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

HTML5, getUserMedia a jeho využití pro práci s kamerou

Vypracoval: Lukáš Hejtmánek

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

České Budějovice 2015

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Lukáš HEJTMÁNEK**
Osobní číslo:	**P12072**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**HTML5, getUserMedia a jeho využití pro práci s kamerou**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Cílem bakalářské práce bude rozbor aktuálního stavu vývoje nové technologie getUserMedia ve webovém prohlížeči, která je součástí HTML5. Tato technologie umožňuje webové stránce přistupovat k webové kameře a mikrofonu. Práce se bude zabývat výhodami a nevýhodami této nové technologie oproti ostatním technologiím, umožňujícím přístup k webkameře. Bude provedeno otestování podpory této nové technologie v dostupných verzích prohlížečů desktopových, tak mobilních (smartphone/tablet). V praktické části bakalářské práce bude zpracována webová aplikace demonstrující využití této technologie v praxi a vytvořena sada příkladů pro její implementaci.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

BIDELMAN, Eric. Creating .webm video from getUserMedia() - Eric Bidelman. Eric Bidelman [online]. 2012 [cit. 2014-03-30]. Dostupné z: http://ericbidelman.tumblr.com/post/31486670538/creating-webm-video-from-getusermedia
BIDELMAN, Eric. Capturing Audio & Video in HTML5. HTML5 Rocks - A resource for open web HTML5 developers [online]. 2012 [cit. 2014-03-30]. Dostupné z: http://www.html5rocks.com/en/tutorials/getusermedia/intro/
DEVLIN, Ian. Using the getUserMedia API with the HTML5 video and canvas elements. HTML5 Hub [online]. 2013 [cit. 2014-03-30]. Dostupné z: http://html5hub.com/using-the-getusermedia-api-with-the-html5-video-and-canvas-elements/
DE ROSA, Aurelio. An Introduction to the getUserMedia API. SitePoint - Learn HTML, CSS, JavaScript, PHP, Ruby & Responsive Design [online]. 2014 [cit. 2014-03-30]. Dostupné z: http://www.sitepoint.com/introduction-getusermedia-api/
HASSMAN, Martin. HTML5, getUserMedia a práce s kamerou. Zdroják - o tvorbě webových stránek a aplikací [online]. 2014 [cit. 2014-03-30]. Dostupné z: http://www.zdrojak.cz/clanky/html5-prace-kamerou/
HTML5.cz - vše co potřebujete vědět o HTML5 [online]. Dostupné z: http://www.html5.cz/
Media Capture and Streams. World Wide Web Consortium (W3C) [online]. 2014 [cit. 2014-03-30]. Dostupné z: http://dev.w3.org/2011/webrtc/editor/getusermedia.html

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

Datum zadání bakalářské práce: **27. března 2014**
Termín odevzdání bakalářské práce: **30. dubna 2015**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

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

V Českých Budějovicích dne 27. března 2014

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 dne 14. dubna 2015

Lukáš Hejtmánek

Anotace

Cílem bakalářské práce bude rozbor aktuálního stavu vývoje nové metody *getUserMedia* ve webovém prohlížeči, která je součástí HTML5. Tato technologie umožňuje webové stránce přistupovat k webové kameře a mikrofonu. V minulosti toto bylo možné pouze pomocí zásuvných modulů, které se musely instalovat. Mezi tyto zásuvné moduly dnes patří převážně Adobe Flash.

Práce se bude zabývat výhodami a nevýhodami této nové technologie oproti ostatním technologiím umožňujícím přístup k webkameře. Bude provedeno otestování podpory této nové technologie v dostupných verzích prohlížečů jak desktopových tak mobilních (smartphone/tablet).

Jako praktická část bakalářské práce poslouží sada příkladů, která bude dostupná na internetu. Hlavním praktickým příkladem bude vytvoření jednoduché webové aplikace umožňující videokonference právě s využitím metody getUserMedia a frameworků třetích stran.

Klíčová slova

HTML5, getUserMedia, webRTC, Media Capture & Stream API, videokonference, video-chat, webkamera

Abstract

The AIM of this thesis is the current analysis of the development of new method *getUserMedia* in web browser, which is a part of HTML5. This technology enables the website to approach to the webcam or microphone. In the past this was possible only via using of plugins, which had to be installed. Nowadays mainly Adobe Flash belongs to these plugins.

This thesis deals with advantages and disadvantages of this new technology against other technologies enabling to approach to the webcam. There will be carried out testing of support of this new technology in available versions of browsers either desktop or mobile (smartphones/tablets).

As a practical part of my work I will create a set of examples, which will be available on the Internet. The main practical example will be a creation of a simple web application enabling videoconferencing using getUserMedia and third-party frameworks.

Keywords

HTML5, getUserMedia, webRTC, Media Capture & Stream API, video-conference, videochat, webcam

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

Multimediální soubory jsou na webu čím dál více využívány. Pro přehrávání videí, či přehrávání zvuku zde ale doposud bylo potřeba využít některého z doplňků a nestačil tedy pouze prohlížeč. Díky novému standardu HTML5, který již implicitně podporuje elementy, jako jsou <video> a <audio>, mohou tvůrci webů využívat videa či zvukové stopy bez použití doplňků pouze s využitím těchto elementů.

S podporou HTML5 přichází i nové Media Capture & Stream API, které umožňuje webové stránce streamovat multimediální objekty do různých elementů. Součástí Media Capture & Stream API je metoda getUserMedia, která umožňuje přístup k webkameře a mikrofonu na straně uživatele, který se pomocí této technologie může např. natáčet, nebo fotografovat. V případě, kdy chceme obraz, zvuk, nebo jakékoliv jiné datové soubory přenášet přes internet přímo k jinému uživateli bez využití služeb třetích stran (typicky servery, databanky apod.), můžeme využít technologie WebRTC, která mezi uživateli vytvoří peer spojení – konkrétně peer2peer.

Technologiím HTML5, WebRTC a hlavně getUserMedia se budu ve své práci věnovat.

1.1 Cíle práce

Cílem bakalářské práce bude rozbor aktuálního stavu vývoje nové technologie getUserMedia, která je součástí HTML 5. Tato technologie umožňuje webové stránce přistupovat k webové kameře a mikrofonu. Díky tomu odpadá nutnost využívání modulů a aplikací třetích stran, které tuto funkci obsluhovaly doposud. Nejpoužívanější je dnes Adobe Flash.

Práce se bude také zabývat výhodami a nevýhodami této nové technologie oproti ostatním technologiím umožňujícím přístup k webkameře. Bude provedeno otestování podpory této nové technologie v dostupných verzích prohlížečů jak desktopových, tak mobilních (smartphone/tablet). Dále bude zpracována webová aplikace demonstrující využití této technologie v praxi a vytvořena sada příkladů pro její implementaci.

1.2 Východiska práce

HTML5 přináší spoustu novinek v podobě podpory nových funkcí, které mohou být implementovány na webových stránkách či aplikacích. Díky těmto novinkám odpadá v mnoha případech nutnost využívání doplňků, které tyto multimediální funkce umožňovaly. Mezi novinky patří i metoda getUserMedia, která umožňuje webové stránce přistupovat přes prohlížeč k webkameře či mikrofonu daného zařízení. Díky tomu jsme schopni pořizovat přes webové stránky fotografie a videozáznamy bez použití zásuvných modulů.

S využitím projektu, který vydal Google pod názvem webRTC, nyní můžeme sdílet multimediální obsah pomocí peer2peer spojení mezi dvěma i více uživateli. Metoda getUserMedia se stala jednou z nejdůležitějších komponent při tvorbě webRTC API.

1.3 Metody práce

V praktické části své bakalářské práce vytvořím sadu příkladů pro demonstraci možností nové metody getUserMedia v praxi. V této sadě budou dostupné zdrojové kódy pro implementaci této metody. Jako hlavní praktický příklad vytvořím webovou aplikaci, která bude využívat této nové technologie pro video-chat a videokonference.

V teoretické části své bakalářské práce se budu v úvodu zabývat zasvěcením do problematiky HTML5. Další částí bude rozbor projektu WebRTC. V návaznosti na projekt WebRTC popíši metodu getUserMedia a její jednotlivé části tak, aby tento text mohl posloužit jako návod pro její implementaci. V této části také otestuji podporu metody getUserMedia a jejích jednotlivých funkcí v aktuálních dostupných verzích prohlížečů. Po tomto rozboru porovnám tuto novou metodu s ostatními možnostmi, jak přistupovat k webkameře a mikrofonu.

2 HTML

HTML neboli Hyper Text Markup Language je značkovací jazyk, který umožňuje vytvářet webové stránky šířené pod standardem World Wide Web. HTML vzniklo z univerzálního jazyka SGML, ze kterého vznikl například i jazyk XML.

HTML používají webové prohlížeče pro interpretování textu, obrázků a dalších částí kódu webové stránky do vizuální podoby. Pro stylování dokumentu můžeme použít jazyk CSS. [1]

2.1 Historie HTML

Jazyk HTML vznikal pro potřeby výzkumného centra CERN od roku 1989, kdy Tim Berners-Lee pracoval spolu s Rebertem Caillau na projektu, který měl vytvořit informační systém pro publikování a sdílení vědeckých článků v rámci počítačové sítě CERNu. První verze jazyka HTML byla publikována v roce 1991 v dokumentu nazvaném HTML Tags. Zároveň s uvedením svého jazyka vytvořil Tim Berners-Lee první webový prohlížeč, který pojmenoval WorldWideWeb. [1]

Od roku 1991 do roku 1999 vyšlo postupně několik verzí jazyka HTML až do verze 4.01. O vydávání standardů tohoto jazyka se v průběhu vývoje začalo starat W3C, které má na starosti vývoj HTML dodnes. O HTML verzi 4 se předpokládalo, že bude poslední verzí jazyka HTML, který nahradí jazyk XHTML využívající XML. Jazyk XHTML vznikl spojením již stávajícího HTML s jazykem XML a přinesl některé nové elementy.

Jazyk XHTML se dělí na tři různé specifikace:

- XHTML Strict,
- XHTML Transitional,

- XHTML Frameset.

