Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Bakalářská práce

2012 **Jaroslav Valdauf**

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta
Ústav aplikované informatiky

Mobilní konzole pro ovládání prezentací v HTML

Bakalářská práce

Jaroslav Valdauf
Školitel: PhDr. Milan Novák, Ph.D.
České Budějovice 2012

Bibliografické údaje:

Valdauf, J. 2012: Mobilní konzole pro ovládání prezentací v HTML. [Mobile console for controlling presentation in HTML. Bc. Thesis, in Czech.] 42p – Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace:

Cílem bakalářské práce je vytvořit kompletní balík aplikací složený z webové prezentační aplikace pro PC a webové aplikace pro mobilní zařízení, která bude sloužit pro ovládání prezentace a zároveň bude zobrazovat poznámky k jednotlivým snímkům. Komunikaci mezi aplikacemi bude zajišťovat program, který bude součástí balíku.

The purpose of this thesis is to create a complete application package consisting of a Web presentation application for PC and Web applications for mobile devices that will be used to control a presentation and also will display the notes for individual slides. Communication between applications will provide a program that will be part of the package.

Prohlášení:

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

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

V Českých Budějovicích, 20. Listopadu 2012

Jaroslav Valdauf

Poděkování:

Děkuji panu Martinu Malému za cenné rady a připomínky. Dále bych chtěl poděkovat PhDr. Milanu Novákovi, Ph.D., že se ujal vedení mé práce.

Obsah

1. Zadání práce

Mobilní konzole pro ovládání prezentací v HTML.

Cíle práce:

1. Připravit vhodný nástroj pro prezentace v HTML z PC
2. Navrhnout a vytvořit HTML aplikaci pro mobilní zařízení, která bude fungovat jako dálkový ovladač, a zároveň jako „nápověda“ pro přednášejícího
3. Navrhnout a vyřešit komunikaci mezi PC a HTML
4. Hotový nástroj (webová prezentační aplikace pro PC + webová aplikace pro mobilní zařízení) připravit do podoby použitelného balíku aplikací (včetně manuálu)

2. Úvod

Českým ekvivalentem slova prezentace je představení či předložení něčeho. Prezentovat se dá v podstatě cokoli, sebe sama, názor, nebo věc. Je tedy zřejmé, že se s prezentacemi nejrůznějších forem setkáváme každý den. V práci, v televizi, v obchodech, ve školách. Nejlépe se samozřejmě prezentuje něco, co se dá ukázat a nejlépe předvést, což není v každém případě úplně možné. Není tedy nic zvláštního, že se začaly využívat vizuální systémy, jakou jsou třeba projektory. Následně s pokrokem vědy přišly na svět počítačové programy, které nám umožňují vytvářet a promítat prezentace tak, jaké je známe teď. Jedním z nejčastěji užívaných prezentačních programů je například PowerPoint z dílny společnosti Microsoft.

Základním elementem takto vytvořené počítačové prezentace je snímek neboli slide, je to v podstatě „arch papíru“, na který můžeme za pomocí nástrojů daného softwaru vkládat nejrůznější prvky. Ať už jde o jednoduchý text, obrázek, písničku, animaci, či video. Výsledná prezentace je posluchačům promítána právě po těchto jednotlivých.

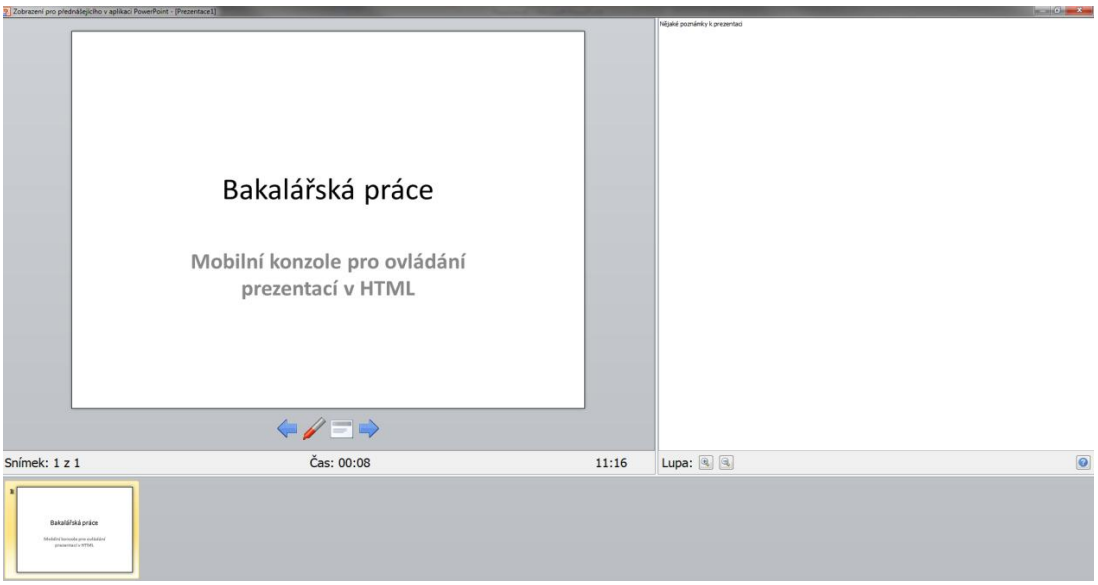
Nevýhodou tohoto na první pohled velmi pohodlného způsobu může být fakt, že prezentující je zároveň obsluhujícím zařízení, na kterém je prezentace spuštěna, tedy nejčastěji počítače, což nemusí být vždy příjemné jemu samotnému ani posluchačům. Jsou situace, kdy to není žádoucí ani z hlediska rozvržení interiéru místnosti, kde prezentace probíhá. Na místě jsou pak dvě řešení – další osoba či dálkové ovládání.

3. Přehled současného stavu

V současné době je na trhu několik platforem pro prezentování a jejich vzdálené ovládání. V následujícím výčtu bylo vybráno několik nejznámějších a nejpoužívanějších technologií a ovladačů používaných k prezentování.

3.1 PowerPoint

Prezentace založená na platformě Microsoft PowerPoint je uživatelsky přívětivá, snadno ovladatelná s možností nastavení *Zobrazení pro přednášejícího*, jak je ukázáno na obrázku 1.



Obrázek 1: PowerPoint – Zobrazení pro přednášejícího

Tato konzole umožňuje zobrazení poznámek, ovládání prezentace, použití zvýrazňovače během přednesu, zobrazuje počet snímků a celkový čas prezentování ve zvláštním okně na monitoru pro prezentujícího. Toto zobrazení musí vždy běžet na počítači, ze kterého je spuštěna prezentace, neexistuje mobilní verze. Řešením dálkového ovládání je využití bezdrátového prezentéru, který umožňuje přepínání jednotlivých snímků, spouštění multimédií, ale nedokáže zobrazovat poznámky k prezentaci. Dalším možným kontrolorem může být program MyPoint PowerPoint Remote od společnosti Didonai LLC. Na mobilní telefon se nainstaluje konzole pro ovládání, která se po síti připojí k počítači s prezentací, kde musí být nainstalován software, umožňující propojení mobilního zařízení a prezentace. Aplikace umožňuje zobrazení aktuálního a následujícího slidu, zobrazení poznámek, možnost ovládat kurzor myši na běžící prezentaci a použití zvýrazňovače. Nevýhodou tohoto řešení je ne-

možnost ovládat videa v prezentaci a občasné problémy s nalezením daemona (program běžící na pozadí) zajišťující navázání komunikace. Ve volně dostupné verzi není možné využívat rozšíření, jsou zpoplatněna. Největším problémem je nemožnost používat toto řešení jako přenosnou verzi na flashdisku nebo paměťové kartě, právě kvůli nutnosti spustit program na pozadí operačního systému.

3.2 Prezentace v PDF

Na vytvoření prezentací ve formátu PDF je možné použít několik specializovaných nástrojů, z nichž lze vybrat i několik volně dostupných. Pro vzdálené ovládání tohoto typu prezentace neexistuje specializovaný nástroj. Jednou z možností bezdrátového ovládání přepínání snímků je využití některého z programů pro simulaci klávesnice počítače na mobilním zařízení. Největším problémem tohoto řešení je nemožnost zobrazení poznámek prezentujícímu na telefonu či tabletu, ze kterého je prezentace ovládána. Dalším z úskalí je nepohodlné ovládání, které není uzpůsobené pro tyto účely.

3.3 HTML5 prezentace

Pro tvorbu HTML5 prezentací s využitím JavaScriptu a CSS existuje mnoho prezentačních systémů, avšak zcela chybí nástroj pro vzdálené ovládání, přepínání snímků, ovládání videa a zobrazení poznámek pro prezentujícího. Právě tento nedostatek bude řešen v této bakalářské práci. Obrázek 2 zobrazuje jedno z možných grafických schémat prezentace.

Obrázek 2: HTML5 prezentace

4. Přehled použitých technologií a postupů

Tato část bakalářské práce se věnuje použitým technologiím a postupům, které vedly k vytvoření funkčního balíku aplikací. Ten se skládá z prezentační platformy vytvořené za pomoci HTML5, Javascriptu a CSS, ovladače prezentace a websocketového serveru využívající platformu Node.js. Aby mohla být aplikace použitá jako nativní v různých mobilních systémech, bylo využito frameworku Phonegap.

