

Slovenská technická univerzita v Bratislave  
Fakulta informatiky a informačných technológií

Bc. Ján Antala

## ADAPTÍVNY WEB DIZAJN

Diplomová práca

Študijný program: Informačné systémy  
Študijný odbor: 9.2.6 Informačné systémy  
Miesto vypracovania: Ústav aplikovanej informatiky, FIIT STU, Bratislava  
Vedúci práce: doc. Ing. Michal Čerňanský PhD.

máj 2014

# **ANOTÁCIA**

Slovenská technická univerzita v Bratislave  
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGII  
Študijný program: INFORMAČNÉ SYSTÉMY

Autor: Bc. Ján Antala

Diplomová práca: Adaptívny web dizajn.

Vedúci diplomovej práce: doc. Ing. Michal Čerňanský PhD.

Máj, 2014

Používanie mobilných zariadení rapídne rastie a tie so sebou prinášajú nie len rôzne veľkosti displejov a nové interakčné spôsoby, ale aj nové druhy pripojenia na internet. Ale na web nemajú prístup len smartfóny a klasické počítače. Sú tu smartfóny, tablety, e-čítačky, netbooky, hodinky, televízie, phablety, herné konzoly a množstvo ďalších. Je preto nevyhnutné prispôsobiť webové stránky rôznym zariadeniam a novým trendom v oblasti interakcie. Rapídne sa vyvíja niekoľko spôsobov dizajnovania webu. My prinášame spôsob adaptácie webových komponentov založený na možnostiach poskytovanými zariadeniami a externých podmienkach. Uvádzame nové spôsoby interakcie s webovými aplikáciami založené na rozpoznaní reči, pohybu alebo rotácie zariadenia.

## **ANNOTATION**

Slovak University of Technology Bratislava  
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES  
Degree Course: INFORMATION SYSTEMS

Author: Bc. Ján Antala  
Diploma Thesis: Adaptive web design.  
Supervisor: doc. Ing. Michal Čerňanský PhD.  
2014, May

Mobile device penetration grows rapidly and it brings not only different display sizes with new human interaction methods, but also high-latency networks. But there are not just smartphones and desktop computers. We now have Web-enabled smartphones, tablets, e-readers, netbooks, watches, TVs, phablets, notebooks, game consoles, cars and more. So it is necessary to adapt web pages for different devices and new interaction methods. Several new methods for designing web are rapidly evolving. We present a new way of web components adaptation based on a device features and external conditions in our work. We introduce new possibilities of interaction with the web applications based on speech recognition, motion or device rotation.

## **Čestné prehlásenie**

Čestne prehlasujem, že záverečnú prácu som vypracoval samostatne, s použitím uvedenej literatúry a na základe svojich vedomostí a znalostí.

Bratislava, máj 2014

.....

*Vlastnoručný podpis*

## **Podakovanie**

Týmto sa chcem podakovať najmä vedúcemu práce, doc. Ing. Michalovi Čerňanskému, PhD., za jeho cenné rady, pripomienky a venovaný čas pri konzultáciách. Vďaka patrí taktiež mojej rodine a všetkým kontribútorom podiaľajúcich sa na zlepšení vytvoreného softvéru.

# **Obsah**

<b>1</b>	<b>Úvod</b>	<b>1</b>
<b>2</b>	<b>Adaptácia webových aplikácií</b>	<b>2</b>
2.1	Možnosti prispôsobenia . . . . .	2
2.1.1	Veľkosť a rozlíšenie . . . . .	2
2.1.2	Interakčné prostriedky . . . . .	4
2.1.3	Pripojenie . . . . .	8
2.1.4	Platforma . . . . .	9
2.2	Adaptačné techniky . . . . .	10
2.2.1	Responsive design . . . . .	10
2.2.2	Mobile First . . . . .	11
2.2.3	Progressive enhancement . . . . .	12
2.2.4	Graceful degradation . . . . .	12
2.2.5	Server-side Adaptation . . . . .	13
2.2.6	RESS (Responsive Design + Server Side Components) . . . . .	14
<b>3</b>	<b>Návrh riešenia</b>	<b>15</b>
3.1	Alternatívne spôsoby ovládania . . . . .	16
3.1.1	Reč . . . . .	16
3.1.2	Pohyb . . . . .	19
3.1.3	Gyroskop . . . . .	23
3.2	Komponenty . . . . .	24
3.2.1	Video . . . . .	25
3.2.2	Mapy . . . . .	27
3.2.3	Lightboxy . . . . .	29
3.2.4	Otvorenie v aplikácii / Stiahnutie aplikácie . . . . .	29
3.3	Nástroje . . . . .	31
3.3.1	Detekcia . . . . .	31
<b>4</b>	<b>Overenie</b>	<b>32</b>
4.1	Adaptívne vstupné metódy . . . . .	33
4.1.1	Ovládanie pomocou reči . . . . .	33
4.1.2	Ovládanie pomocou gyroskopu . . . . .	34
4.1.3	Ovládanie pomocou webkamery . . . . .	39
4.2	Adaptívne webové komponenty . . . . .	41
4.2.1	Modul youtube videa . . . . .	41
4.2.2	Modul google maps . . . . .	43
<b>5</b>	<b>Záver</b>	<b>46</b>
<b>Príloha A</b>	<b>Obsah CD nosiča</b>	<b>52</b>
<b>Príloha B</b>	<b>Technická dokumentácia a inštalačná príručka</b>	<b>53</b>
<b>Príloha C</b>	<b>Článok prezentovaný na IIT.SRC2014</b>	<b>81</b>

