovnána časová náročnost vývoje, výsledná datová velikost a obtížnost pro vývojáře z hlediska znalosti syntaxe.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **KADLEC, Tim. Responzivní design. Brno: Zoner Press, 2014. ISBN 978-80-7413-280-3.**
2. **SHARKIE, Craig, Andrew FISHER a Ondřej BAŠE. Responzivní webdesign okamžitě. Brno: Computer Press, 2015. ISBN 9780987332165.**
3. **GRUNT. Grunt: The JavaScript Task Runner [online]. 2012 [cit. 2015-04-16]. Dostupné z: http://gruntjs.com/**
4. **GULP. Gulp.js - the streaming build system [online]. 2013 [cit. 2015-04-16]. Dostupné z: http://gulpjs.com/**
5. **BOOTSTRAP. Bootstrap The world's most popular mobile-first and responsive front-end framework. [online]. 2010 [cit. 2015-04-16]. Dostupné z: http://getbootstrap.com/**
6. **ZURB. Foundation — The Most Advanced Responsive Front-end Framework from ZURB [online]. 1998-2015 [cit. 2015-04-16]. Dostupné z: http://foundation.zurb.com/**
7. **MARCOTTE, Ethan. RESPONSIVE WEB DESIGN. 2011. ISBN 978-0-9844425-7-7. Dostupné z: http://www.petronet.ir/documents/10180/2323248/Responsive_Web_Design**

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

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

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

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

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

V Českých Budějovicích dne 27. dubna 2015

Prohlášení

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

V Českých Budějovicích dne 24. června 2016.

Podpis studenta

Abstrakt

Cílem bakalářské práce je zpracovat problematiku vývoje frontendu responzivních webových stránek s využitím HTML5 frameworků (Bootstrap, Foundation, HTML5 Boilerplate aj.) a buildovacích nástrojů (Grunt, Gulp aj.). Tyto nástroje usnadňují vývojářům práci, automatizují opakující se úlohy (kompilace CSS preprocesorů, spojování a minifikace souborů, generování dokumentace) a mají mnoho dalších výhod. HTML5 frameworky obsahují šablony kódu, které vývojář již nemusí vymýšlet, šablony implementuje, případně modifikuje dle potřeby do projektu. V rámci práce bude provedena komparace s vývojem responzivních stránek bez pomoci těchto nástrojů a frameworků.

Výstupem práce bude kompletní webová prezentace, na které budou vidět rozdíly jednotlivých frameworků a nástrojů, bude srovnána časová náročnost vývoje, výsledná datová velikost a obtížnost pro vývojáře z hlediska znalosti syntaxe.

Klíčová slova

Frontend, responzivní, framework, Grunt, Gulp, Bootstrap, Foundation

Abstract

The aim of this thesis is to handle issues of development of the frontend of responsive websites using HTML5 frameworks, including Bootstrap Foundation, HTML5 Boilerplate and others. Furthermore, also the help of task runners Grunt and Gulp is used in this work. As part of the work is performed comparison with the development of responsive web pages without using these tools and frameworks mentioned above.

Grunt, Gulp and others of the tool range make it easier for developers to save working time. They automate repetitive tasks, such as compilation of CSS preprocessors, file linking and minification, documentation generation, and many other tasks. HTML5 frameworks are templates of the codes that the developer no longer has to invent. These templates are implemented or modified as needed in the project.

The outcome of the work is a complete web presentation, which shows the differences between various frameworks and tools. These differences are used for the comparison of time-consuming development, data size and difficulty for developers in terms of the knowledge of syntax.

Keywords

Frontend development, framework, Grunt, Gulp, Bootstrap, Foundation

Poděkování

Rád bych poděkoval PaedDr. Petru Pexovi, Ph.D., za odborné vedení při zpracování mé bakalářské práce a jeho cenné rady. Dále bych rád poděkoval své rodině, která mi umožnila studium na vysoké škole.

Obsah

1 Úvod

Responzivní design je v současnosti stále více rozšířenou technikou využívanou k tvorbě webových stránek a aplikací. Počet přístupů na internet z mobilních zařízení (smartphony[1], tablety) rapidně roste. Tento fakt je třeba respektovat a jako frontend designér[2] jej akceptovat. V současné době si bohužel nevystačíme se znalostí značkovacího jazyka HTML a CSS stylů pro dosažení co nejefektivnějšího postupu při tvorbě webového projektu. Konkurence na trhu v tomto oboru je vysoká, proto je třeba hledat způsoby jak čas strávený nad prací co nejvíce minimalizovat a to bez negativního dopadu na výsledný projekt.

Jednou z mnoha možností je využití HTML5 responzivních frameworků (frontend frameworků). Jedná se o kompletní řešení v podobě předdefinovaných bloků kódu (HTML, CSS, Javascript a podobně), které typicky obsahují: CSS třídy pro zarovnání bloků, textu, kontextuální třídy, responzivní mřížku (grid), UI komponenty[3] a podobně. Důležitým faktem je, že se veškeré uvedené části chovají responzivně a některé mají dynamickou povahu díky Javascriptovým pluginům. Není třeba vymýšlet zdrojový kód a psát jej od začátku, ale stačí implementovat funkční řešení. Mezi jedny z nejznámějších frameworků patří Twitter Bootstrap a Foundation. Framework je pojem zavádějící – pokud podporuje snadnější tvorbu, lze mluvit o frameworku. Někdy je lze použít jako finální řešení – v tomto případě se jedná o knihovnu. Fluidní mřížky, CSS styly pro formulářové elementy a podobně se již objevily v CSS frameworku jakým je například Blueprint. Z něj se pak frontendový stává přidáním výše zmíněných UI komponent a Javascriptu. [1]

Další možností je použití programů umožňujících kompilace preprocesorového kódu, minifikace CSS nebo JS souborů a mnoho dalších užitečných funkcí. Ty mohou být využívány paralelně (pro jednotlivé operace zvláštní program) což je nepraktické. V současné době jsou využívány v praxi task runnery (buildovací nástroje) zastupující funkcionalitou výše zmíněných programů – jeden pro všechny operace. Jedná se automatizační nástroje obsahující Javascriptové soubory, v kterých si definujeme vlastní funkce nebo využíváme již hotové naprogramované

[1] Smartphone – chytrý telefon vybavený OS, jehož prostřednictvím lze instalovat, upravovat programy (aplikace)
[2] Frontend designér – pracuje s HTML, CSS, Javasscript, nejedná se o grafického designéra
[3] User Interface komponenty – komponenty uživ. prostředí - tlačítka, seznamy atd.

či nakonfigurované, jejichž zdrojový kód lze stáhnout zdarma. Zajišťují automatické spouštění výše uvedených akcí. Nebudeme je muset neustále spouštět příkazy či jiným způsobem (kliknutí tlačítka v GUI prostředí po každé úpravě kódu). Mezi nejznámější z nich patří například Grunt a Gulp. [16]

Obecně řečeno se jedná tedy o automatizační nástroje, které mají na starost opakující se práci za frontend designéra. Výsledkem je více času na vývoj.

2 Cíle práce

Cílem práce je kompletní rozbor a představení HTML5 frameworků (především Bootstrap a Foundation) a buildovacích nástrojů (Grunt, Gulp). Dále rozbor jejich výhod a nevýhod, seznámení se s jejich instalací a s následným používáním. Hlavním cílem je dokázat, že vývoj webových aplikací či webových stránek je rychlejší, pokud používáme frameworky případně v kooperaci s task runnery. Ať už se jedná o tvorbu webových prototypů či kompletních projektů. Výsledky jsou prezentovány formou tabulek v jednom z výstupů mé bakalářské práce. Komparace je provedena z několika hledisek.

Výstupem práce je webová prezentace vytvořená pomocí Bootstrapu, kde jsou taktéž prezentovány výsledky jednotlivých frameworků a nástrojů, je srovnána časová náročnost vývoje, obtížnost pro vývojáře z hlediska znalosti syntaxe a další vlastnosti.

Dalším výstupem je webová prezentace napsaná pomocí frameworku Foundation. Tato prezentace bude zároveň sloužit jako průvodce studentům nastupujících do prvních ročníků Pedagogické fakulty Jihočeské univerzity.

3 Východiska práce

V současné době se HTML5 frameworky stávají standardem, které by frontend designér měl znát a měl by je umět používat. Obsahují předdefinované šablony CSS stylů, Javascriptu a HTML kódu, které jsou otestované napříč prohlížeči. Tyto šablony stačí pak pouze implementovat, případně je modifikovat. Můžeme je považovat za blackboxy.

Čas při vývoji projektu je faktor, který se podílí na výsledné ceně. Díky frameworkům Bootstrapu či Foundation můžeme tento faktor obrátit v náš prospěch, mohou se snížit časové náklady vložené do projektu. Navíc mají integrovanou podporu CSS preprocesorů, což nám umožňuje například vytváření proměnných a jejich opakované použití. Lze vytvářet například tzv. mikrobreakpointy místo breakpointů[4] díky zanořování CSS v preprocesorech, což je v případě potřeby responzivního designu užitečná věc.

Buildovací nástroje neboli task runnery automatizují různé opakující se úkoly. Může se jednat o minifikaci souborů, kterou zvládají lépe než preprocesory, odstranění nadbytečného CSS kódu, který není aktuálně využíván. Obzvláště pokud je používán Bootstrap a to špatným způsobem – import nevyužitých CSS stylů. Dále pak kompilaci preprocesorů, spojování JavaScriptových souborů, minifikace obrázků, převody SVG do PNG. Pomáhají také při generování náhledů obrázků. Generují dokumentace a mnoho dalších úloh.

Typickým příkladem pro využití může být vývoj webové prezentace pro firmu, která se nechce omezovat jen na desktopová zařízení. Ze statistik má získané informace ohledně přístupů na jejich webové stránky z mobilních zařízení. Pokud by se měl ladit takový web do výsledné podoby responzivního designu s využitím samotného CSS kódu bez frameworků a jiných nástrojů, čas vývoje by rapidně vzrostl.

Na internetu se objevuje čím dál tím více projektů založených na těchto kombinacích (framework / task runner). Navíc s frameworkem Bootstrap úzce spolupracuje redakční systém Wordpress, jenž je velice oblíbený. V současné době se také objevuje čím dál tím více pracovních nabídek, kde jsou právě tyto předpoklady.

[4] Zlomový bod kdy dochází k aplikaci jiných CSS stylů pomocí @media queries.

4 Metody práce

V teoretické části popíši problematiku responzivního webu. Uvedu zde postupy, které by se měly dodržovat. Naproti tomu zde zmíním chyby, kterých by se vývojář při vývoji měl vyvarovat. Představím základní pilíře responzivního designu, rozeberu problematiku HTML5 frameworků – o co se jedná a proč je dobré je využívat. Dále představím relativní novinku v podobě buildovacích nástrojů – Grunt, Gulp. Zmíním se také o dalších jim podobných nástrojích.

Praktická část se bude sestávat ze dvou webových projektů. Prvním bude responzivní webová prezentace, která bude představovat online manuál, kde bude vysvětlen postup instalace a návod na používání frameworků a task runnerů. V dílčích částech HTML a CSS kódu porovnám rozdíly mezi použitím výše zmíněných nástrojů a vývojem s použitím běžného CSS kódu a bez použití nástroje Gulp a Grunt. Budu zde porovnávat časové rozdíly, náročnost z hlediska znalosti syntaxe, datovou velikost výsledných souborů a rychlost načítání.

Druhým výstupem bude responzivní web sloužící jako průvodce pro studenty prvních ročníků Pedagogické fakulty. Budou se zde nacházet užitečné informace pro nové studenty, jako jsou například šablony formálních dopisů vedoucím kateder, vedoucím fakult a podobně, informace o přestupech mezi fakultami, přerušení, prodlužování a ukončení studia. V první fázi provedu sběr dat. Z tohoto výstupu pak přesně stanovím hlavní body obsahu tohoto webu. Důležitým parametrem pro kvalitní výstup jsou studenti vyšších ročníků, kteří si zpětně uvědomují, jaké informace postrádali v případě jejich nástupu do studia.

5 Responzivní design

V minulosti byly webové stránky navrhovány tak, aby se jejich šířka přizpůsobovala velikosti okna prohlížeče. V této době nebyla tato technika tak propracovaná a weby se začaly psát v šířkách fixních (pevných). Tyto weby byly navrhovány technikou: pixel-perfect. Jedná se o dodržování sebemenších detailů, kdy kodér převede grafický design a dodrží jednotlivé mezery, odsazení a zarovnání. Obrázky se nerozpadají při změně rozlišení a podobně. Oblíbenou se stala šířka 960 px, případně její odvozené alternativy 940 px nebo 980 px. Byla totiž ideálním rozvržením pro rozlišení 1024 x 768 px. S příchodem širokoúhlých obrazovek a naopak menších displejů (tablety, smartphony) bylo potřeba vymyslet jiný postup. Na monitorech s vysokým rozlišením vznikalo mnoho nevyužitého místa po stranách webu, který měl fixní šířku 960 px. Na malých displejích se zobrazoval horizontální posuvník, což bylo nepraktické, pokud si musel návštěvník neustále posunovat obsah webové stránky.

Responzivní design je schopný adaptivně se zobrazovat na jakémkoliv zařízení. Přizpůsobí se velikosti jeho obrazovky. Poprvé se tento pojem objevil v článku, který napsal Ethan Marcotte, jenž je považován za otce tohoto designu. Definoval jej ve svém článku na blogu A List Apart takto: „*za pomoci fluidního layoutu, fluidních obrázků a media queries vyrobíme webovou stránku, která je schopná přizpůsobit se všem možným velikostem obrazovky*.“ Jiná definice jej představuje jako stylování dokumentu prostřednictvím CSS stylů a s pomocí @media-queries je možné detekovat zařízení s následnou aplikací správných stylů. Je jedno, zda se jedná o starší zařízení s nízkým rozlišením – Samsung Galaxy S2 (480 x 800 px) nebo novější zařízení – Samsung Galaxy S6 (2560 × 1440 px). Pokud správně nadefinujeme potřebné náležitosti layoutu stránky, přizpůsobí se nám správně web či aplikace nejen na výše zmíněných dvou zařízeních, ale také na mnoha ostatních. Pro kvalitní výstup responzivního projektu je třeba dodržet základních pilířů a dalších technik, které jsou popsány v samostatných kapitolách této práce.[1]

5.1 Základní pilíře responzivního designu

Responzivní design je postaven na třech respektive čtyřech základních pilířích, díky jejichž kombinaci lze dosáhnout téměř stoprocentního požadovaného řešení. Téměř proto, že se vždy objeví nějaké zařízení nebo typ obrazovky / displeje, na kterém

Javascript, CSS a podobně nebude funkční, a tak se hledá alternativní postup pro daný typ. Výše zmíněnými pilíři jsou:

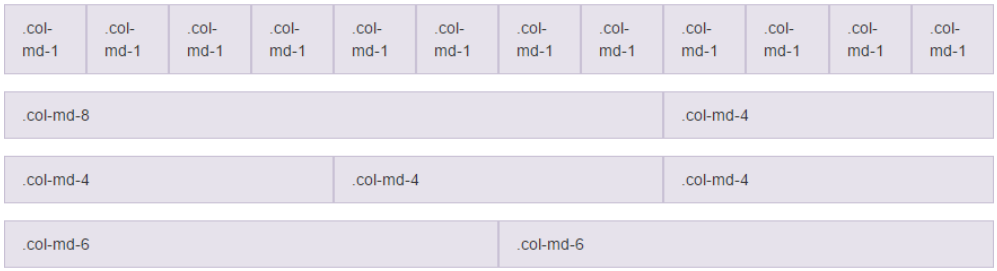
- Fluidní / flexibilní mřížka – grid,
- media queries,
- responzivní obrázky,
- responzivní obsah.

Jednotlivým pilířům jsou věnované samostatné kapitoly.

5.1.1 Fluidní mřížky – grid

Fluidní mřížky jsou základem layoutu[5] webové stránky. Jsou rozděleny na sloupce a řádky. Do sloupců lze řadit jednotlivé sekce webu. Klasickým případem je tří sloupcový layout. Každý ze sloupců bude mít proměnlivou šířku měnící se v závislosti na velikosti okna prohlížeče. Pro úplnost představím kompletní přehled layoutů:

1. Layout s pevnou šířkou (fixed – width),
2. plovoucí layout (fluid, liquid),
3. elastický layout. (elastic),
4. hybridní layout (hybrid). [1, 2]



Obrázek 1: Grid.

5.1.1.1 Layout s pevnou šířkou

Tento typ layoutu má pevně danou šířku v pixelech. Její doporučená hodnota je 960 px. Toto je optimální šířka z důvodu dělitelnosti beze zbytku. V době, kdy narůstal počet oblíbenosti pro rozlišení 1024 px, byla optimální šířka mezi 964 px až 984 px pro experimentování s layoutem vůči okrajům prohlížeče. Číslo 960 je však dělitelné beze zbytku čísly: 3, 4, 5, 6, 8, 10, 12, 15, 16, což je praktické pro různé varianty s různým počtem sloupců. V důsledku toho se šířka 960 px velice rychle prosadila. Tyto layouty jsou na webu v současnosti stále běžnou

[5] Layout – grafické rozvržení webové stránky

implementací. K ukázce jsem vybral webové stránky obce Střezimíř. Hlavní stránka na následujících obrázcích má fixní šířku a neexistuje zde varianta s více soubory CSS pro implementaci na různá zařízení. [1]

Obrázek 2: Webová stránka - fixní mřížka.

Tyto layouty se potýkají s několika problémy:

- Když volíme pevnou šířku layoutu, měli bychom zjistit ze statistik, jaké rozlišení se nejvíce využívá pro přístup na web.
- Problémy na desktopových zařízeních:
    - Uživatel může mít své oblíbené rozlišení pro přístup na web.
    - Nainstalovaný plugin – postranní panel v prohlížeči, který ubere pixely.
    - Není maximalizované okno prohlížeče.
    - Zmenšené okno pod hranici 960 px – zobrazí se nevhodný posuvník, narušující design.

Zobrazení ve full HD – kolem stránky zbyde nevyužitý prostor.[1, 2]

5.1.1.2 Plovoucí layouty (fluid / liquid)

Tento typ layoutu využívá místo pixelů procenta, která jsou flexibilnější jednotkou. Na obrázku č. 3 je uveden příklad kde má stránka následující rozměry:

- Levý sloupec: 60 %.
- Pravý sloupec: 30 %.
- Mezera mezi sloupci: 10%.

Obrázek 3: Webová stránka načtená v různých zařízeních – fluidní layout.

5.1.1.3 Elastické layouty

Tento typ layoutu je podobný fluidním layoutům. Velikosti se určují pomocí jednotky em. Ta má definovanou velikost z nadřazeného stylu (font-size). Pokud bude základní velikost písma nastavena na 16 px, potom 1 em = 16 px. Tento typ layoutu disponuje dobrou kontrolou nad typografií. Pokud nadefinujeme DIVU šířku 60 em, bude se adaptovat v závislosti na velikosti fontu.

Příklad:

- velikost fontu – 16 px,
- hlavní obsah 60 x 16 px = 960 px.