4.1 HTML5 prezentace

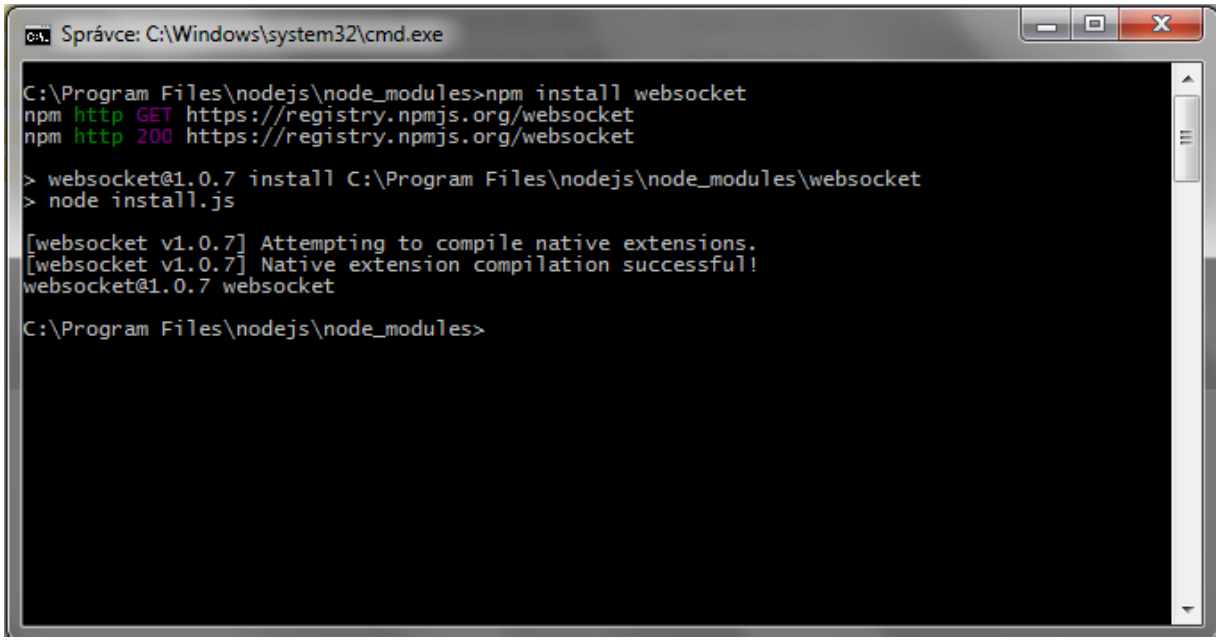
HTML5 je specifikace značkovacího jazyka HTML (někdy se takto označují souhrnně i další technologie, jako CSS3, Websocket a další, takovéto označení je ovšem nepřesné). V současné době je ve verzi draftu a podle plánu by stabilní verze měla být schválena do konce roku 2014.[1. Node.js, 2012] Pro potřeby této práce jsem zvolil prezentační platformu vytvořenou společností Google a dostupnou na adrese http://slides.html5rocks.com. V této prezentaci je využito moderních webových technologií, jako jsou CSS3, HTML5 a JavaScript. Tvorba prezentace probíhá v jakémkoliv textovém editoru za využití značkovacího jazyka HTML, jak ukazuje následující úsek kódu. Je možné používat značky jako v jakémkoliv jiném dokumentu HTML, umožňuje vkládat různé elementy, například obrázky, videa, zvukové soubory, odkazy, tlačítka a další.

```
<div class="slide" id="landing-slide">
  <section class="middle">
    <h3>Bakalářská práce</h3>
    <h2>Mobilní konzole pro ovládání prezentací v HTML</h2>
    <h3>Jaroslav Valdauf</h3>
    <h3>Vedoucí bakalářské práce: Martin Malý</h3>
  </section>
</div>
```

4.2 Node.js

Na oficiálních stránkách projektu Node.js je tato platforma specifikovaná následovně: „Node.js je platforma postavená na JavaScript runtime Chrome pro snadné budování rychlých, škálovatelných síťových aplikací. Node.js používá řízení událostí, neblokující I/O model, který ho dělá lehkým a výkonným, ideální pro datově náročné aplikace v reálném čase, které jsou spuštěny v distribuovaných zařízení.“ [1. Node.js, 2012] Node.js umožňuje využívat JavaScript na straně serveru, díky čemuž není nutné, aby programátor ovládal další programovací jazyk. V případě této bakalářské práce byl využit pro vytvoření HTTP a následně i Websocket serveru pro distribuci ovládací konzole na mobilní zařízení a zajištění komunikace mezi tabletem a prezentací běžící na počítači.

Node.js využívá pro instalaci modulů balíčkovací systém NPM – Node Package Manager. Seznam všech dostupných balíčků je k vidění na adrese http://npmjs.org/, který obsahuje více než 17 000 různých položek. Práce s balíčky probíhá přes příkazový řádek pomocí příkazu *npm*. Instalace modulu websocket, který je klíčový pro správnou funkčnost serveru této bakalářské práce, se spustí pomocí příkazu *npm install websocket*, viz obrázek 3.



Obrázek 3: Instalace modulu Websocket v příkazovém řádku

4.3 Websockets

Během příchodu HTML5 se objevila i nová technologie Websockets, která umožňuje navázání obousměrné komunikace mezi klientem a serverem. Pokud je spojení navázáno, může probíhat výměna informací, aniž by bylo nutné, aby se klient dotazoval serveru, zda má pro něj nějaká data. Tím dojde ke snížení objemu přenesených dat, ke snížení latence a vytížení serveru.

Websocket protokol byl navržen tak, aby pracoval s existující webovou infrastrukturou. Specifikace protokolu definuje, že spojení Websocket začíná svůj život jako připojení http zaručující plnou zpětnou kompatibilitu se světem před použitím Websocketu. Přechod protokolu z HTTP na Websocket se označuje jako Websocket handshake.

Prohlížeč odešle požadavek na server, který naznačuje, že chce přejít z protokolu HTTP na protokol Websocket. Klient žádá o změnu protokolu pomocí Upgrade header:

```
GET ws://echo.websocket.org/?encoding=text HTTP/1.1
Origin: http://websocket.org
Cookie: __utma=99as
Connection: Upgrade
Host: echo.websocket.org
Sec-WebSocket-Key: uRovscZjNol/umbTt5uKmw==
Upgrade: websocket
Sec-WebSocket-Version: 13
```

Pokud server používá protokol Websocket, schválí změnu protokolu pomocí Upgrade header:

```
HTTP/1.1 101 WebSocket Protocol Handshake
Date: Fri, 10 Feb 2012 17:38:18 GMT
Connection: Upgrade
Server: Kaazing Gateway
Upgrade: WebSocket
Access-Control-Allow-Origin: http://websocket.org
Access-Control-Allow-Credentials: true
Sec-WebSocket-Accept: rLHCkw/SKsO9GAH/ZSFhBATDKrU=
Access-Control-Allow-Headers: content-type
```

V tomto okamžiku připojení HTTP končí a je nahrazeno Websocket připojením přes stejné TCP / IP spojení. Připojení Websocket používá stejné porty jako HTTP (80) a HTTPS (443). [2. Websocket.org, 2012] Možné je ovšem použít libovolný port, než jeden z přednastavených.

V současné době je standardizovaný protokol RFC 6455 z prosince 2011.
[3. w3.org, 2012]

Problémem Websocketu je nekompatibilita jednotlivých verzí mezi sebou.

V následujících tabulkách je názorně zobrazeno, jaká je podpora ze strany jednotlivých desktopových [Tabulka 1] i mobilních prohlížečů [Tabulka 2] s poslední verzí tohoto protokolu.

Desktopové prohlížeče					
	Internet Explorer	Firefox	Chrome	Safari	Opera
Websockets RFC 6455	10.0	Od verze 6.0	Od verze 14.0	6.0	Od verze 12.1

Tabulka 1: Přehled podporovaných desktopových prohlížečů

Mobilní prohlížeče					
	iOS Safari	Android browser	Opera mobile	BlackBerry browser	Chrome for Android
Websockets RFC 6455	6.0	4.1	X	7.0	18.0

Tabulka 2: Přehled podporovaných mobilních prohlížečů

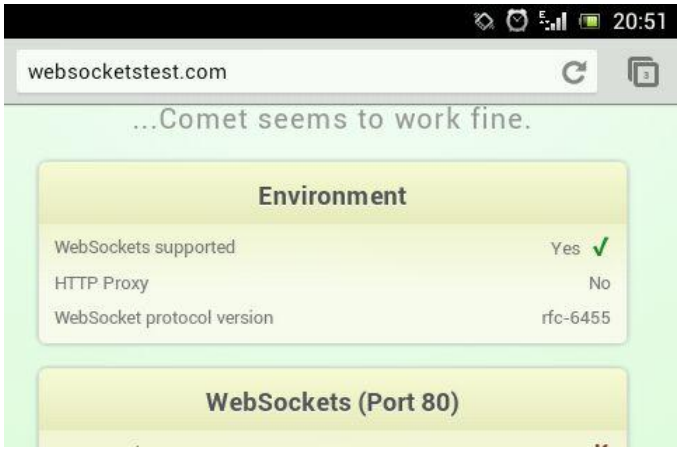
U mobilního systému Android, který podporuje protokol Websocket až od verze 4.1 Jelly Bean v integrovaném prohlížeči, se dá vyřešit tento problém pomocí prohlížeče internetu Google Chrome, který je dostupný i na starší verze systému, popřípadě pomocí Opery Mobile či Firefox pro Android.