Tyto verze se od sebe liší převážně v množství povolených elementů, možností využívat rámce, či různými metodami stylování pomocí CSS. [2]

2.2 HTML5

HTML5 vzniklo díky společnému vývoji W3C a iniciativy WHATWG. Jako základ pro práci posloužila specifikace vytvořená právě WHATWG. Práce na HTML5 začaly již v roce 2008 a tento standard byl schválen v říjnu 2014. Syntaxe v HTML5 již nevychází z jazyka SGML, jako tomu bylo u starších verzí, přesto byla navržena tak, aby umožňovala zpětnou kompatibilitu právě se staršími verzemi HTML. HTML5 dále obsahuje rozšíření Web Forms 2.0, které bylo dříve vyvíjeno zvlášť iniciativou WHATWG.

HTML5 přináší spoustu nových elementů, které vycházejí z využití na moderních stránkách. Některé z nich v mnoha případech nahradí přemnožené elementy a na elementy, jako jsou <nav>, <footer> a další.

HTML5 také přináší podporu multimediálních souborů a mnoha nových elementů, jako jsou <audio>, <video>, <canvas> apod. Všechny tyto nové elementy umožňují vytvářet webové stránky mnohem jednodušeji, rychleji a přesněji, než tomu bylo doposud. Změnou prošel i samotný DOCTYPE, který je v současné verzi mnohem jednodušší na zápis. Základní struktura stránky v jazyce HTML5 může tedy vypadat takto:

```
<!DOCTYPE html>
    <head>
        <title> </title>
    </head>
    <body>
        <header> </header>
```

```
            <nav> </nav>
            <article>
                 <section> </section>
            </article>
            <aside> </aside>
            <footer> </footer>
      </body>
 </html>
```

Díky této změně je nyní základní kostra webu mnohem přehlednější. [3]

2.2.1 Nová API

HTML5 s sebou přináší podporu mnoha nových funkcí, které jsou publikovány jak na webu W3C, tak na webu WHATWG. Nových API je velké množství, a tak se v následujícím seznamu pokusím vypsat ty nejzajímavější, které HTML5 přináší a ve stručnosti je popsat.

2.2.1.1 App Cache

Toto API umožňuje uložení webových aplikací do cache paměti a umožňuje takto uchovaným aplikacím běžet v režimu offline – bez připojení k internetu. Tuto funkci momentálně využívá například Google v jejich Google Drive Apps. Díky App Cache mají webové aplikace kromě výhody běhu v režimu offline i další výhody. Pro uživatele je to jednoznačně rychlost běhu aplikace a pro server je to menší zatížení v důsledku načítání jednotlivých částí aplikace opakovaně. [4]

2.2.1.2 Drag & Drop

Funkce Drag & Drop umožňuje přesouvat objekty mezi jednotlivými elementy na stránce s využitím myši, a to v reálném čase bez nutnosti opětovného načtení stránky. Funkce Drag & Drop se dá využít například při nahrávání souborů na server, kde uživatel nemusí pomocí okna zadávat

cestu k souboru, který chce nahrát, ale stačí, když soubor z počítače přetáhne přímo do okna prohlížeče do místa, které tento soubor zachytí. [5]

2.2.1.3 Geolocation

Geolocation API patří do kategorie funkcí, které spadají pod JavaScriptový objekt Navigator. Tento objekt dokáže zaznamenávat informace o uživateli, konkrétně o verzi prohlížeče, jazyku operačního systému nebo třeba získat přístup k webkameře. Geolocation API zajišťuje získání polohy uživatele. Zmiňované přístupy k webkameře, či k poloze uživatele jsou však podmíněny schválením této funkce v prohlížeči. [6]

2.2.1.4 Microdata

Microdata nám umožňují označovat obsah webové stránky tak, aby byl pro webové vyhledávače a prohlížeče lépe pochopitelný. Jednotlivé tagy Microdata API umožňují zařazovat obsah do kategorií, jako jsou jméno, fotografie, url a další. Microdata API je vyvíjeno iniciativou WHATWG, nikoliv W3C. [7]

2.2.1.5 Media Capture & Stream

Media Capture & Stream API bylo původně vyvíjeno pod projektem WebRTC, avšak později se oddělilo a nyní je vyvíjeno samostatně. Funkce tohoto API – getUserMedia – patří stejně jako getLocation z Geolocation API pod objekt Navigator a zajišťuje přístup ke kameře a následně případný stream do elementu na webové stránce. Bez uživatelova souhlasu tento přístup není možný. [8]

2.2.1.6 Local Storage

Díky Local Storage mohou webové aplikace ukládat informace na lokální úrovni – v prohlížeči uživatele. Dříve musela být tato data ukládána

v cookies. Cookies se uchovávaly na serveru, odkud se musely vždy načíst. Dále pomocí Local Storage můžou webové aplikace ukládat větší množství dat, než tomu bylo doposud. [9]

2.2.2 Podpora v prohlížečích

Mezi prohlížeči jsou stále velké rozdíly v podpoře jazyka HTML5. Největší podpory HTML5 dosahuje prohlížeč Chrome od Googlu, zatímco Internet Explorer od Microsoftu podporuje pouze zlomek nových funkcí.

Řada nových technologií, které HTML5 přináší, je prohlížeči podporována jen částečně, nebo s využitím prefixů. Dobrým zdrojem pro kontrolu podpory v prohlížečích může posloužit http://caniuse.com, kde si můžeme ověřit podporu různých funkcí jak v mobilních, tak desktopových prohlížečích.

3 Úvod k WebRTC

WebRTC je open-source projekt, který umožňuje komunikaci mezi prohlížeči v reálném čase s využitím jednoduchého API. Za tímto projektem stojí zejména tři společnosti – Google, Mozilla a Opera. Celý projekt je ale vyvíjen pod záštitou W3C. Společnostmi, které se také zajímají o tento projekt, jsou Apple, Microsoft, Cisco a další. Díky API projektu WebRTC je mezi prohlížeči možné posílat soubory, provádět videohovory, či vytvořit chat. Všechny tyto funkce jsou vytvářeny tak, aby je bylo možné využívat bez jakýchkoliv zásuvných modulů v prohlížeči. [15]

Projekt WebRTC byl poprvé zveřejněn v roce 2011 firmou Google. Zdrojové kódy tohoto projektu poskytla firma Google jako open-source. Na vývoji se od začátku nepodílel pouze Google ale i společnosti Mozilla a Opera. Díky tomuto zveřejnění se o tento projekt začaly zajímat další společnosti včetně organizace W3C. [16]

Podpora v prohlížečích u projektu WebRTC je dána podporou jeho jednolivých částí. V případě, kdy prohlížeč nepodporuje některou část, WebRTC nebude fungovat korektně. Projekt WebRTC je momentálně podporován pouze v prohlížečích Chrome, Firefox a Opera. [17]

WebRTC API se skládá z několika JavaScriptových API, kterými jsou:

- PeerConnection API,
- DataChannel API
- a Media Capture & Stream API (getUserMedia).

Všechna tato API vznikají jako součást technologie WebRTC, a tudíž je jejich vývoj primárně řízen směrem k WebRTC. Všechna tato API jsou dále v jazyce JavaScript a každé API má svoji roli. MediaStream API má za úkol získat přístup ke kameře, nebo mikrofonu a vytvořit poté pro zvolený zdroj

stream. PeerConnection API má za úkol navázat spojení mezi dvěma uživateli a posílání dat ze streamu, která mají formát videa či audia. A naposled DataChannel API má za úkol s pomocí existujícího spojení posílat a přijímat data jako soubory, či textové zprávy. [18]

3.1 PeerConnection API

RTCPeerConnection je komponenta WebRTC, která zajišťuje spojení mezi dvěma uživateli (peery) a dále přenos dat mezi nimi. Druh dat, který PeerConnection API přenáší, je audio, či video stream. Pro ostatní druhy dat je potřeba pro přenos využít DataChannel API. PeerConnection API obsahuje několik metod, pomocí kterých můžeme vytvářet nabídky na vytvoření spojení, inicializovat samotné spojení a samozřejmě toto spojení i zrušit. Pro tyto účely obsahuje PeerConnection API soubor metod. [16]

3.1.1 Metody

PeerConnection API obsahuje spoustu metod, které mají na starosti základní potřeby, jako jsou vytvoření spojení, nebo jeho zrušení, ale zároveň se starají i o správné kódování přenášeného obsahu a neustálé kontrolování spojení. Níže vypíši základní metody a k čemu slouží. [19]

3.1.1.1 Metoda PeerConnection.createOffer()

Tato metoda vytvoří požadavek na nalezení vzdáleného peeru s danou konfigurací. Metoda PeerConnection.createOffer() obsahuje tři parametry, jedním je successCallback, dalším parametrem je errorCallback a posledním parametrem je volba druhu spojení. [19]

Základní syntaxe metody PeerConnection.createOffer() vypadá takto:

```
pc.createOffer(function(desc){
```

```
 pc.setLocalDescription(desc, function() {
      // odeslání požadavku na server pro spojení
se vzdáleným klientem
   });
}
```
[1]

3.1.1.2 Metoda PeerConnection.createAnswer()

Tato metoda zajišťuje příjem metody PeerConnection.createAnswer() a vytváří na ni odpověď. Metoda PeerConnection.createOffer() obsahuje stejné parametry jako metoda předchozí. [19]

```
pc.createAnswer(function(answer) {
   pc.setLocalDescription(answer, function() {
      // odeslání odpovědi na spojení vzdálenému peeru
  })
})
```
[2]

3.1.1.3 Metoda PeerConnection.getLocalStreams()

Metoda PeerConnection.getLocalStreams() obsahuje Array list všech lokálních streamů, tedy těch, které poskytujeme přes peer spojení. [19]

3.1.1.4 Metoda PeerConnection.getRemoteStreams()

Protože potřebujeme uchovávat informace i o vzdálených streamech, tak využíváme metodu PeerConnection.getRemoteStreams(), která uchovává informace o vzdálených streamech – také v Array listu. [19]

---

[1] Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection#Example

[2] Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection#Example_2

3.1.1.5 Metoda PeerConnection.getStreamById()

Tato metoda vrací objekt MediaStream podle ID, které je spojeno právě s jedním streamem, ať už se jedná o lokální, či vzdálený. [19]

3.1.1.6 Metoda PeerConnection.addStream()

V případě, kdy potřebujeme přidat stream, voláme tuto metodu. Potřeba této metody může nastat v případě, kdy se při vytváření spojení zapomene vytvořit stream pro audio, či video, ale jsou potřeba oba. [19]