## **Zoznam tabuliek**

1	Podpora rozpoznávania reči . . . . .	34
2	Natívna podpora rečovej syntézy . . . . .	34
3	Podpora polyfillu rečovej syntézy . . . . .	34
4	Podpora získavania orientácie v prehliadačoch . . . . .	35
5	Alpha rotácia gyroskopu . . . . .	36
6	Beta rotácia gyroskopu . . . . .	36
7	Gamma rotácia gyroskopu . . . . .	37
8	Podpora prístupu ku kamere v prehliadačoch . . . . .	40
9	Konfigurácie internetových pripojení . . . . .	41
10	Množstvo požiadaviek a prenesených dát pre youtube video . . . . .	41
11	Množstvo požiadaviek a prenesených dát pre google mapy . . . . .	44

## Zoznam obrázkov

1	Porovnanie zariadení vzhľadom na veľkosť displeja . . . . .	3
2	Schopnosť ovládania mobilných telefónov . . . . .	5
3	Schopnosť ovládania tabletov . . . . .	5
4	Schopnosť ovládania hybridných počítačov . . . . .	6
5	Accelerometor a gyroskop . . . . .	7
6	Fázy spracovania požiadavky na server . . . . .	8
7	Čas potrebný na spracovanie požiadavky na server . . . . .	9
8	Progressive enhancement vs Graceful degradation . . . . .	13
9	Webová stránka vytvoreného projektu angular-adaptive . . . . .	15
10	Algoritmus ovládania aplikácie pomocou reči . . . . .	18
11	Vizualizácia videa zachyteného web kamerou . . . . .	19
12	Vizualizácia videa po aplikovaní filtra detekcie kože . . . . .	20
13	Vizualizácia videa po aplikovaní filtra detekcie hrán . . . . .	20
14	Algoritmus ovládania aplikácie pomocou pohybu . . . . .	22
15	Algoritmus ovládania aplikácie pomocou gyroskopu . . . . .	24
16	Prenášané dátá pri požiadavke na video zo servera YouTube . . . . .	25
17	Prenášané dátá pri požiadavke na video zo servera Vimeo . . . . .	26
18	Podmienené otváranie máp . . . . .	28
19	Chybová hláška po otvorení custom URL schémy, pokiaľ aplikácia neexistuje	29
20	Otvorenie natívnej aplikácie v iOS 6 . . . . .	30
21	Prototypová aplikácia vytvoreného modulu adaptive-speech . . . . .	33
22	Prototypová aplikácia vytvoreného modulu adaptive-scroll a rozšírenie Gyrocopter . . . . .	39
23	Prototypová aplikácia vytvoreného modulu adaptive-motion . . . . .	40
24	Časová os načítavania stránky s použitím youtube embed videa . . . . .	42
25	Časová os načítavania stránky s použitím youtube modulu . . . . .	42
26	Porovnanie rýchlosťi plného načítania stránky s youtube komponentom . . . . .	42
27	Porovnanie rýchlosťi zostavenia dom stromu s youtube komponentom . . . . .	43
28	Prototypová aplikácia vytvoreného modulu adaptive-googlemaps . . . . .	44
29	Porovnanie rýchlosťi plného načítania stránky s google maps komponentom . . . . .	45
30	Porovnanie rýchlosťi zostavenia dom stromu s google maps komponentom . . . . .	45
31	Umiestnenie projektu na serveri github . . . . .	46

# 1 Úvod

Web sa neustála vyvíja a my sa musíme prispôsobiť. S príchodom mobilných zariadení, ktoré postupne nahradzujú klasické počítače, vznikol problém pri správnom zobrazovaní webového obsahu. Web na ne bol pripravený, a tak sa im ponúkala verzia stránky pre klasické počítače. Dnes však neriešime web len stolové počítače, ale aj pre mobilné zariadenie, tablety, televízie, nositeľné prístroje a ďalšie. Aj veci pre priemernú webovú stránku ako sú podiel prehliadačov, operačných systémov či rozlíšení sa za posledné roky v mnohom zmenili [20].

Len za posledných pár rokov sa vo svete predalo viac ako 1 bilión mobilných zariadení, 1.038 bilióna celkovo [37]. Mobilné telefóny a tablety sa stali ešte viac personálnymi a používajú sa neustále počas celého dňa pri rôznych činnostiach [7, 3, 29]. Ich predajnosť sa zvýšuje každým dňom. V porovnaní vývoja trhového podielu osobných počítačov typu WINTEL a mobilných zariadení Apple a Android za posledné roky je tento trend ešte viac viditeľný.