Problém nastává u mobilního operačního systému iOS, který podporuje Websocket verze RFC 6455 ve vestavěném prohlížeči Safari až od verze systému 6.0. V tomto případě se nedá použít stejné řešení jako u Androidu, neboť společnost Apple nedovoluje napsat prohlížeč pro svůj mobilní systém na jiném jádře prohlížeče, než je Safari. Z tohoto důvodu, není možné provozovat ve starších verzích tohoto mobilního systému aplikace, které využívají v současné době poslední verzi protokolu Websocket.

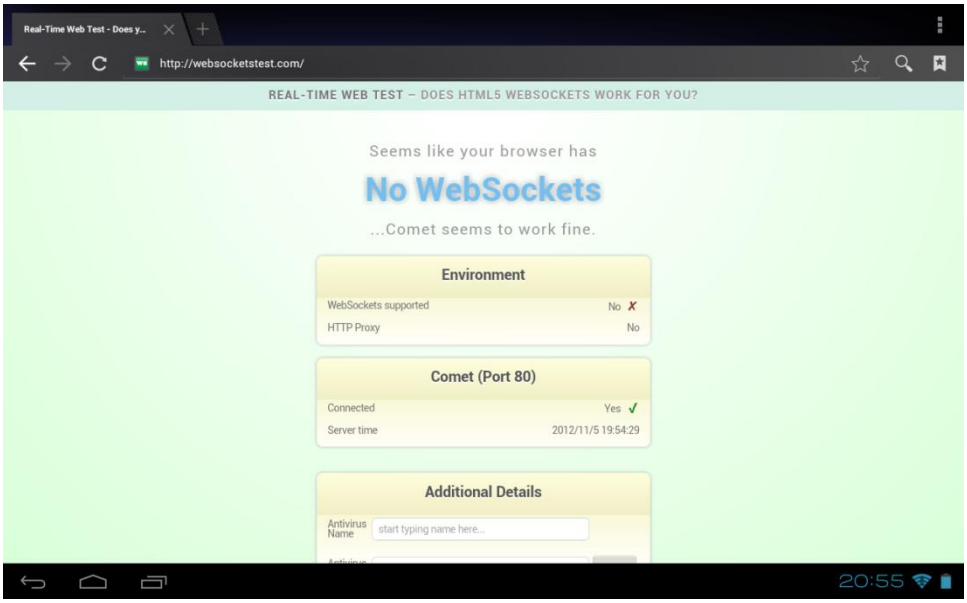
V době tvorby této bakalářské práce nebyl k dispozici mobilní systém Windows RT, proto bylo testováno použití Windows 8 v 64-bitové verzi na tablet PC s prohlížečem Internet Explorer 10 a Google Chrome.

4.3.1 Ověření verze Websocket v prohlížeči

Pro ověření správné funkčnosti protokolu websockets se zobrazením používané verze je vhodné využít webové stránky http://websocketstest.com/. Pomocí nich lze snadno zjistit, zda bude fungovat na používaném zařízení ovládací konzole, která využívá aktuálně poslední verzi protokolu Websocket a to RFC 6455.



Obrázek 4: Zobrazení verze protokolu na systému Android 4.0.4 v prohlížeči



Obrázek 5: Zobrazení verze protokolu na systému Android 4.0.3 na tabletu

4.4 Phonegap

Phonegap je nástroj, který slouží k vytvoření nativní aplikace za použití HTML5, JavaScriptu a CSS3 pro mobilní platformy uvedené v tabulce 3. Největší výhodou tohoto řešení je to, že stačí vytvořit jednu aplikaci v HTML a pomocí Phonegapu z ní udělat nativní aplikaci na většinu mobilních platforem. Základní webová aplikace zůstane stejná na všech platformách, lišit se bude pouze v té části, která z ní bude dělat nativní aplikaci. Pro použití Phonegapu je vhodné mít vývojové prostředí pro každou z platforem, na které bude třeba aplikaci používat. Pro iOS je to XCode v OSX, pro Android Android SDK, je ovšem možné využít i službu Phonegap Build. Princip této služby spočívá v tom, že nahrajete zdrojové soubory webové aplikace a pomocí této online služby je vytvořena aplikace nativní, takže není nezbytně nutné vlastnit každé z vývojových prostředí. Phonegap zároveň umožňuje programátorovi přístup k různým funkcím mobilního systému jako je akcelerometr, fotoaparát, kompas a jiným, které jsou uvedeny v tabulce níže.

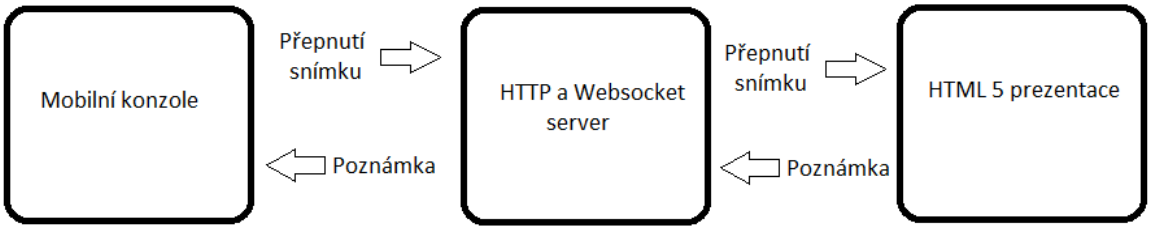
	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 5.x	Blackberry OS 6.0+	WebOS	Windows Phone 7	Symbian	Bada
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓	✓
Compass	x	✓	✓	x	x	✓	✓	x	✓
Contacts	✓	✓	✓	✓	✓	x	✓	✓	✓
File	✓	✓	✓	✓	✓	x	✓	x	x
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	x	x	x	✓	x	x
Network	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	✓	x

Tabulka 3: Tabulka podporovaných funkcí v jednotlivých mobilních systémech [5. Phonegap.com, 2012]

5. Návrh řešení

Při návrhu řešení bylo využito technologií uvedených v předchozí kapitole. Prvotní návrh byl takový, že bude vytvořena aplikace pro mobilní zařízení na platformě iOS pomocí Phonegapu, která bude komunikovat přímo s prezentací, jež poběží na počítači. Toto řešení bylo nakonec upraveno tak, že byl vytvořen server, který zajišťuje komunikaci mezi klienty a zároveň může distribuovat ovládací rozhraní do mobilního zařízení, takže není nutná instalace. Za použití Phonegapu byla vytvořena nativní aplikace pro Apple iOS 6, která je součástí této práce. Pro vytvoření aplikace pro jiné mobilní operační systémy je v ovladači zaváděn potřebný Javascriptový soubor, který je nutný pro použití Phonegapu.

Jako základ komunikace bylo nutné vytvořit server, který bude zajišťovat distribuci ovladače do prohlížeče mobilního zařízení a komunikaci mezi touto konzolí a HTML5 prezentací běžící na počítači. Pro zajištění komunikace byla vybrána technologie Websocket díky své malé náročnosti na datové přenosy a výkon počítače, na kterém běží server. Pomocí této technologie se posílají zprávy od klientů na server, který je poté distribuuje dál klientům. Následující obrázek zobrazuje zjednodušené schéma posílání zpráv mezi mobilní konzolí a prezentací. Zprávy obsahují informace o posunutí na předchozí nebo následující snímek, ovládání videa a hlavně zaručují distribuci poznámek pro přednášejícího na obrazovku ovladače.



Obrázek 6: Schéma komunikace jednotlivých částí

Pro zobrazování prezentací bylo vybráno řešení od společnosti Google, které je dostupné na webové adrese http://slides.html5rocks.com. Prezentační platforma je šířena pod licencí BSD. Toto řešení bylo zvoleno proto, že jej lze volně použít a vytvořit odvozená díla, což vyhovuje potřebám této práce. Podrobně budou rozebrány možnosti výběru jednotlivých částí v podkapitolách níže.

5.1 Real-time komunikace

Jako základ celé práce bylo nutné vybrat technologii, která bude zajišťovat výměnu dat v reálném čase mezi ovladačem na mobilním zařízení a prezentací běžící na počítači. Bylo rozhodováno mezi třemi způsoby, a to Polling (dotazování), Comet a Websocket.

5.1.1 Polling

Tento způsob využívá HTTP protokol pro odpověď na dotaz klienta, proto není možné, aby server doručil zprávu klientovi, aniž by byl předtím dotázán. Opakovaným zasíláním dotazů na server však vzniká zbytečný síťový provoz, neboť dotaz i odpověď musí obsahovat HTTP hlavičky. A protože tedy získání dat od serveru není možné bez dotazu klienta, dochází samozřejmě ke zvyšování latence.

5.1.2 Comet

Server Comet funguje na protokolu http, ale nevyužívá opakovaného dotazování pro zajištění komunikace v reálném čase. Princip funkce je takový, že klient se dotáže serveru, server mu odpoví, ale nechá spojení otevřené. Pokud se objeví zpráva, zašle ji klientovi a pak až je spojení ukončeno. Klient neobdrží od serveru další zprávu, pokud se ho nejprve nedotáže a tím znovu neotevře spojení.