3.1.1.7 Metoda PeerConnection.removeStream()

Naopak od minulé metody může nastat situace, kdy chceme přestat sdílet mikrofon, či video. V takovém případě se vyvolá metoda removeStream, která zajistí zrušení streamu požadovaného typu. [19]

3.1.1.8 Metoda PeerConnection.close()

Metoda, která zajišťuje ukončení spojení, se jmenuje PeerConnection.close() a díky ní je možné ukončit spojení mezi dvěma uživateli. [19]

3.1.1.9 Metoda PeerConnection.createDataChannel

PeerConnection.createDataChannel() vytvoří nový objekt RTCDataChannel. Při vytváření tohoto objektu je nutné dodržet jeho syntaxi. [19]

```
var pc = new PeerConnection();
var channel = pc.createDataChannel("Mydata");
channel.onopen = function(event) {
   channel.send('sending a message');
}
```

```
channel.onmessage = function(event) {
    console.log(event.data);
}
```
[3]

3.2 DataChannel API

DataChannel API je součást WebRTC, která zajišťuje přenos dat mezi dvěma uživateli. Typem dat, která toto API dokáže přenášet, jsou jakékoliv soubory, ať už se jedná o fotografii, nebo dokument v jakémkoliv formátu. Kromě posílání souborů dokáže DataChannel API přenášet textové zprávy. Toto API má využití v mnoha webových projektech a aplikacích. Dá se využít pro chatovací aplikace, nebo přenos souborů přímo mezi uživateli bez nutnosti posílat data přes jakýkoliv server. Dalším využitím tohoto API by mohly být také HTM5 hry. Díky DataChannel API by se dal naprogramovat multiplayer, který by běžel v prohlížeči bez nutnosti jakéhokoliv doplňku, či nutnosti serveru pro přenos dat. [16]

DataChannel API obsahuje pouze tři metody, kterými jsou:

- createDataChannel(),
- send()
- a close().

První metoda vytvoří objekt DataChannel, který bude zajišťovat přenos dat mezi uživateli. Další metodou je send(), která zajišťuje odesílání souboru mezi uživateli. Poslední metodou tohoto API je close(), která ukončí posílání souborů a zavře spojení mezi uživateli na úrovni tohoto API. [20]

[3] Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection#Example_5

3.3 Media Capture & Stream API

Poslední částí projektu WebRTC je Media Capture & Stream API. Toto API zajišťuje přístup k webkameře, či mikrofonu daného zařízení a stará se o vytvoření streamu z webkamery, či mikrofonu do jednotlivých objektů a elementů na webové stránce. Celé Media Capture & Stream API se skládá ze dvou částí. První je získání vstupu z webkamer, či mikrofonu a druhou předání tohoto vstupu objektu MediaStream.

Objekt MediaStream se skládá ze vstupu, který se získává pomocí metody getUserMedia a dále z výstupu. Tento výstup můžeme zobrazit buď v některém elementu na webové stránce, nebo můžeme výstup předat dále do objektu PeerConnection. Pokud předáváme výstup do PeerConnection, PeerConnection se postará o přenos streamu mezi uživateli a následně předá stream opět objektu MediaStream na straně příjemce, který se postará o zobrazení výstupu do elementu na webové stránce. MediaStream rozlišuje jednotlivé streamy podle toho, zda se jedná o lokální, či o vzdálené. Díky tomu je možné zobrazit dané streamy v různých elementech na webové stránce. [21]

MediaStream je asi nejdůležitějším objektem celého Media Capture & Stream API, a tudíž se pokusím popsat jeho jednotlivé metody.

3.3.1 Metoda MediaStream.addTrack()

Tato metoda umožňuje ukládání kopie stopy z objektu MediaStreamTrack do MediaStream. V případě, že stopa byla již přidána, neprovede nic. [8]

3.3.2 Metoda MediaStream.clone()

Pro vrácení kopie objektu MediaStream slouží metoda clone(). Tato kopie bude mít vlastní id a bude figurovat jako samostatný nový objekt. [8]

3.3.3 Metoda MediaStream.getAudioTracks()

Metoda MediaStream.getAudioTracks() vrací seznam stop, které jsou uchovávány v objektu MediaStream a které mají atribut nastaven na audio. V případě, že je atribut nastaven na video, je potřeba volat metodu MediaStream.getVideoTracks(). Pořadí objektů v seznamu není specifikováno a záleží tedy jak na prohlížeči, tak na této metodě, v jakém pořadí vrátí dané objekty. [8]

3.3.4 Metoda MediaStream.getTrackById()

Tato metoda vrátí stopu, která se shoduje s požadovaným ID. Pokud není nalezena žádná stopa podle požadovaných kritérií, vrátí null. Pokud je nalezeno více stop se stejným ID, pak vrátí tu, kterou tato metoda nalezla jako první. [8]

3.3.5 Metoda MediaStream.getVideoTracks()

Pro vrácení seznamu video stop slouží metoda MediaStream.getVideoTracks(), která vrátí nesetříděný seznam video stop. Tato metoda funguje stejně jako metoda MediaStream.getAudioTracks(), která má jako parametr vyhledávání nastaveno audio, kdežto metoda MediaStream.getVideoTracks má jako parametr vyhledávání nastaveno video, tudíž vrací pouze stopy, u kterých se jedná o video. [8]

3.3.6 Metoda MediaStream.removeTracks()

Tato metoda smaže audio, či video stopu z objektu MediaStream. [8]

4 Metoda getUserMedia

Hlavním cílem mé bakalářské práce je popsat metodu getUserMedia a otestovat její podporu v dostupných prohlížečích. Tato metoda vznikla jako součást projektu WebRTC. Avšak její využití nekončí pouze u spojení s technologií WebRTC. Díky této metodě můžeme vytvářet webové stránky, které budou schopny streamovat video zachycené z webkamery, pořizovat fotografie, nahrávat video tutoriály či vytvářet podcasty. Toto vše pouze s využitím webového prohlížeče a webové stránky bez využití doplňků třetích stran, jako je Adobe Flash, či Microsoft Silverlight.

Metoda getUserMedia spadá pod objekt *window.navigator*, který zajišťuje přístup k informacím o klientovi, konkrétně o verzi jeho prohlížeče, jazyce operačního systému a dalších.[10]

Základní syntaxe metody getUserMedia je velmi jednoduchá a skládá se pouze ze tří parametrů. Samotná syntaxe vypadá takto:

```
navigator.getUserMedia(
   constraints, successCallback, errorCallback
);4
```

4.1 Parametry

Parametry se dělí do dvou skupin, v první skupině je pouze jeden parametr s názvem *constraints*, který definuje omezení metody getUserMedia. Ve druhé skupině jsou dva parametry *Callback*, jeden parametr pro každý případ – parametr *successCallback* pro kladnou odezvu od uživatele

[4] Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia

a parametr *errorCallback* pro zápornou odezvu uživatele, tedy odmítnutí přístupu, nebo např. pro nepodporující prohlížeč či zařízení. [11]

4.1.1 Parametr constraints

Parametr constraints je MediaStreamConstraints objekt, který obsahuje dva atributy. Prvním atributem tohoto objektu je video a druhým audio. Atributy jsou typu Boolean a mohou nabývat pouze hodnoty *true*, nebo *false*. Parametr *constraints* je u metody getUserMedia povinný a nelze ho vynechat. Pro správné fungování metody getUserMedia musí alespoň jeden z těchto atributů nabývat hodnoty *true*. Pokud je ve výčtu jeden atribut vynechán, jeho hodnota se automaticky nastavuje na *false*. Zde vidíme příklad, kdy si metoda getUserMedia žádá o přístup jak k videu, tak audiu, neboli webkameře a mikrofonu. Oba atributy proto nastavíme na hodnoty true. [11]

```
{
   video: true,
   audio: true
}5
```

4.1.2 Parametr successCallback

Parametr *successCallback* je stejně jako parametr *constraints* povinný a také ho nelze vynechat. Tento parametr obsahuje kód, který se provede v případě, že webová stránka získá od prohlížeče povolení pro přístup k zařízení uživatele. Zde je příklad kódu, který vytvoří objekt pro streamování videa z webkamery do elementu <video> a následně toto video spustí. [11]

[5] Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia

```
function (stream) {
    var video = document.querySelector('video');
    video.src = window.URL.createObjectURL(stream);
    mediaStream = stream;
    video.play(); }6
```

4.1.3 Parametr errorCallback

Tento parametr je jediným volitelným parametrem metody getUserMedia a jeho použití je tedy na programátorovi dané aplikace. Parametr errorCallback, jak již název napovídá, se vyvolá při předem definované chybě, která nastala při získávání přístupu k webkameře, či mikrofonu. Obsah tohoto parametru nám umožňuje daný problém nějakým způsobem vyřešit, vypsat do konzole, či ignorovat. Tento parametr rozeznává tři druhy chyb a těmi jsou:

- PERMISSION_DENIED,
- NOT_SUPPORTED_ERROR
- a MANDATORY_UNSATISFIED_ERROR. [11]

Co se týče první chyby PERMISSION_DENIED, tak té je dosaženo v případě, kdy webová stránka nedostane povolení přistupovat k zařízení, ke kterému si dle prvního parametru metody getUserMedia – *constraints* – přístup vyžádala. [11]

Další možnou chybou při přístupu k webkameře, či mikrofonu je NO_SUPPORTED_ERROR. Tato chyba se vyvolá v případě, kdy prohlížeč, přes který uživatel přistupuje na webové stránky, nepodporuje přístup

[6] Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia

k zařízení, které jsme si vyžádali v parametru *constraints*, nebo v případě, kdy nepodporuje daná specifika, která jsme si vyžádali. [11]

Poslední chybovou hláškou parametru *errorCallback* je chyba MANDATORY_UNSATISFIED_ERROR. Tato chyba se vyvolá v případě, kdy webová stránka žádá o přístup k webkameře, či mikrofonu, které však na daném zařízení nejsou k dispozici. V případě, kdy webová stránka žádá o přístup k webkameře i mikrofonu, avšak na daném zařízení je dostupná pouze webkamera, nebo pouze mikrofon, daná chyba nenastane a webová stránka získá přístup k zařízení, které je dostupné. [11]

4.2 Jak probíhá přístup ke kameře a mikrofonu

Webová stránka nemůže bez souhlasu uživatele přistupovat k webkameře, ani mikrofonu. V každém prohlížeči musí být naprogramovaná metoda, která uživatele vyzve k reakci na žádost o poskytnutí webkamery a mikrofonu. U některých prohlížečů jsou stále rozdíly v podpoře tohoto bezpečnostního opatření a výběru konkrétního zařízení pro sdílení.