Postupom času sa vyvinuli rôzne spôsoby ako prispôsobiť web aj na mobilné zariadenia. Najlepší spôsob prispôsobenia obsahu je taký, ktorý pokrýva všetky zariadenia od najmenších mobilných po veľké televízne obrazovky a nevytvára pre rôzne zariadenia vlastné samostané stránky.

Ignorovanie webu na mobilných zariadeniach väčšinou spoločnosťí však vedie k vytváraniu samostatných natívnych aplikácií pre každú platformu. Okrem vedúceho postavenia Androidu a iOS existuje množstvo ďalších platform, pre ktorú treba vytvoriť vlastnú aplikáciu, a tým sa vývoj predražuje. Nevýhodou natívnych aplikácií je, že neotvárajú webové odkazy a stále nemáme istotu, že si ich používateľ stiahne a nainštaluje. Taktiež vzniká problém pri aktualizáciách, používateľ ju môže manuálne spustiť. Nestačí len otvoriť aplikáciu, ktorá bude automaticky obsahovať najnovšiu verziu ako web. S tým je spojený aj problém s ich udržiavaním.

Práve tu je neoddeliteľná súčasť webu a mobilných zariadení. V súčasnosti populárne sociálne siete, ale aj emaily či qr kódy obsahujú množstvo odkazov na web. Na rovnaké odkazy je však možné pristúpiť aj z klasických počítačov. Je tak dôležité, aby sa obsah používateľom zobrazil správne bez ohľadu na to, na akom zariadení k nemu pristupujú.

Optimalizácia webového obsahu pre mobilné zariadenia má veľmi krátku história. Dnes však už existujú základné vzory, podľa ktorých je možné prispôsobiť zobrazenie webového obsahu na ich malé displeje [14, 40]. Problémom je, že neustále rastú rozmery mobilných zariadení, ale aj veľké televízne obrazovky sa stávajú prístupovým bodom k webovému obsahu. Len za posledné 3 mesiace bolo predaných 29% android zariadení s obrazovkou väčšou ako 4.5 palca [36] a k podobnému trendu pristupujú aj iní výrobcovia.

Optimalizácia webového obsahu zariadeniam však neznamená len jeho vizuálne prispôsobenie rôznym veľkosťiam displejom. Nemenej dôležitým prvkom je aj pripojiteľnosť zariadenia na sieť s cieľom čo najrýchlejšieho stiahnutia, zobrazenia stránky a šetrenia používateľových prenášaných dát a financií. Nielen práve preto je vhodné použiť princípy adaptívneho web

dizajnu. Medzi jeho hlavné charakteristiky patria všadeprítomnosť, flexibilita, výkonnosť, rozšíriteľnosť a priateľskosť k budúcnosti [12]. V súčasnosti nevieme povedať aké zariadenia sa budú predávať o päť rokov, aké budú mať vlastnosti, ale s celkom veľkou pravdepodobnosťou budú obsahovať webový prehliadač.

Adaptívny web dizajn je v podstate „Progressive enhancement”, len je aplikovaný na mnoho širšie a diverzifikovanejšie spektrum. Pripojenie na internet sa v súčasnosti nachádza na smartfónoch, tabletach, e-čítačkach, netbookoch, hodinkách, televízoroch, phabletoch, notebookoch, herných konzolách, autách a na mnohých iných zariadeniach. Taktiež máme mnoho druhov internetových pripojení s rozdielnými rýchlosťami, latenciami či kvalitou. „Responsívny Web dizajn” je taktiež jednou z techník používanou v stretežii adaptívneho web dizajnu. Vytváranie flexibilných rozvrhnutí stránok je dôležité, ale je tu ešte mnoho ďalších faktorov na ktoré musíme myslieť. Je dôležité brať do úvahy ergonómiu, možnosti dotyku či iných interakčných metód, internetové pripojenie a množstvo ďalších faktorov, ktoré vieme detektovať.

Adaptívny web dizajn považujeme za zhodný s vytvorením jednotného webového zážitku. Môžeme ho prispôsobiť na základe možností ponúkaných konkrétnym zariadením či prehliadačom. Webové aplikácie môžu pristúpiť k senzorom v zariadeniach a použiť ich na zlepšenie používateľského zážitku.

Výzvou sa tak stáva nie len vytvorenie celkového používateľského rozhrania, ale aj jednotlivých prvkov ako je navigácia či ovládacie prvky, ktoré by pomohli správne zobraziť obsah na malých zariadeniach a zároveň aby sa obsah prispôsobil aj tabletom, klasickým počítačom či pre veľké obrazovky televíznych príjmačov. Kedže rozlíšenia mobilných zariadení sa začínajú prelínat s klasickými počítačmi a aj klasické počítače pridávajú nové spôsoby ovládania dotykom, je dôležité rozoznať spôsob ovládania zariadenia a prispôsobiť navigáciu a obsah na dotyk alebo klávesnicu a myš. Rovnako je potrebné rozoznať aj internetové pripojenie zariadenia a automaticky mu odovzdať taký obsah, aby sa mu čo najrýchlejšie načítal a aby to používateľa nestalo zbytočný čas a financie za prenášané dátá.