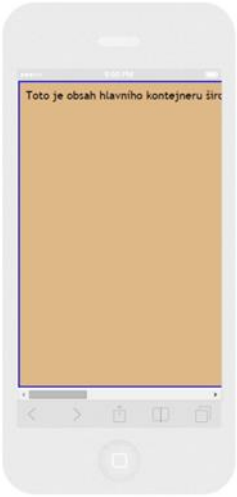
Musíme dát však pozor na správné řazení elementů kvůli dědičnosti v kaskádových stylech. [1]

5.1.1.4 Hybridní layouty

Využívají kombinace procentuálních jednotek a pixelů. Jedná se o kombinace předchozích layoutů. Levý sloupec má šířku 70 %, pravý sloupec 300 px. [2]

Obrázek 4: Hybridní layout.

Pokud šířka okna prohlížeče bude menší než 1000 px, pravý sloupec přesáhne 30 % z celkové šířky a zobrazí se horizontální posuvník jak je vidět na obrázku č. 5.



Obrázek 5: Detail hybridního layoutu na telefonu.

5.1.1.5 Velikost písma – jednotky fontu

Pokud plánujeme tvorbu responzivního designu, musíme uvažovat také o změnách velikosti písma (fontu) – musí být flexibilní. Jako přednostní jednotky bychom měli používat pixely, procenta, jednotky EM nebo REM.

- Pixely – tato jednotka byla dříve dominantní variantou (dnes stále hojně využívána) díky skvělé kontrole nad typografií. Každý prohlížeč zobrazuje velikost textu stejně, pokud je nastavená v této jednotce. Nevýhodou je absence dědičnosti (pokud budeme měnit velikost fontu, musíme ji měnit v jednotlivých elementech).
- Em – jedná se o flexibilnější jednotku. Využívá dědičnosti. Pokud změníme velikost u nadřazeného elementu, změní se velikost u všech dceřiných

(vnořených) elementů využívajících této jednotky. Musíme si dát pozor v některých případech na dědičnost samotnou. [1]

Příklad:

Nadpis H1 se odvíjí od velikosti písma nastavené v elementu body:

```
body{font-size: 16px;}
#inner{font-size: 2em;}
span{font-size:0.5em; color:blue;/* 16px*/}
<div id="main">
      <div id="inner">toto je nějaký text
            <span>uvnitř řádkového elementu SPAN</span>
      </div></div
```

Vysvětlení příkladu:

2 x 16 px = 32 px (velikost fontu v DIVU #inner), pomocí dědičnosti přebírá jako hlavní kontext právě tuto velikost jako výchozí. Pro docílení výchozí velikosti musíme zpětně nastavit násobek elementu na hodnotu: 0,5 em (0,5 x 32 = 16).

- Procenta – vlastnosti mají podobné jako jednotka em. Využívají dědičnosti. Pokud je základních 16 px = 100 %, pak 200 % se pak rovná 32 pixelům.
- REM („root em“) – princip je podobný jako u jednotky em s tím rozdílem, že nedědí z jeho rodičovského elementu, ale vztahuje se k velikosti fontu hlavního elementu <html>. Tuto jednotku nepodporuje prohlížeč IE 8. Pokud není podporována jednotka REM, automaticky se počítá v pixelech. [2, 3]

5.1.1.6 Výchozí velikost fontu

Výchozí velikost desktopových prohlížečů je 16 px. Výjimkou jsou některé typy zařízení, například: Blackberry s OS BlackBerry 6.0, kde je výchozí velikost 22 px nebo Kindle Touch s výchozí velikostí 26 px. Důvodem je větší hustota rozlišení – font o velikosti 16 px by byl malý. Zařízení odešle fiktivní data o rozlišení prohlížeči a ten pak zobrazí výsledek v rozlišení, které obdržel. Například iPhone4 má rozlišení 640 x 960, prohlížeči však oznámí rozlišení 320 x 480 (hustota pixelů – popsáno v další kapitole). V praxi je nejdůležitější ve finále čitelnost fontu. Velikost 16 pixelů nemusí být stejná ve všech prohlížečích. Doporučuje se tedy nastavit v elementu <body> velikost fontu na 100%. [2]

5.1.1.7 Přepočet pixelů

Pokud potřebujeme předělat layout, který je postaven na pixelech tak aby používal velikosti v procentech, použijeme k výpočtu jednoduchý vzorec:

CÍL / KONTEXT = VÝSLEDEK

Příklad:

DIV #wrapper má šířku 960px.

DIV #main má velikost 620px.

```
<div id="wrapper">
      <div id="main">
      Nějaký text
</div></div>
```

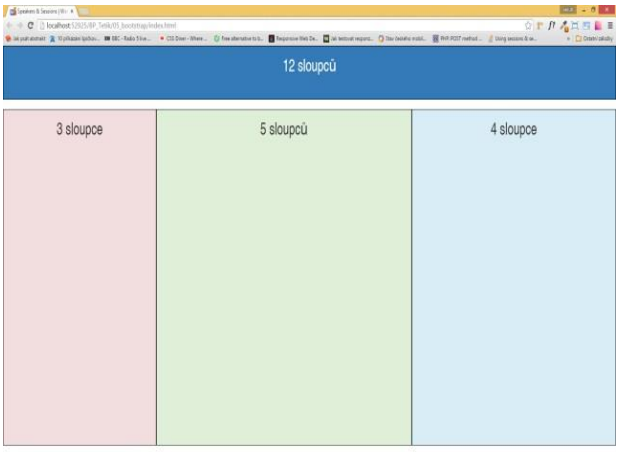
Pokud bychom chtěli velikost DIVU #main převést na procenta provedeme to výše uvedeným vzorcem:

620 / 960 = 0,645833333 x 100 % = 64,5833333 %

Velikosti se nezaokrouhlují na desetiny ani na setiny. Používáme pokud možno přesné výsledky. Stejný postup aplikujeme na výpočet jednotek em, rem.

5.1.1.8 Role fluidních mřížek

Mřížky představují na webu způsob, jak zacházet s volným místem pro rozvržení layoutu. Obsah rozdělený do dvou nebo do tří sloupců lze většinou umístit do jednoho řádku. Pokud máme větší počet sloupců, obsah je rozdělen do více řádků.



Obrázek 6: Třísloupcový layout.

Obrázek 7: Vícesloupcový layout.

Fluidní mřížky mají tu schopnost, že dokáží intuitivně měnit svou velikost podle velikosti viewportu[6]. Skládají se ze sloupců, řádků a mezer mezi jednotlivými sloupci případně řádky. Šířky těchto sloupců bývají relativní k velikostem jejich nadřazených elementů. Ty mohou mít fixní nebo fluidní šířku ale elementy v nich obsažené se díky dědičností velikostně přizpůsobují. Z toho důvodu mohou být označovány jako fluidní.[2]

5.1.1.9 Fluidní typografie

Cílem je nastavit veškeré aspekty písma v relativních jednotkách. Mezi relativní jednotky patří: em, ex, rem a procenta. Díky používání těchto jednotek se typografické prvky dokáží přizpůsobovat dle zařízení, velikosti viewportu a podobně. Návštěvníci webu si mohou nastavit základní velikost písma, od toho se pak odvíjí naše nastavení. Nejedná se jen o CSS vlastnost: font-size. Relativní jednotky se pak musejí používat u vlastností: margin, padding, border a podobně. Díky relativním jednotkám umožní kodér měnit uživateli velikost písma prohlížeče.

Příklad – nevhodné řešení:

```
html{
font-size:16px; /** výchozí velikost moderních prohlížečů **/
}
H1{
font-size:24px; /** nadpis první úrovně: 24 px.**/
}
```

Příklad – vhodné řešení:

```
html{
font-size:1em; /** defaultně bude 16px **/
}
```

[6]Viditelná velikost okna prohlížeče – průhled okna prohlížeče.

```
H1{
font-size:1.5em; /** v poměru ku 16px**/
}
```

Jedním ze způsobů, jak pracovat pohodlně s velikostí fontu, je nastavit elementu <body> velikost písma na 10 px = 62.5 % z 16 px. Potom se jednoduše pracuje s násobky při použití například em jednotek:

- 1 em = 10 px,
- 1.2 em = 12 px,
- 1.6 em = 16 px.

5.1.2 Adaptivní obrázky

Umožňují variabilní přizpůsobení obrázku vzhledem k velikosti viewportu. Reagují intuitivně na změnu velikosti rozlišení. Přizpůsobením je myšleno načtení obrázku s jiným kontextovým obsahem nebo stejným kontextovým obsahem ale s jinou datovou velikostí (jiným rozlišením), případně stejný obrázek s adaptivní velikostí. Existuje zde několik problémů – fluidního textu lze dosáhnout mnohem snadněji než fluidních obrázků. Další problém jsou dnes displeje s vyšším poměrem rozlišení – 1.5 či 2 násobné rozlišení, kterými disponují Retina displeje od Applu či Amoled od Androidu. Zde musí být specifikovány detekce rozlišení. Neexistuje jednoznačné řešení, ale několik doporučených technik: CSS styly, Javascript nebo jejich kombinace.

Obrázky lze do prohlížeče ze serveru stahovat podle určitých aspektů:

- Výběr varianty obrázku podle velikosti okna prohlížeče.
- Výběr varianty podle velikosti obrázku v rámci layoutu stránky. V některých situacích totiž můžeme na větším displeji potřebovat menší obrázky.
- Výběr podle device-pixel-ratio, neboli poměru mezi hardwarovým a CSS rozlišením.
- Výběr podle art-direction, neboli podle režie. Jedná se o způsob, kdy na smartphonu potřebujeme oříznout obrázek jinak než na desktopu. [1, 3]

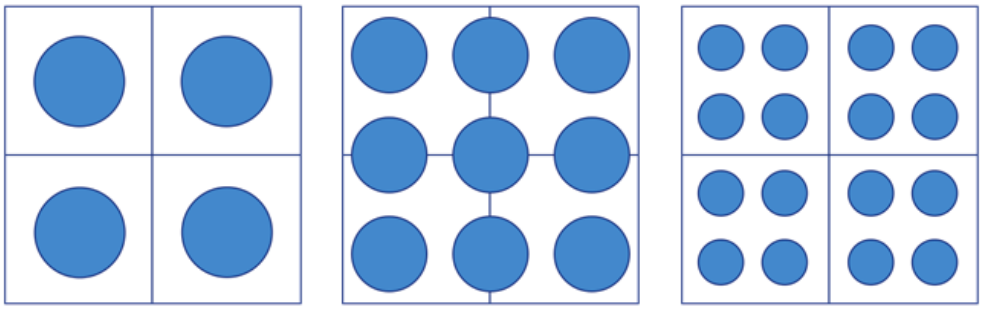
5.1.2.1 CSS pixel a fyzický pixel

Důležité je brát v potaz, že existují dva druhy pixelů. Se vzrůstajícím rozlišením se zvětšuje takzvaný poměr fyzických a CSS pixelů a udává se následujícím vzorcem:

fyzický pixel / CSS pixel = 4/2 = 2

Poměr pixelů je vlastnost, která umožní popsat rozlišení obrazovky v počtu bodů na pixel. CSS pixely jsou odlišné od fyzických tím, že nejsou závislé na hustotě rozlišení. Jedná se o DIP pixely (density independent pixels) – odpovídají fyzickým bodům na palec DPI. Příkladem je iPhone 5S s rozlišením 640 x 1136 pixelů. Prohlížeč zařízení obdrží výsledné rozlišení: 320 x 568 pixelů. Obrázek č. 7 ukazuje jednotlivé poměry mezi CSS a hardwarovým rozlišením:

- 1 x 1,
- 1,5 x 1,
- 2 x 1.



Obrázek 8: Poměry hustoty pixelů.

Fyzických pixelů při poměru 2 x 1 je dvakrát více než pixelů CSS. Například IPhone 5S s rozlišením rozlišení 640 x 1136 px v landscape[7] módu by načítal styly podle breakpointu 1024 px. Prohlížeč však obdrží informaci, že rozlišení je: 320 x 568 px. [2]

5.1.2.2 CSS řešení

Adaptivní obrázky lze řešit pomocí CSS stylů – obrázek nastavujeme jako pozadí určitého elementu. Případně lze šířku a výšku obrázku definovat v procentuální velikosti. Příkladem je blok CSS kódu pro mobile-first řešení:

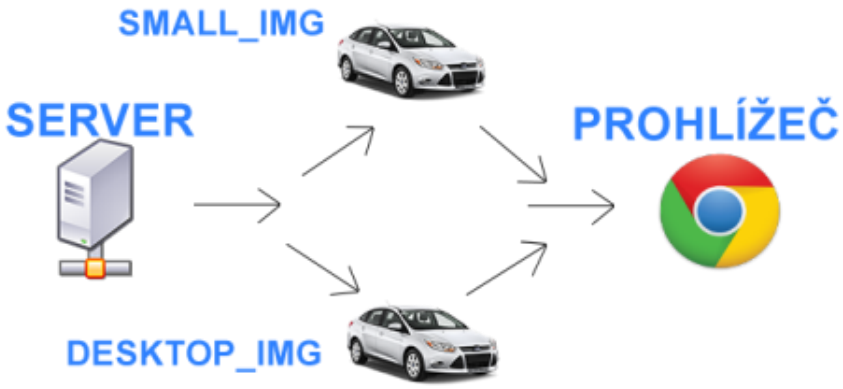
Příklad s obrázkem jako pozadím

```
.small_img{ background: #fff url('desktop_img.png') no-repeat; }
@media (min-width: 768px){
.small_img{ background: #fff url('mobile_img.png') no-repeat; }
}
```

[7] Telefon orientovaný na šířku displeje

Výše uvedený kód zobrazí na mobilním zařízení nepřesahující šířku 768 px obrázek: mobile_img.png. Jakmile šířka přesáhne tuto hranici, zobrazí se obrázek destop_img.

Toto řešení je spolehlivé, jednoduché a funguje ve většině prohlížečů. Podmínkou je podpora v prohlížeči vlastnosti @media-queries. Nevýhodou však je, že se fyzicky stáhnou oba obrázky (2 HTTP požadavky), jak ukazuje obrázek číslo 9:



Obrázek 9: Varianta pomocí CSS řešení.

V případě, že stylujeme rozměry obrázku v procentech, bude prohlížeč stahovat obrázek stejné datové velikosti jak pro mobilní zařízení, tak pro desktop. Rozdíl bude pouze v zobrazení vzhledem k viewportu.

5.1.2.3 Skriptované adaptivní obrázky

CSS styly nenabízí stoprocentní správné řešení adaptivních obrázků. Použitím Javascriptu lze dosáhnout v některých případech lepších výsledků. Nevýhodou je, že může dojít k zablokování Javascriptového kódu.

V responzivním designu často potřebujeme volit mezi různými variantami obrázku se stejným obsahem. Nejčastěji proto, že chceme ušetřit datovou velikost stránky na mobilním připojení. Výše uvedené varianty jsou popsány v následujících kapitolách.

5.1.2.4 Image-Set()

Jedná se o funkci, kterou zavedla Firma Apple. Ta je schopná detekovat hustotu pixelů jednotlivých displejů. Slouží k načítání kvalitnějších obrázků pro displeje s vysokým rozlišením – Retina, Amoled.

Zápis funkce:

image-set() + modifikátor

Příklad:

```
#pravySloupec{
      background-image:url(../images/obr1.png);
      background-image: -webkit-image-set(
      url(../images/obr2.png) 1x,
      url(../images/obr3.png) 2x);
}
```

Tento příklad zobrazí na displejích Retina obrázek obr3.png. Modifikátor 2x detekuje dvojnásobnou hustotu CSS pixelů na jeden fyzický pixel.

5.1.2.5 Atribut srcset

Jedná se o nové atributy elementu img, které umožňují zobrazování správných variant obrázků v závislosti na rozlišení obrazovky. Prohlížeči předáme informace, jaké varianty obrázků má k dispozici v atributu srcset.

Tento atribut zavedla pracovní skupina WHATWG[8] s podporou společnosti Apple. Vznik atributu prosadil z Applu Ted O´Connor – na základě své metody image-set(). V atributu jsou definované zdroje – varianty obrázků. Každý z těchto zdrojů má celkem až 3 možné deskriptory (popisovače) – jejich kombinací vznikají jedinečné identifikátory pro použití na specifickém zařízení. Jednotlivé deskriptory:

- w – šířka displeje v pixelech,
- h – výška displeje v pixelech,
- 1x – device – pixel – ratio (poměr pixelů).

Příklad:

```
<img src = "normalni.jpg" alt="obrazek"
      srcset="velkyobrazek.jpg" 1024w 200h 1x,
      srcset="velkyobrazek2x.jpg" 1024w 200h 2x,
      srcset="mobilobrazek.jpg" 320w 200h 1x >
```

5.1.2.6 Atribut sizes

Umožňuje vybírat obrázky nejen podle technických záležitostí okna prohlížeči, ale také podle jeho velikosti v rámci layoutu, ve kterém je aktuálně použit.

[8] Pracovní skupina, která se snaží pomocí nových technologií umožnit psaní a nasazení webových projektů jednotlivým autorům.

Příklad:

```
<img src="small.png"
srcset="small.png 600w, medium.png 1024w, large.png 1600w"
sizes="(min-width: 768px) 300px, 100vw"
alt="Obrázek" width="200" height="200">
```

Tímto blokem sdělujeme prohlížeči informaci: pokud bude mít layout stránky šířku větší než 768 px, bude obrázek v tomto layoutu mít šířku 300 px. Ve všech ostatních případech, (šířka < 768 px) bude obrázek široký 100 %. [3]

5.1.2.7 Velikost obrázků podle velikosti layoutu

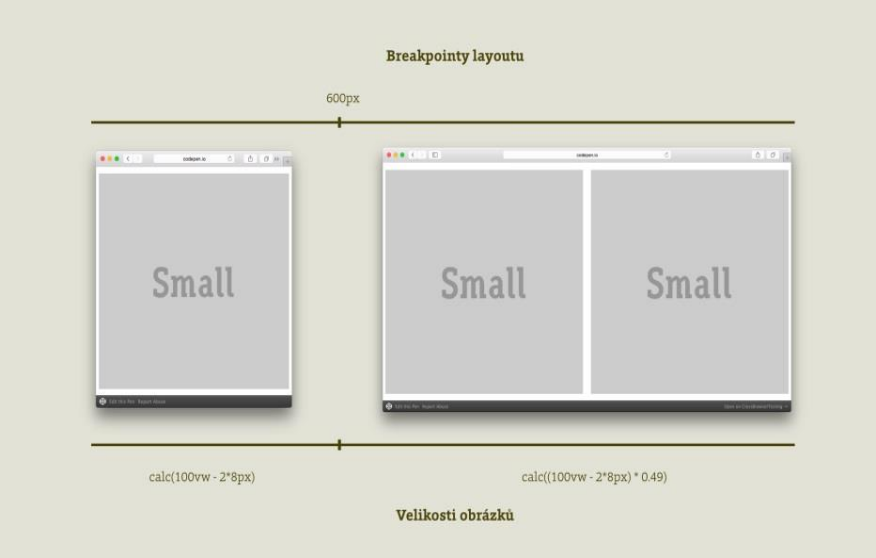
V responzivním layoutu nedokážeme předem určit, jakou velikost bude mít obrázek v rámci jednotlivých šířek viewportu. Pomocí funkce calc() lze tento problém vyřešit.