Problém této technologie spočívá v možnosti vypršení timeoutu, pokud server dlouho nedostane zprávu, kterou by poslal klientovi. Udržování aktivního spojení je pro servery velice náročné. Výhodou je opět silná podpora na straně prohlížečů.

5.1.3 Websocket

Websocket na rozdíl od předchozích technologií nevyužívá dotazování, ale je vytvořeno spojení, po kterém se mohou posílat informace od klienta k serveru i obráceně, aniž by bylo nutné neustálé dotazování ze strany klienta. Tento způsob ulehčuje síťovému provozu, snižuje latenci a nejsou kladeny velké nároky na výkon počítače, na kterém běží serverová aplikace. Nevýhodou tohoto řešení je ovšem nekompatibilita jednotlivých verzí websocketu mezi sebou, jak už bylo napsáno v kapitole výše.

5.2 HTTP a Websocket server

Jako základ celého návrhu je server, který bude zajišťovat komunikaci mezi klienty a zároveň bude distribuovat ovládací konzolu do internetového prohlížeče mobilního zařízení. Musel být vytvořen tak, aby byl snadno přenositelný, tzn., aby balík nebyl vázán na jeden konkrétní počítač, ale na flashdisku nebo paměťové kartě byl schopný funkce na jakémkoliv počítači se systémem Windows. Pro vytvoření serveru byl vybrán Node.js. Tato technologie umožňuje tvorbu HTTP a Websocket serveru za použití Javascriptu, nebylo tedy nutné využívat další programovací jazyk, než který je použit v prezentaci a mobilním ovladači. Node.js je možné používat na operačních systémech Windows, Linux a Mac OS.

5.2.1 HTTP server

Distribuce ovládacího rozhraní prezentace do mobilního zařízení probíhá pomocí HTTP serveru, který bude poskytovat HTML stránky, Javascript i CSS. Díky využití Node.js stačí spustit jeden skript, který obslouží jak HTTP, tak Websocket. Pokud nebude v adrese specifikován soubor, webserver automaticky distribuuje úvodní stránku mobilního ovladače. HTTP server naslouchá na portu 8080.

5.2.2 Websocket server

Jak bylo napsáno výše, jeden javascriptový soubor obsluhuje HTTP i Websocket server. Aby bylo možné navázat websocketovou komunikaci, bude nutné nainstalovat do Node.js balík websocket, který umožní vytvořit Websocket server. Server bude sloužit k přeposílání zpráv mezi klienty.

6. Implementace

V následující kapitole budou popsány jednotlivé komponenty, a to mobilní konzole, server a prezentace, u které bude navíc jednoduchý návod na vytvoření vlastní prezentace. Pro vytváření aplikace bylo využito programu Adobe Dreamweaver ve volně dostupné verzi.

6.1 Mobilní konzole

Mobilní konzole využívá javascriptové knihovny jQuery [6. jQuery.com, 2012], která je volně šiřitelná na základě duální licence MIT a GPL. Konkrétně se jedná o soubory *jQuery.mobile.min.js* a *jQuery.min.js*. Mimo to je nutné využívat soubor *cordova.js*, aby bylo možné použít PhoneGap pro vytvoření nativní aplikace. V následujících odstavcích budou popsány jednotlivé soubory, které byly vytvořeny.

6.1.2 Soubor index.html

Tento soubor je základem mobilního ovladače, obsahuje elementy popsané níže, vložení Javascriptových souborů a načtení souboru *jquery.mobile.css*, který obsahuje styl této stránky.

6.1.2.1 Hlavička html souboru

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
   maximum-scale=1.0, user-scalable=no;" />
  <meta name="apple-mobile-web-app-capable" content="yes">
  <title>Presentation Controller
  </title>
  <link href="js/jquery.mobile.css" rel="stylesheet" type="text/css">
  <script type="text/javascript" src="cordova-2.1.0.js"></script>
  <script type="text/javascript" src="js/jquery.min.js"></script>
  <script type="text/javascript" src="js/jquery.mobile.min.js"></script>
  <script type="text/javascript" src="js/controller.js"></script>
</head>
```

V hlavičce souboru je pomocí nepárového tagu *<meta charser="utf-8">* charakterizována využitá znaková sada, v tomto případě se jedná o UTF-8. V dalším meta tagu je specifikováno zobrazení webu na celou šířku obrazovky, zachování měřítka a pomocí *user-scalable=no* je zakázáno využití zoomu na stránce. Použití tagu *<meta name="apple-mobile-web-app-capable" content="yes">* nám zajistí schování ovládacích prvků prohlížeče Safari v případě, že si „připneme“ stránku na plochu.

V případě, že budeme chtít mít zástupce na ploše v systému iOS, je možné nastavit ikonu, která se zobrazí a obrázek na startovací stránce. Pomocí link *tagu <link rel="apple-touch-icon" href="./ icon.png">* určíme ikonu, která musí mít v případě použití na iPhonu starším, než model 3GS rozlišení 57x57 pixelů, od modelu 4 až po v současnosti poslední iPhone5 rozlišení 114x114 pixelů. Rozlišení ikony na iPadu je 57x57 pixelů, při použití retina displaye je nutné použít rozměr 114x114.

Podobný tagem, *<link rel="apple-touch-startup-image" href="./apple-touch-startup-image.png">*, lze nastavit obrázek startovací obrazovky. Při použití Ipadu první a druhé generace musí mít rozlišení 768x1004 pixelů, v případě využití novějších modelů iPadu je to 1536x2008 pixelů. Rozlišení startovací obrazovky u iPhonu je 320x480 pixelů, respektive 640x960 pixelů u modelů od verze 4.

Dále je v hlavičce určen pomocí párového tagu *<title>* titulek stránky a pomocí link tagu určen CSS soubor obsahující informace o vzhledu stránky. Následuje načtení Javascriptových souborů, které budou popsány dále.

6.1.2.2 Tělo html souboru

```
<body>
<div data-role="page">
    <div data-role="header" data-theme="b">
    <h1>Presentation Controller</h1></div>
    <div data-role="content">
        <div id="textControl"></div>
    </div>
    <div data-role="footer" data-theme="b">
        <a href="#prev">Previous</a>
        <a href="#play">Play</a>
        <a href="#pause">Pause</a>
        <a href="#stop">Stop</a>
        <a href="#next">Next</a>
    </div>
</div>
</body>
```

Tělo je rozděleno na hlavičku, obsahující nadpis stránky, vlastní obsah stránky, kde je umístěn div s atributem id *textControl* sloužící k zobrazování poznámek k prezentaci. Na závěr je specifikována patička stránky v tagu *<div data/role=”footer” data-theme=“b“>*, která obsahuje pět tlačítek k ovládání prezentace a to přechod na další či předchozí stránku, spuštění, zastavení a pozastavení přehrávače videa. Každé z těchto tlačítek má v parametru *href* hodnotu označující funkci tohoto tlačítka.

6.1.3 Soubor controller.js

Tento Javascriptový soubor obsahuje všechny funkční komponenty, které jsou nutné pro správnou komunikaci se serverem, výpočet rozmístění prvků na obrazovce a úpravu příchozích poznámek, aby byly korektně zobrazeny.

6.1.3.1 Zabránění posunu

```
document.ontouchmove = function(e) {
    if(e.target.id != "textControl"){
        e.preventDefault()
    }};
```

Tento úsek kódu zabraňuje uživateli posouvat stránku. Pokud by nebyla funkce uvedena, bylo by možné u systému iOS posunovat stránku mimo obraz, což by způsobilo zobrazení nevzhledných černých okrajů. Uvedená podmínka dovoluje posun pouze v části s id *textControl*, což je úsek stránky zobrazující poznámky pro přednášejícího. Pokud podmínka splněna není, tzn., že uživatel se pokouší posunout stránku jinde než v tomto elementu, je mu v tom zabráněno.

6.1.3.2 Změna orientace

```
window.onorientationchange = function(e) {
    setSize();
};
```

Tato funkce, která je vyvolána při otočení zařízení na výšku či na šířku, zajišťuje zavolání funkce *setSize* bez parametru, její fungování bude popsáno níže.

6.1.3.3 Po načtení aplikace

```
$(document).ready(function() {

var IP = prompt("Please insert IP address", window.location.hostname);

var connection = new WebSocket('ws://' + IP + ':1367');

setSize();

connection.onmessage = function(event) {
    switch(event.data.substring(0, 4)) {
        case 'note':