## 2 Adaptácia webových aplikácií

### 2.1 Možnosti prispôsobenia

Existuje mnoho možností, na základe ktorých môžeme prispôsobiť webové aplikácie jednotlivým zariadeniam. Hlavným prvkom adaptácie je veľkosť a rozlíšenie displeja cieľového zariadenia. Ďalšími ale nemej dôležitými sú spôsoby ovládania zariadenia, jeho pripojenie na internet alebo samotná platforma.

#### 2.1.1 Veľkosť a rozlíšenie

Jednou zo základných možností prispôsobenia webových aplikácií je adaptácia na základe zobrazovacieho displeja zariadenia. Displeje môžu mať rozličné fyzické veľkosti ale aj rozlíšenia.

Časy s „rovnakým“ displejom na všetkých zariadeniach a podporou jednotného statického rozlíšenia na webových stránkach sú už dávno preč. S príchodom nových malých mobilných zariadení sa potreba prispôsobenia ešte viac umocnila. Veľkosti a rozlíšenia zariadení sa postupne začínali prelínat, dokonca v súčasnosti sa na trh uvádzajú zariadenia s väčším rozlíšením ako majú monitory stolových počítačov.



Obr. 1: Porovnanie zariadení vzhľadom na veľkosť displeja [40].  
Prevzaté z <http://www.lukew.com/ff/entry.asp?1649>

Kedže sa začínajú vyskytovať rovnaké rozlíšenia displeja na rozlične veľkých zariade-

niach alebo opačne, je potrebné medzi zariadeniami rozlíšovať aj inými spôsobmi. Je potrebné správne porozumieť jednotkám „pixel” a „viewport”.

Na rozdiel od pixelu definovaného W3C pomocou pozorovacieho uhlu a vzdialenosťi [34] existujú rôzne iné bežne používané jednotky „CSS pixel”, „device pixel” a „density-independent pixel” [24]. CSS pixel je abstraktný, môže sa zvyšovať alebo znižovať, používa sa bežne v kóde na definovanie rozmerov elementov. Device pixel je fyzický pixel nachádzajúci sa na zariadení. Pretože zariadenia majú stále viac fyzických pixelov a tým aj ich väčšiu hustotu, zaviedol sa pojem density-independent pixel. Ten je opäť abstraktný a predstavuje počet CSS pixelov optimálnych na prezeranie obsahu. Pokiaľ by nebol zavedený, tak zariadenia s veľkou hustotou pixelov by sa nedali použiť na bežné prezeranie obsahu, nakoľko pixely sú veľmi malé a zobrazený text alebo elementy by tak boli nečitaťné.

Viewport je celkové miesto potrebné na zobrazenie webovej stránky. Na mobilných zariadeniach je situácia komplikovanejšia, pretože stále existuje množstvo stránok, ktoré nie sú na ne optimalizované. Preto ho výrobcovia mobilných prehliadačov rozdelili na „layout viewport” a „visual viewport”. [24] Layout viewport je pre rozmiestnenie elementov celej stránky a visual viewport je definovaný pre elemty po priblížení stránky tak, že sa nezmestila na displej zariadenia.

### 2.1.2 Interakčné prostriedky

Postupom času začína upadať používanie klasických stolových počítačov ovládaných pomocou klávesnice a myši a presadzujú sa nové druhy mobilných zariadení so vstupným interfejsom v podobe množstva senzorov, ale hlavne s dotykovou plochou. Tá sa ako ovládaci prostriedok začína presadzovať okrem mobilných zariadení aj v notebookoch. Pri dizajne aplikácie je preto potrebné myslieť aj na takýchto používateľov. Okrem nich však existujú aj iné možnosti ovládania ako sú napríklad sledovanie pohybu pomocou web kamery, natočanie a posúvanie zariadenia získané pomocou gyroskopu a accelerometra či počúvanie hlasových povelov zaznamenaných cez mikrofón. Je tak potrebné brať do úvahy aj ďalšie možnosti.

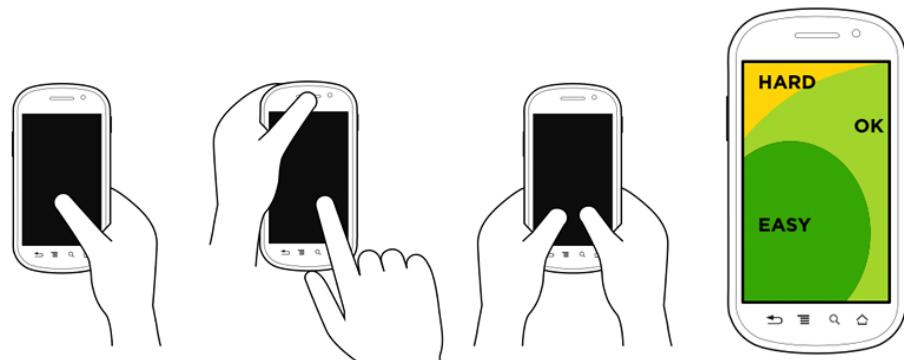
Webové prehliadače na televízoroch sú celkom dobré, ale interakcia s nimi je otrasná, tak ich nikto nepoužíva. Vstupné metódy sú tak oveľa dôležitejšie pri tvorbe dizajnu než veľkosť displejov. Vstup definuje ako musí vyzerať dizajn pre splnenie úloh [18]. Taktiež máme na tele nositeľné zariadenia. Súčasný interfejs Google Glass je limitovaný len na pári vstupných metód.