Příklad:

```
<img src="small_600.png"
srcset="small_600.png 600w, medium_1024.png 1024w, large_1600.png
1600w"
sizes="(min-width: 600px) calc((100vw - 2*8px) * 0.49), calc(100vw -
2*8px)" alt="Obrázek" width="200" height="200">
```

Tento kód sděluje prohlížeči následující informace:

- Viewport široký maximálně 599 px – obrázek bude mít šířku 100 % – 2 * 8 px (margin layoutu),
- Viewport dosáhne šířky 600 px – obrázek bude mít velikost: 100 % 2 * 8 px * 0,49 – poloviční velikost.



Obrázek 10: Ukázka funkce calc().

5.1.2.8 Element <picture>

Jedná se o nový tag, jehož pomocí lze definovat jednotlivé varianty obrázků v závislosti na odlišných stavech responzivního designu. Element picture obsahuje elementy source, které jsou jednotlivým zdroji.

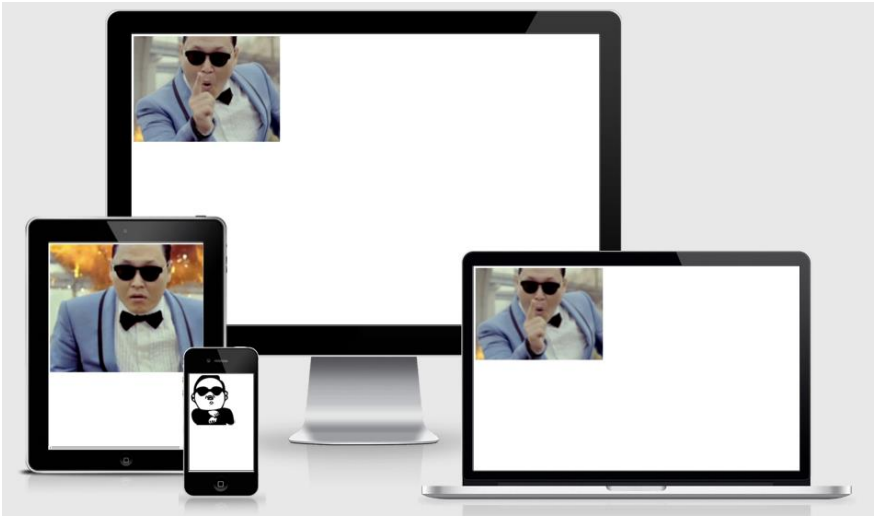
Příklad:

```
<picture>
<source media="(min-width: 1024px)" srcset="large.jpg">
<source media="(min-width: 600px)" srcset="medium.jpg">
<img src="4.jpg" alt="…">
</picture>
```

Obrázek 4.jpg se načte v prohlížečích, které nepodporují media queries, nebo při maximální šířce viewportu 599 pixelů. Musíme zde dodržet správné pořadí kvůli korektnímu chování. Prohlížeč načítá do elementu img jednotlivé zdroje z atributů srcset u elementů <source>. Tento element je vhodné použít v těchto případech:

- obrázek je například jinak oříznutý na mobilním zařízení než na tabletu,
- obrázek je v různých souborových formátech,
- obrázky mají různý kontext v rámci různých rozlišení.

Toto řešení stahuje na mobilních zařízeních obrázek s menší datovou velikostí, navíc je prováděn jeden HTTP request – na obrázek, který je aktuálně stahován. Je nutné stejně jako v ostatních případech připojit Picturefill (Polyfill[9]). Při testování tato varianta fungovala nejlépe. Ostatní varianty měli problém v jednotlivých prohlížečích. [3]



Obrázek 11: Použití elementu <picture> v praxi.

[9] V kontextu frontendu se jedná o Javascriptový kód, který nám umožní zprovoznit HTML5 komponentu ve straších prohlížečích. Například Respond.js zprovozní CSS3 media queries na IE8

5.1.3 @media-queries

Media queries jsou dalším ze základních pilířů responzivního designu. Název bychom mohli do češtiny přeložit jako dotazy na médium, dotazy na výstupní médium a podobně. Díky nim jsme schopni detekovat vlastnosti jednotlivých zařízení. Například šířku mobilního telefonu v portrait módu (na výšku) nebo landscape módu (na šířku), hustotu pixelů tabletu a podobně. Prohlížeč na základě stanovených pravidel načte jednotlivé sady CSS stylů. Lze tak například detekovat mobilní telefon iPhone 5S v režimu landscape módu a jiné konkrétní specifikace.

Formálně se jedná o specifikaci W3C. Prvně se Media queries objevily v CSS2, kde se konkrétně jednalo o tiskovou funkci print. Původní myšlenka použití byla prostá: schopnost detekovat tisková zařízení, projektory, desktopy a mobilní telefony. V jazyce CSS3 byla tato pravidla prohloubena vzhledem ke zvyšujícím se nárokům na vlastnosti zařízení.

5.1.3.1 Struktura @media-queries

Obecný příklad:

```
@media not|only typ and (výraz){
CSS pravidla
}
```

Konkrétní příklad:

```
@media screen and (min-width:320px){
div#blogs{
      width:30%;
      display:block;
      background: green;
      }
}
```

Media-queries se skládají z jednotlivých sekcí:

- Mediální typ – určuje typ zařízení, na kterém je web zobrazen,
- mediální výrazy – vlastnosti zařízení vyhodnoceny - true nebo false,
- logické operátory – lze vytvářet složitější výrazy,
- pravidla – jedná se o základní CSS styly, které jsou aplikovány, pokud je výsledek true.

5.1.3.2 Mediální typ

Jednou z mnoha výhod webových stránek je, že mohou být zobrazeny na mnoha různých zařízeních, nemusí se jednat jen o počítačové obrazovky. Můžeme je zobrazovat na tabletech, smartphonech, může se jednat také jednat o syntetizátory řeči, projektory, televizory a další typy zařízení. Mediální typy byly vyvinuty proto, aby se upřesnila pravidla v používání těchto zařízení. Sdělují prohlížeči, na jakém zařízení je web aktuálně prohlížen. [2]

Přehled jednotlivých typů:

- All – všechna zařízení, defaultní volba pokud není uvedeno jinak,
- Braille – zařízení s hmatnou zpětnou vazbou v Braillově písmu,
- Embossed – tiskárna Braillova písma,
- Handheld – přenosná zařízení (typicky malé obrazovky),
- Print – tisk nebo náhled před tiskem,
- Projection – projektor,
- Screen – barevná počítačová obrazovka,
- Speech – syntetizátor řeči,
- Tty – media která používají neproporciální znaky (terminál),
- Tv – televize.

V praxi se využívají převážně: all, screen a print. Mediální typy sami o sobě selektují velmi široký rozsah zařízení, proto se používají mediální výrazy, jimiž je možné konkrétněji specifikovat zařízení. V praxi si převážně vystačíme s width, height, orientation, resolution, někdy také aspect-ratio.

5.1.3.3 Mediální výrazy

Výsledek je reprezentován hodnotami true nebo false.

Příklad:

```
@media screen and (min-width:320px) {
Seznam CSS vlastností;
}
```

Tento blok testuje dvě věci:

- Zda se jedná o počítačovou obrazovku (screen).
- Zda je šířka viewportu větší nebo rovna 320 px. (min-width).

Následující tabulka vyjadřuje vlastnosti zařízení, které se mohou testovat:

Charakteristika	Definice	Hodnota	Min / Max
Width	Šířka zobrazovací části zařízení	<lenght> (př. 320)	Ano
Height	Šířka zobrazovací části zařízení	<lenght> (př. 600)	Ano
Device-width	Šířka renderovacího povrchu zařízení	<lenght> (př. 600)	Ano
Device-height	výška renderovacího povrchu zařízení	<lenght> (př. 600)	Ano
Orientation	Indikace zařízení na šířku /výšku	Portrait /landscape	Ne
Aspect-ratio	Poměr hodnot width /height	<ration> (př. 16/9)	Ano
Device-aspect-ration	Poměr hodnot device-width /device-height	<ration> (př. 16/9)	Ano
Color	Počet bitů zařízení na komponentu (vrací 0 pokud je barevné zařízení)	<integer> (př. 0)	Ano
Color-index	Počet položek ve vyhledávací (look-up) tabulce zařízení	<integer > (př. 250)	Ano
Monochrome	Počet bitů na pixel u monochronního zařízení (vrací 0 pokud je zařízení jednobarevné)	<integer > (př. 8)	Ano
Resolution	Rozlišení (hustota pixelů dpi, dpcm)	<resolution> (př 118dpcm)	Ano
Scan	Skenovací proces „televizních zařízení“	Progressive | interlace	Ne
Griud	Mřížkové zařízení (1 – ano, 0 – není)	<inetger> (př. 1)	1ne
device-pixel-ratio	Detekce vysokokapacitních displej.	(př. 2)	ano

Tabulka 1: Media queries - přehled vlastností.

V praxi se však používají převážně tyto výrazy:

- Width,
- height,
- orientaton,
- resolution,
- aspect-ratio,
- device-pixel-ratio.

5.1.3.4 Logická klíčová slova

Pomocí nich můžeme vytvářet sofistikovanější mediální dotazy. Můžeme například testovat v jeden okamžik více vlastností pomocí mediálních výrazů.

Příklad:

```
@media screen and (color){
CSS vlastnosti;
}
```

- Testujeme, zda se jedná o počítačovou obrazovku. (screen)
- Testujeme, zda je tato obrazovka barevná. (color)

Seznam logických klíčových slov:

- AND – současně se testuje platnost několika výrazů.

  Příklad:

  ```
  @media screen and (min-width:1200px){
  Width:60%;
  float:left;
  background: purple;
     }
  ```

  Detekce barevné obrazovky + min šířka obrazovky 1200 px:

- NOT – negace výrazu, vrací hodnotu true když podmínka neplatí. Neguje se celý výraz, nikoliv jen část za klíčovým slovem NOT.

  Příklad:

  ```
  @media not screen and (min-width:533px){
  div#left{
           Width:60%;
           float:left;
           background: purple;
           }
  ```

  - Detekce mobilního telefonu.

    - Detekce konkrétního rozlišení (Samsung Galaxy S3 mini) v landscape módu.
    - Alternativně je možné testovat vlastností orientation.
- OR – logický výraz vyjadřující analogicky slovo „nebo“ (pokud platí alespoň jedna z podmínek, výsledek vrátí hodnotu true). V zápise se toto klíčové slovo nepoužívá, místo toho je reprezentováno čárkou.

  Příklad:

```
@media print ,(min-width:992px){
div#left{
         float: left;
         width:60%;
         height: 30em;
         background: purple;
         }
```

- ONLY – Díky tomuto logickému slovu můžeme skrýt CSS3 styly před prohlížeči, které jim nerozumí.

  Příklad:

```
@media only screen{
Div#left.wrapper{
   Box-shadow: black 4px 6px 20px}}
```

Tuto sadu stylů s vlastnosti CSS3 box-shadow (vržený stín) skryjeme tak například před prohlížečem IE8 který nepodporuje @media queries.

5.1.3.5 CSS pravidla

Samotná CSS pravidla jsou ve složených závorkách. Ta jsou aplikována, pokud výsledek nabývá hodnoty true. Tento výsledek se vztahuje vždy k @media queries, jímž začíná tento blok CSS kódu.

5.1.4 Připojení mediálních dotazů

Existují tři způsoby připojení do html dokumentu:

- Externí styly,
- Přímo v CSS souboru – podmínění celého bloku stylů,
- Přímo v CSS souboru – podmínění jednotlivých pravidel.

1) **Externím stylopisem**

   Tento druh připojení se píše přímo do elementu: link:

   ```
   <link href="mobile.css" media="screen and (min-width:480px)"
   />
   ```

   U tohoto typu připojení jsou mediální dotazy vloženy do jednoho souboru. Stačí pouze jeden HTTP požadavek pro stažení jednoho souboru CSS pravidel. Pokud však máme externích souborů více, provede se odpovídající počet HTTP požadavků. Musí být totiž staženy, pokud by se například změnila orientace displeje.

   Příklad:

   Provedou se tři HTTP požadavky:

   ```
   1) <link rel="stylesheet" type="text/css" href="media.css">
   2) <link rel="stylesheet" href="portrait.css" media="screen
      and (min-width:320px)"/>
   3) <link rel="stylesheet" href="landscape.css" media="screen
      and (min-width:321px)"/>
   ```

2) **Přímo v CSS souboru** – podmínění celého stylopisu.

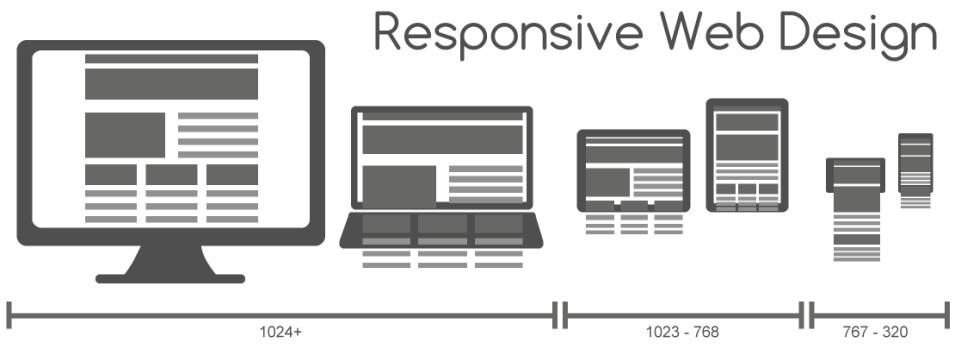
   Využíváme pravidlo @import. Můžeme jej zapsat přímo v HTML souboru nebo jej vnoříme na začátek CSS souboru. (IE 6 a IE 7 toto pravidlo nepodporuje)

3) **Přímo v CSS souboru** – podmínění jednotlivých pravidel.

   Jednotlivé mediální dotazy píšeme do CSS souboru v pořadí, v jakém mají být vlastnosti detekovány. To se může odvíjet od techniky postupu, pokud například využíváme techniku mobile first či desktop first. Tyto techniky jsou popsány v dalších kapitolách.

5.1.5 Breakpointy

Abychom mohli využívat vlastností media queries, je třeba definovat tzv. breakpointy. Jedná se o hraniční rozměry obrazovky nebo displeje. Jsou to zlomové body, ve kterých dochází k aplikaci jiných CSS pravidel definovaných v mediálním dotazu – změní se vzhled stránky.



Obrázek 12: Znázornění rozdělení zařízení podle breakpointů.

Jedním z problémů breakpointů je jejich definice. Nejsou totiž pevně definovány. Lze je dělit několika způsoby:

1. Podle předem daných rozlišení jednotlivých zařízení:
   - 320 px – zařízení s malou obrazovkou, př. telefon v portrait módu,
   - 480 px – zařízení s malou obrazovkou, př. telefon v landscape módu,
   - 768 px – př. tablet v landscape módu,
   - 800 px – př. 10“ tablet (iPad 768 x 1024) v portrait módu,
   - 1024 px – př. tablety v landscape módu, netbooky, osobní počítače,
   - 1200 px – př. desktopy, širokoúhlé displeje.
2. Předdefinované v určitém frameworku:
   - Příklad – Bootstrap breakpointy:
     - 480 px – mobilní telefony,
     - 768 px – tablety,
     - 992 px – desktopy,
     - 1200 px – velké displeje.
3. Vlastní breakpointy – jsou definovány následujícím postupem:
   1. Layout nadefinujeme ve flexibilních jednotkách (em, rem, procenta).
   2. Zmenšujeme / zvětšujeme okno prohlížeč, dokud se nám „nerozsype“ grafický design.
   3. Tento bod definujeme jako breakpoint.[4]

5.1.5.1 Používání mikrobreakpointů

Nevýhodou klasických breakpointů je jejich nepřesná detekce zařízení. Šířku 480 px může mít totiž iPhone 3 a iPhone 4 v landscape módu nebo starý Android tablet v portrait módu. Na následující stránce je přehled rozlišení jednotlivých vybraných zařízení: http://mydevice.io/devices/#sortOthers

Klasickými breakpointy rozdělíme CSS na celé logické části, díky možnosti zanořování preprocesorového kódu můžeme však vytvořit mikrobreakpointy, díky nimž dokážeme lépe kontrolovat chování layoutu.

Příklad:

1. Bez použití zanořování – klasické CSS:

Definice dvou tříd:

```
.nav { /* … */ }
.content  { /* … */ }
```

Řešení pro tablety:

```
@media screen and (min-width: 481px) and (max-width: 1023px) {
  .nav{}
  .content{}
}
```

Řešení pro smartphony:

```
@media screen and (max-width: 480px) {
  .nav{}
  .content{}
}
```

2. Zanořování CSS – použití preprocesoru (mikrobreakpointů)

```
.nav {
  @media screen and (max-width: 560px){
}
}
.content  {
  @media screen and (max-width: 900px) {
}
}
```

5.1.6 Mobile / Desktop First

Jedná se o techniky použité při postupu návrhu webu. Použitím mobile first nejdříve přizpůsobujeme rozvržení layoutu pro nejmenší obrazovky (mobilní zařízení) a postupně řešíme pro větší displeje (desktopy). K této technice existuje opačný postup – technika, která se nazývá desktop first. V té nejdříve navrhujeme design pro desktopy a postupně přidáváme pravidla pro menší obrazovky – „omezujeme design“.

Obrázek 13: Znázornění mobile first metody.

5.1.6.1 Mobile first

Postup tvorby je ve své podstatě takový, že nejprve nastavíme stránce základní barvy fonty a naformátujeme bloky. Následně začneme zvětšovat velikost okna prohlížeče. Jakmile se nám v konkrétním breakpointu layout zobrazí nekonzistentně, pomocí @media queries aplikujeme CSS styly, jimiž ho spravíme do odpovídajícího stavu.

Analogicky provádíme postup pro větší obrazovky. Výhodou tohoto řešení je, že prohlížeče (konkrétně IE 8), které nepodporují media queries, zobrazí základní layout. U mobile first metody bychom se měli zamyslet nad obsahem. Malé obrazovky totiž nutí autora stránky zvážit, jaký obsah zobrazí uživateli – minimalizace obsahu. Při této technice se využívá výrazu min-width, jak ukazuje následující příklad. [1,5]

Příklad:

```
@media screen and (min-width:480px){
    div#obsah{
Width:60%;
Float:left;
}
div#sidebar{
width:30%;
margin-left:10%;
}
}
```

Snadnější je z jednoduchého webu dělat složitější než opačně. Díky minimalistickým CSS stylům pro mobily dosáhneme menší datové náročnosti.

5.1.6.2 Desktop first

Tato metoda je opačným postupem k mobile first metodě. Nejprve tedy navrhujeme pro desktopy a následně pro menší obrazovky přizpůsobujeme styly. U této metody převažují nevýhody nad výhodami:

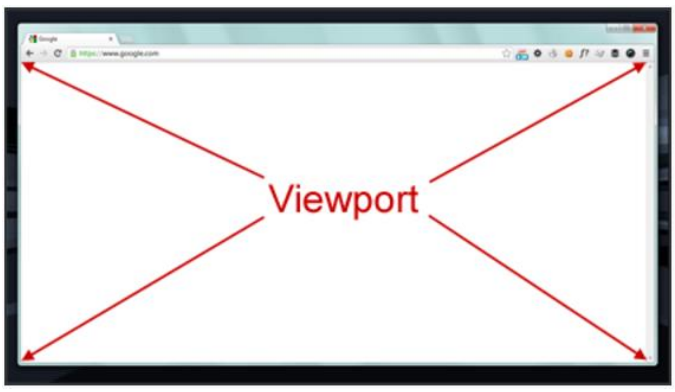
- Při stylování pro menší displeje používáme také vlastnost: display: none (není vhodné z hlediska SEO).
- Styly nejsou vidět na menších zařízeních, ale přesto se zbytečně stahují.
- Větší problém je web zjednodušovat, než jej dělat složitější.