            $("#textControl").html(removeSpaces(event.data.substring(5)));
        break;
    }
};
```

Funkce, která je zavolána po načtení stránky *index.html*. Při načítání stránky z webového serveru je zobrazeno vyskakovací okno s žádostí o vložení IP adresy websocketového serveru, která je shodná s IP adresou webového serveru, pokud ovšem není využito řešení jiného výrobce pro distribuci webových stránek do prohlížeče mobilního zařízení, například Apache. V okně je již předvyplněna stejná adresa, jako je v adresním řádku prohlížeče, aby uživatel nemusel adresu opisovat. Její hodnota je získána z vlastnosti *hostname* objektu *window.location*.

Následuje vytvoření instance třídy websocket a tím dojde k otevření spojení mezi klientem a serverem, za využití protokolu websocket. Jako prefix je využito „ws://“, v případě, že by bylo potřeba využít spojení SSL je prefix „wss://“, což je v našem případě zbytečné, neboť jsou posílány pouze zprávy k ovládání prezentace. Jako IP adresa je využita proměnná, do které byla uložena hodnota z vyskakovacího okna popsaného výše. Port websocketového serveru je v tomto konkrétním případě nastaven na 1367. Nutné je před hodnotou portu uvést znak dvojtečka, aby byla celá adresa ve správném tvaru, například: *ws://192.168.0.1:1367*.

Následně se vypočítá rozměr prvků, dle rozlišení obrazovky, zavoláním funkce *setSize();* a následuje úsek, který pracuje se zprávami, jenž klient obdržel od websocketového serveru. Pokud přijde zpráva, jsou použity první čtyři znaky k identifikaci, zda se jedná o poznámky nebo cokoliv jiného. Zde je příprava pro budoucí možné rozšíření,
proto je využito přepínače *switch*. Pokud zpráva obsahuje „parametr“ *note* je zbytek textu od pátého znaku považován za poznámky a vložen do elementu *textControl*. Ještě před samotným vložením je zavolána funkce *removeSpaces*, která v parametru obsahuje text

poznámek, její přesný popis bude následovat níže.

6.1.3.4 Posílání zpráv

```
$("div[data-role=footer] a").on("click", function(e) {
    e.preventDefault();
    connection.send($(this).attr("href").split("#")[1]);
    });
});
```

Pomocí této části kódu se po kliknutí na tlačítko zabrání standardnímu chování, které by zavolalo stránku uvedenou v parametru *href*. Místo toho je odeslána zpráva pomocí příkazu *connection.send(zprava)* na websocketový server. Hodnota parametru je shodná s hodnotou parametru tlačítka *href*, jen je odstraněna mřížka na začátku. Odesílané zprávy ovládají prezentaci a to posunutí na předchozí snímek v případě, že je odeslána zpráva *prev*, na následující snímek při zprávě *next* a samozřejmě i ovládání videa, které se přehraje, pokud je ode-

sláno *play*, pozastaví v případě *pause*, zastaví a vrátí na začátek, pokud je stisknuto tlačítko *stop*. Zde je možnost budoucího rozšíření, neboť stačí vytvořit tlačítko s jakoukoliv hodnotou v parametru *href* a ta je přeposlána na server. Není tedy nutné zasahovat do tohoto kódu pro přidání další funkce, stačí upravit příjem zprávy na serveru.

6.1.3.5 Funkce setSize

```
var iOS = navigator.userAgent.match(/(iPad|iPhone|iPod)/i)?true:false;

if(iOS == true) {
    switch(orientation) {
        case 0:
  $("div#textControl").css("width", $(window).width() - 50);
  $("div#textControl").css("height", $(window).height() -
        ($("div[data-role=header]").height() +
        $("div[data-role=footer]").height() + 55));
        break;
        case 180:
            $("div#textControl").css("width", $(window).width() - 50);
            $("div#textControl").css("height", $(window).height() -
            ($("div[data-role=header]").height() +
            $("div[data-role=footer]").height() + 55));
        break;
        case 90:
    $("div#textControl").css("width", $(window).width() - 50);
            $("div#textControl").css("height", $(window).height() -
            ($("div[data-role=header]").height() +
            $("div[data-role=footer]").height() + 55));
        break;
        case -90:
            $("div#textControl").css("width", $(window).width() - 50);
            $("div#textControl").css("height", $(window).height() -
            ($("div[data-role=header]").height() +
            $("div[data-role=footer]").height() + 55));
        break;
    }
} else {
    $("div#textControl").css("width", $(window).width() - 100);
    $("div#textControl").css("height", $(window).height() -
        ($("div[data-role=header]").height()
        + $("div[data-role=footer]").height() + 75));
}
```

Následující funkce počítá rozměry elementu *textControl* při načtení stránky či otočení zařízení. Do proměnné *orientation* je uložena hodnota otočení, která u zařízení se systémem iOS a prohlížečem Safari nabývají hodnot 0, 180, což jsou hodnoty pro pozici Portrait a -90 a 90 pro pozici Landscape. V dalším kroku je zjištěno pomocí objektu navigator a vlastnosti userAgent, zda se dané zařízení identifikuje jako iPhone, iPad či iPod, pokud ano, proměnná *iOS*, obsahuje hodnotu *true*. Pokud je splněna podmínka *if*, tak je podle natočení zařízení určena velikost elementu *textControl* a to jak výška, tak šířka. Tyto hodnoty byly nastaveny

podle testování na iPadu druhé generace, jedná se o výšku či šířku obrazovky, od které se odečte hlavička a patička stránky a část pixelů pro zachování okrajů. V případě nasazení na jiném zařízení je potřeba tyto hodnoty upravit pro dané rozlišení obrazovky. Pokud podmínka splněna není, tzn., že zařízení se neidentifikuje jako mobilní zařízení od společnosti Apple, je výpočet rozměrů stejný při jakémkoliv natočení. Výpočet výšky a šířky elementu je nastaven tak, že je zjištěna výška, případně šířka obrazovky a od této hodnoty se odečte část pixelů, které zabírá lišta prohlížeče, hlavička a patička stránky. U výpočtu šířky se odečítá taková hodnota, aby se okraje „nedotýkaly“ stran obrazovky.

6.1.3.6 Funkce removeSpaces

```
function removeSpaces(strInputText){
    strInputText = strInputText.replace(/(\n\r|\n|\r)/gm,"<1br />");
    strInputText = strInputText.replace(/\t/g,"");

    re1 = /\s+/g;
    strInputText = strInputText.replace(re1," ");

    re2 = /\<1br \/>/gi;
    strInputText = strInputText.replace(re2, "\n");

    return strInputText;
}
```

Funkce *removeSpaces* slouží k odstranění tzv. whitespace a jejich nahrazením jednoduchým odřádkováním či mezerou. V první řádce funkce jsou nahrazeny znaky nové řádky nahrazeny textem „<1br />“, s nímž se pracuje později. Přepínač */gm* zajišťuje nahrazení opakovaně a na více řádcích. Odstranění tabulátoru, které je hned v následujícím řádku, je opět opatřeno přepínačem */g* pro opakované použití. Nahrazení nežádoucích dvojitých mezer, jejichž escape sekvence je „\s“, která je uložena v proměnné *r1*, probíhá opět za využití příkazu *replace.* Na závěr je text vytvořený na začátku této metody nahrazen odřádkováním. Celá funkce vrací upravený text, který je uložen v proměnné *strInputText.*

6.2 HTTP a websocket server

Jak již bylo uvedeno, nezbytnou součástí je server, který distribuuje rozhraní ovladače do mobilního zařízení a zároveň zajišťuje komunikaci pomocí protokolu websocket mezi klienty a serverem. Pro vytvoření bylo využito frameworku Node.js, aby bylo možné vytvořit server založený na Javascriptu. Při použití Node.js je nutné stáhnout správné balíčky, v tomto případě je to package websocket.

6.2.1 Soubor server.js

Jediný soubor, který zajišťuje distribuci HTML ovladače do mobilního zařízení a zároveň slouží pro zajištění komunikace mezi klienty.

6.2.1.1 Zavedení modulů

```
var WebSocketServer = require('websocket').server,
    http = require('http'),
    fs = require('fs'),
    path = require('path');
```

Jako první věc je nutné zavést všechny moduly, které budeme potřebovat. V tomto případě je to pro websocket *require('websocket')*, pro použití http serveru *require('http')*, pro možnost pracovat se soubory File System *require('fs')* a modul pro práci s cestami *require('path')*.

6.2.1.2 HTTP server

```
var server = http.createServer(function(request, response) {

    console.log('request starting for: ' + request.url);
    var filePath = path.join('./Ovladac/', request.url);
    if (filePath === "Ovladac\\" ) {
      filePath = './Ovladac/index.html';
    }
```

Základem http serveru je vytvoření jeho instance s parametrem *function(request,response)*. Žádost je instancí třídy *http.ServerRequest* a odpověď je instancí třídy *http.ServerResponse*. [7. Node.js, 2012] Při přijetí žádosti konzole zapíše z jaké url adresy byla žádost přijata. Do proměnné *filePath* je uložena cesta, která vede do složky s názvem Ovladac a připojí se k ní název souboru, který byl vyplněn v žádosti. V případě, že nebyl za adresou v prohlížeči specifikován žádný soubor, je do proměnné doplněna adresa vedoucí ke spouštěcímu souboru ovladače a to *index.html*.