Univerzálny interakčný prostriedok pre web neexistuje. Musíme sa rozhodovať medzi dotykom, klávesnicou, myšou, rečou a ďalšími. Výzvou pre nasledujúce roky sa stáva nedefinovať dizajn len pre jeden druh interakcie ale pre mnoho.

## Dotyk

Dôležitým prvkom v prípade mobilných zariadení je možnosť ovládania aplikácií jednou rukou. Mobilné zariadenia sú využívané takmer pri každej príležitesti či vo vonkajšom alebo vnútornom prostredí, preto je potrebné správne prispôsobiť vzhľad a rozmiestnenie ovládacích prvkov. Pre ne platí, že najjednoduchšie dosiahnutelné časti sú v spodnej strane zariadenia a s postupom k hornému okraju možnosť dosiahnutia klesá [9].

Dotyk však už nie je izolovaný len pre mobilné zariadenia. V súčasnosti množstvo veľkých obrazoviek či notebookov disponuje dotykovým ovládaním. Teda dizajn aj pre väčšie displeje musí byť prispôsobený na dotyk. Optimálna veľkosť elementu by mala byť aspoň 7mm čo predstavuje 30 pixelov [8]. Ale veci nie sú len také jednoduché pretože existujú dynamické veľkosti displejov oproti fyzikálnym.



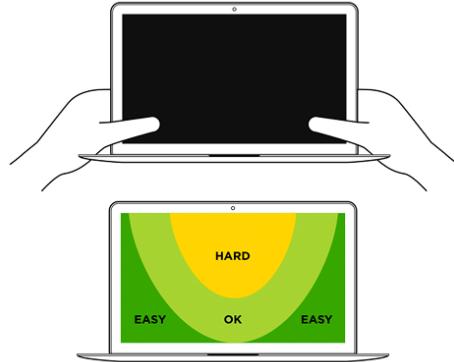
Obr. 2: Schopnosť ovládania mobilných telefónov [40].  
Prevzaté z <http://www.lukew.com/ff/entry.asp?1649>

Väčšie mobilné zariadenia alebo tablety už nie je možné pohodlne udržať v jednej ruke a tak je dôležité prispôsobiť ovládanie na dve ruky. Tablety sú držané v dvoch rukách za hranu a tak najlepšie dosiahnutelné miesta sú na jeho okrajoch [9].



Obr. 3: Schopnosť ovládania tabletov [40].  
Prevzaté z <http://www.lukew.com/ff/entry.asp?1649>

Podobné výsledky [9] sú aj pri novej kategórii zariadení, tzv. hybridných notebookov, ktoré okrem klávesnice obsahujú aj dotykový displej.



Obr. 4: Schopnosť ovládania hybridných počítačov [40].

Prevzaté z <http://www.lukew.com/ff/entry.asp?1649>

Najlepší interfejs na ovládanie webu je však niekedy úplne bezdotykový. Môžeme použiť ďalšie senzory a rečový vstup, ktorý je ďalším zlepšením interakčného dizajnu.

## Reč

Ovládanie aplikácií a zariadení pomocou reči môže byť užitočné pre prípady spojené s indispozíciou používateľov najmä v spojení so slabším zrakom či fyzickým postihnutím [4].

Možnosť rozpoznania reči vo webovej aplikácii je v súčasnosti experimentálna novinka a jej štandardizácia je zatiaľ veľmi ďaleko. Používateľ samozrejme musí explicitne povoliť použitie mikrofónu pre webovú aplikáciu. Špecifikácia je zatiaľ len vo forme návrhu a nie je ani zaradená do W3C štandardu HTML5 [35]. Podpora v prehliadočoch s enginom Blink sa však už nachádza od začiatku roku 2013 a základná funkčnosť bola odprezentovaná v rámci konferencie Google IO 2013. Rozpoznanie reči prebieha vzdialene na serveroch patriacich Googlu a zatiaľ neexistuje možnosť rozpoznania lokálne [5].

S rozpoznaním reči je spojená aj jeho syntéza, alebo preklad textu na reč. Tá je na tom v súčasnosti čo sa týka podpory a implementácie v prehliadačoch ešte horšie. Experimentálna funkčnosť existuje len v posledných buildoch prehliadočov. Naštastie sa táto funkcionala dá čiastočne nahradniť volaniami vzdialených webových služieb ako je napríklad „Google Translate“. Takáto syntéza však už neprebieha na zariadení, ale len sa zo serveru príjme zvuková nahrávka, ktorá sa následne prehrá.