V prohlížečích Opera, Chrome a Firefox dostane uživatel na výběr, jakou konkrétní webkameru, či mikrofon chce sdílet, kdežto u českého prohlížeče Seznam.cz uživatel může pouze zvolit, zda chce, či nechce sdílet. Prohlížeč mu již nijak nedá vědět, jaké zařízení se sdílí s webovou stránkou, což považuji za velký bezpečnostní nedostatek.

4.3 Aktuální stav vývoje

Jak jsem již uváděl výše, metoda getUserMedia je součástí Media Capture & Stream API, které se začalo vyvíjet jako samostatné API v roce 2011. Toto API bylo do té doby součástí projektu WebRTC, pro který je metoda getUserMedia jednou ze tří klíčových funkcí. Původní

název nesl označení getUserMedia API, které se ale později, jak se metoda rozšiřovala, přejmenovalo na Media Capture & Stream API. [8]

Metoda getUserMedia je stále ve vývoji. Stále se opravují chyby, které nastávají převážně při spojení této metody s WebRTC. Pro hlášení chyb, které souvisí s WebRTC a getUserMedia, vytvořila pracovní skupina, která se stará o vývoj těchto technologií, veřejný Google dokument, kde může každý nahlásit chybu. Tabulka pro hlášení chyb je dostupná na této adrese: https://code.google.com/p/webrtc/issues/list. [12]

Pokud se podíváme na stránky W3C a na těchto stránkách si najdeme Media Capture & Stream API, zjistíme, že toto API je stále ve fázi *„working draft“,* to znamená, že na jeho finální podobě se stále pracuje a vývoj ještě nebyl dokončen. [8]

4.4 Podpora v prohlížečích

V každém zařízení, ať už se jedná o desktop, tablet, smartphone, či třeba chytré hodinky, máme k dispozici operační systém, který definuje funkce daného zařízení a pro který existují různé webové prohlížeče. Těchto prohlížečů jsou spousty a jsou vyvíjeny velkými společnostmi, organizacemi, i malými studii.

Záleží na vývojářích prohlížečů, jak rychle dokáží přidat nové funkce do svého prohlížeče. Z tohoto důvodu je pro vývojáře webových aplikací a prezentací klíčová znalost podpory jednotlivých technologií podporovaných v prohlížečích. Některé prohlížeče jsou na podporu getUserMedia připravovány již delší dobu (Chrome, Opera, Firefox), jiné o této metodě nejspíš ještě neslyšely (IE, Safari). V testu prohlížečů jsem se zaměřil jak na desktopové prohlížeče, tak na jejich mobilní verze.

4.4.1 Testování

Pro testování prohlížečů jsem vytvořil jednoduchou webovou stránku dostupnou na adrese: http://bp.lukashejtmanek.cz/pages/gum.html, která uživateli zobrazí, zda jeho prohlížeč, ze kterého na stránku přistupuje, podporuje getUserMedia, či nikoli.

Samotný kód, který používám na testovací stránce pro otestování podpory, je následující:

```
if (!navigator.getUserMedia) {
    document.getElementById('gumko').style.display =
'block';
} else {
    document.getElementById('gumok').style.display =
'block';
}
```

Tento kód funguje na principu jednoduché podmínky. Když není nalezen navigator.getUserMedia, tak se uživateli zobrazí element informující o nepodpoře stávajícího prohlížeče. V opačném případě, tedy pokud je navigator.getUserMedia nalezen, informuje uživatele o jeho podpoře.

4.4.1.1 Desktopové prohlížeče pro Mac

Z prohlížečů pro operační systém OS X jsem pro testování vybral čtyři nejznámější zástupce z kategorie prohlížečů, kterými jsou Safari, Opera, Firefox a Chrome. Pro zajímavost jsem přidal český prohlížeč Seznam.cz a dále Maxthon a Torch.

Prohlížeč	Verze	Podpora
Safari	8.0.3	Ne
Opera	27.0.1689.76	Ano
Firefox	34.0.5	Ano
Chrome	41.0.2272.76	Ano
Seznam.cz	1.0.17	Ano
Maxthon	4.1.3.5000	Ano
Torch	29.0.0.7181	Ano

Tabulka č. 1 – Podpora getUserMedia v prohlížečích pro Mac

4.4.1.2 Desktopové prohlížeče pro Windows

Mezi prohlížeči pro Windows jsem testoval nejznámější čtyřku prohlížečů včetně Internet Explorer od Microsoftu a pro zajímavost jsem přidal i výsledek prohlížečů Seznam.cz, Torch a Maxthon. Jediný prohlížeč Seznam.cz nepodporuje výběr zdroje videa.

Prohlížeč	Verze	Podpora
Internet Explorer	11.0.9600	Ne
Opera	27.0.1689	Ano
Firefox	36.0.1	Ano
Chrome	40.0.2214	Ano
Seznam.cz	1.1.0	Ano
Maxthon	4.4.4.2000	Ano
Torch	39.0.0	Ano

Tabulka č. 2 – Podpora getUserMedia v prohlížečích pro Windows

4.4.1.3 Desktopové prohlížeče pro Linux

Na systému Linux, konkrétně distribuce Ubuntu, jsem testoval opět hlavní trojici prohlížečů (Firefox, Chrome, Opera), které jsou na daném systému dostupné. Dále jsem testoval prohlížeč Chromium, ze kterého vychází prohlížeč Chrome a jako poslední jsem otestoval prohlížeč QupZilla.

Prohlížeč	Verze	Podpora
Firefox	36.0	Ano
Chrome	41.0.2272	Ano
Opera	27.0.1689	Ano
QupZilla	1.8.6	Ne
Chromium	40.0.2214	Ano

Tabulka č. 3 – Podpora getUserMedia v prohlížečích pro Linux

4.4.1.4 Mobilní prohlížeče pro iOS

Z prohlížečů pro operační systém iOS 8 jsem vybral celkem 8 zástupců, z nichž pouze jeden vyšel z testu s kladným výsledkem. Avšak v tomto případě se jedná o webový prohlížeč Bowser, který je vytvářen přímo pro testování této metody v praxi. Bohužel tento prohlížeč ani nenabízí možnost výběru zdroje.

Prohlížeč	Verze	Podpora
Safari	8.1.3 (iOS)	Ne
Bowser	0.4	Ano
Mercury Browser	8.9.4	Ne
Seznam.cz	1.0.4	Ne
Kaspersky Safe Browser	1.5.1	Ne
Chrome	40.0.2214.73	Ne
Opera Mobile	9.2.0	Ne
Opera Coast	4.03	Ne

Tabulka č. 4 – Podpora getUserMedia v prohlížečích pro iOS

4.4.1.5 Mobilní prohlížeče pro Android

Na operačním systému Android jsem testoval tři hlavní zástupce z kategorie prohlížečů – Chrome, Firefox a Operu. Dále jsem testoval defaultní prohlížeč Internet Browser (HTC). U tohoto prohlížeče jsem nepředpokládal, že by metodu getUserMedia podporoval. Tuto domněnku výsledková tabulka dokazuje. Dalšími prohlížeči byly opět mobilní verze alternativních prohlížečů Maxthon, UC Browser, Seznam.cz a Dolphin.

Prohlížeč	Verze	Podpora
Internet (prohlížeč HTC)	7.0.2513	Ne (Ano)
Chrome	40.0.2214	Ano
Firefox	36.0.1	Ano
Opera	28.0.1764	Ano
Maxthon	4.4.1	Ne
UC Browser	10.2.0	Ano
Dolphin Browser	11.4.3	Ne (Ano)
Seznam.cz	2.0.5	Ne

Tabulka č. 5 – Podpora getUserMedia v prohlížečích pro Android

Z prohlížečů pro mobilní zařízení fungujících na OS Android bych rád vyzdvihl několik zástupců, kteří svou podporou předčily i některé desktopové prohlížeče. Mezi tyto výjimečné prohlížeče patří Firefox, který dokonce nabízí i možnost volby, jakou kameru chceme použít jako zdroj. Stejnou podporu jako u prohlížeče Firefox jsem zaznamenal u prohlížečů Opera a UC Browser. Naopak prohlížeč Chrome nenabídl žádnou možnost výběru zdroje videa a rovnou zobrazoval video z přední kamery. Zajímavostí je, že prohlížeče Internet a Dolphin se snaží tvrdit, že metodu getUserMedia znají, avšak při žádosti o přístup k zařízení neodpovídají.

4.4.1.6 Mobilní prohlížeče pro Windows Phone

Na mobilním zařízení s operačním systémem Windows Phone 8 jsem testoval kromě prohlížeče Internet Exploter ještě několik nejstahovanějších

prohlížečů. Z výsledků testů vyplynulo, že pro mobilní operační systém od Microsoftu zatím není žádný prohlížeč, který by podporoval metodu getUserMedia.

Prohlížeč	Verze	Podpora
Internet Explorer Mobile	11	Ne
Opera Mini - beta	0.9.0.8	Ne
Browser	1.2.5.0	Ne
Maxthon	2.0.1.1000	Ne
Surfy	5.12.3	Ne
Touch Browser	1.10.0.0	Ne
UC Browser	4.2.1.1	Ne

Tabulka č. 6 – Podpora getUserMedia v prohlížečích pro Windows Phone

4.4.2 Prefixy

Díky tomu, že webové prohlížeče implementují funkce, které ještě nemají hotový standard, jako je tomu třeba u metody getUserMedia, vznikla potřeba vývojářů těchto prohlížečů oddělit funkce, které jsou standardizované a které ještě nejsou. Pro tyto účely se využívají tzv. prefixy. Každý prohlížeč využívá jiný druh prefixu, který se odvíjí od jádra daného prohlížeče. Pro vývojáře to znamená, že se musejí naučit všechny prefixy a na stránkách jednotlivých prohlížečů zjišťovat, s jakým prefixem tu, či onu funkci implementovat. Prefixy se převážně používají v jazyce CSS, kde je spousta vlastností, které nejsou standardizovány W3C, avšak ani nová API v jazyce JavaScript nejsou výjimkou.