```
}).listen('8080','0.0.0.0');
```

Na závěr je specifikováno na jakém portu a jaké IP adrese server naslouchá. V tomto případě je to port 8080 a IP adresa není specifikována, takže připojení může přijít z libovolného připojení.

6.2.1.3 Zjištění existence souboru

```
fs.exists(filePath, function(exists) {
    if (exists) {
        var extname = path.extname(filePath);
        var contentType = 'text/html';
        switch (extname) {
         case '.js':
            contentType = 'text/javascript';
            break;
         case '.css':
            contentType = 'text/css';
            break;
        }
     fs.readFile(filePath, function(error, content) {
            if (error) {
                response.writeHead(500);
                response.end(); }
            else {
                response.writeHead(200, {
                    'Content-Type': contentType
                });
                response.end(content, 'utf-8');
            }); }
    else {
        response.writeHead(404);
        response.end();
    }});
```

V následujícím kódu probíhá zjištění, zda daná cesta k souboru existuje. V případě, že neexistuje, je status hlavičky odpovědi nastaven na hodnotu *400 Bad Request*. Pokud je podmínka splněna, ukládá se do proměnné *extname* koncovka ze souboru. Funkce *path.extname(nazev_souboru.html)* odstraní vše před koncovkou a vrací například *.html*. Proměnná *contentType* uchovává informaci o tom, o jaký druh souboru se jedná, zda html, Javascript či CSS. Následuje asynchronní přečtení souboru, jehož cesta je uložena v proměnné filePath, pomocí fs.readFile(nazev_souboru,). Pokud nastane chyba při čtení, je v hlavičce odpovědi serveru nastaven status *500 Internal Server Error*. V případě, že je vše v pořádku, nastaví se status na *200 OK*. Do hlavičky odpovědi se zapíše *Content-Type*, který specifikuje typ souboru, z proměnné *contentType* a znaková sada utf-8.

6.2.1.4 Websocket server

```
wsServer = new WebSocketServer({
    httpServer: serverForWs
});
```

Vytvoření instance websocketového serveru, hodnoty parametrů jsou využity z HTTP serveru.

6.2.1.4 Websocketová komunikace

```
wsServer.on('request', function(request) {

var connection = request.accept(null, request.origin);
console.log('Connection accepted');

connection.on('message', function(message) {
    if (message.type === 'utf8') {
       switch(message.utf8Data) {
            case 'next':
                console.log('Received: Next slide');
                wsServer.broadcast( "next");
                break;
            case 'prev':
                console.log('Received: Previous slide');
                wsServer.broadcast("prev");
                break;
            case 'play':
                console.log('Received: Play');
                wsServer.broadcast("play");
                break;
            case 'pause':
                console.log('Received: Pause');
                wsServer.broadcast("pause");
                break;
            case 'stop':
                console.log('Received: Stop');
                wsServer.broadcast("stop");
                break;
            default:
                console.log('Received: Note');
                wsServer.broadcast('note' + message.utf8Data);
                break;
     }}});});
```

Funkce *wsServer.on('request', function('request'));* je zavolána pokaždé, když se někdo snaží připojit na websocketový server. Pokud připojení proběhne, je vytvořena nová instance tohoto připojení a do konzole se vypíše, že bylo spojení navázáno.

Následuje nejdůležitější část a to zpracování zpráv přijímaných od klientů. Pokud je na server doručena zpráva proběhne nejprve kontrola, zda obsahuje pouze text, tuto kontrolu provádí příkaz *if (message.type === 'utf8').* V případě, že je vše v pořádku, pokračuje zpra-

cování zprávy dál, pokud podmínce není vyhověno, zpráva je zahozena. V podmínce přepínače switch je zpráva uložena jako text s kódováním UTF-8. V případě přijetí nějaké z definovaných zpráv je vypsáno do konzole jaká zpráva byla přijata a je následně přeposlána všem klientům. Pokud text neodpovídá žádné podmínce, pokládá se přijatá zpráva za poznámky. V tom případě se před text připojí *„note“*, aby ovladač poznal, že je zpráva určena pro něj a odešle se opět všem klientům. Způsob distribuce zpráv sice zvyšuje náročnost na datové přenosy, ale nezáleží na pořadí připojení klientů – prezentace a ovladače. V případě výpadku připojení je možné po opětovném připojení pokračovat, stačí pouze obnovit stránku s prezentací či stránku s ovládáním.

6.3 Prezentace

V následující kapitole bude popsána tvorba prezentace a funkce, které byly přidány do prezentace, která je volně dostupná na http://slides.html5rocks.com.

6.3.1 Soubor slides.html

V tomto souboru je uložen obsah jednotlivých snímku, postup tvorby snímku bude popsán níže.

6.3.1.1 Hlavička html souboru

```
<!DOCTYPE html>
<html manifest="cache.appcache">
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=Edge;chrome=1" />
<title>HTML5 Presentation
</title>
<link href="src/prettify/prettify.css" id="prettify-link" rel="stylesheet"/>
<link href="css/default.css" class="theme" rel="stylesheet" />
<link href="css/sand.css" class="theme" rel="stylesheet" />
<link href="css/sea_wave.css" class="theme" rel="stylesheet" />
<link href="css/moon.css" class="theme" rel="stylesheet" media="screen" />
<script src="js/jquery-1.8.0.js" type="text/javascript"></script>
<script src="js/main.js" type="text/javascript"></script>
</head>
```

V hlavičce html souboru je definována znaková sada UTF-8 a pomocí meta tagu *<meta http-equiv="X-UA-Compatible" content="IE=Edge;chrome=1" />* je určeno, aby byl použit nejnovější dostupný mód prohlížeče Internet Explorer, který podporuje HTML5 a Websocket. Následuje načtení několika vzhledů prezentace, mezi kterými je možné přepínat. Na závěr hlavičky je nutné určit, jaké javascriptové soubory se budou používat, v tomto případě je to *jQuery* a soubor *main.js*, který obsahuje funkce pro správný chod prezentace.

6.3.1.2 Syntaxe prezentace

Tato kapitola bude popisovat vytváření snímků prezentace, jednotlivé prvky, které je možné do prezentace vložit a jejich možné formátování.

```
<body>
<div class="presentation">
 <div id="presentation-counter">Loading...
  </div>
  <div class="slides">
      <!--Zde budou jednotlivé snímky prezentace-->
  </div>
 </div>
</body>
```

Toto je základ obsahu, kde v *divu presentation* jsou obsaženy všechny snímky prezentace, nezbytným je prvek s id *presentation-counter*, který zajišťuje počítání snímků při přepínání. Poté je již možné přejít k samotnému vytváření snímků ve třídě *slides*, jak ukazuje následující úsek kódu, tvorba slidů je jednoduchá a uživateli stačí základy HTML.

```
<div class="slide" id="landing-slide">
<section class="middle">
<h3>Bakalářská práce</h3>
<h2>Mobilní konzole pro ovládání prezentací v HTML</h2>
<h3>Jaroslav Valdauf</h3>
<h3>Vedoucí bakalářské práce: Martin Malý</h3>
</section>
<aside class="note">
Poznámky k prezentaci
</aside>
</div>
```

Celý snímek je zabalen do jedné třídy *slide*, úvodní snímek musí mít id *landing-slide* z důvodu správné inicializace celé prezentace, bez této podmínky se nenačtou žádné snímky. U následujících snímků již nezáleží na jejich názvu, musí být ovšem unikátní. Sekce *<section class=”rozložení snímku”>* určuje, jak chce mít snímek rozložen, máme možnost použít *class=“middle“*, což zaručuje, že text bude zobrazen v jednom sloupci uprostřed nebo jeden snímek rozdělit vertikálně na dvě poloviny, levou polovinu jako *class=“left“*, a pravou jako *class=“right“*. Tvorba obsahu snímku využívá HTML značky pro formátování textu, je možné vložit i vlastní CSS kód, pro úpravu jednotlivých elementů. *<aside class=“note“>* obsahuje poznámky, které jsou odesílány do mobilního ovladače. U poznámek je opět možné využívat značky *<p>* pro odstavec, *<b>* pro tučný text, *<i>* pro kurzívu nebo *<u>*pro podržení.

```
<div class="slide" id="video">
  <section class="middle">
  <div id="video-column" class="center">
  <video width="800" id="clip">
  <source src="src/chrome_japan.mp4" type='video/mp4' />
      </video>
    </div>
  </section>
```

V kódu zobrazeném výše je ukázáno vložení videa do snímku. Syntaxe je stejná jako u jakéhokoliv jiného HTML5 dokumentu. Je vytvořen div s id *video-column*, následuje nastavení šířky videa a zarovnání na střed snímku. Následuje načtení souboru v elementu <*source*> a nastavení typu videa, v tomto případě jde o formát mp4. HTML5 podporuje videa ve formátu *.ogv, *.mp4 a *.webm. [4. whatwg.org, 2012]

```
<img src="obrazek.gif">
```

Pokud je potřeba vložit do prezentace obrázek, je možné opět využít syntaxi HTML, jak předvádí kód výše. Jednotlivé snímky mohou obsahovat veškeré prvky, které je možné vložit do HTML souboru, kromě výše zmíněných je možné použít zvukové soubory, vytvářet funkční tlačítka, odkazy, flash, mapy a další.

6.3.2 Soubor main.js

Soubor *main.js* obsahuje funkce, které ovládají zobrazení prezentace, přepínání snímků, připojení k websocket serveru. Tento soubor je součástí prezentační platformy od společnosti Google, proto zde budou popsány pouze změny, které byly nutné udělat, aby se prezentace dala ovládat vzdáleně.

6.3.2.1 Navázání komunikace

```
var connection = new WebSocket('ws://localhost:1367');
```

Vytvoření instance websocket, připojuje se na *localhost*, neboť serverová aplikace běží na počítači, na kterém je zároveň zaplá prezentace. Komunikace probíhá na portu *1367*. Pokud by server běžel na jiném zařízení či jiném portu, stačí změnit tyto údaje.

6.3.2.2 Pole snímků