Příklad:

```
@media screen and (max-width:992px){
div#obsah{
      width:65%;
      float:left;
      padding:2%;
      background:#sd45s45;
}
div#side.cols{
      display:none;
}
}
```

5.1.7 Viewport

Viewport z hlediska desktopového počítače představuje viditelnou oblast prohlížeče. U mobilních telefonů je koncipovaný jinak. Přestože mají mobilní telefony menší displeje, snaží se zobrazit kompletní web v plné šířce, což dělá web nečitelným. Uživatel si jej pak musí zvětšit zoomováním, což je nekomfortní.



Obrázek 14: Viewport na desktopu.

Mobilní prohlížeč v orientaci na výšku, který má šířku 320 px, zobrazuje web 960 px široký. Aby měl návštěvník po načtení přehled o celé stránce a mohl si následně přiblížit, co potřebuje, musí jej přiblížit – zazoomovat.

Obrázek 15: Rozdíl mezi viewportem zařízení a viewportem webové stránky.

Tomu zamezíme, pokud v hlavičce definujeme meta tag – viewport:

```
<head>
<meta name=viewport content="direktiva, direktiva">
</head>
```

- Direktiva width - umožňuje nastavit viewport na konkrétní šířku nebo na šířku zařízení:
  - `<meta name="viewport" content="320px" />`
    - Viewport webu se přizpůsobí visual viewportu (zařízení) konkrétně na šířku 320 px.
    - Nevhodné řešení – vysoká rozmanitost zařízení s různým rozlišením.
- Direktiva device-width:
  - `<meta name="viewport" content="width-device-width">`
    - Toto řešení je optimální,
    - Šířka viewport layoutu se přizpůsobí vždy visual viewportu zařízení.
- Ostatní direktivy
  - Height – hodnoty se nastavují stejně jako u width (absolutní např. 480 px, nebo device-height). Tuto direktivu bychom použili jen v případě,

pokud bychom nechtěli posunovat obsah stránky dolů. V praxi se nepoužívá.

- User-scalable – definuje prohlížeči, zda uživatel bude moct stránku přibližovat / oddalovat. Nabývá hodnot Yes / no. Typicky se jedná o pixel perfect weby. Bez uvedení je výchozí hodnota Yes. V praxi se běžně nevyužívá.
- Initial-scale – nastavuje se počáteční úroveň přiblížení stránky v rozsahu 0.1 (10 %) do 10.0 (1000 %). Pokud tuto hodnotu nastavíme na 1,

  pak se layout široký 320 px bude zobrazovat 1 : 1 na zařízení s šířkou 320 px.

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

- Minimum - scale – určuje, jak moc bude schopen uživatel stránku oddalovat (rozsah hodnot 0.1 – 10.0) Pokud nastavíme na 1.0 – 100 %, uživatel si nikdy nebude moci stránku zmenšit – oddálit. [1]

Obrázek 16: RWD řešením s nastavenou direktivou a bez direktivy.

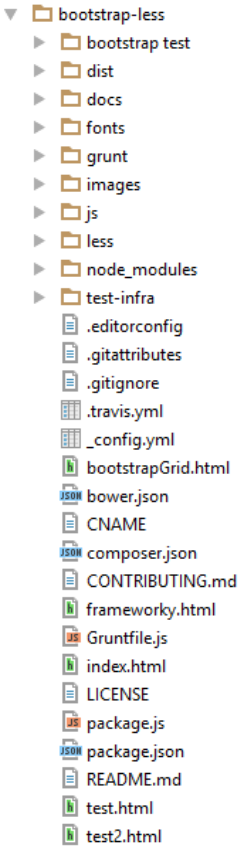
6 HTML5 frameworky

Pojem framework lze přeložit jako sada knihoven, aplikační rámec a podobně. V praxi se jedná o základní kostru určitého kódu, který může použít frontend designér jako výchozí bod pro svůj budoucí projekt. Jedná se standardizovaný soubor pojmů, postupů a kritérií pro konkrétní postup při řešení specifické problematiky. Framework usnadňuje vývojářům velkou část práce přebíráním základních problematických oblastí. Může obsahovat dále různé podpůrné programy, API knihovnu, podporu pro návrhové vzory – obecná řešení využívající podporu programů v softwarovém inženýrství. [6]

6.1 Frontend frameworky

Typicky můžeme rozdělit frameworky na dva typy – backend a frontend frameworky (CSS / HTML5 frameworky).

Frontend frameworky se skládají ze souborů, které jsou uspořádány v určité hierarchické struktuře a obsahují zdrojový kód, který je napsaný tak, aby se choval ve výsledku stejně ve všech prohlížečích.



Obrázek 17: Struktura Bootstrapu.

V tomto případě se jedná o HTML, CSS a Javascript. V poslední době je na vzestupu vývoj responzivních frameworků, které se skládají typicky z těchto segmentů:

- CSS styly pro vytvoření gridu[10] – umožňují jednoduché a konzistentní rozmístění prvků v rámci layoutu, který se chová responzivně.
- Definici typografie pro HTML elementy.
- Alternativní řešení pro případ, kdy prohlížeč nepodporuje určitou část kódu (IE 8).
- Univerzální třídy, které můžeme využívat pro stylování user-interface[11] komponent.

Dělíme je spíše dle subjektivního pohledu. Neexistuje žádné standardní rozdělení.

- Jednoduché frameworky:
    - Převážně nabízejí základní CSS styly,
    - obsahují grid,
    - 960 Grid System, Base, Titan, …
- Kompletní:
    - Výše uvedené,
    - konfigurovatelné funkce,
    - UI komponenty,
    - Javascripty pro UI komponenty
    - Bootstrap, Foundation, Gumby, Skeleton, ….

6.2 Bootstrap

Jedná se o frontendový framework s MIT[12] licencí. Vytvořili jej Mark Otto a Jacob Thornton na Twitteru jako interní podporu. Jeho původní název zní: Twitter Blueprint. Před Bootstrapem sloužily pro tuto podporu různé knihovny, ty však nepřinášely takové výsledky, jaké od nich byly očekávány. Na vývoji Bootstrapu se podíleli také ostatní vývojáři v rámci Hack week. Vznikl v polovině roku 2010. Oficiálně byl však uvolněn až 19. srpna 2011. Je kompatibilní téměř se všemi desktopovými prohlížeči, dokonce i s IE 8, což je dnes nejproblémovější prohlížeč z hlediska podpory.

[10] Mřížka – grafické rozmístění layoutu na stránce
[11] Tlačítka, rolovací menu, dynamické formuláře
[12] Svobodná licence

Jedná se o modulární systém složený z LESS[13], CSS a Javascriptových souborů. Díky použití preprocesoru LESS je možné využívání proměnných, funkcí, vnořených selektorů, zanořování media queries – vytváření mikrobreakpointů a podobně. Lze tak pomocí funkce @import využívat pouze soubory, které opravdu využijeme.

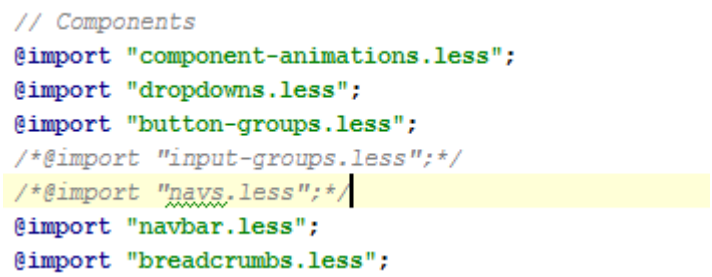
Od verze 2.0 obsahuje techniky pro podporu responzivního designu. S touto verzí přišla také varianta balíčku ke stažení s názvem: „Customize“ díky níž je možné konfigurovat Bootstrap před stažením – zvolíme si komponenty, které budeme chtít stáhnout. Můžeme si také konfigurovat jednotlivé vlastnosti: (typografii – výchozí písmo, základní velikost písma a podobně). Je tak zajištěna redukce přebytečného kódu, který bychom nevyužili, tím i velikosti, která ovlivňuje rychlost načítání stránek. Bootstrap může být pro každý projekt přizpůsoben jinak díky výběrem komponent, které budeme skutečně potřebovat. Ve verzi 3 byly styly pro mobile-friendly řešení přepsány od základů a místo přidávání různých stylů byly přepsány přímo v jádru. Vznikla tak plná podpora techniky mobile-first. V současné době je používána verze 3.3.6, ke stažení je již beta verze 4. Ve verzi 4 bude Bootstrap pracovat s preprocesorem SASS.

6.2.1 Struktura Bootstrapu

Bootstrap je modulární systém složený z LESS souborů představujících jednotlivé UI komponenty, grid, pomocné třídy atd. Obsahuje potřebné zdroje pro task runner (Gruntfile.js). LESS soubory jsou kompilovány do CSS. Webovou stránku pak lze snadno stylovat použitím těchto CSS stylů. Je možné využívat CSS nebo mnohem efektivnější variantu – LESS.

LESS soubory jsou importovány v základním konfiguračním souboru Bootstrap.less. Výhoda tohoto řešení spočívá v tom, že využíváme styly, které budeme pro vývoj skutečně potřebovat. Ty, které nepotřebujeme, jednoduše zakomentujeme. [9]

[13] soubory preprocesoru LESS



Obrázek 18: Struktura Bootstrap.less souboru.

Webové stránky jsou pak rychlejší, jelikož se nemusí stahovat přebytečné množství dat. Další změny lze provádět přímo v samotných souborech LESS. Jelikož se jedná o kompletní frontendový framework, obsahuje také Javascriptové soubory a J-Query pluginy pro dynamické chování komponent, automatické funkce, slideshow a podobně.

- JS komponenty:
    - Modal (modální dialogová okna),
    - Dropdown (rozbalovací políčka / menu s nabídkou),
    - Scrollspy (dynamický obsah),
    - Tab (přepínatelné prvky, z nichž je jeden právě aktivní),
    - Tooltip (bublinková nápověda),
    - Popover (otevírací okna),
    - Alert (výstražné a informativní boxy),
    - Button (tlačítka),
    - Collapse (efekt pro změnu obsahu),
    - Carousel (slideshow),
    - Typeahead (našeptávač).

6.2.2 Instalace

Bootstrap stačí stáhnout z oficiálních stránek, nakopírovat si ho do projektu a začít jej využívat. Verze 3.3.6 nabízí tři varianty ke stažení:

1. Bootstrap – obsahuje minifikované CSS soubory, Javascript a webové fonty s piktogramy.
2. Source code – obsahuje LESS soubory, Javascript, gruntfile pro automatizační nástroj.

   Struktura složek:

   - less, js, fonts – zdrojové kódy LESS, JS.

- dist – kompilované soubory CSS, JS,
- docs – dokumentace,
- examples – ukázkové příklady (grid, UI komponenty).

3. Sass – Bootstrap se styly napsanými v SASS preprocesoru.

Bootstrap lze použít také prostřednictvím sítí CDN.

6.2.3 CSS styly

6.2.3.1 Grid – mřížka

Bootstrap obsahuje responzivní grid rozdělený do 12 sloupců, které se automaticky přizpůsobují. K definování jednotlivých sloupců a řádků používáme třídy. Grid Bootstrap využívá předdefinované třídy (soustavy tříd) pro specifické breakpointy rozděleny do 4 základních kategorií:

- Extra malá zařízení (< 768 px) – smartphony.
- Malá zařízení (≥ 768 px) – tablety.
- Střední zařízení (≥ 992 px) – desktopy.
- Velká zařízení (≥ 1200 px) – desktopy s velkými obrazovkami.

Hlavní třída pro grid má název **container**, má nastavené fixní šířky. Ty jsou aplikovány pomocí @media queries. Můžeme také využít třídu s označením **container-fluid** kde je velikost definována jako 100% – webová stránka je zobrazena přes celý viewport. Definice jednotlivých sekcí:

- Řádek:

```
<div class="row">
  <div class="col-xs-3></div>
  <div class="col-xs-9"></div>
</div>
```

- Sloupce – jsou definovány breakpointů. Písmena XX nabývají hodnoty 1 až 12 – počet sloupců.

```
<div class="col-xs-xx">
<div class="col-sm-xx">
<div class="col-md-xx">
<div class="col-lg-xx">
```

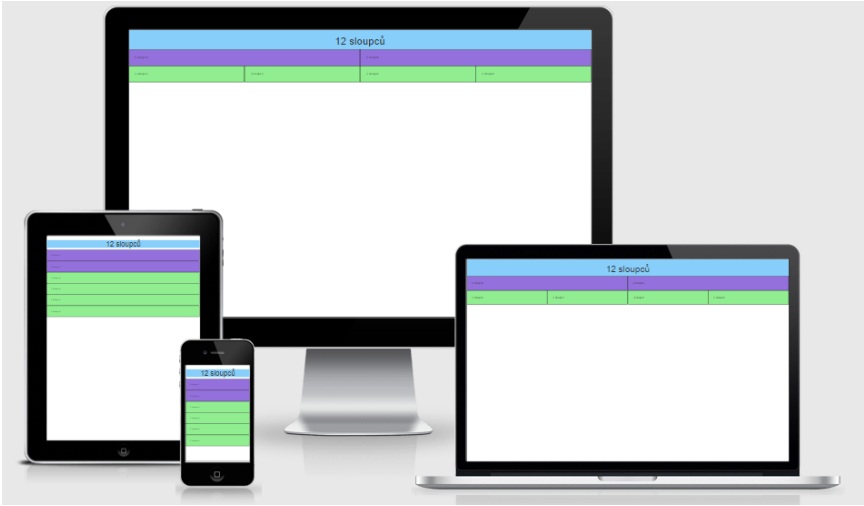
Následující tabulka vysvětluje význam použití jednotlivých kategorií.

	Extra malá zař.	Malá zař.	Střední. Zař	Velká zař.
Chování mřížky	Horizontální po celou dobu	horizontální, menší než breakpoint SM - blok	horizontální, menší než breakpoint MD - blok	horizontální, menší než breakpoint LG - blok
Šířka containeru	Auto	750 px	970 px	1170 px
Prefix třídy	.col-xs-	.col-sm-	.col-md-	.col-lg-
Šířka sloupce	Auto	62px	81px	97px
Šířka mezery mezi sloupci	30px (padding 15px na každé straně)			
Posuny jednotlivých sloupců	Ano			
Změna pořadí sloupců	Ano			

Tabulka 2: Přehled jednotlivých kategorií.

Na obrázku číslo 19 vidíme, jak se zobrazí responzivní grid na různých zařízeních. Je zvolena varianta pro malá zařízení – třída: <col-md-xx>.

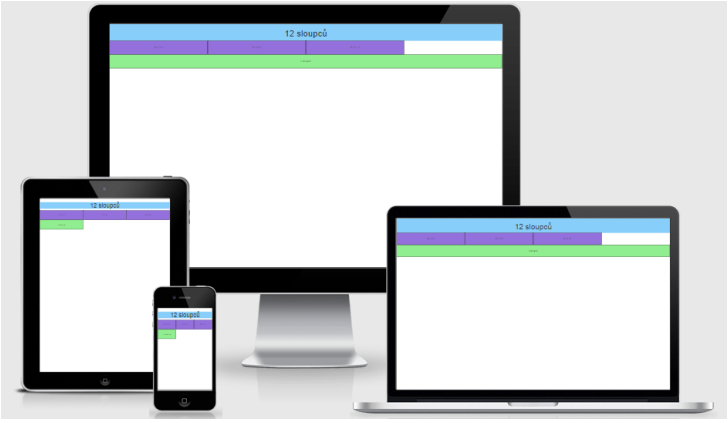
:



Obrázek 19: Responzivní grid – stejné velikosti.

Třídy lze kombinovat tak, aby se adaptivně zobrazovaly dle velikosti zařízení. V tomto případě jsou kombinovány třídy:

- col-xs-xx,
- col-md-xx,
- col-lg-xx.

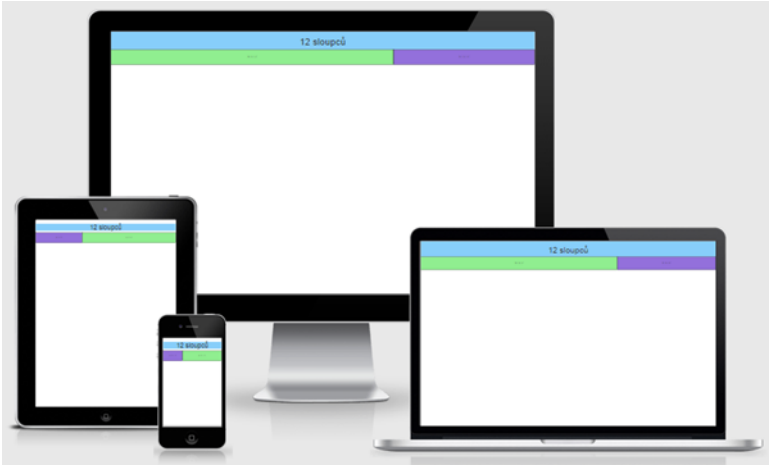


Obrázek 20: Responzivní grid, rozdílné velikosti.

Pořadí sloupců v horizontálním směru lze pohodlně měnit použitím následujících tříd.

- .col–xx-push – sloupec je odsazen zleva CSS vlastností float:left.
- .col-xx-pull – využívá opačnou CSS vlastnost float: right.

Na obrázku číslo 21 je znázorněna funkčnost výše uvedených tříd.



Obrázek 21: Použití tříd push a pull.

Sloupec, který má nastavenou zelenou barvu pozadí, mění pořadí v horizontálním směru na smartphonu a tabletu jinak než na desktopu. [9]

6.2.3.2 Typografie

Výchozí velikost textu v Bootstrapu je nastavena na 14 px a výška řádku je nastavena na 1,428. V případě že používáme preprocesor LESS používáme proměnné:

- @font-size-base – velikost písma,
- @line-height-base – výšku řádku.

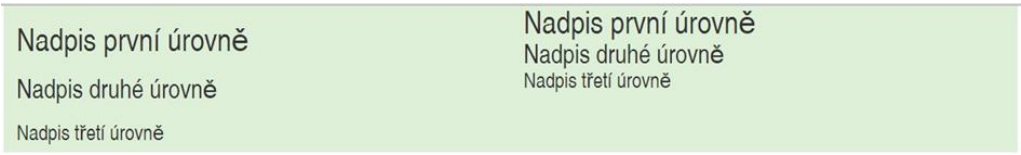
Bootstrap zajišťuje silnou podporu týkající se nadpisů <h1> až <h6>. Pro stylování textu je možné využít třídy:

- .h1 až .h6.

Tyto třídy využijeme v případě, že potřebujeme nastylovat text tak, aby vypadal jako nadpis. Chceme však zachovat chování typické pro řádkový element. Rozdíly ukazuje následující příklad společně s obrázkem.