## Pohyb používateľa

Rozpoznanie pohybu v reálnom čase je možné určiť podľa udalostí uskutočnených používateľom. Jednou z možností je použiť algoritmus na základe zmeny osvetlenia objektu [2] a tým určiť smer jeho pohybu.

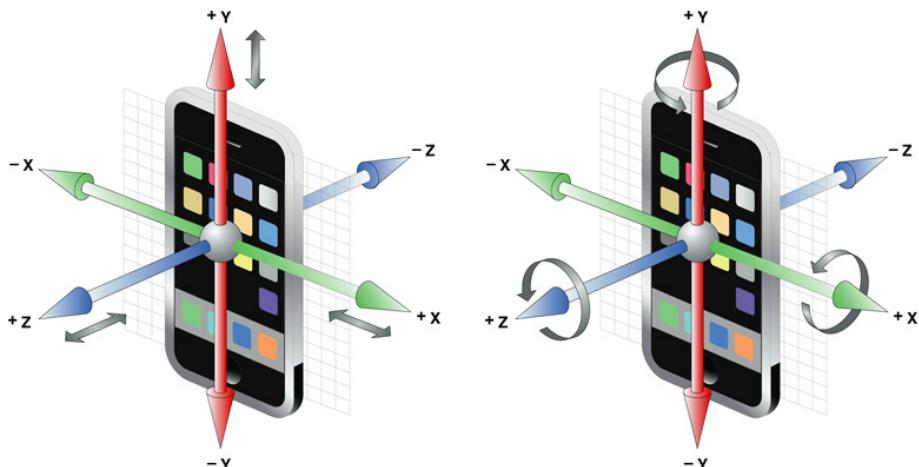
Moderné webové prehliadače umožňujú získať prístup aj k video streamu z web kamery používateľa. Na prístup je rovnako potrebné povolenie od používateľa. Tento video stream prebieha vo zvolenej frekvencii a sa dá odchytiť a následne uložiť do elementu canvas, kde už sa k jednotlivým vzorkám pristupuje ako k obrázku. Je možné odfiltrovať okolie a zachovať len podstatné informácie na základe ktorých sa získa pohyb, respektíve gesto od používateľa.

## Pohyb zariadenia

Na detekovanie pohybu zariadenia môžme použiť množstvo senzorov. Jednou z možností je použitie kombinácie vždy snímajúcich senzorov ako sú accelerometer, gyroskop a magnetometer, ktoré nám povedia ako sa zariadenie pohybuje v priestore. Schopnosť použitia týchto senzorov na poskytnutie presných informácií o pohybe zariadenia nám prináša množstvo nových možností v dizajne webových aplikácií [41].

Medzi dlhodobo podporované senzory najmä v mobilných zariadeniach patria accelerometer a gyroskop. Je k nim umožnený prístup priamo z webovej aplikácie aj bez priameho povolenia od používateľa. Dáta z nich získané je zároveň možné použiť na vytvorenie a rozpoznanie giest použitých na ovládanie aplikácie [25].

Accelerometer slúží na získanie zrýchlenia zariadenia v osiach x, y a z, gyroskop na získanie uhlového zrýchlenia okolo týchto osí. Ich rozdiel je znázornený na nasledujúcom obrázku:



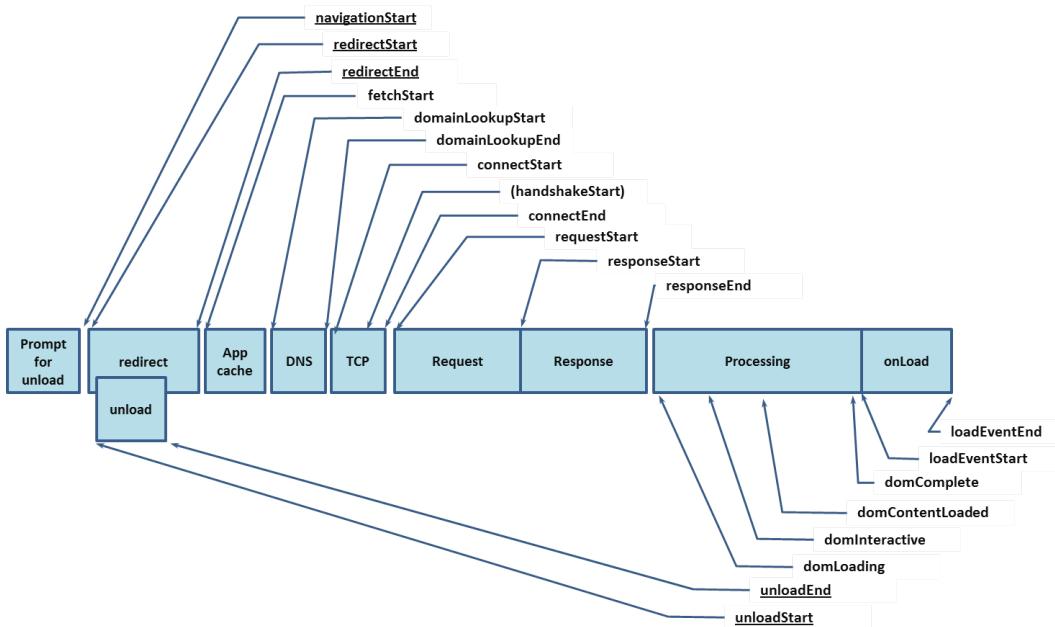
Obr. 5: Zaznamenávané údaje pomocou accelerometra a gyroskopu