4.4.2.1 Druhy prefixů

Jak jsem psal výše, tak každé vykreslovací jádro prohlížeče má vlastní prefix, prohlížeče na jádře WebKit používají prefix *webkit-*. Jádro Gecko, na kterém je postavený prohlížeč Firefox využívá prefix *moz-*. A v neposlední řadě tu máme Internet Explorer s jeho vykreslovacím jádrem Trident, které využívá prefixu *ms-*. Opera dnes již využívá jádro WebKit, tudíž se pro ni používá stejný prefix jako pro Chrome. Do verze 15 Opera běžela na vlastním jádře Presto, které využívalo prefixu *o-*. Toto jádro se ale již nevyvíjí. [13]

Jelikož je metoda getUserMedia stále ve vývoji a k jejímu standardizování ještě nedošlo, v prohlížečích, které tuto metodu implementovaly, je nutné použití prefixů. Abychom nemuseli při každém volání této metody vypisovat všechny prefixy, můžeme na začátku scriptu použít tento kód:

```
navigator.getUserMedia = (
    navigator.getUserMedia ||
    navigator.webkitGetUserMedia ||
    navigator.mozGetUserMedia ||
    navigator.msGetUserMedia
);7
```

Tento jednoduchý kód nám zajistí, aby si prohlížeč, ze kterého přistupujeme na webové stránky, vybral prefix, který momentálně používá. V tomto kódu jsou použity všechny prefixy, kterých využívají dnešní prohlížeče.

---

[7] Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia

4.5 Využití getUserMedia na internetu

Přestože metoda getUserMedia ještě není implementována ve všech prohlížečích a její podpora tedy není úplná, již nyní je spoustu webových aplikací a stránek, které tuto metodu využívají na úkor doplňků Adobe Flash, nebo MS Silverlight.

Možností využití této metody ve webových projektech je spousta, ať už by se jednalo o vytvoření avataru pro nějakou webovou aplikaci, které by bylo velmi jednoduché pro implementaci, nebo by se jednalo třeba o online video-chat se zákazníkem implementovaný do webových stránek dané společnosti.

Za odvětví, kde by se dala tato nová metoda využít, považuji biometrii, tedy vědu zabývající se fyzickými rozdíly mezi jednotlivci, ať už se jedná o otisky prstů, oční sítnici a další. Metoda getUserMedia by se dala využít například při ověřování uživatele podle obličeje, či podle hlasu. Použitím těchto metod autentizace by se zvýšila bezpečnost webových aplikací a pro uživatele by odpadla nutnost pamatovat si nějaké heslo.

Dalším příkladem využití metody getUserMedia by mohla být tvorba video-tutoriálů se souběžným záznamem z webkamery, mikrofonu a samozřejmě také obrazovky zařízení. Funkce pro zachycení obrazovky zařízení je u metody getUserMedia teprve v experimentální fázi a podporuje ji pouze prohlížeč Chrome, a to pouze v případě manuálního povolení této experimentální metody ve speciálním nastavení Chromu na adrese *chrome://flags*. Toto povolení s sebou nese i možnost nenadálého pádu Chromu při využití zachycení obrazovky zařízení. [14]

4.5.1 Fotografie přes Web Toy

Vytváření fotografií pomocí prohlížeče a jejich snadné sdílení na sociálních sítích. Přesně k tomuto účelu vznikla webová aplikace WebToy dostupná na adrese www.webcamtoy.com, která využívá právě metody getUserMedia pro zachycení streamu z webkamery. Tato webová stránka funguje jako Facebooková aplikace a má svoji konkurenci ve starší službě Seenly, dostupné na www.seenly.com. Ta ale pro získání přístupu k webkameře používá Adobe Flash, tudíž na uživatele klade větší nároky ohledně softwaru.

4.5.2 Videokonference přes Appear.in

Služba, která nabízí videokonference přímo v prohlížeči, se jmenuje Appear.in, a je dostupná na webové adrese www.appear.in. Služba nabízí možnost vytvoření vlastní místnosti a dále sdílení adresy této místnosti s přáteli. Tato služba funguje na podobném principu, na jakém funguje aplikace, kterou jsem vytvořil v praktické části své bakalářské práce.

4.5.3 Čtení QR kódu – Web QR

Webová stránka umožňující skenování a čtení QR kódu? Ano, i taková stránka již existuje a je možné ji spustit přímo z webového prohlížeče bez nutnosti cokoli instalovat jak do mobilu, tak desktopu a je dostupná na adrese www.webqr.com.

5 Porovnání getUserMedia s Adobe Flash

Metoda getUserMedia není první, která podporuje přístup k webkameře a mikrofonu daného zařízení pomocí prohlížeče. Ještě před HTML5, WebRTC a metodou getUserMedia bylo možné pomocí webové stránky přistupovat k webkameře i mikrofonu a pořizovat díky tomu fotografie a videozáznamy. Technologiemi, díky kterým toto vše bylo možné již v minulosti, jsou Adobe Flash a Microsoft Silverlight. Avšak tyto technologie měly jednu velkou nevýhodu. Tou nevýhodou je fakt, že pro jejich fungování bylo potřeba, aby si uživatel dané služby do svého počítače, respektive prohlížeče, stáhnul a nainstaloval zásuvný modul.

Metoda getUserMedia přichází jako nová alternativa k těmto starším formám pro přístup k webkameře a mikrofonu. Existence této metody je podmíněna příchodem právě HTML5, které implicitně podporuje nové multimediální objekty na webových stránkách. Konkrétně díky elementům <audio> a <video> můžeme záznam z kamery a mikrofonu zobrazit přímo v těchto elementech. U technologií Flash a Silverlight bylo nutné vložit do webové stránky objekt, který byl vytvořen ve vývojovém prostředí daného doplňku za využití jeho vlastního programovacího jazyka.

Při porovnání metody getUserMedia jsem jako alternativu zvolil pouze Adobe Flash, neboť Microsoft Silverlight je v dnešní době tak málo používaný, že není důvod ho do srovnání zahrnovat.

5.1 Výhody a nevýhody Adobe Flash

Adobe Flash byl dlouhou dobu jedinou možností pro využití webové kamery a mikrofonu na webových stránkách. Pro otestování přístupu přes Adobe Flash jsem si vyhledal jednoduchý příklad, který umožňoval přístup k webkameře a mikrofonu právě pomocí Adobe Flash a při použití tohoto

příkladu jsem měřil zatížení CPU, hodnotil složitost implementace, rychlost načítání webové stránky a v neposlední řadě také to, jaké jsou na uživatele kladené nároky.

Asi největším nedostatkem při využití Adobe Flash pro přístup k webkameře je již několikrát zmiňovaná nutnost nainstalovaného doplňku, což může být v mnoha případech problém, vezmeme-li v potaz, že spousta uživatelů – převážně firemních – nemá na svých počítačích právo instalovat. Dalším problémem, který se týká spíše tvůrců webových stránek, je ten, že samotná implementace a vytvoření funkční aplikace, která dokáže přistupovat k webkameře a mikrofonu zařízení pomocí ActionScriptu[8], není tak jednoduchá jako pomocí metody getUserMedia.

Zatížení CPU vlivem spuštěné aplikace Adobe Flash je něco, co zná snad každý uživatel. Tento problém nastává i při využití Adobe Flash pro přístup k webkameře a mikrofonu. Zatímco u webové stránky, která k webkameře a mikrofonu přistupovala pomocí metody getUserMedia, bylo zatížení procesoru 4 %, tak u webové stránky, která přistupovala k webkameře a mikrofonu pomocí Adobe Flash, bylo zatížení procesoru 6 %. Dále jsem testoval i zatížení paměti RAM. Větší zatížení paměti RAM při využití Adobe Flash se mi ale prokázat nepodařilo. V obou případech bylo vytížení paměti RAM shodné.

Veškeré své poznatky jsem shrnul v následující tabulce, která obsahuje mnou zjištěné výhody a nevýhody při využití přístupu k webkameře a mikrofonu pomocí Adobe Flash.

[8] ActionScript je programovací jazyk, využívaný pro animace, hry a aplikace vytvářené v Adobe Flash.

Výhody	Nevýhody
Podpora ve všech prohlížečích.	Potřeba doplňku Adobe Flash v prohlížeči.
	Složitější implementace.
	Vyšší zatížení CPU.
	Nepodpora mobilních zařízení značky Apple.
	Pomalejší načtení doplňku Adobe Flash a jeho spuštění.
	Několik kroků při schválení přístupu k webkameře a mikrofonu.

Tabulka č. 7 – Výhody a nevýhody využití Adobe Flash pro přístup k webkameře

5.2 Výhody a nevýhody getUserMedia

Metoda getUserMedia na rozdíl od Adobe Flash nepotřebuje pro svůj běh žádný doplněk v prohlížeči, avšak vyžaduje podporu jazyka HTML5 ve webovém prohlížeči. Další překážkou pro využívání metody getUserMedia může být fakt, že tato metoda je stále v pracovní verzi a není tedy schválena W3C.

Velkou výhodu naopak vidím při použití metody getUseMedia převážně ve velmi jednoduché implementaci a nízkých nárocích na uživatele. Nespornou výhodou je také potencionální podpora na mobilních zařízeních od firmy Apple, kterými jsou iPhone a iPad. Na těchto zařízeních je řešení přístupu k webkameře a mikrofonu pomocí Adobe Flash neuskutečnitelné z důvodu nepodpory doplňku Adobe Flash na mobilním systému iOS.

Ve využití CPU není v současné době, kdy jsou procesory na velmi vysoké úrovni, rozdíl nijak znatelný. Díky metodě getUserMedia se také webová stránka umožňující přístup k webkameře a mikrofonu načítá na rozdíl od webové stránky využívající Adobe Flash o několik desetin vteřiny rychleji.

Pro otestování při využití metody getUserMedia jsem zvolil stejný postup jako při testování přístupu pomocí Adobe Flash s tím rozdílem, že jsem využil svůj vlastní příklad. Veškeré mnou zjištěné poznatky shrnuji v následující tabulce.

Výhody	Nevýhody
Funguje bez jakýchkoliv doplňků.	Není zatím schváleno W3C.
Jednoduchá implementace.	Není implementováno ve všech prohlížečích.
Menší využití CPU.	Vyžaduje podporu HTML5.
Podpora v mobilních zařízeních značky Apple.	
Rychlejší načtení webové stránky.	
Pouze jedno schválení od uživatele k přístupu k webkameře a mikrofonu.	