Příklad:

```
<h1>Nadpis první úrovně</h1> <span class="h1">Nadpis první úrovně</span>
<h2>Nadpis druhé úrovně</h2> <span class="h2">Nadpis druhé úrovně</span>
<h3>Nadpis třetí úrovně</h3> <span class="h3">Nadpis třetí úrovně</span>
```



Obrázek 22: Rozdíl v použití nadpisu a tříd .h1-.h6.

6.2.3.3 Tabulky

Bootstrap nabízí několik variant pro tvorbu tabulek. Základem je třída .table, která se přiřazuje elementu <table>:

Příklad:

```
<table class="table"></table>
```

Tato základní třída definuje základní nastavení vlastnosti padding pro jednotlivé buňky a horizontální oddělení jednotlivých řádků.

Ostatní třídy:

- .table-striped – definuje každému lichému řádku odlišnou barvu.

First Name	Last Name	Username
Mark	Otto	@mdo
Jacob	Thornton	@fat
Larry	the Bird	@twitter

Obrázek 23: Znázornění třídy .table-striped.

- .table-bordered – každá buňka v tabulce je ohraničená,
- .table.hover – na každý řádek tabulky je použita CSS pseudotřída :hover,
- Použití pseudotříd – lze použít pro jednotlivé řádky:

    - .active – barva pseudotřídy :hover,
    - .success – úspěšná, pozitivní akce,
    - .info – informativní změna / akce,
    - .warning – varování,
    - .danger – negativní akce,
- .table-responsive. – responzivní chování.

6.2.3.4 Formuláře

Jednotlivým formulářovým elementům (input, textarea, a podobně) je přiřazena třída:

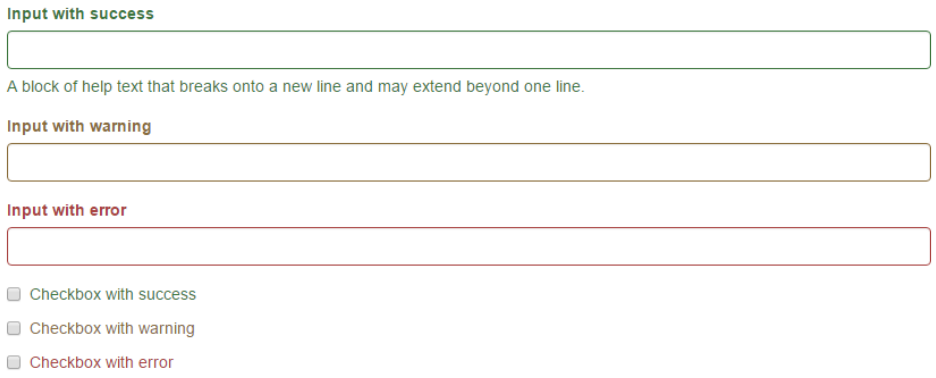
- .form-control – nastavuje defaultní šířku na 100%.
- .form-group – zajišťuje optimální rozložení mezi jednotlivými elementy.
- .form-inline – formulář je zarovnán jako řádkový element. Jakmile velikost viewportu přesáhne 768 px.

Formulářům lze nastavit také atributy pomocí proměnných typu boolean:

- Disabled – aplikuje se na jednotlivé formulářové elementy. Na obrázku číslo 24 je názorná ukázka jak se bude zobrazovat kurzor myši při najetí na formulářový element, který má nastaven tento atribut.
- Readonly – nelze zapisovat do formuláře.

Obrázek 24: Znázornění třídy .disabled.

Pro formuláře jsou předdefinovány třídy, které vizuálně signalizují specifické stavy. Nejedná se o kontextové použití tříd – nemají sémantický význam (čtečka pro nevidoucí či barvoslepé nebude registrovat tuto třídu odlišným způsobem). [9]



Obrázek 25: Vizuální třídy pro formuláře.

6.2.3.5 Tlačítka

Tlačítka, hypertextové odkazy, formulářové vstupní elementy a podobně můžeme stylovat několika třídami, aby dostaly vzhled jako tlačítko:

- .btn – základní vzhled tlačítka,
- .btn-default – defaultní vzhled tlačítka.

Pokud je element <a> využíván funkčně stejně jako element <button>, měl by mít přidělen následující atribut.

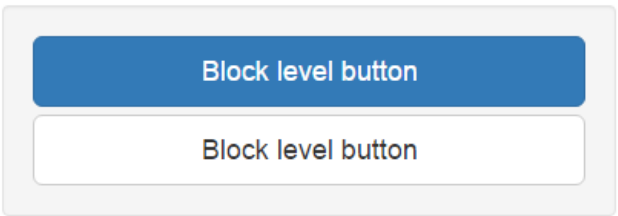
- role=“button“

Třídy pro změnu velikosti tlačítka:

- .btn-lg,
- .btn-sm,
- .btn-xs,

Využít lze také tříd symbolizujících jednotlivé stavy – výstraha, informace a podobně, jak je vidět na obrázku č. 26. Stylovat je také můžeme proto, aby vypadaly jako blokové elementy – obrázek č. 27.

Obrázek 26: Tlačítka 1.



Obrázek 27: Tlačítka 2.

6.2.3.6 Obrázky

Základního responzivního chování obrázku – přizpůsobení velikosti (datová velikost zůstává stejná) docílíme přidáním třídy elementu <img>:

- .img-responsive – obsahuje CSS styly:
  - max-width:100%,
  - height-auto,
  - display:block.

Pro základní stylování obrázků používáme třídy:

- .img-rounded – rohy mají radius,
- .img-circle – kulatý obrázek,
- .img-thumbnail – odsazený rámeček.

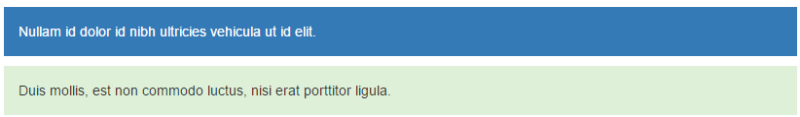
Obrázek 28: CSS třídy pro náhledy obrázků.

6.2.3.7 Pomocné třídy

Umožňují rychlé úpravy. Pokud potřebujeme využít text v nějakém kontextu, zarovnat určitý element a podobně.

- Kontextuální třídy – slouží k určení kontextu obsahu. Nastavují barvu specifické barvy:
  - bg-primary,
  - bg-success,
  - bg-info,
  - bg-warning,
  - bg-danger.

Příklad:



Obrázek 29: Použití kontextových tříd primary (modrá) a success (zelená)

- Obtékání:
    - pull-left,
    - pull-right.
- Zrušení obtékání:
    - Clearfix.
- Skrývání obsahu – tyto třídy používají klíčovou vlastnost !important:
    - .show – CSS vlastnost: display: block,
    - .hidden – display: none,
    - .invisible – visibility: hidden.
- Ikony – používají se pro element :
    - .close – křížek signalizující zavření okna,
    - .caret – šipka signalizující vysouvací menu.

6.2.3.8 Responzivní utility

Tyto třídy obsahují styly, které na základě breakpointů umožňují blok skrýt nebo zobrazit. Obsah, který má být viditelný pouze na tabletu, se nezobrazí na smartphonu. Teoreticky se jedná o čtvrtý pilíř responzivního obsahu. [9]

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
.visible-xs-*	Visible	Hidden	Hidden	Hidden
.visible-sm-*	Hidden	Visible	Hidden	Hidden
.visible-md-*	Hidden	Hidden	Visible	Hidden
.visible-lg-*	Hidden	Hidden	Hidden	Visible
.hidden-xs	Hidden	Visible	Visible	Visible
.hidden-sm	Visible	Hidden	Visible	Visible
.hidden-md	Visible	Visible	Hidden	Visible
.hidden-lg	Visible	Visible	Visible	Hidden

Tabulka 3: Přehled responzivních utilit.

Ve verzi 3.2.0 tyto třídy mají dále pro každý breakpoint nastavenou rozdílnou CSS vlastnost display, tím se stávají výše uvedené třídy zastaralými:

- .visible-*-block - display: block;
- .visible-*-inline - display: inline;
- .visible-*-inline-block - display: inline-block;

Další pomocnou třídou jsou třídy tiskové:

Třídy	Prohlížeč	Tisk
.visible-print-block **.visible-print-inline** **.visible-print- inline-block**	skryté	viditelné
.hidden-print	Visible	hidden

Tabulka 4: Tiskové třídy.

6.2.3.9 Využití preprocesoru LESS v Bootstrapu

- Kompilace a použití – Použití preprocesoru LESS dělá Bootstrap použitelnějším a efektivnějším nástrojem. Pokud potřebujeme napsat sofistikovanější úpravy, je vhodné zvolit tuto variantu. Bootstrap podporuje task runner – automatizační nástroj Grunt, kterému je věnovaná samostatná kapitola. V této kapitole není nutná znalost problematiky task runnerů. Postačí nám informace, že LESS soubory mohou být kompilovány s každou změnou do CSS. Zpětný postup kompilace není možný, proto je nutné provádět změny v souborech LESS a ne CSS. Varianta LESS navíc umožňuje využití některých již předdefinovaných proměnných a mixinů[14].
- Proměnné – barvy, mezery mezi sloupci, velikosti písma a podobně.
    - Barvy – Lze využívat dvou schémat
        - Stupně šedi – umožňují rychlý přístup k běžným barvám odstínu šedé:
            - @gray-darker: lighten(#000, 13.5%); // #222,
            - @gray-dark: lighten(#000, 20%); // #333,
        - Sémantická schémata – obdobné jako kontextuální třídy:
            - @brand-primary: darken(#428bca, 6.5%); // #337ab7
    - Základní proměnné pro rychlé použití:
        - @body-bg: #fff, – barva pozadí

[14] Mixin – preprocesory umožňují zanořování CSS vlastností

        - @text-color: @black-50 – barva fontu.
    - Odkazy:
        - @link-color,
        - @brand-primary,
        - @link-hover-color: darken (@link-color, 15%).
    - Typografie:
        - @font-size-base: 14px,
        - @font-size-large: ceil((@font-size-base * 1.25)); // ~18px[15]
    - Komponenty – tyto proměnné lze využívat pro nastavení běžných hodnot:
        - @padding-base-vertical: 6px,
        - @padding-base-horizontal: 12px.
- Mixiny: Díky nim lze snadno využívat některé CSS vlastnosti mnohem rychlejší cestou. Jsou již napsané, v případě použití stačí napsat jen název mixinu a hodnotu, kterou chceme použít. Ukázku uvádím v příkladu s kulatými rohy:

  Příklad:

```
.border-top-radius(@radius){
      border-top-right-radius: @radius;
      border-top-left-radius: @radius;
}
```

Analogickým postupem lze použít transformace, animace, přechody a podobně.

6.2.4 Javascript

Bootstrap obsahuje Javascriptové pluginy – moduly, které umožňují dynamické chování jednotlivých UI komponent. Detailně jsou popsány v následujících kapitolách. Využívají JQuery pluginů.

6.2.4.1 Připojení souborů

Tyto pluginy mohou být použity dvěma způsoby:

- Připojením jednotlivých Javascriptových souborů – v závislosti na tom, jaký plugin zrovna potřebujeme:
    - alert.js,

[15] Ceil() – funkce pro zaokrouhlování v integeru

    - button.js,
    - a podobně.
- Připojením souboru bootstrap.js nebo zkompilovaného souboru bootstrap.min.js, který je minifikovaný. Oba v sobě zahrnují ostatní pluginy, jejichž funkčnost může být závislá na jiných. Proto je důležité kontrolovat jejich vzájemnou provázanost. Všechny jsou také závislé na JQuery pluginu, ten musí být vždy importován před Javascriptovými pluginy.