```
var arrSlidesID = new Array();
$('div.presentation div.slides div.slide').each(function() {
    arrSlidesID.push($(this).attr('id'));
});
```

Po načtení stránky je vytvořeno pole, které je naplněno id jednotlivých snímků. Tyto údaje jsou později využívané pro určení snímku, na kterém se má video přehrát, pozastavit či zastavit úplně.

6.3.2.3 Websocket komunikace

```
connection.onmessage = function(event){
switch(event.data) {
    case 'next':
        simulateKeypress({'keyCode':39});
    break;
    case 'prev':
        simulateKeypress({'keyCode':37});
    break;
    case 'play':
        funcVideoControl(arrSlidesID[intCurrentSlide], event.data);
    break;
    case 'pause':
        funcVideoControl(arrSlidesID[intCurrentSlide], event.data);
    break;
    case 'stop':
        funcVideoControl(arrSlidesID[intCurrentSlide], event.data);
    break;
    }}});
```

Při přijetí zprávy klientem, se pomocí přepínače *switch* rozhodne, které z funkcí se má zavolat, podle obsahu zprávy. Pokud je obdržena zpráva *next*, která označuje přesun na následující slide, je zavolána funkce *simulateKeypress* s parametrem *keyCode:39*. Tento keycode označuje šipku vlevo. Při obdržení zprávy *prev*, označující návrat na předchozí slide je opět zavolána funkce *simulateKeypress* s parametrem *keyCode:37*, označující šipku vlevo. Ovládání videa, které proběhne po přijmutí zprávy *play*, *pause* nebo *stop*, je řízeno funkcí *funcVideoControl* se dvěma vstupními parametry. Jedním je id slidu, které je načteno z pole snímků a druhým je text příchozí zprávy.

6.3.2.4 Ovládání videa

```
function funcVideoControl(strSlideID, strAction) {
if($("div#" + strSlideID + " section.middle video").length > 0) {
var objCurrentVideoClip = $("div#" + strSlideID + " section.middle video")[0];
try {
    switch(strAction){
        case 'play':
        objCurrentVideoClip.play();
        break;
        case 'pause':
        objCurrentVideoClip.pause();
        break;
        case 'stop':
        objCurrentVideoClip.currentTime = 0;
        objCurrentVideoClip.pause();
        break;}
} catch(error) {
alert("Few problems with video playback have occured.\n" + error); }
}}
```

Funkce *funcVideoControl* slouží k ovládání videa vloženého v prezentaci, má dva vstupní parametry, prvním je id snímku a druhým je akce, která se má provést. Nejprve dojde k ověření, jestli je na snímku přítomné video, pokud není, nebude se spouštět přehrávání. Pokud je video přítomné, je uložen zdroj videa do proměnné *objCurrentVideoClip*. Poté následuje try catch blok, obsahující ovládání videa. Pomocí přepínače *switch* je rozhodnuto, jaká zpráva byla přijata. Pokud byla přijata zpráva *play* je spuštěn příkaz *objCurrentVideoClip.play()*, jenž spustí přehrávání videa. V případě přijmutí zprávy *pause*, která označuje pozastavení videa je použit příkaz *objCurrentVideoClip.pause()*. A v posledním případě, což je zastavení videa, označovaného zprávou *stop*, je čas přehrávání videa nastaven na počátek a následuje pozastavení, jako v předchozím případě.

6.3.2.5 Funkce simulateKeypress

```
function simulateKeypress(params) {
    var paramsDefault = {'keyCode':'39'};
    for(var index in paramsDefault) {
        if(params[index] == 'undefined') {
            params[index] = paramsDefault[index];
        }}
    var event;

    event = document.createEvent('HTMLEvents');
    event.initEvent('keydown',true, true);
    event.keyCode = params['keyCode'];
    document.dispatchEvent(event);
```

Tato funkce simuluje stisk klávesy, dle toho, s jakým parametrem je zavolána. V proměnné *paramsDefault* je uložena hodnota *keycode 39*, označující šipku doprava.

For-each cyklem se projde pole *paramsDefault* a pokud je index ve vstupním poli *params* s prázdnou hodnotou, označovanou *undefined,* je vložen index z pole *paramsDefault*. Následuje úsek simulující stisk tlačítka podle hodnoty uložené v poli *params* u identifikátoru *keyCode.*

6.3.2.6 Video při posunu snímku

Prezentace při posunu snímku nechává přehrávat video, bylo proto nutné upravit funkci posunu snímku tak, aby bylo video zastaveno.