Tabulka č. 8 – Výhody a nevýhody využití getUserMedia pro přístup k webkameře

5.3 Hodnocení

Metoda getUserMedia je ve všech ohledech zlepšením stávajících postupů a principů, které platily v případě přístupu k webkameře, či mikrofonu přes webové stránky. I v případě, že tato metoda bude implementována do všech moderních prohlížečů, bude její reálné využití omezeno právě na tyto nové verze a uživatelé starších verzí – například Internet Exploreru do verze 10, které tuto metodu nepodporují, budou nadále odkázaní na využití právě Adobe Flash.

Pro použití ve webových aplikacích vzhledem k faktu nepodpory ve starších prohlížečích bych doporučoval využít obě metody dohromady. Primárně nastavit přístup k webkameře nebo mikrofonu přes metodu getUserMedia a v případě, kdy není metoda getUserMedia implementována v prohlížeči uživatele, nastavit přístup k webkameře nebo mikrofonu pomocí Adobe Flash. Tento přístup by pro uživatele nových verzí prohlížečů byl mnohem pohodlnější na užívání než přístup přes Adobe Flash, na druhou stranu ale umožníme přístup k webkameře nebo mikrofonu i uživatelům starších prohlížečů.

V oblasti výkonu není na desktopu mezi getUserMedia a Adobe Flash zásadně velký rozdíl, avšak v mobilních zařízeních může být tento rozdíl mnohem zásadnější. Právě v mobilních zařízeních bude metoda getUserMedia mnohem lepší variantou.

Celé hodnocení bych ukončil myšlenkou, že použití pouze metody getUserMedia ve webových aplikacích je stále velmi omezené právě z důvodu nepodpory ve všech prohlížečích. Adobe Flash díky tomu ještě nějaký čas bude hrát důležitou roli ve webových aplikacích umožňujících přístup k webkameře a mikrofonu.

6 Praktická část

V praktické části své bakalářské práce jsem se zabýval vytvořením sady příkladů využívajících metody getUserMedia. Kromě těchto příkladů jsem měl za úkol vytvořit webovou aplikaci umožňující videokonference. V první části jsem se zabýval tvorbou příkladů. Po zpracování několika základních příkladů jsem začal pracovat na webové aplikaci pro video-chat a videokonference.

6.1 Příklady využití

Při tvorbě příkladů jsem se zamýšlel nad jejich případným reálným využitím a také nad tím, aby se daly jednoduše implementovat do jakékoliv webové aplikace či prezentace. U každého příkladu uvádím kód, který zajišťuje funkčnost. V této části bakalářské práce tento kód rozeberu a vysvětlím podrobněji, než je tomu na webu http://bp.lukashejtmanek.cz/.

Pro každý příklad využívám prefixy, které mi zajistí podporu metody getUserMedia v prohlížečích, které běží na jádře WebKit a Gecko. Dále také, spíše do budoucnosti, využívám prefixu pro prohlížeč Explorer, který běží na jádře Trident.

Ve všech příkladech uvádím pouze JavaScriptový kód. Veškerý kód, který uvádím, je dále součástí Media Capture & Stream API.

```
navigator.getUserMedia = (navigator.getUserMedia ||
   navigator.webkitGetUserMedia ||
   navigator.mozGetUserMedia ||
   navigator.msGetUserMedia);
```
[9]

[9] Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia

Prefix *webkit-* využívají prohlížeče na jádře WebKitu – tedy Chrome, Opera, Safari a další. Prefix *moz-* využívá jediný prohlížeč, a tím je Firefox. Pro budoucí potřeby přidávám i prefix *ms-*, který je využíván prohlížečem Internet Explorer od Microsoftu.

6.1.1 Video stream do elementu video

Díky novému elementu v jazyce HTML5, kterým je <video>, je možné využívat na stránce multimediální video-formáty. Do tohoto elementu je možné také přenášet stream z webkamery pomocí metody getUserMedia a následně MediaStream objektu.

```
navigator.getUserMedia(
    {
       video: true,
       audio: false
    },
    function (stream) {
       var video = document.querySelector('video');
       video.src = window.URL.createObjectURL(stream);
       mediaStream = stream;
       video.play();
    },
    function (error) {
       console.log("ERROR: " + error);
    }
);
```

V první části tohoto kódu si definujeme parametr constraints, který má v tomto případě hodnotu pro audio *false* a pro video *true*. Díky tomu požádá webová stránka prohlížeč o přístup pouze k webkameře, nikoliv k mikrofonu. Dalším parametrem je successCallback, který zde zastupuje funkce stream. V prvním řádku této funkce si pouze přiřadíme element video k proměnné video. V dalším řádku nastavíme zdroj videa na právě získaný stream z webkamery, který je jako objekt dostupný

na vlastní adrese. Poslední dva řádky zajišťují pouze snazší zápis pro stream a dále spuštění elementu video.

Posledním parametrem je errorCallback, který se provede v případě, kdy webová stránka nezíská přístup k webkameře. Jediná věc, která se zde provede, je, že se chyba vypíše do konzole. Je samozřejmě možné chybu zobrazit uživateli pomocí různých metod, ale já jsem zde zvolil tichou variantu.

6.1.2 Audio stream do elementu audio

Pro streamování audio záznamu můžeme využít elementu <audio>, který je stejně jako element <video> součástí nového HTML5.

```
navigator.getUserMedia(
    {
       video: false,
       audio: true
    },
    function (stream) {
       var audio = document.querySelector(audio);
       audio.src = window.URL.createObjectURL(stream);
       mediaStream = stream;
       audio.play();
    },
       function (error) {
       console.log("ERROR: " + error);
    }
);
```

V tomto příkladu si v prvním parametru opět definujeme constraints, tentokrát ovšem s opačnými hodnotami než v minulém příkladu. Zbytek kódu se liší jen v rozdílném druhu streamu a jiným elementem pro stream.

6.1.3 Fotografie

Pro zachycení fotografie ze streamu je zapotřebí využít elementu <canvas>, který je také novinkou v HTML5. Prvním krokem pro fotografování pomocí metody getUserMedia s využitím elementu <canvas> je jeho inicializace a nastavení základních parametrů.

```
var canvas = document.querySelector('canvas');
var ctx;
var width = 480;
var height = 0;
ctx = canvas.getContext("2d");
```

Po tomto zápisu nastoupí na řadu klasický video-stream, jenž bude přenášet obraz z webkamer do elementu <video>, ze kterého zachytíme snímek, který pošleme do elementu <canvas>.

```
height = video.videoHeight / (video.videoWidth /
width);
canvas.setAttribute('width', width);
canvas.setAttribute('height', height);
ctx.drawImage(video, 0, 0, width, height);
```

V prvních třech řádcích si definujeme velikosti plátna. Poslední řádek slouží již pro samotné vykreslení obrazu do elementu <canvas>.

6.1.4 Fotobudka

Příklad simulující fotobudku funguje na stejném principu jako příklad zabývající se fotografiemi s tím rozdílem, že místo jednoho elementu <canvas> jsou na stránce tři a zachycení fotografií je řízeno časem.

```
var canvas1 = document.querySelector('#canvas1');
var canvas2 = document.querySelector('#canvas2');
var canvas3 = document.querySelector('#canvas3');
var width = 480;
var height = 0;
ctx1 = canvas1.getContext("2d");
```

```
ctx2 = canvas2.getContext("2d");
ctx3 = canvas3.getContext("2d");
```

Po inicializaci základních proměnných následuje kód, který je totožný s kódem pro přístup k webkameře se streamem do elementu <video> a následuje kód, který zajistí sekvenční snímání z elementu <video> do elementu <canvas>.

```
height = video.videoHeight / (video.videoWidth /
width);
canvas1.setAttribute('width', width);
canvas1.setAttribute('height', height);
ctx1.drawImage(video, 0, 0, width, height);
setTimeout(function () {
   canvas2.setAttribute('width', width);
   canvas2.setAttribute('height', height);
   ctx2.drawImage(video, 0, 0, width, height);
   }, 2000);
setTimeout(function () {
   canvas3.setAttribute('width', width);
   canvas3.setAttribute('height', height);
   ctx3.drawImage(video, 0, 0, width, height);
}, 4000);
```

Pro sekvenční snímání zde využívám funkce setTimeout(), která mi umožňuje provést část kódu po uplynutí určité doby.

6.1.5 Stylování videa pomocí CSS3

U příkladu, který demonstruje možnosti využití technologie getUserMedia spolu s novými filtry dostupnými v CSS3, je kód pro přístup k webkameře totožný s kódem, který je u příkladu s video streamem. Přidanou hodnotou v tomto případě je možnost aplikace různých filtrů.

Nedostatkem, který jsem u tohoto příkladu zjistil, je, že při použití filtru se filtr neaplikuje na samotné video, ale pouze na element <video> a díky tomu se filtr projeví i na ovládacích tlačítkách, jsou-li zobrazena.

Filtry, které jsem použil, jsou následující: Blur, Grayscale, Sepia, Brightness, Contrast, Hue, Invert a Saturation. Implementace těchto filtrů je realizována přes třídy, které se na element video aplikují při stisknutí tlačítka daného filtru. Každý filtr má své vlastní tlačítko.

6.1.6 Audio záznam a následné stažení

Pro příklad s nahráváním audia jsem se rozhodl využít framework RecordRTC, který zajišťuje veškeré kódování, obnovovací frekvence i převod do použitelného formátu.

Pro správné fungování nahrávání je nutné přidat do HTML dokumentu odkaz na soubor RecordRTC.js. Pokud chceme mít vždy aktuální verzi tohoto frameworku, tak je možné přidat do dokumentu odkaz na webobou verzi souboru. V případě, že nám stačí stávající verze, je možné soubor stáhnout a v dokumentu ho přidat.

```
<script src="http://RecordRTC.org/latest.js">
</script>[10]
```

Samotné spuštění nahrávání se provádí v těle metody getUserMedia v parametru successCallback. V této části kódu je nutné definovat si zdrojový stream pro nahrávání a dále zjistit, zda se jedná o audio, či video stream.

```
recorder = window.RecordRTC(
   stream, {
      type: 'audio'
   }
);
```

Po definování streamu pro nahrávání je možné spustit nahrávání jednoduchým příkazem *recorder.startRecording()*.