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

- Lze také použít import z CDN serveru

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquer
y.min.js"></script>
```

Použití Javascriptových pluginů zajistíme použitím data-atributů u jednotlivých komponent (popsány v další kapitole). [9]

6.2.4.2 Přehled pluginů

Uvádím zde souhrnný přehled základních Javascriptových pluginů:

- Modal.js – jedná se o takzvané pop up boxy s jednoduchou a efektivní grafikou. Jsou flexibilní. V překladu se jedná o „vyskakovací“ okna. Používají se pro různá hlášení stavů na stránce, potvrzení vstupu na stránku a podobně. Defaultně lze v jeden okamžik vyvolat pouze jedno okno.
- Dropdown.js – vysouvací menu.
- Scrollspy.js – dynamický obsah se signalizací kde se aktuálně nacházíme v rámci stránky. Pokud například stránku posuneme na sekci kontakty, zobrazí se v navigační liště záložka kontakty jako aktivní.
- Tab.js – dynamické chování tlačítek sloužících k přepínání obsahu. V principu jsou to dynamické přepínatelné karty /záložky.
- Tooltip.js – dynamické popisky.
- Popover.js – Podobné tooltipu. Pop-up okno, které se objeví při uživatelské akci, například po kliknutí na nějaké informační tlačítko.
- Alert.js – dynamické chování stavů (varovných, informačních a podobně).
- Collapse.js – dynamické chování obsahu (skrývání obsahu, přechody, zobrazování skrytého obsahu).
- Carousel.js – slideshow – cyklické přepínání obrázků nebo jejich náhledů.

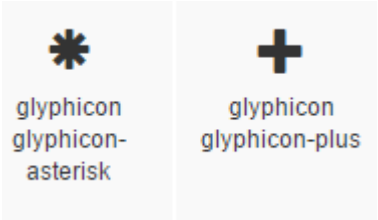
6.2.5 Komponenty

Bootstrap obsahuje mimo standardních CSS stylů ještě prvky uživatelského prostředí – znovupoužitelné komponenty (UI komponenty). Jsou obsaženy v samostatných LESS souborech. Pokud například potřebujeme responzivní rozbalovací menu fungující téměř ve všech prohlížečích, nemusíme od základů psát CSS styly a Javascripty. Jejich použití je jednoduché – stačí zkopírovat zdrojový kód. Detailní návod případně modifikovat tyto komponenty je na webových stránkách Bootstrapu. [9]

6.2.5.1 Přehled komponent

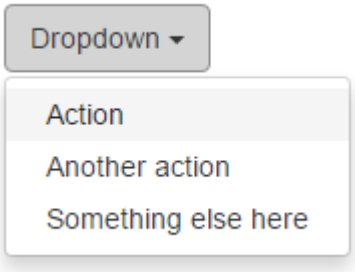
Tyto UI komponenty jsou stavební bloky využívající CSS / LESS stylů a Javascriptových pluginů. Díky této kombinaci se jedná o modulární blok, který může být interaktivního charakteru. Lze tak efektivně navrhnout prototyp nebo vytvořit základ pro webovou stránku bez nadměrného ladění kódu.
Přehled jednotlivých komponent:

- Glyphicons – k dispozici je přes 250 ikon ve formě piktogramů.



Obrázek 30: Ukázka piktogramů.

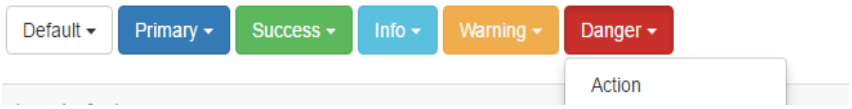
- Dropdown menu – vysouvací kontextové menu, které může obsahovat nějaký seznam, případně další vysouvací menu.



Obrázek 31: Dropdown menu v základním tvaru.

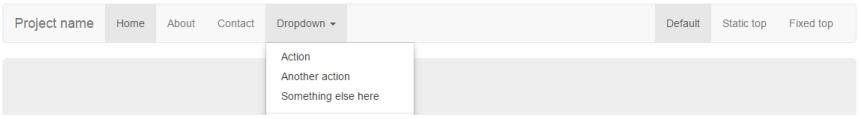
  - Potřebný plugin: dropdown.js
- Buttongroups – tato komponenta seskupí tlačítka dohromady do logického celku (jeden řádek).

- Button dropdowns – pomocí libovolného tlačítka lze vyvolat rozbalovací menu.



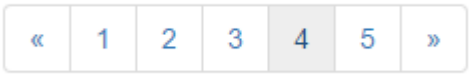
Obrázek 32: Dropdown tlačítka.

  - Potřebný plugin: dropdown.js.
- Input groups – rozšiřuje funkčnost formulářových elementů. Mezi dvě textová pole lze přidat symbol: @ (slouží k předvyplnění e-mailové adresy).
- Nav – jedná se o navigační komponentu, která zajistí ucelený vzhled ovládacích prvků.
- Navbar – jedná se o responzivní navigační panel. Defaultní navbar je na mobilních zařízeních (na malých obrazovkách skrytý – je přepínatelný). Na širokých obrazovkách se panel postupně horizontálně přizpůsobuje velikosti viewportu. [9]



Obrázek 33: Jednoduchý navbar.

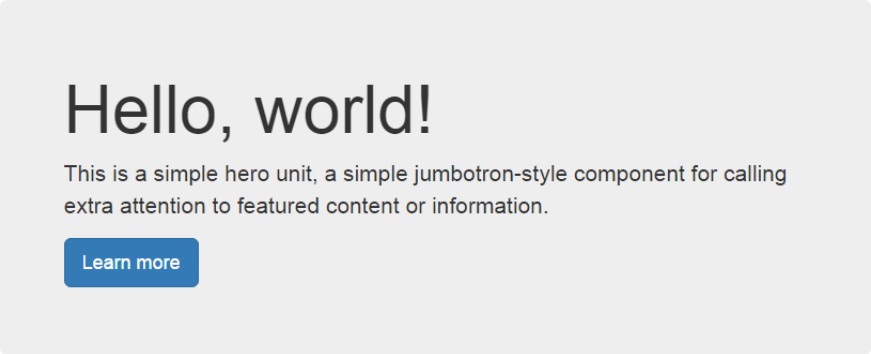
  - Potřebný plugin: dropdown.js, collapse.js
- Breadcrumbs – drobečková navigace.
- Pagination – tato komponenta slouží k stránkování – můžeme jí využít například při stránkování příspěvků.



Obrázek 34: Komponenta Pagination.

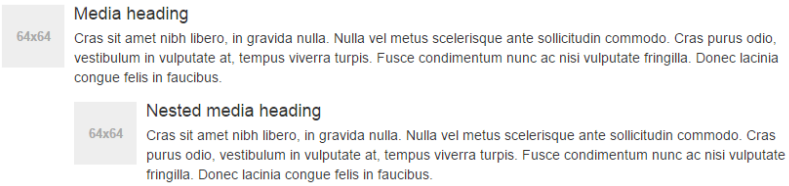
- Labels – v překladu se jedná o označení nebo značku. Label je v podstatě štítek sloužící ke zvýraznění určitého HTML elementu nebo jen textu. Lze jim přiřazovat barvy dle kontextu.
- Jumbotron – jedná se o komponentu, která může například zobrazovat důležitý obsah na webových stránkách.
- Badges – pomocí nich lze snadno například upozornit na nově příchozí zprávy v emailové schránce a podobně.

- Jumbotron – flexibilní komponenta, která opticky zvýrazní část stránky, do které potřebujeme umístit například klíčový obsah.



Obrázek 35: Komponenta Jumbotron.

- Page Header – vyčleněný prostor pro nadpis H1, který je umístěn do sekce <header>. Použitím této komponenty lze dosáhnout vzdušné koncepce.
- Thumbnails – tato komponenta nám zajistí jednoduché a efektivní rozmístění náhledů obrázků či videa v gridu.
- Togglable tabs – jedná se o přepínatelné položky. Jsou to jakési dynamické karty přepínající se v závislosti na menu.
- Tootips – jsou insiprovány podobným jQuery.tipsy pluginem. Jedná se o jakousi plovoucí nápovědu.
- Alerts – poskytuje zpětnou vazbu formou informačních panelů, které mají přiřazenou barvu pozadí v závislosti na kontextu.
- Progress bar – poskytuje informaci o aktuálním stavu dat. Tyto panely se používají například při zobrazení stavu instalace, kapacity datové schránky a podobně.
- Media object – obrázky k příspěvkům.



Obrázek 36: Komponenta Media object.

- List group – seznamy, které lze snadno kombinovat s ostatními třídami tak, aby vypadaly například jako seznamy tlačítek, badgetů a podobně.
- Panel – tuto komponentu lze využít například pro DOM model.

- Responsive embed – lze využít například pro responzivní chování videa. Lze například nastavit chování v poměru stran 16:9 případně 4:3. [9]

6.2.6 Šablony

K dispozici je několik šablon – templaty. Ty lze snadno aplikovat. Pokud potřebujeme základní layout celého webu, můžeme je použít. Nalezneme zde různé prefabrikáty zdrojových kódů například:

- Grid,
- Menu,
- Jumbotron,
- Kompletní rozmístění templatu a podobně.

6.2.7 Závěr Bootstrap

Bootstrap je jeden z nejvhodnějších frameworků pro nasazení do jakéhokoliv projektu. Ať už se jedná o tvorbu prototypu, prezentačního webu, nebo použití pro rozhraní sloužící k editaci, Bootstrap poskytuje v tomto ohledu komplexní řešení ze strany frontendu. Díky MIT je vhodným kandidátem i pro malá začínající vývojářská studia. [6]

Výhody:

- Licence MIT,
- výborná dokumentace,
- podpora preprocesorů,
- UI komponenty,
- JS moduly – dynamika komponent,
- podpora task runneru Grunt.

Nevýhody:

- verze 3.3.6 podpora LESS („nevýhoda“ pro uživatele SASS),
    - od verze 4 podpora SASS.
- Pokud je vyžadována prostá funkcionalita komponent – rozsáhlejší úpravy kódu.

6.3 Foundation

Jedná se o další velice propracovaný frontendový framework od pracovní skupiny ZURB. Číslo aktuální verze je 6. Na rozdíl od současné verze Bootstrapu (číslo 3), používá preprocesor SASS. Princip používání Foundationu je analogický jako u přechozího frameworku – skládá se z jednotlivých komponent, které jsou reprezentovány SASS soubory. Obsah lze kódovat prostřednictvím CSS stylů nebo můžeme použít preprocesory. Tento framework je sofistikovanější. Další rozdíl je z hlediska rozdělení. Foundation lze rozdělit podle typu projektu:

- Foundation for Sites – responzivní webové stránky,
- Foundation for Emails – responzivní emaily,
- Foundation for Apps – responzivní aplikace (využití Angularu a Flexboxu).

Stejně jako Bootstrap je vhodný k tvorbě prototypů nebo komplexních projektů. Podporuje vývoj pomocí techniky mobile-first. Pracuje s HTML, CSS a Javascriptem. Díky využívání jQuery, dalšího HTML5 frameworku Boilerplate, návrhových vzorů a dalších technologií se jedná o velice propracovaný projekt. [10]

6.3.1 Foundation for sites

Tato verze nám umožní jak rychlé prototypování, tak kompletní tvorbu webových stránek. Toho lze snadno dosáhnout díky jednoduchým flexibilním stavebním blokům. Jejich používání je analogické jako v Bootstrapu (UI komponenty). Oproti předchozí verzi číslo 5 je kód zredukován přibližně o 50 %.

Všechny komponenty používají ARIA atributy. Díky nim dokáží asistivní technologie korektně zobrazovat web na většině zařízení. Tyto technologie jsou využívány handicapovanými návštěvníky webů (hlasové čtečky a podobně). Foundation disponuje aplikací Notable code – jedná se o utilitu, která umožňuje kontrolu, zda je náš web responzivní.

Obsahuje také knihovnu Motion UI, která je napsaná v preprocesoru SASS. Ta zajišťuje efektivní a snadnou práci s animacemi a transformacemi. Obsahuje třídy přímo určené k tomuto účelu. Díky SASS mixinům je možné tyto třídy jednoduše modifikovat.

Verze 6 přináší i nové responzivní šablony. Díky nim můžeme rychle a efektivně navrhnout layout. Nemusíme tak vymýšlet základní kostru webu. Novinkou je

vylepšený nástroj CLI (command line tool), který umožňuje pohodlné nastavení některých specifikací.

Dalším propracovaným nástrojem je Yeti Launch, který slouží k automatizaci práce na projektu. Odpadá tak nutnost používání příkazů prostřednictvím příkazové řádky. Disponuje správou projektů. Lze se mezi nimi snadno přepínat, pokud máme například projekt pro webové stránky, emaily či aplikaci. Zatím je vydaná verze pro OS X. Jeho prostřednictvím lze snadno a rychle nasdílet projekty, pokud je například potřebujeme kooperovat. [10]

6.3.1.1 Přehled variant ke stažení

Před stažením frameworku si lze vybrat z několika instalačních balíků:

- Complete – kompletní balík.
- Essential – odlehčený balík (typografie, grid, základní responzivní CSS), velikostně menší přibližně o 40 kB.
- Custom – je možné si nastavit, co budeme stahovat (grid, navigace, barvy a podobně).
- SASS – pracuje na rozdíl od Bootstrapu se soubory SCSS.

6.3.1.2 Instalace

V případě, že využíváme SASS preprocesor, je postup instalace odlišný. Můžeme volit z několika variant jak Foundation nainstalovat.

- Prostřednictvím GUI (aplikace Yeti launch) – tato varianta je zatím přístupná jen pro systémy OS X.
- Prostřednictvím NPM – pro instalaci použijeme příkazový řádek, kde je postup instalace následující:
    1. Zadáme příkaz pro instalaci Foundationu:
       `npm install --global foundation-cli`
    2. Zadáme příkaz pro vytvoření projektu:
       `Foundation new`
       Zde budeme dotázáni na několik vstupní informací:
        - Výběr verze Foundation (Sites, Apps, Emails)
            - Vybereme Sites
        - Název projektu

        - Zvolíme název projektu, kde budou obsaženy všechny soubory.
    - Výběr varianty:
        - Basic – kompilace SASS preprocesoru.
        - ZURB – kompilace SASS / Javascript – tato instalace trvá mnohem díle než varianta Basic.
    - Vytvoří se projekt se stromovou strukturou, jejíž součástí jsou SASS soubory.
3. Posledním krokem je spouštění Gulpfilu příkazem: npm start.
    - Nyní se vytvoří složka s CSS soubory

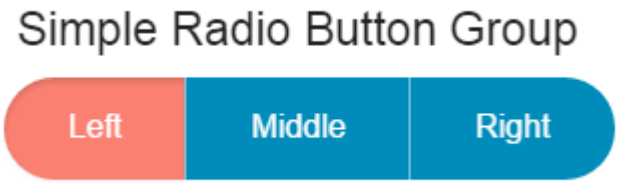
6.3.1.3 UI komponenty – Building blocks

Zurb building block je návrhová knihovna obsahující UI komponenty frameworku Foundation. V roce 1998 byla vytvořena a od té doby se razantně naplnila. Obsahuje stavební bloky například:

- Dropdown menu,
- Header submenu,
- Hover efekty – přechody a animace,
- Nastylovaná tlačítka,
- Navigační lišty,
- A další.

V této knihovně je vždy zobrazena daná komponenta včetně jejích zdrojových kódů. Můžeme si zde vyzkoušet i její funkčnost.

Na obrázku č. 37 uvádím příklad jednoduché komponenty: radio button group selection. Ostatní komponenty jsou zobrazeny na webu bakalářské práce. [10]



Obrázek 37: UI komponenta radio button group selection.

6.3.1.4 Dokumentace

V této části nalezneme kompletní návod, jak pracovat s frameworkem. Dokumentace je rozdělena na jednotlivé sekce. Od instalace až po používání SASS mixinů či knihovnu Motion UI.

- Instalace – lze ji provést několika způsoby v závislosti na tom, jakým způsobem budeme Foundation používat:
    - Pomocí nástroje Yeti Launch – varianta SASS.
    - Nástrojem CLI (příkazová řádka) – varianta SASS.
    - Stažením CSS – varianta CSS.
    - CDN odkazy – online knihovny.
- Kitchen sink – všechny komponenty jsou dostupné na jedné stránce včetně jejich zdrojových kódů.
- Veškeré komponenty jsou podobné jako v Boostrapu. Jsou například využívány kontextuální třídy, dropdown menu a podobně. Liší se grafickým designem a zápisem kódu.
- Detailní přehled komponent a jejich způsob použití je dostupný na tomto odkazu: Foundation for sites - dokumentace. [10]

6.3.2 Foundation for emails

Tato verze slouží primárně k tvorbě responzivních emailů. Ty lze snadno a rychle zobrazit na mnoha zařízeních. Pro jejich tvorbu se používají podobná pravidla jako pro webové stránky. Hlavní rozdíl je v principu psaní kódu. Emaily se sestavují pomocí tabulek a kaskádové styly se nepřipojují externím způsobem.

Pomocí gridu a responzivních bloků lez docílit vhodného řešení – pokud například potřebujeme poslat prostřednictvím emailu obrázky. K dispozici jsou také UI komponenty[16]:

- Tlačítka,
- Navigační panel a podobně.

6.3.3 Responsive Email inliner

Na webových stránkách Foundationu je k dispozici nástroj, díky kterému si můžeme otestovat responzivitu a korektní chování emailů. Jeho prostřednictvím můžeme

[16] Prvky uživatelského prostředí

email odeslat přímo ze stránek. Zjistíme tak například funkčnost v určitých emailových klientech. Postup testování je velice jednoduchý:

1) Vložíme kód, který chceme otestovat.
2) Konvertujeme email.

6.3.3.1 Šablony

K dispozici máme čtyři šablony pro ideální rozložení emailů.

- Basic – základní šablony.
- Hero – k dispozici je blok pro obrázek.
- Sidebar – obsahuje postranní panel.
- Sidebar Hero – kombinace druhého a třetího typu.
- A další

Lze je stáhnout v jednom souboru. K dispozici je soubor ink.css, který připojíme externím způsobem, slouží ale jen pro testování.

6.3.3.2 Dokumentace

V kompletní dokumentaci nalezneme návod pro tvorbu emailů. Jsou zde popsány jednotlivé CSS třídy, které lze použít. Nalezneme zde přehled použitelných UI komponent, nastavení gridu a podobně. Je zde zobrazena základní kostra kódu. Zde uvádím odkaz na tuto dokumentaci:

- Dokumentace pro Foundation for emails

6.3.4 Foundation for Apps

Tato verze je novinkou ve frontendových frameworcích. Umožňuje vytvářet layout responzivní aplikace. K dispozici je verze 1.1.0. K psaní kódu aplikace je využíván SASS, Motion UI, Angluar, JS moduly a GULP. [10]

6.3.4.1 Šablony

Foundation for Apps nám umožňuje vytvořit typově 5 různých aplikací:

- Emailový klient – aplikace se základním třísloupcovým layoutem. Zprávy lze na mobilním zařízení přesouvat mezi jednotlivými sloupci gestem.
- Chat – interaktivní chatová aplikace.
- Aplikace pracující s muzikou – lze napsat stream aplikace.

- Aplikace pro cestování – díky responzivitě lze skrýt nepotřebný obsah na zařízeních s malou obrazovkou.
- Marketingová aplikace – je možné napodobit povahu marketingových aplikací.

6.3.5 Závěr Foundation

Tak jako Bootstrap je Foundation vhodným kandidátem pro využití k tvorbě prototypů i kompletních projektů – weby, aplikace a podobně. Díky vynikající a přehledné dokumentaci je snadné se s tímto frameworkem poměrně rychle naučit. Licence je MIT. Navíc je zde vynikající podpora preprocesoru (SASS) a různých užitečných nástrojů – Grunt, Gulp.

Výhody:

- Licence MIT,
- UI komponenty,
- JS moduly,
- přehledná dokumentace,
- podpora SASS,
- v určitých případech – omezená funkcionalita některých UI komponent.

Nevýhody:

- v některých případech – omezená funkcionalita UI některých komponent (pokud bychom vzali kód z dokumentace, může to být jen kostra bez dynamické povahy kódu). [10]

6.4 Ostatní frameworky

V následující kapitole představím jedny z dalších frameworků patřících do kategorie frontendu. Nebudou zde již popsány tak podrobně jako předchozí dva vzhledem k analogickému způsobu používání.

6.4.1 HTML5 Boilerplate

Obsahuje několik nástrojů pro vývoj webových stránek. Prošel již také dlouholetým vývojem, ve kterém bylo provedeno od počátku několik změn za účelem co nejefektivnějšího výsledku. Za zmínku stojí soubor Normalize.css, který resetuje CSS nastavení pro zobrazení v různých prohlížečích – docílíme tak stejného

výchozího vzhledu ve většině moderních prohlížečů. Jedná se o novější alternativu k souboru reset.css. Dále používá Javascriptovou knihovnu Modernizr pro testování kódu v prohlížečích.

Instalace je jednoduchá: stačí stáhnout HTML5 Boilerplate z oficiálních stránek nebo z repozitáře na GitHubu, zkopírovat do projektu a začít jej používat Následující seznam uvádí podporu jednotlivých prohlížečů:

- Chrome – poslední dvě verze,
- Edge – poslední dvě verze,
- Firefox – poslední dvě verze,
- Internet Explorer – od verze 8 (včetně),
- Opera – poslední dvě verze,
- Safari – poslední dvě verze.

Obsahuje minifikované soubory, JQuery knihovny pro dynamické chování. Nalezneme zde také užitečné CSS třídy:

- Tiskové třídy.
- Třídy pro skrývání obsahu.
- Třídy pro reset obtékání.
- @media-queries s definovanými brekapointy – podpora mobile first.
- Základní nastavení typografie a další. [11]

6.4.2 Semantic UI

Tento framework, jak lze již odvodit z názvu, používá odlišné názvy tříd pro stylování dokumentu než ostatní konkurenti. Názvy jednotlivých tříd jsou sémanticky označeny. Například oproti Bootstrapu je označení tříd mnohem jasnější, jak ukazují následující bloky kódu:

Zápis v Bootstrapu:

```
<div class="row">
      <div class="col-lg-4">1</div>
      <div class="col-lg-4">2</div>
      <div class="col-lg-4">3</div>
</div>
```

Zápis v Semantic UI:

```
<div class="ui three column grid">
  <div class="column">1</div>
```

```
  <div class="column">2</div>
  <div class="column">3</div>
</div>
```

Semantic UI obsahuje několik šablon, které můžeme okamžitě začít využívat. Ty jsou rozděleny podle typu:

- Starter – na těchto šablonách je znázorněno chování jednotlivých komponent.
- Pages – šablony pro webové stránky. Nalezneme zde například: homepage, formulář pro přihlášení nebo stránku s fixním menu.

Dále obsahuje UI Komponenty, které lze umístit do gridu, který je složen z 16 sloupců. Dynamické chování zajišťují jednotlivé JS moduly. Obsahuje poslední verzi souboru Normalize.css.

Semantic UI má snadnou instalaci – stejně jako předchozí CSS varianty. K dispozici je také varianta s preprocesorem LESS. V tomto případě využíváme pro práci s tímto frameworkem Grunt. Jak CSS styly, tak Javascriptové soubory lze již defaultně ve stažené verzi používat prostřednictvím minifikovaných souborů.
Tento framework je velice propracovaný. Nabízí více nastavení než HTML Boilerplate. Je podobného charakteru jako Bootstrap a Foundation. [12]

6.4.3 Srovnání HTML 5 frameworků

V této kapitole jsou umístěny výsledky, v nichž jsou porovnány výše popsané frameworky. Hodnocení je provedeno v několika fázích:

- Srovnání vlastností – výstup ANO /NE
- Škálovatelné porovnání vlastností – výstup nabývá hodnot 1 až 5. Hodnota 1 reprezentuje nejpříznivější výsledek, 5 naopak nejméně vhodné řešení.

6.4.4 Kritéria pro hodnocení frameworků

Do tabulky jsou pro srovnání přidány další frontendové frameworky. Hlavními kritérii pro hodnocení jsou:

- časová náročnost na vývoj určité části kódu,
- obtížnost z hlediska znalosti syntaxe,
- dostupnost a přehlednost dokumentace,
- podpora preprocesorů,
- podpora mobile first,
- custom mód – volba / nastavení,

- UI komponenty.

6.4.5 Výsledky frameworků

V této subkapitole uvádím konkrétní data. Celkem jsou zde tři tabulky. V tabulce č. 1 jsou frameworky srovnány dle obecných vlastností (výstup ANO / NE). Tabulka č. 2 reprezentuje výsledky frameworků nabývajících hodnot 1 až 5. Těmito hodnotami jsou hodnoceny jednotlivé důležité vlastnosti. [13, 14, 15]

	Bootstrap	Foundation	HTML5 Boilerplate	Semantic UI	Skeleton
Verze	3.3.6 (4.0)	6	5.3.0	2.1.8	2.0.1
Licence	MIT	MIT	MIT	MIT	MIT
Preprocesor	LESS(SASS)	SASS	LESS/SASS	LESSS	Ne
JS moduly	Ano	Ano	Ne	Ano	Ne
Grid	12	12	Ne	16	12
UI komponenty	Ano	Ano	Ne	Ano	Ano
Mobile first	Ano	Ano	Ne	Ano	Ano
Custom mód	Ano	Ano	Ne	Ne	Ne

Tabulka 5: Srovnání frameworků - obecné vlastnosti.

	Bootstrap	Foundation	HTML5 Boilerplate	Semantic UI	Skeleton
Dokumentace	2	1	4	2	2
UI komponenty – kvantita / kvalita	1	1	5	3	4
JS pluginy – kvantita / kvalita	1	1	5	3	5
Čas – vývoj	1	1	5	1	1

Tabulka 6: Srovnání vlastností – škála.

Celkové umístění	
4. místo	Foundation
5. místo	Bootstrap
6. místo	Semantic UI
7. místo	Skeleton
8. místo	HTML5 Boilerplate

Tabulka 7: Srovnání frameworků – umístění.

7 Task runnery

Pojem task runner lze přeložit jako buildovací nástroj. Název je odvozen od hlavního úkolu – sestavení aplikace. Task runnery pracují s pluginy (balíčky), které se pak pomocí konfiguračního souboru nastaví a následně spustí tak, aby automatizovaly procesy. Ty bychom museli v opačném případě manuálně provádět neustále dokola. Například kompilace LESS preprocesoru do CSS stylů: pokud například změníme během práce několikrát po sobě hodnotu LESS proměnné, nemusíme se dále starat o převod do CSS souboru – po každé změně se sám CSS soubor znovu uloží (Watch plugin).

Pracuje se s nimi přes příkazový řádek. Je třeba mít nainstalovaný balíčkovací systém Node Package Manager (NPM), který je součástí Javascriptové knihovny Node.js.[17]. Můžeme provádět například akce:

- Spojování souborů – ušetří nám počet HTTP požadavků na server. Pokud spojíme například 6 CSS souborů do jednoho, proběhne místo šesti pouze jeden HTTP požadavek. Na mobilním připojení bude tato redukce znatelná.
- Minifikaci souborů – ušetříme datovou velikost.
- Kompilaci preprocesorů (SASS / LESS / STYLUS).
- Doplňování prefixů v CSS kódu pro webové prohlížeče a další.

Princip funkce task runnerů je u všech v podstatě stejný – jsou definovány zdroje – soubory, v kterých jsou prováděny změny. Ty jsou v některých případech (kompilace LESS) sledovány pomocí pluginu Watch a následně se tyto změny okamžitě provádějí. Dále lze snadno pomocí jednoduchých příkazů spojit Javascriptové soubory do jednoho, minifikovat výsledné CSS soubory a podobně. Plugin Watch nemusíme používat pro všechny zdroje [16]

7.1 Grunt

Tento task runner provádí veškeré výše zmíněné úkony a mnoho dalších. Vznikl v roce 2012 a jeho autem je Ben Alman. Princip práce s Gruntem je v instalaci balíčků a jejich konfigurace v souboru Gruntfile.js. Těchto balíčků jsou stovky. Grunt od verze 0.4.x vyžaduje verzi Node.js minimálně 0.8.0 a vyšší. Pro úplnost, současná verze Node.js je 5.8. [17]

[17] Knihovna, která slouží pro práci s Javascriptem na serveru

7.1.1 Instalace

Před samotnou instalací je třeba zajistit, aby byla nainstalována aktuální verze Node.js knihovny. Případnou aktualizaci provedeme příkazem – veškerá komunikace probíhá přes příkazový řádek (CMD):