```
if($("div#" + arrSlidesID[this._getCurrentIndex() - 2] + " section.middle video").length > 0) {
funcVideoControl(arrSlidesID[this._getCurrentIndex()-2], 'stop');
}
```

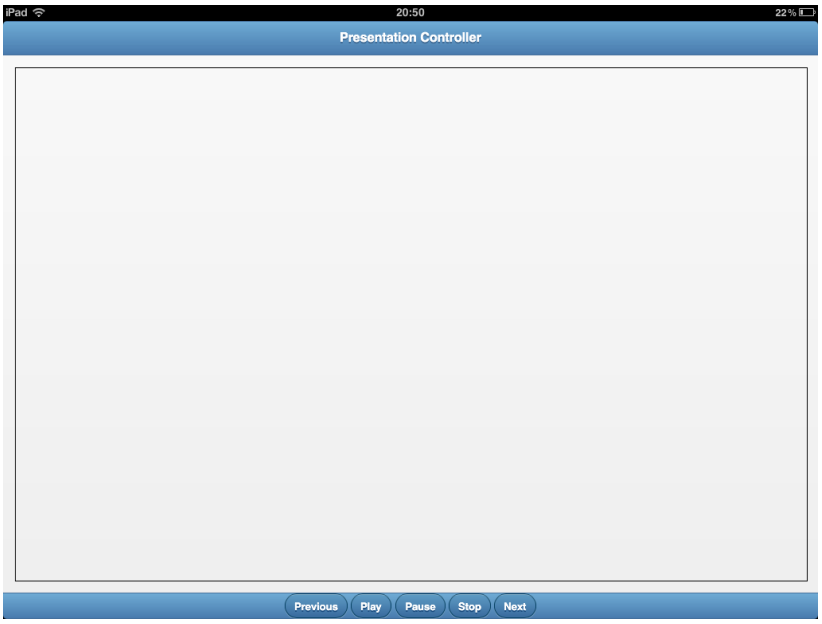
Úsek kódu výše zjistí, zda je na aktuálním snímku video. Pokud ano, je zavolána funkce *funcVideoControl* s dvěma parametry, číslem aktuálního snímku a textem *stop*, značící zastavení přehrávání. Ověření, zda je na snímku video, probíhá při posunu na následující i předchozí slide.

7. Testování

Testování mobilního ovladače probíhalo na třech různých platformách, celkově na pěti zařízeních. Následující kapitola bude rozdělena na test konzole při použití různých mobilních systémů, a test serveru. Zkoušky na všech zařízeních probíhaly stejně, přičemž bylo testováno správné vykreslování prvků ovladače při otáčení, připojení k HTTP a Websocket serveru, posílání a příjem zpráv a správné formátování příchozích poznámek. Test funkčnosti serverové části probíhal pod systémy Windows a Debian. Testování na mobilních zařízeních bylo prováděno třemi dobrovolníky.

7.1 Mobilní konzole

Testování na platformě iOS probíhalo na tabletu iPad druhé generace s poslední verzí mobilního operačního systému iOS6. V případě tohoto zařízení proběhly veškeré testy v pořádku a nedocházelo k žádným chybám. Testování proběhlo i v programu XCode pomocí simulátoru iPadu. I v tomto případě bylo vše v pořádku. Připojení k HTTP i Websocket serveru proběhlo v mobilním prohlížeči v pořádku. Nižší verze systému iOS testovány nebyly z důvodu nekompatibility s technologii Websocket. Na obrázku níže je náhled na mobilní konzolu v iPadu 2.



Obrázek 7: Ovládací konzole na iPadu

Další platformou pro zkoušení byl Google Android. Jako zařízení byl vybrán mobilní telefon Sony Ericsson Xperia Mini s verzí systému 4.0.4 a tablet Acer Iconia Tab A500

s verzí 4.0.3. Vestavěné mobilní prohlížeče v těchto systémech nepodporují technologii Websocket poslední verze, byl proto nainstalován prohlížeč Google Chrome, jenž je schopen navázat spojení. Během testování se nevyskytovali žádné problémy ani na jednom zařízení.

Pro testování na platformě Windows byl vybrán tablet Fujitsu Stylistic Q550. Jako operační systém byl zvolen Windows 8 Pro 64 bit, neboť v době tvorby této práce nebylo možné testovat na Windows RT. Jako prohlížeč byl zvolen Internet Explorer 10 a Chrome 23. Přestože IE10 podporuje Websocket verze RFC 6455, konzole nebyla schopna navázat komunikaci se serverem. Testování proto proběhlo v prohlížeči Google Chrome, kde veškeré testy byly úspěšné.

Testování na mobilních platformách bylo úspěšné a všechny funkční prvky pracovaly správně. Během testování mobilní konzole se nevyskytly žádné problémy či anomálie. Připomínka od dobrovolníka, který testoval aplikaci na tabletu Acer Iconia Tab A500 byla k velikosti ovládacích prvků. Tento problém byl ovšem ojedinělý a na žádném jiném zařízení se nevyskytoval.

7.2 Testování serverové části

Pro testování HTTP a Websocket serveru byly zvoleny operační systémy Microsoft Windows ve verzích XP (32 a 64 bitů), Vista (32 a 64 bitů), 7 Professional (64 bitů) a Windows 8 Pro (64 bitů). Testování v těchto systémech spočívalo v zapnutí serveru a prezentace a připojení mobilní konzole. Jako největší problém se ukázalo systémové omezení z důvodu nedostatečných práv běžného uživatele počítače. Firewall, obsažený v systému Windows, vyžaduje administrátorská práva pro povolení komunikace Node.js skrz síť. Pokud byla práva uživateli přidělena, proběhlo testování ve všech verzích systému korektně.

Jako alternativa k systému Windows byl postaven server, na který byl nainstalován Debian Squeeze. Do tohoto systému byl zaveden balíček Node.js a překopírovány soubory s prezentací, ovladačem a serverem. Takto vytvořený celek byl připojen k internetu s veřejnou IP adresou. Díky tomuto řešení bylo možné spustit prezentaci ve webovém prohlížeči na libovolném počítači připojeném k internetu. Pro ovládání této prezentace stačilo otevřít stejnou adresu, ovšem s portem 8080. Po navázání komunikace bylo možné ovládat prezentaci bez nutnosti přenášet soubory na jakémkoliv paměťovém médiu.

8. Možná vylepšení

Při tvorbě této práce byl brán zřetel na možná rozšíření ať už v rozhraní ovladače, prezentace či v serverové části. Z tohoto důvodu jsou některé části kódu připraveny na dopsání dalších funkcí.

8.1 Rozšíření ovladače

V případě mobilní konzole by bylo možné přidat další tlačítka na ovládání prezentace, ať už by to byla změna velikosti, která je prozatím možná pouze přímo v prezentaci pomocí klávesové zkratky *Ctrl* + *či Ctrl* –, nebo změna CSS souboru, které nyní ovládá tlačítko *T* na klávesnici. Dalším nápadem bylo vytvoření lišty s posuvníkem, který bude ukazovat stav aktuálně přehrávaného videa s možností posunu přehrávání dozadu či dopředu. A v neposlední řadě upravit nativní aplikaci tak, aby si sama našla vytvořený server a připojila se k němu a nebylo tak nutné zadávat adresu serveru ručně.

8.2 Úprava serverové části

Současné řešení serverové části je podmíněno spuštěním na počítači s prezentací. Ovšem ne vždy má prezentující dostatečná práva v systému Windows, aby mohl spouštět exe soubory. Aby bylo přesto možné použít toto řešení, musela by serverová část být v mobilním zařízení. Na internetu je několik možností, jak v systému Android spustit Node.js a tím pádem rozběhnout server na mobilním zařízení. Takovéto vylepšení, které by ovšem bylo funkční pouze na mobilním systému od Googlu, by umožnilo prezentujícímu spouštět v počítači pouze HTML soubor s prezentací a ovladač se serverem by byl součástí aplikace v přenosném zařízení.

8.3 Vylepšení prezentace

Jedna z možností vylepšení prezentace je vytvoření nových CSS stylů, které změní vzhled prezentace. Další možností je odesílání počtu snímků a čísla aktuálního snímku do mobilního ovladače, aby měl prezentující přehled o tom, v jaké fázi přednášky je. Jak bylo zmíněno výše, v případě ukazatele stavu přehrávání na obrazovce mobilního zařízení je nutné zajistit odesílání času přehrávání z prezentace.

9. Závěr

Na závěr této práce lze říci, že byly splněny dané cíle. Díky použití moderních technologií Node.js, Websocket a dalších se podařilo vytvořit rychlý a snadno ovladatelný nástroj, který zjednodušuje prezentování díky dálkovému ovládání a zobrazení poznámek pro přednášejícího.

V první části práce byly přestaveny technologie, které byly použity při tvorbě softwaru této práce. V těchto pasážích je možné nalézt informace o Node.js, tabulky zobrazující podporu technologie Websocket v různých mobilních i desktopových prohlížečích a kompatibilitu mobilních operačních systémů s nástrojem Phonegap. V části zabývající se implementací byly představeny jednotlivé úseky kódu s popisem jejich funkce, popřípadě možného rozšíření.

Testování v ostrém provozu proběhlo na 3. lékařské fakultě Univerzity Karlovy. Po tomto testu byl vyjádřen zájem o nasazení serveru zajišťující ovládání prezentací ze strany pracovníků fakulty.

Tvorba této práce pro mne měla přínos z hlediska práce s novými technologiemi jak na počítačích, tak na mobilních zařízeních s různými operačními systémy.

10. Použité zdroje

1. Node.js [online]. 2012 [cit. 2012-11-08]. Dostupné z: http://nodejs.org/.

2. Websocket.org [online]. 2012 [cit. 2012-11-08]. Dostupné z: http://www.websocket.org/aboutwebsocket.html

3. W3.org [online]. 2012 [cit. 2012-11-08]. Dostupné z: http://www.w3.org/TR/websockets/.

4. Node.js [online]. 2012 [cit. 2012-11-08]. Dostupné z. http://nodejs.org/api/http.html#http_event_request.

5. Whatwg.org. Whatwg.org [online]. 2012 [cit. 2012-11-08]. Dostupné z: http://www.whatwg.org/specs/web-apps/current-work/multipage/the-video-element.html#video

6. Phonegap.com [online]. 2012 [cit. 2012-11-08]. Dostupné z: http://phonegap.com/about/feature

7. jQuery.com [online]. 2012 [cit. 2012-11-08]. Dostupné z: http://jquery.com/.

Seznam obrázků

11. Seznam příloh

Příloha A: Obsah přiloženého CD

Příloha B: Manuál k balíku aplikací

Příloha A

- Adresář *Prezentacni_software* obsahuje podadresáře a soubory:
  - *GoogleChromePortable* – obsahuje přenosnou verzi prohlížeče Chrome
  - *Node* – obsahuje platformu Node.js
  - *Ovladac* – obsahuje mobilní ovládací konzoli
  - *Prezentace* – obsahuje prezentační platformu
  - *Server* – obsahuje serverovou část aplikace
  - *Start_server* – batch soubor, který spustí prezentaci a server
  - *Manual* – PDF soubor s návodem k použití
- Adresář *Zdrojove_kody* – obsahuje vyexportované zdrojové kódy
- Adresář *MacOS-Workspace* – obsahuje workspace ovladače z XCode
- Text bakalářské práce *Mobilní konzole pro ovládání prezentací v HTML* ve formátu PDF a doc

Příloha B

Návod k použití

1. Instalace a spuštění

Balík programů je vytvořen jako přenosná verze, není tudíž nutné spouštět instalaci. Stačí pouze překopírovat do počítače či na externí paměťové médium všechny soubory a z tohoto úložiště je možné spustit program.

Pro spuštění prezentace a serveru slouží soubor *start_server.bat*, který spustí prezentaci v obsaženém prohlížeči Google Chrome a zároveň spustí server, který zajistí komunikaci mezi klienty.

2. Ovládací rozhraní

Pokud chcete ovládat prezentaci z mobilního zařízení, je nutné nejprve zapnout server, jak bylo popsáno výše. Pro zobrazení ovládací konzole stačí otevřít v prohlížeči internetu na mobilním zařízení IP adresu počítače s běžícím serverem na portu 8080. Po načtení stránky je zobrazeno vyskakovací okno se žádostí o zadání IP adresy s předvyplněnou adresou. Stačí potvrdit tlačítkem *OK* a proběhne připojení k serveru. Ovládací tlačítka dole umožňují posun na předchozí či následující snímek prezentace a ovládání videa na aktuálním slidu. Při posunu snímku s přehrávaným videem je automaticky přehrávání zastaveno.

3. Tvorba prezentace

Pro tvorbu prezentace je možné využít jakýkoliv textový editor. Po otevření souboru *slides.html* ze složky *Prezentace* v některém z editoru je možné vytvářet jednotlivé snímky ve vyznačené oblasti. Následující úsek kódu zobrazuje strukturu jednoho snímku.

```
<div class="slide" id="landing-slide">  <!-- Specifikace snímku -->
  <section class="middle">    <!-- Obsah snímku a rozložení -->
  <h3>Bakalářská práce</h3>
  <h2>Mobilní konzole pro ovládání prezentací v HTML</h2>
  <h3>Jaroslav Valdauf</h3>
  <h3>Vedoucí bakalářské práce: Martin Malý</h3>
  </section>
  <aside class="note">    <!-- Sekce s poznámkami -->
  Poznámky k prezentaci
  </aside>
</div>
```

V prvním řádku je specifikováno *ID* snímku, je to jednoznačný identifikátor a nesmí se opakovat. Podmínkou je, aby první slide byl pojmenován *landing-slide.* Na následujícím

řádku je možné nastavit uspořádání snímku, v případě parametru *middle* je text psán doprostřed stránky, pokud chceme rozdělit stránku na dva sloupce, vytvoříme dvě sekce s parametry *class="left"* a *class="right"*. Poté již následuje samotný obsah snímku, který je tvořen HTML značkami, jako jakákoliv jiná webová stránka. Na závěr je možné vytvořit oddíl s poznámkami pro prezentujícího, které budou zobrazovány na mobilním zařízení.