### 2.1.3 Pripojenie

Spoločnou vlastnosťou webových aplikácií je, že pristupujú k rôznym zdrojom, ktoré sa môžu nachádzať okrem lokálneho úložiska aj na serverom, pomocou internetu. Spôsoby prenosu dát sú rôzne, od pevného pripojenia cez bezdrôtové až po mobilné, a každé z nich má iné vlastnosti.

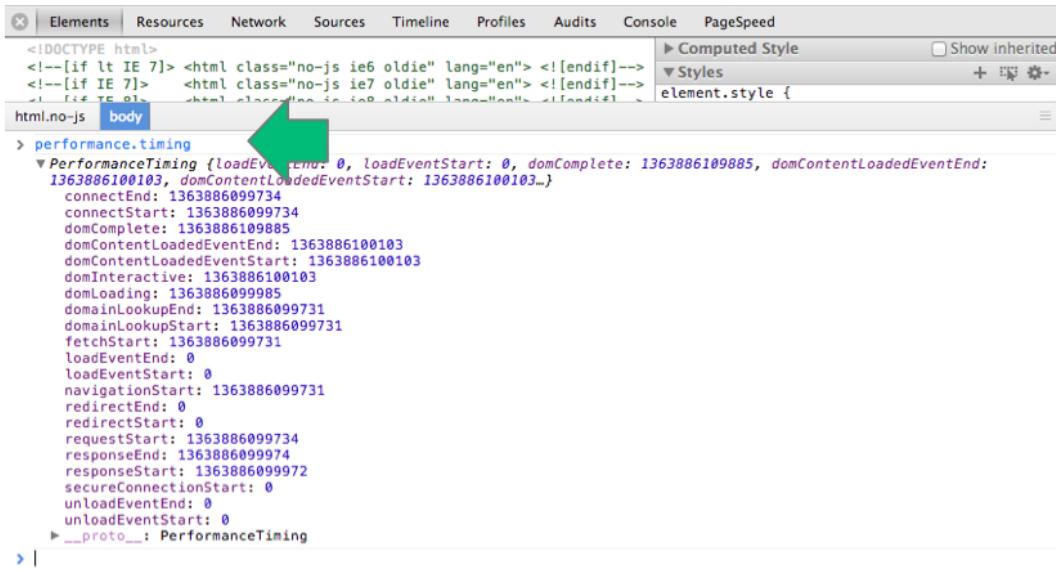
V poslednom období sa spolu s mobilnými zariadeniami rozširuje aj používanie mobilného internetu. Kedže našim cieľom je, aby sa webová stránka načítala používateľovi čo najrýchlejšie, respektívne ak chceme aby sa používateľ na našu stránku prišiel a príchod si nerozmyslel pri jej dlhom načítavní, tak musíme šetriť množstvom prenášaných dát. Takéto šetrenie dát zároveň šetrí aj peňaženky používateľov [11], hlavne pokiaľ sa jedná o roamingové dátá v zahraničí.

Načítanie webovej stránky sa skladá z viacerých fáz. Okrem samotnej konektivity používateľa aj spracovanie požiadavky serverom a následne vykonanie akcie v prehliadači. To je jediná fáza, ktorú môžeme ovplyvniť.



Obr. 6: Fázy spracovania požiadavky na server [33].  
Prevzaté z <http://www.w3.org/TR/navigation-timing/>

Tieto fázy môžme aj priamo merať pomocou interfacu `performance.timing`, ktorý ich zobrazuje priamo v podobe času [15, 17]. Vďaka tomu máme dokonalejší prehľad o používateľovom pripojení a vieme mu tak prispôsobiť jednotlivé komponenty stránky. W3C špecifikácia je vo fáze „Recommendation“ [33], ale stále hlavnou nevýhodou je chýbajúca podpora v niektorých prehliadačoch. Čiastočnou náhradou je meranie rozdielov dvoch časov, ale tak získame dátá len zo spracovania požiadavky na strane klienta.



```

<!DOCTYPE html>
<!--[if lt IE 7]> <html class="no-js ie6 oldie" lang="en"> <![endif]-->
<!--[if IE 7]>   <html class="no-js ie7 oldie" lang="en"> <![endif]-->
<!--[if IE 8]>   <html class="no-js ie8 oldie" lang="en"> <![endif]-->
<!--[if gt IE 8]> <html class="no-js" lang="en"> <![endif]-->
html.no-js body
  > performance.timing
    > PerformanceTiming {loadEventEnd: 0, loadEventStart: 0, domComplete: 1363886109885, domContentLoadedEventEnd: 1363886100103...}
      connectEnd: 1363886099734
      connectStart: 1363886099734
      domComplete: 1363886109885
      domContentLoadedEventEnd: 1363886100103
      domContentLoadedEventStart: 1363886100103
      domInteractive: 1363886100103
      domLoading: 1363886099985
      domainLookupEnd: 1363886099731
      domainLookupStart: 1363886099731
      fetchStart: 1363886099731
      loadEventEnd: 0
      loadEventStart: 0
      navigationStart: 1363886099731
      redirectEnd: 0
      redirectStart: 0
      requestStart: 1363886099734
      responseEnd: 1363886099974
      responseStart: 1363886099972
      secureConnectionStart: 0
      unloadEventEnd: 0
      unloadEventStart: 0
      > __proto__: PerformanceTiming

```

Obr. 7: Čas potrebný na spracovanie požiadavky na server.