```
npm update –g npm.
```

Samotný Grunt se instaluje globálně, tím zajistíme, že jej lze používat ve všech projektech. Pluginy jsou pak instalovány lokálně – v každém projektu používáme jiné (někde potřebujeme Javascriptové soubory spojit, jinde zase spojit a minifikovat). Instalace Gruntu a pluginů sestává z následujících kroků:

1) Samotný Grunt nainstalujeme globálně příkazem:

   ```
   npm install –g grunt
   ```

   Modifikátor –g umožní budoucí spuštění Gruntu z jakéhokoliv adresáře (projektu).

2) Instalace pluginů provádíme jednotlivými příkazy, které lze nalézt v dokumentaci na oficiálních stránkách projektu Grunt. Abychom mohli zadávat příkazy pro instalaci balíčků, musíme se přepnout do kořenové složky projektu. Pluginů jsou stovky, uvádím zde jen pár vybraných – analogie instalace je u všech stejná:

- Sledování změn v souborech:

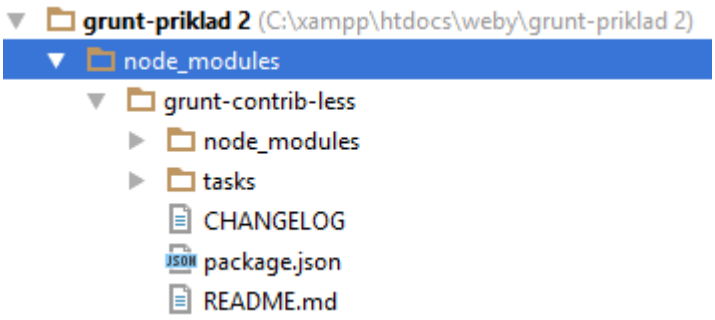
  ```
  npm install grunt-contrib-watch --save-dev
  ```

- Kompilace LESS do CSS

  ```
  npm install grunt-contrib-less --save-dev
  ```

- Příkazy jsou velmi podobné, liší se názvem balíčku. Struktura příkazu:
  - npm – použití Node Package Manager,
  - install – instalace,
  - grunt – použití Gruntu,
  - –contrib-watch – název balíčku,
  - –save-dev – instalace aktuální verze.

  Vytvoří se složka node_modules v které jsou nainstalovány jednotlivé pluginy.



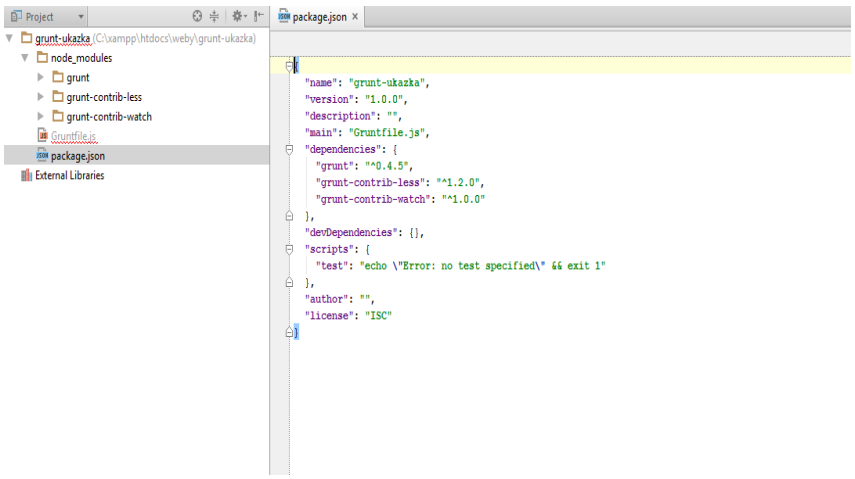
Obrázek 38: Struktura grunt - instalace pluginu.

3) Vytvoříme soubor package.json, kde jsou definovány závislosti jednotlivých pluginů (obdoba registrace), které jsme nainstalovali. Tento soubor v případě exportu projektu na GitHub a jeho opětovného použití v jiném projektu umožňuje jejich automatickou instalaci zadáním jednoho příkazu. Nemusíme tak instalovat opětovně pluginy jeden po druhém. Příkaz pro vytvoření souboru Package.json:

   ```
   npm init.
   ```

   Na následujícím obrázku je vidět ukázka souboru Package.json. Nainstalovány jsou v tomto konkrétním příkladu dva pluginy:

   - Contrib-less.
   - Contrib-watch.



Obrázek 39: Package.json.

4) Poslední fází je vytvoření souboru Gruntfile.js, v kterém se konkrétně definují jednotlivé pluginy (kompletní návod najdeme v dokumentaci na oficiálních stránkách).

7.1.2 Konfigurace úloh

Soubor Gruntfile.js slouží ke konfiguraci jednotlivých úloh. Vytvoříme ho stejně tak jako vytváříme například HTML soubor. Skládá se ze 3 částí:

1. Obalovací funkce – funkce pro nativní chování Gruntu.
   `Module.exports = function(grunt{/*konfigurace úlohy*/}`
2. Načtení pluginu – příkazem načteme plugin který jsme předtím nainstalovali.
   `grunt.loadNpmTasks('grunt-contrib-less');`
3. Registrace pluginu – definujeme, v kterém pořadí se budou spouštět:
   `grunt.registerTask('default', [less']);`
   Jednoduchý Gruntfile.js vypadá následovně:

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

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

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

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

};
```

Obrázek 40: Jedna ze základních konfigurací souboru Gruntfile.js.

U konfigurací lze vytvářet další konfigurace (podkonfigurace). Toho docílíme použitím nativních příkazů:

- `Foo`
- `Bar`

```
grunt.initConfig({
    concat: {
        options: {
            // Task-level options may go here, overriding task defaults.
        },
        foo: {
            options: {
                // "foo" target options may go here, overriding task-level options.
            },
        },
        bar: {
            // No options specified; this target will use task-level options.
        },
    },
});
```

Obrázek 41: Konfigurace úloh.

Princip konfigurace:

1) Budeme uvažovat dva projekty.
2) Na GitHubu máme vyexportovaný package.json, vytvořený z prvního projektu, kde jsou uvedené závislosti. Máme zde také uložený Gruntfile.js s výše uvedenou kostrou konfigurace.
3) Gruntfilu.js obsahuje úlohy:
    - concat – spojování souborů,
    - uglify – minifikace souborů.

   Ty obsahují dále konfigurace:
    - Foo – zde mohou být uvedeny například Javascriptové soubory, které potřebujeme spojit v rámci nahrávání ostré verze projektu na web.
    - Bar – zde mohou být umístěny také soubory, které potřebujeme spojit pro práci na lokální pracovní verzi.

7.1.3 Závěr Grunt

Grunt je skvělý nástroj pro automatizaci. Má ovšem také své nevýhody. Jsme odkázáni na dokumentaci a na předdefinované konfigurace úloh. Dalším problémem může být u tohoto nástroje rychlost. Tento faktor je negativně ovlivňován ukládáním mezivýsledků do filesystému u větších projektů.

7.2 Gulp

Dalším propracovaným nástrojem pro automatizaci je Gulp, který vznikl v roce 2013 pod hlavičkou společnosti Fractal. Autorem je Eric Schoffstall. Princip práce je oproti nástroji Grunt rozdílný.

Gulp využívá Node.js streamu, to znamená, že mezivýsledky si ukládá do paměti – úlohy se provádí paralelně a tím i rychleji. Práce s ním probíhá spíše pomocí programování než přes konfigurování. Díky tomu nejsme omezení na dokumentaci. [18]

7.2.1 Instalace

Stejně tak jako v přechozím případě se Gulp instaluje přes balíčkovací systém NPM. Postup instalace:

1) Otevřeme příkazový řádek a zadáme příkaz pro instalaci (instalace globální):
   `npm install --global gulp-cli`
2) Vytvoříme projekt a v příkazovém řádku se do něj přepneme (nutná lokální instalace):
   `C:/xampp/htdocs/weby/gulp-projekt`
3) Dále vytvoříme soubor Package.json, který bude obsahovat metadata o projektu a závislosti.
4) Dalším krokem je zanesení Gulpu do projektu, který máme již vytvořený dle výše uvedeného postupu. Tuto operaci provedeme příkazem:
   `npm install gulp --save-dev`
5) Do projektu přidáme plugin browser-sync:
   `Npm install browser-sync --save-dev`
6) Na stejné adresářové úrovni, kde máme vytvořený package.json, vytvoříme soubor Gulpfile.js, kde budeme mít veškerou konfiguraci úloh.

7.2.2 Konfigurace úloh

Stejně tak jako v předchozím případě, má i tento nástroj svůj konfigurační soubor Gulpfile.js. Zde programováním přizpůsobíme jednotlivé gulp pluginy. V následujícím seznamu uvádím některé z nich:

- gulp-concat – spojování souborů,
- gulp-cleancss – minifikace CSS souborů,
- gulp sass – kompilace SASS do CSS a další.

7.2.3 Závěr Gulp

Gulp je skvělým nástrojem, který podporuje kompletní automatizaci úloh, které bychom museli opakovaně spouštět. Stejně tak jako Grunt obsahuje nespočet pluginů, které lze nastavit dle vlastní potřeby. Gulp díky tomu že využívá streamů (pracuje paralelním způsobem), je trochu rychlejší v provádění jednotlivých úloh než Grunt.

7.3 Srovnání Task Runnerů

V této kapitole uvádím tabulku, kde jsou porovnány výše popsané task tunnery. Srovnání je rozděleno na dvě fáze:

1) Obecný přehled – rok vzniku, počet pluginů a podobně, přidán navíc nástroj Broccoli
2) Srovnání časových údajů pro provedení úlohy na totožných souborech
    a. Minifikace CSS souboru – CSS min
    b. Watch – kompilace LESS preprocesoru – less test.

Tyto výsledky nelze určit se stoprocentně, závisí například na kvalitě napsaných pluginů. [15]

Parametr	Grunt	Gulp	Broccoli
Oficiální stránky	http://gulpjs.com/	http://gruntjs.com/	http://broccolijs.com/
Rok vzniku	2012	2013	2014
Počet Pluginů	5580	2266	-
Licence	MIT	MIT	MIT
Github stars	10 748	20 700	2 747

Tabulka 8: Srovnání Task runnerů – obecná data.

Parametr	Grunt	Gulp
Dat. Velikost původní (CSS – min)	144,67 kB	144,67 kB
Dat. Velikost výsledná (CSS – min)	116,17 k B	116,40 kB
Čas – watch / less	5,7 s	2,3 s

Tabulka 9: Srovnání Task runnerů - konkrétní data

Srovnat tyto dva nástroje je složité. Nelze přesně stanovit, který je lepší a který naopak horší. Grunt používá konfigurace pluginů a díky dokumentaci lze nakonfigurovat téměř vše, co budeme k práci potřebovat. Pracuje sekvenčním způsobem – výsledky si ukládá do mezipaměti. Tato operace nepatrně zpomalí čas provedení úlohy.

Gulp je oproti Gruntu využíván spíše skrz programování. Pracuje s vlákny paralelním způsobem, tím pádem je práce rychlejší. Díky programování nejsme závislí na dokumentaci. Oba nástroje jsou vynikající volbou pro automatizaci.

8 Workflow pro design

Jednotlivé kroky procesu nejsou tak důležité jako potřeba dodržovat následující klíčové koncepce:

- Web je interaktivní médium – veškeré nástroje a postupy na to musí brát ohled a zřetel.
- Nutná je kooperace v průběhu procesu (například grafik a frontend designér).
- Musíme koukat na projekt jako na systém, nikoliv jako na webovou statickou stránku.

8.1 Web je interaktivní médium

Web není statické médium – je flexibilní. V současné době se stále ještě v některých případech přístup k designu webu spíše podobá designu pro tisk. V potaz se musí brát skutečnost, že lidé s webovými stránkami komunikují, některé neslouží jen k prohlížení. Někteří designéři přirovnávají web k „živému dýchajícímu plátnu“. Z toho plyne, že prostřednictvím našich vývojových nástrojů a technik musíme lépe brát v úvahu dynamickou povahu webu jako takového. [1]

8.2 Kooperace

Převážná část vývoje spíše směřuje lineární cestou. To znamená, že grafik nejprve web navrhne pomocí grafických nástrojů a předá jej vývojáři jako statický prototyp – například v PNG formátu. V tomto procesu se ztrácejí určité důležité informace (chybové stavy, stavy kurzoru – hover atd.), otevřená / zavřená navigace a podobně. Tato klíčová fakta se při předávání informací někdy ztrácí. Spolupráce nám zajišťuje, že neopomeneme důležité věci. Ethan Marcotte je zastáncem jisté teorie, která říká, že vývojáři a grafici by měli kooperovat v jedné pracovní skupině. Pokud je vytvořená tato pracovní skupina, vývojář může rychle interagovat na jednotlivé scénáře.

Příklad:

- Co se stane, když kliknu na toto tlačítko?
- Jaká animace nastane, pokud kliknu zde?
- A podobně.

Obecně můžeme říci že: spolupracující workflow (collaborative workflow) vede k inovativním řešením. [1]

8.3 Komunikace

Grafici a vývojáři by měli být zapojeni do raných fází projektu – první meetingy. Během nich by se měl web prohlížet na co největším počtu zařízení s různou velikostí obrazovky a různým rozlišením. Mohou nastat důležité otázky:

- Jsou místa určená pro dotyk na obrazovce smartphonu dostatečně veliká?
- Při jakých velikostech a při jakém rozlišení se stává design webu nečitelným?

8.3.1 Iterace

Iterace nezbytným krokem při vývoji softwaru:

1) Navrhneme design.
2) Upravíme design podle kritérií.
3) Prověří / zkontrolují se změny a pokračuje se další iterací.

Přitom je důležité v každé iteraci provádět jen malé změny. Tyto drobné změny se mnohem lépe editují než rozsáhlé zásahy do projektu.

8.3.2 Prosazování responzivního webdesignu

Důležité je klienty seznámit s přednostmi, které přináší responzivní webdesign. Dokonce i v případě, že projekt nemá být responzivní je vhodné klientům klást otázky typu:

- Jak se bude řešit design a chování v případě přístupu z mobilního zařízení.
- Co když určité zařízení nepodporuje konkrétní typ písma atd.

8.3.3 Přemýšlení o projektu jako o systému

Web musí být konzistentní a soudržný. Ne z hlediska stránky ke stránce, ale z hlediska od zařízení k zařízení. Aby bylo dosaženo této soudržnosti, je třeba přemýšlet o projektu jako o systému – rozdělit stránku na:

- headery,
- footery,
- menu,
- a podobně.

O výše uvedených blocích bychom měli přemýšlet jako o samostatných komponentách. Tento přístup pomáhá udržovat konzistenci webu.

8.3.4 Brát v potaz mobilní zařízení

Průkopníkem techniky mobile first je Luke Wroblewski. Uvádí tři důvody, proč by měly být vytvořeny nejprve mobilní prožitky z webu.

1) **Exploze mobilních zařízení**: Mobilní zařízení v současné době přibývají jak do kvality, tak do kvantity. Některé informace uvádějí nárůst až na 12 miliard mobilních zařízení v roce 2020. V současné době přibývá lidí, kteří přistupují na internet přes mobilní zařízení (smartphony, tablety, …).
2) **Design pro mobilní zařízení zaměřuje pozornost na podstatné**: Pokud máme například web společnosti, kde z jednotlivých oddělení chtějí všichni umístit informace na web, může tato varianta vést k layoutům, které obsahují zbytečné informace, díky nimž se stránka stává nepřehlednou – zbytečné záložky a podobně. Návštěvníka webu tím zahlcujeme. Na mobilním zařízení je tato varianta nepřípustná: menší obrazovka, kde není prostor pro tyto věci. Je tedy nutné zaměřit se na skutečně důležitý obsah.
3) **Mobilní zařízení rozšiřují schopnosti přístupu k webu**: V současnosti je přístup firem k webovým stránkám spíše brán jako přístup k nějaké omezené verzi desktopu. To je špatný způsob myšlení, můžeme využívat například:
    - geolokaci,
    - přepínání layouty v závislosti jak toto zařízení držíme,
    - podporu multidotykových rozhraní (vnímání více dotyků najednou).

    Tato zařízení poskytují často mnohem bohatší zážitky než desktopy. Mobilní zařízení by neměla být chápána jako jednodušší verze desktopu. [1]

9 Plánování

Plánování je klíčovým faktorem. Volba, aby byl projekt responzivní, záleží na několika faktorech. Je důležité si uvědomit, zda je responsivní design na projektu nutný, užitečný nebo zda je to od projektu očekáváno jen proto, že je to moderní trend.

9.1 Faktory ke zvážení

1) Výkon

Řešení, které je z hlediska výkonu dobré na jednom zařízení, může být omezeno použitím na zařízeních jiných – nelze jej použít. Typickým případem je optimalizace výkonu pro mobilní sítě. Na kabelovém připojení, WIFI a podobně je vhodné mít externí styly a skripty mimo zobrazovaný HTML dokument. Na mobilním zařízení je však díky pomalejšímu připojení vhodné někdy zakomponovat styly interně včetně skriptů, aby se snížil počet HTTP požadavků na server.

Obrázky zde hrají obrovskou roli. Uživatel musí kolikrát načítat různé velikosti obrázků, i když v onen okamžik potřebuje jen jeden typ rozlišení – na mobilním zařízení stahuje obrázky pro desktopovou verzi. Pokud se tento faktor nevezme v potaz, web bude přeplněný a pomalý, bude vykonávat mnohem více HTTP požadavků, což rapidně sníží výkon. [1]

2) Kontext

User experience budou na webu závislé na kontextu. Weby na mobilních zařízeních mohou být používány jinak než na desktopech. Například využití různých geolokačních služeb – weby nabízející rozdílné informace vzhledem k místu, kde se právě nacházíme a podobně.