[10] Odkaz na nejnovější verzi. Dostupné z: http://recordrtc.org

6.1.7 Video záznam a následné stažení

Pro nahrávání videa nám poslouží stejný základ jako pro nahrávání audia. Jedinými rozdíly jsou:

- jiný cílový element pro streamování videa,
- rozdílná definice streamu pro nahrávání
- a dále jiný formát výsledného souboru.

```
recorder = window.RecordRTC(
   stream, {
      type: 'video'
   }
);
```

V současné době zatím nefunguje žádná spolehlivá metoda, která by umožňovala nahrávání zároveň video i audio streamu. Žádný z dostupných prohlížečů toto zatím nepodporuje. Částečná podpora se již objevila ve Firefoxu, ale ve většině případů se bohužel uložení přesto nepodaří.

6.2 Webová aplikace pro video-chat a videokonference

Celý projekt WebRTC má za úkol umožnit peer2peer video-chat, pomocí kterého budou moci uživatelé komunikovat v reálném čase pouze za pomoci webového prohlížeče. Z tohoto důvodu jsem si také jako jeden z cílů své práce zvolil vytvoření takovéto aplikace s využitím některých frameworků. Pro jádro aplikace využívám frameworku PeerJS, který zajišťuje spojení mezi uživateli pomocí STUN[11] serveru, přenos dat pomocí PeerConnection API a dále pomocí DataChannel API. Dalším frameworkem, který využívám

[11] Server umožňující přímou komunikaci mezi uživateli, kteří jsou v různých podsítích. Server dokáže zjistit cestu od veřejné IP, přes router, až po poslední port na switchi.

v této aplikaci je Bootstrap 3, který umožňuje vytvářet responsivní stránky pomocí předpřipravených tříd se styly.

Samotná aplikace běží na technologiích PHP a MySQL, které zajišťují registrace uživatelů, jejich přihlašování a dále přidávání přátel.

USERS
ID (int)
Username (text)
Email (text)
Password (text)

BUDDIES
User (int)
Buddy (int)

Takto vypadá struktura databáze, která obsahuje informace o uživatelích a jejich přátelích. Databáze obsahuje pouze dvě tabulky, z nichž jedna uchovává informace u uživateli a druhá o přátelích. Heslo v tabulce USERS je kódováno pomocí md5, díky tomu je uchováváno v šifrované podobě.

Po přihlášení do aplikace si každý uživatel může zvolit, zda chce spojit s jedním uživatelem, či chce vytvořit místnost pro videokonferenci a následně název místnosti sdílet s těmi, které chce pozvat.

V případě, že uživatel žádá o spojení pouze s jedním uživatelem, vytvoří se požadavek na STUN server, který zmapuje cestu k uživateli a tuto cestu poskytne vytočenému uživateli, u kterého také zmapuje cestu. Cestu vytáčeného vrátí volajícímu a pokud ani u jednoho uživatele není problém s firewallem, tak uživatele spojí.

Pro spojení více uživatelů je možné využít videokonferenci, tedy vytvořit místnost s určitým názvem a tento název sdílet s ostatními uživateli, aby se tito uživatelé mohli připojit. Založení místnosti vytvoří na dané stránce session, která naslouchá všem novým streamům, které se na webové stránce objeví a tyto streamy posílá automaticky do elementů <video>. Lokální video stream odesílá do hlavního elementu <video>.

Vytvořením této aplikace byl splněn jeden z cílů bakalářské práce. Screenshoty aplikace jsou dostupné v příloze této práce a zdrojové kódy jsou dostupné na CD.

7 Závěr

Závěrem bych shrnul, čemu se má bakalářská práce věnovala a k jakým výsledkům jsem dospěl při testování a rozboru metody getUserMedia.

Cílem bakalářské práce byl rozbor aktuálního stavu vývoje nové technologie getUserMedia, která je součástí HTML5. Práce se také zabývala výhodami a nevýhodami této nové technologie oproti ostatním technologiím umožňujícím přístup k webkameře, konkrétně Adobe Flash. Bylo také provedeno důkladné otestování podpory této nové technologie v dostupných verzích prohlížečů jak desktopových, tak mobilních (smartphone/tablet). Na závěr byla zpracována webová aplikace demonstrující využití této technologie v praxi a vytvořena sada příkladů pro implementaci této technologie ve webových aplikacích.

V první části jsem provedl rozbor nové technologie getUserMedia včetně otestování této technologie v dostupných prohlížečích. Otestování dopadlo na desktopových prohlížečích velmi kladně, až na výjimky všechny prohlížeče tuto technologii podporují. Co se týče mobilních prohlížečů, tak u těch je podpora velmi omezená a jediná platforma, která má alespoň nějaké prohlížeče, které tuto technologii podporují, je Android.

Další částí bakalářské práce bylo porovnání metody getUserMedia a Adobe Flash. Jak jsem již shrnul v hodnocení oné kapitoly, použití technologie gtUserMedia zatím nedoporučuji používat bez záložního přístupu k webkameře a mikrofonu pomocí Adobe Flash.

Jako praktickou část bakalářské práce jsem vytvořil funkční webovou aplikaci pro videokonference a video-chat pomocí frameworku PeerJS dostupnou na adrese http://video.lukashejtmanek.cz/ a dále sadu příkladů, které demonstrují různé možnosti využití technologie getUserMedia

ve webových aplikacích. Sada příkladů je dostupná na adrese http://bp.lukashejtmanek.cz.

Na závěr bych tedy zopakoval, že všechny cíle mé bakalářské práce byly splněny.

8 Literatura a zdroje

[1] HTML. Wikipedia, the free encyclopedia [online]. San Francisco (CA), 2001, 2015-04-06 [cit. 2015-04-06]. Dostupné z: http://en.wikipedia.org/wiki/HTML

[2] XHTML. Wikipedia, the free encyclopedia [online]. San Francisco (CA), 2001, 2015-01-28 [cit. 2015-04-06]. Dostupné z: http://en.wikipedia.org/wiki/XHTML

[3] HTML5. *Wikipedia, the free encyclopedia* [online]. San Francisco (CA), 2006, 2015-03-17 [cit. 2015-04-06]. Dostupné z: http://en.wikipedia.org/wiki/HTML5

[4] HTML5 Application Cache. *W3Schools Online Web Tutorials* [online]. © 1999-2015 [cit. 2015-04-06]. Dostupné z: http://www.w3schools.com/html/html5_app_cache.asp

[5] HTML5 Drag and Drop. *W3Schools Online Web Tutorials* [online]. © 1999-2015 [cit. 2015-04-06]. Dostupné z: http://www.w3schools.com/html/html5_draganddrop.asp

[6] HTML5 Geolocation. *W3Schools Online Web Tutorials* [online]. © 1999-2015 [cit. 2015-04-06]. Dostupné z: http://www.w3schools.com/html/html5_geolocation.asp

[7] HTML Standard: Microdata. *Web Hypertext Application Technology Working Group* [online]. 2004-, 2015-03-14 [cit. 2015-04-06]. Dostupné z: https://html.spec.whatwg.org/multipage/microdata.html

[8] Media Capture and Streams. *World Wide Web Consortium (W3C)* [online]. 2011, 2015-02-15 [cit. 2015-04-06]. Dostupné z: http://www.w3.org/TR/mediacapture-streams/

[9] HTML5 Local Storage. *W3Schools Online Web Tutorials* [online]. © 1999-2015 [cit. 2015-04-06]. Dostupné z: http://www.w3schools.com/html/html5_webstorage.asp

[10] The Navigator Object. *W3Schools Online Web Tutorials* [online]. © 1999-2015 [cit. 2015-04-06]. Dostupné z: http://www.w3schools.com/jsref/obj_navigator.asp

[11] Navigator.getUserMedia(). *Mozilla Developer Network* [online]. © 2005-2015, 2015-04-01 [cit. 2015-04-06]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia

[12] Report bugs - WebRTC. *WebRTC* [online]. © 2011-2014 [cit. 2015-04-06]. Dostupné z: http://www.webrtc.org/report-bug

[13] Prefix. *Vzhůru dolů — webový frontend ze všech stran* [online]. 2013 [cit. 2015-04-06]. Dostupné z: http://www.vzhurudolu.cz/prirucka/prefix

[14] Issue 1757: Enable screen capture support in getUserMedia by default. *Webrtc - Web-based real-time communication - Google Project Hosting* [online]. 2013 [cit. 2015-04-06]. Dostupné z: https://code.google.com/p/webrtc/issues/detail?id=1757

[15] WebRTC - Web developer guide | MDN. *Mozilla Developer Network* [online]. © 2005-2015, 2015-01-23 [cit. 2015-04-06]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Guide/API/WebRTC

[16] DUTTON, Sam. Getting Started with WebRTC - HTML5 Rocks. *HTML5 Rocks - A resource for open web HTML5 developers* [online]. 2012, 2014-02-21 [cit. 2015-04-06]. Dostupné z: http://www.html5rocks.com/en/tutorials/webrtc/basics/

[17] WebRTC Peer-to-peer connections. *Can I use... Support tables for HTML5, CSS3, etc* [online]. 2013, 2015-02-21 [cit. 2015-04-06]. Dostupné z: http://caniuse.com/#search=webrtc

[18] WebRTC FAQs. *OpenTok | WebRTC Platform for Video, Voice and Messaging from TokBox* [online]. © 2007-2015 [cit. 2015-04-06]. Dostupné z: https://tokbox.com/about-webrtc

[19] RTCPeerConnection. *Mozilla Developer Network* [online]. © 2005-2015, 2014-11-27 [cit. 2015-04-06]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection

[20] RTCDataChannel. *Mozilla Developer Network* [online]. © 2005-2015, 2015-03-08 [cit. 2015-04-06]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/RTCDataChannel

[21] MediaStream API. *Mozilla Developer Network* [online]. © 2005-2015, 2015-03-08 [cit. 2015-04-06]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/MediaStream_API

[22] BIDELMAN, Eric. Creating .webm video from getUserMedia() - Eric Bidelman. *Eric Bidelman* [online]. 2012 [cit. 2014-03-30]. Dostupné z: http://ericbidelman.tumblr.com/post/31486670538/creating-webm-video-from-getusermedia

[23] BIDELMAN, Eric. Capturing Audio & Video in HTML5. *HTML5 Rocks - a resource for open web HTML5 developers* [online]. 2012 [cit. 2014-03-30]. Dostupné z: http://www.html5rocks.com/en/tutorials/getusermedia/intro/

[24] DEVLIN, Ian. Using the getUserMedia API with the HTML5 video and canvas elements. *HTML5 Hub* [online]. 2013 [cit. 2014-03-30].