3) Struktura webu

Pokud máme například dvousloupcový layout webu, na desktopu se nám zobrazuje standardně postranní panel, kde máme důležité informace. Pokud tento web zobrazíme na mobilním zařízení, kde se postranní sloupec zařadí pod primární, nebudeme mít hned na očích důležité informace, jako to bylo na desktopu.

4) Časové investice

Responzivní web je na přípravu časově náročnější. Je třeba zvážit, na kterých zařízení bude web zobrazován. Jaké zařízení vůbec má podporovat, jak budou

uživatelé komunikovat s webem prostřednictvím těchto zařízení. V budoucnu nám však RW ušetří čas (místo dvou a více CSS souborů nám stačí upravit pouze jeden).

5) Podpora

Budování webů technikou desktop first přináší problémy v podobě prohlížečů na mobilních zařízeních. Prohlížeče na desktopu mají jádro WebKit a díky němu podporují @media queries. To však nemusí platit pro mobilní prohlížeče. Proto je důležité nejprve navrhovat pro nejméně schopné prohlížeče.

9.2 Prototypy

Prototypy jsou opakem vytváření dokonalých designů, které jsou perfektní na pixelové úrovni. Jedná se o vytváření prototypů, kde si můžeme dovolit drobné nedokonalosti. Jsou sice nedokonalé, ale jsou flexibilní a efektivnější než dokonalé statické prototypy. Existuje totiž mnoho proměnných, které mohou kdykoliv narušit pixelovou perfektnost. Uživatel si může obsah přiblížit, oddálit. Někteří si prohlížejí obsah, aniž by měli maximalizované okno prohlížeče, používají libovolné prohlížeče atd.

- Statické prototypy – Typickým přístupem je grafický návrh v Adobe Photoshop. Výstupem je statický prototyp, který je předveden klientovi, následně se provede revize, která se zpětně dělá v grafickém programu. Jakmile je design hotov, předá se grafická podoba kodérům. Tento postup není zcela vhodný z důvodu eliminace dynamické povahy flexibilního média – webu. Při používání statických prototypů může mít zákazník mylnou představu o tom, jak bude vypadat výsledný web. Nelze předvést, jak se bude web chovat na jiných velikostech obrazovek (pokud nejsou hotové další návrhy v PS). Problém nastává, pokud komunikujeme s klientem a předáme mu dokonalý statický prototyp – v jiných prohlížečích však vypadá design odlišně. To pak může způsobit, že je klient nespokojen. Designér předává vývojáři statický prototyp – neříká nic o komunikaci mezi objekty na webu.
- Dynamické prototypy – ideální prostředí pro navrhování je prohlížeč. Zde bude web fungovat. Maximálně se eliminují problémové záležitosti. Díky živému prototypu můžeme předvést, jak se bude web chovat na různých zařízeních a různě velkých obrazovkách. Web je totiž interaktivní médium.

9.3 Responzivní obsah

Před budováním samotného webu je důležité znát obsah nebo alespoň část obsahu webu. Důležité je mít připravenou alespoň částečnou strukturu webu. Primární otázkou je, jaký typ obsahu bude web zobrazovat (zpravodajský web, blog, a podobně). Všechny weby mají obsah, který by se měl určit do různých typů. Důležitý je také účel webu, struktura obsahu – například článek by měl mít nějaký titulek a perex, autora a datum publikace. Tyto fáze se určuje tzv. modelováním obsahu (content modeling). [2]

9.3.1 Modelování obsahu

Každý úsek textu by měl mít specifický obsah, text by měl být strukturován do samostatných entit. Obsah by se měl modelovat až poté co bude známa cílová skupina uživatelů komunikujících s obsahem.

Mnoho webů dnes obsahuje v mobilních verzích omezené množství informací s odkazem na zobrazení desktopové verze, což je špatné řešení. Jedním z možných správných řešení je zjednodušená verze, která se bude rychle načítat. Dříve se telefon používal v omezené míře, co se týče prohlížení webových stránek, dnes slouží k plnohodnotnému prohlížení webu. Pokud se nějaký web prohlíží na mobilním zařízení, měl by být jednoduchý a rychlý, ne však webem, který snižuje intelektuální úroveň. [2]

10 Praktická část

Praktická část je rozdělena na dvě části. První je internetový průvodce responzivním designem. Ten se dále dělí na několik sekcí:

- Popis RWD,
- popis frameworků,
- popis task runnerů,
- výsledky testů,
- postupy a techniky návrhu.

10.1 Internetový průvodce

Tento průvodce slouží jako podpora pro prezentaci bakalářské práce. Internetový manuál je umístěn na adrese: http://www.bp.jakubtetik.cz. Úvodní stránka obsahuje menu s jednotlivými sekcemi, které jsou popisovány v teoretické části BP.

Web je napsán pomocí frameworku Bootstrap. Na webu jsou prezentovány rozdíly mezi Bootstrapem a Foundationem. Jsou zde vyobrazeny UI komponenty včetně Javascriptových pluginů, které jednotlivé frameworky nabízejí včetně jejich zdrojových kódů.

Nejdůležitější část tohoto webu – výsledky obsahuje tabulky, kde jsou zobrazeny jednotlivé testy a srovnání. Jsou zde umístěny zároveň bloky kódu, na kterých bylo provedeno testování.

10.2 Web PF JCU – průvodce prváka

Tento web vznikl jako druhý zdroj pro porovnání. Je postaven na frameworku Foundation. Tento web je menšího rozsahu, není zde nutné tedy využívat automatizační nástroj. Jedná se o jednoduchý web obsahující základní užitečné informace pro nastupující studenty do prvních ročníků Pedagogické fakulty Jihočeské Univerzity v Českých Budějovicích. Mezi tyto informace jsou zahrnuty například informace o struktuře katedry, informace ohledně studia a další cenné rady. Web je umístěn na adrese: http://pruvodce.jakubtetik.cz/

11 Závěr

V závěru bych rád shrnul, čím se tato bakalářská práce zabývá a k jakým výsledkům jsem došel.

Cílem práce bylo provést komparaci HTML5 responzivních (frontend) frameworků a task runnerů. Dále jsem chtěl dokázat, že používání těchto nástrojů je mnohem efektivnější než používání samotného HTML s CSS. Komparace byla provedena mezi jednotlivými nástroji a dále s ručním vývojem.

V teoretické části jsem jako první provedl rozbor responzivního webového designu. Jsou zde představeny základní pilíře RWD: Fluidní mřížky, adaptivní obrázky a @media-queries. K jednotlivým pilířům jsou popsány doporučené rady ohledně používání. Dále jsou představeny frontend frameworky (detailně Bootstrap a Foundation). Provedl jsem analýzu jejich celkové koncepce: od instalace po představení UI komponent a jejich využití. Další obsáhlou kapitolou jsou task runnery – jejich využití, instalace a způsoby použití. V posledních dvou kapitolách teoretické části jsem představil workflow pro design a jeho plánování.

V praktické části jsem vytvořil dva responzivní weby. První z nich slouží jako internetový manuál k prezentaci bakalářské práce. Web se nachází na adrese: https://bp.jakubtetik.cz. Na tomto webu jsou shrnuty zásady pro budování RWD webů. Jsou zde popsány návody, jak začít využívat například Bootstrap v kombinaci s Gruntem. K dispozici jsou zdrojové kódy některých UI komponent včetně jejich prezentace. Umístil jsem zde výsledky jednotlivých testů, které byly provedeny dle zadání práce. Tento web je psaný sám o sobě v Bootstrapu. Výsledky dokazují, že ruční vývoj je krokem zpět oproti vývoji s pomocí frameworků či automatizovaných opakujících se úkolů.

Druhý web slouží jako průvodce pro studenty prvních ročníků Pedagocické fakulty Jihočeské Univerzity v Českých Budějovicích. Je napsán pomocí Foundationu. Web je umístěn na adrese: http://pruvodce.jakubtetik.cz/.

Zda si vybrat Foundation či Boostrap, Grunt či Gulp je spíš subjektivní pohled. Ve výsledku jsou srovnatelné. Na sto procent však mohu stanovit, že díky nim lze web vyvinout mnohonásobně rychleji.

12 Seznam použité literatury a zdrojů

[1] KADLEC, Tim. *Responzivní design profesionálně*. Brno: Zoner Press, 2014, 246 s. Encyklopedie Zoner Press. ISBN 978-80-7413-280-3.

[2] SHARKIE, Craig a Andrew FISHER. *Responzivní webdesign: okamžitě*. Brno: Computer Press, 2015. ISBN 978-80-251-4384-1.

[3] MARCOTTE, Ethan. Responsive Web Design · An A List Apart Article. In: *A List Apart: For People Who Make Websites* [online]. A List Apart, 2010 [cit. 2016-05-02]. Dostupné z: http://alistapart.com/article/responsive-web-design.

[4] Responzivní obrázky. *Vzhůru dolů — webový frontend ze všech stran* [online]. Martin Michálek, 2016 [cit. 2016-05-05]. Dostupné z: http://www.vzhurudolu.cz/prirucka/responzivni-obrazky

[5] Mylné představy o responzivním designu | Interval.cz. *Interval.cz | Svět Internetu, Technologií a Bezpečnosti* [online]. Brno: ZONER software, a.s, 2016 [cit. 2016-05-05]. Dostupné z: https://www.interval.cz/clanky/mylne-predstavy-o-responzivnim-designu/

[6] Vzhůru dolů - Jaké breakpointy zvolit v responzivním webdesignu? *Vzhůru dolů — webový frontend ze všech stran* [online]. Martin Michálek, 2016 [cit. 2016-05-05]. Dostupné z: http://kratce.vzhurudolu.cz/post/46416507703/jak%C3%A9-breakpointy-zvolit-v-responzivn%C3%ADm-webdesignu

[7] Proč (ne)použít Mobile First. *Je čas.cz – moderní tvorba webových stránek* [online]. Bohumil Jahoda, 2013 [cit. 2016-05-02]. Dostupné z: http://jecas.cz/mobile-first.

[8] K čemu je dobrý Bootstrap a frontend frameworky? - Zdroják. *Zdroják – o tvorbě webových stránek a aplikací* [online]. Praha 4: Devel.cz Labs, s.r.o., 2016 [cit. 2016-05-05]. Dostupné z: https://www.zdrojak.cz/clanky/k-cemu-je-dobry-bootstrap-frontend-frameworky/

[9] *Bootstrap The world's most popular mobile-first and responsive front-end framework.* [online]. Core team of Bootstrap, 2016 [cit. 2016-05-02]. Dostupné z: http://getbootstrap.com/

[10] *Foundation | The most advanced responsive front-end framework in the world.* [online]. ZURB, 2016 [cit. 2016-05-02]. Dostupné z: http://foundation.zurb.com/

[11] Html5-boilerplate/TOC.md at 5.3.0 · h5bp/html5-boilerplate · GitHub. *How people build software · GitHub* [online]. GitHub, 2016 [cit. 2016-05-02]. Dostupné z: https://github.com/h5bp/html5-boilerplate/blob/5.3.0/dist/doc/TOC.md

[12] *Semantic UI* [online]. 2016 [cit. 2016-05-05]. Dostupné z: http://semantic-ui.com/

[13] CSS Front-end Frameworks with comparison - By usabli.ca. *GitHub Pages - Websites for you and your projects, hosted directly from your GitHub repository. Just edit, push, and your changes are live.* [online]. 2016 [cit. 2016-05-02]. Dostupné z: http://usablica.github.io/front-end-frameworks/compare.html.

[14] Responsive CSS Framework Comparison: Bootstrap, Foundation, Skeleton. *Vermilion > Vermilion Design + Digital | Nonprofit + Natural Foods* [online]. Colorado: Vermilion Inc., 2015 [cit. 2016-05-02]. Dostupné z: http://responsive.vermilion.com/compare.php.

[15] *Skeleton: Responsive CSS Boilerplate* [online]. Dave Gamache [cit. 2016-05-02]. Dostupné z: http://getskeleton.com/

[16] Gulp vs. Grunt: souboj bez vítěze a poraženého - Zdroják. *Zdroják – o tvorbě webových stránek a aplikací*[online]. Praha 4: Devel.cz Labs, s.r.o., 2016 [cit. 2016-05-05]. Dostupné z: https://www.zdrojak.cz/clanky/gulp-vs-grunt-souboj-bez-viteze-a-porazeneho/

[17] *Grunt: The JavaScript Task Runner* [online]. Development Team of Grunt, 2012 [cit. 2016-05-02]. Dostupné z: http://gruntjs.com/

[18] Gulp/getting-started.md at master · gulpjs/gulp · GitHub. *How people build software · GitHub* [online]. GitHUb, 2016 [cit. 2016-05-02]. Dostupné z: https://github.com/gulpjs/gulp/blob/master/docs/getting-started.md

[19] VERNAJUNE, Stefano. BazaarJS: the Node.js task runner diaspora. In: *Web agency Ruby Rails full-stack developers* [online]. London: LEANPANDA, 2015 [cit. 2016-05-02]. Dostupné z: http://www.leanpanda.com/blog/2015/06/15/nodejs-grunt-gulp/.

13 Seznam obrázků

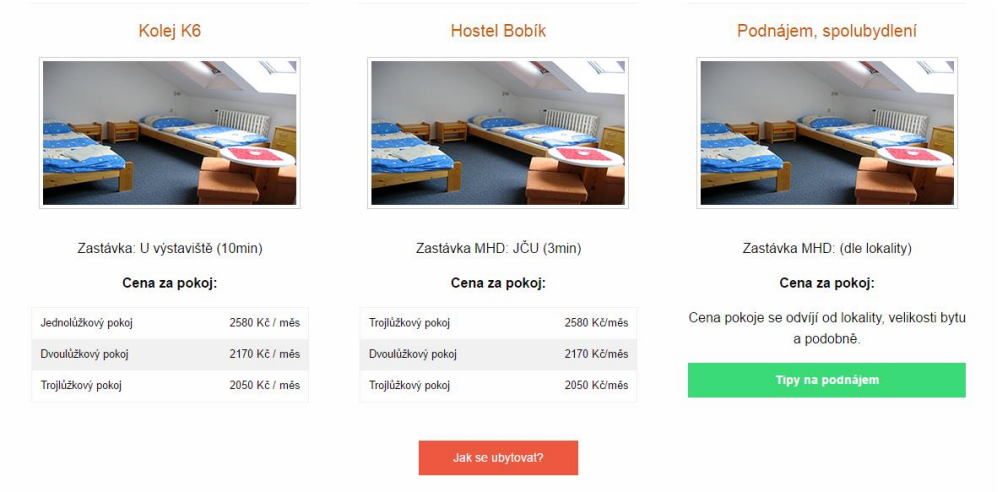
14 Seznam tabulek

15 Přílohy

Příloha č. 1 – CD – obsahuje text bakalářské práce a zdrojové kódy k webovým prezentacím.

Příloha č. 2 – Screenshoty z webových prezentací.

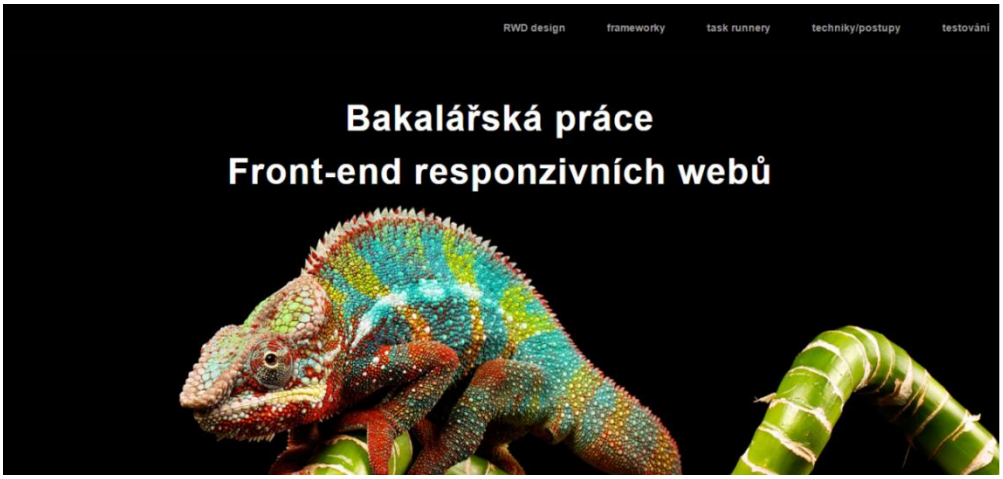
Obr. 1 Příloha obrázek 1



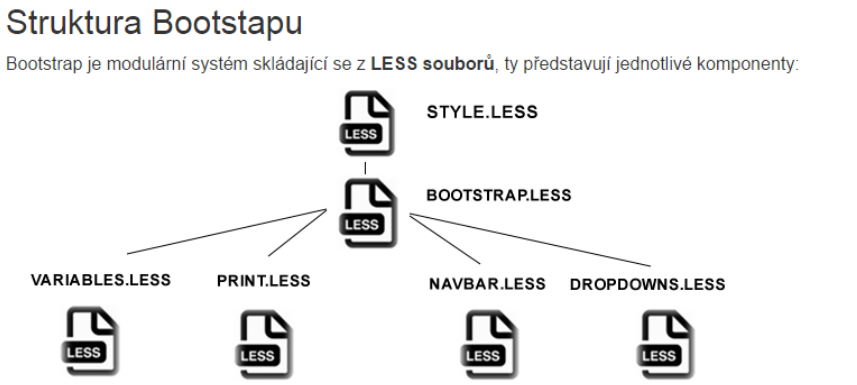
Obr. 2: Příloha obrázek 2

Obr. 3: Příloha obrázek 3

Obr. 4: Příloha obrázek 4

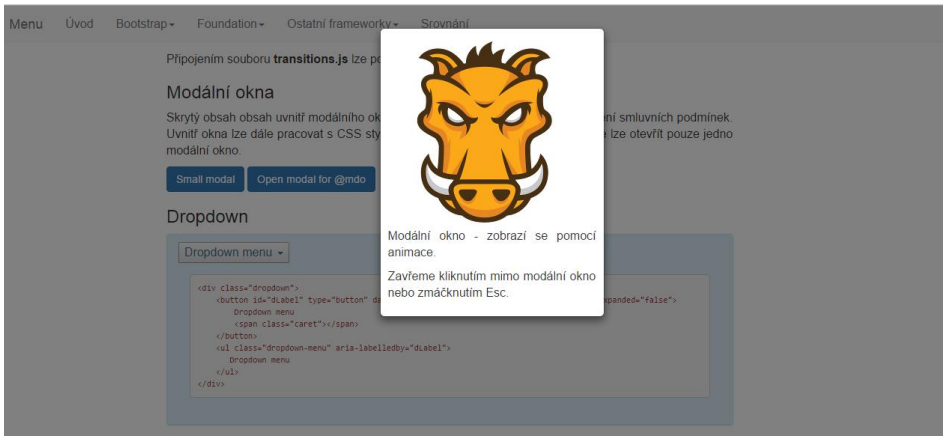


Obr. 5: Příloha obrázek 5



Obr. 6: Příloha obrázek 6

Obr. 7: Příloha obrázek 7



Obr. 8: Příloha obrázek 8