Dostupné z: http://html5hub.com/using-the-getusermedia-api-with-the-html5-video-and-canvas-elements/

[25] DE ROSA, Aurelio. An Introduction to the getUserMedia API. *SitePoint – Learn HTML, CSS, JavaScript, PHP, Ruby & Responsive Design* [online]. 2014 [cit. 2014-03-30]. Dostupné z: http://www.sitepoint.com/introduction-getusermedia-api/

[26] HASSMAN, Martin. HTML5, getUserMedia a práce s kamerou. *Zdroják - o tvorbě webových stránek a aplikací* [online]. 2014 [cit. 2014-03-30]. Dostupné z: http://www.zdrojak.cz/clanky/html5-prace-kamerou/

[27] *HTML5.cz - vše co potřebujete vědět o HTML5* [online]. Dostupné z: http://www.html5.cz/

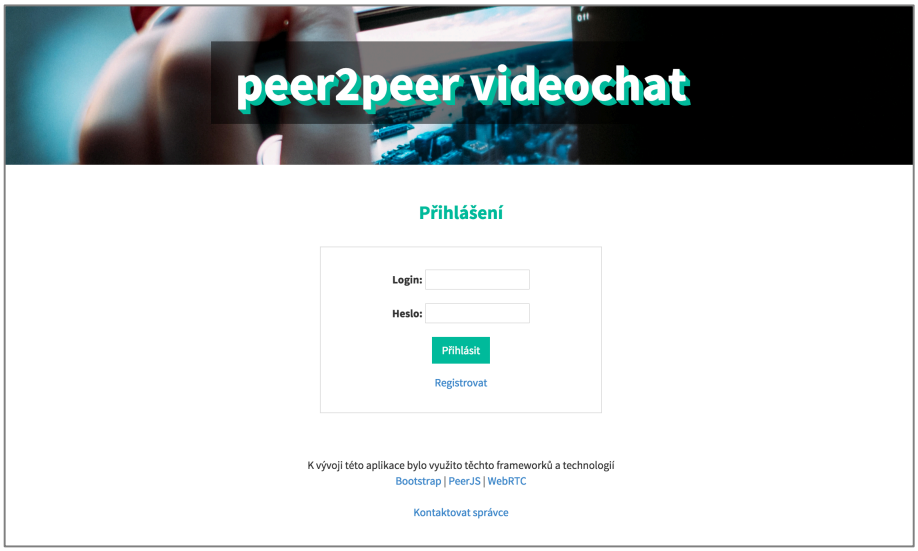
[28] Media Capture and Streams. *World Wide Web Consortium (W3C)* [online]. 2014 [cit. 2014-03-30]. Dostupné z: http://dev.w3.org/2011/webrtc/editor/getusermedia.html

9 Seznam tabulek

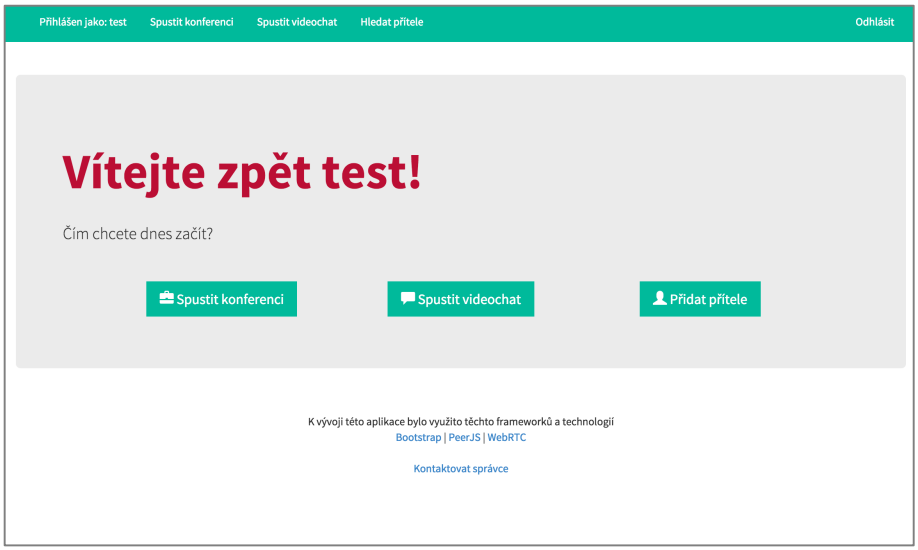
10 Přílohy

Příloha č. 1 – CD s textem práce a zdrojovými kódy k webové aplikaci

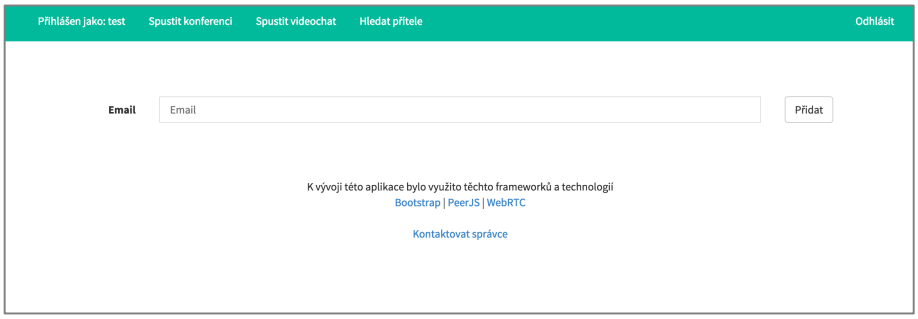
Příloha č. 2 – Screenshoty webové aplikace



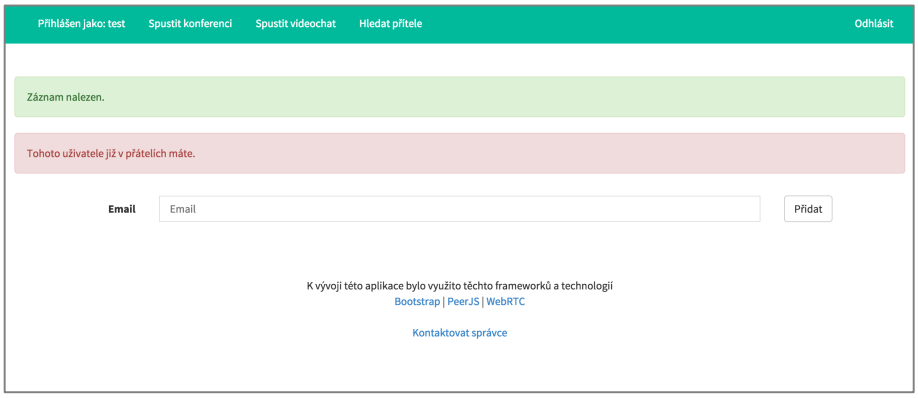
Obr. 1 – Přihlašovací obrazovka webové aplikace



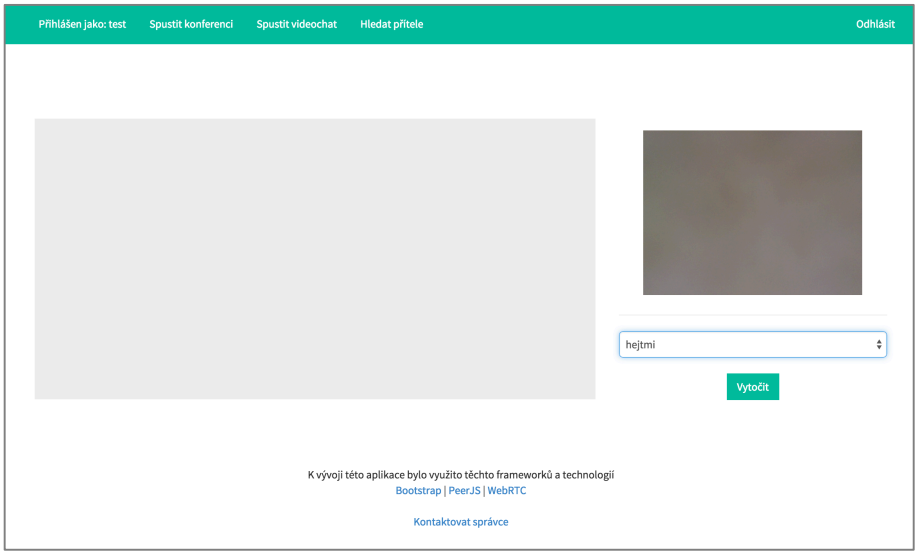
Obr. 2 – Úvodní obrazovka webové aplikace



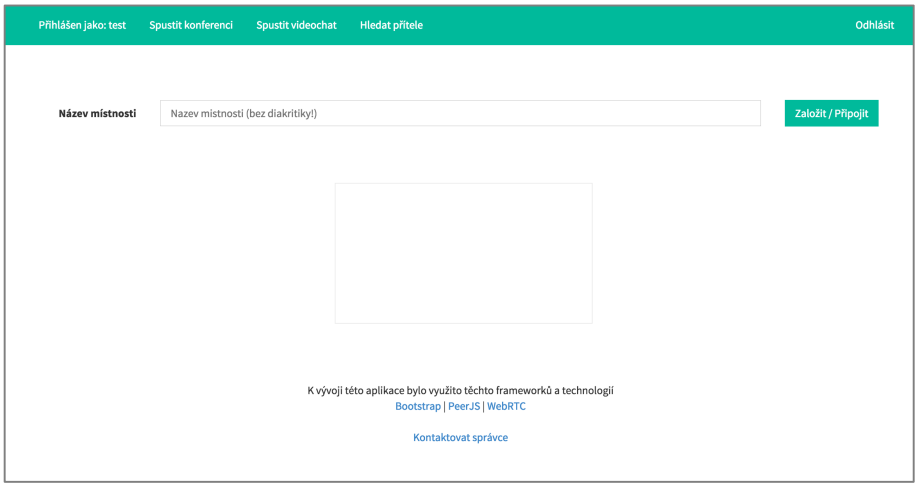
Obr. 3 – Obrazovka pro přidávání přátel



Obr. 4 – Obrazovka po provedení přidání/nepřidání přítele



Obr. 5 – Obrazovka pro video-chat mezi 2 uživateli



Obr. 6 – Obrazovka pro vytvoření/přihlášení se do místnosti

Obr. 7 – Obrazovka pro videokonference