Každá požiadavka na server niečo stojí. Ideálny prípad je taký, že medzi zariadením a serverom sa neprenášajú žiadne dátá a všetky prístupy ku zdrojom sa riešia len z lokálneho úložiska. V prípade internetových aplikácií to však väčšinou nie je úplne možné, pretože používateľ chce pristupovať k čo najčerstvejším dátam. Cieľom je však čo najviac limitovať požiadavky na server.

To, či je vôbec používateľ pripojený na internet vieme zistiť pomocou interfacu `navigator.onLine`, ktorý vráti hodnotu „true“ alebo „false“ a takisto môžeme počúvať na zmeny pripojenia vďaka „event listenerom“ na `window.online` a `window.offline`.

Rýchlosť, akou je používateľ pripojený na internet, je dostupná v objekte `navigator.connection` pod atribútom „bandwidth“ charakterizujúcej pripojenie v MB/s [32]. V prípade zmeny rýchlosťi je rovnako vyvolaný „event“. Nevýhodou je zatiaľ slabá podpora zo strany prehliadačov.

#### 2.1.4 Platforma

Rozhodovanie sa na základe platformy je taktiež veľmi dôležité. Umožňuje nám jednoduši dizajn a zmeniť počet potrebných komponentov na webovej stránke, ale taktiež vytvárať cielenú reklamu. Rozlišovanie prebieha na základe pola „user agent“ špecifickom pre každý prehliadač, respektíve operačný systém.

V prípade zisťovania podpory jednotlivých vlastností je však lepšie priamo zisťovať podporu komponentu zo strany prehliadača ako zisťovaním a porovnaním s platformou. Nemusíme si tak udržiavať databázu a neustále ju aktualizovať. Takéto riešenie je preto z pohľadu vývoja lepšie pre budúcnosť.

## 2.2 Adaptačné techniky <sup>1</sup>

S príchodom prvých mobilných zariadení existoval rozdiel medzi mobilným webom a webom pre desktopy a tak bolo pomocou servera jednoduché zistiť, aká verzia sa má zariadeniu zobraziť.

Pretože dnes už existuje mnoho zariadení od mobilných cez tablety až po klasické počítače a vzájomne sa prelínajú, je potrebné zabezpečiť, aby sa webový obsah zobrazoval správne na každom z nich.

*„There is no Mobile Web. There is only The Web, which we view in different ways. There is also no Desktop Web. Or Tablet Web. Thank you.” Stephen Hay [21]*

Tento výrok bol vyslovený už pred niekoľkými rokmi a v súčasnosti pri prelínaní rôznych zariadení sa stále viac potvrdzuje. Pre vývoj webovej stránky alebo aplikácie existuje viacero spôsobov [10], každý má svoje výhody a nevýhody. Správnosť výberu konkrétnej metódy záleží od toho, či ideme upravovať už existujúcu webovú verziu na rôzne zariadenia alebo či vytvárame novú aplikáciu a v neposlednom rade aj od vynaloženého úsilia či financií.

### 2.2.1 Responsive design

Pojem „Responsive design“ pôvodne zahrňal pravidlá tvorby dizajnu pre rôzne rozlíšenia a bol špecifikovaný v roku 2010. Všetkým zariadeniam sa posielala rovnaký HTML a javascript, rozdiel je len v dizajne. Dizajn sa zakladá na používaní flexibelného vzhľadu stránky, ktorý sa prispôsoboval rôznym zariadeniam, flexibilných obrázkoch, ktorých veľkosti sa prispôsobujú vzhľadu a CSS media queries. [26, 31] Až neskôr bol označený ako metóda na dosiahnutie výsledku.

Tvorba vzhľadu pomocou Responsive design znamená aplikáciu rozmerov jednotlivých elementov v percentuálnom pomere namiesto statických hodnôt, ktoré sa tak automaticky prispôsobujú rôznym veľkostiam. Ďalej sa vytvára webová stránka pre referenčné rozlíšenie a pomocou CSS media queries sa pridávajú ďalšie pravidlá pre elementy ako by mali vyzerat pri iných rozlíšeniach, orientácii či pomere strán. Pri použití takéhoto prístupu však nastávajú problémy správnosti zobrazenia na zariadeniach s menším či väčším rozlíšením displeja vzhľadom k ich fyzickej veľkosti, napríklad keď má televízor menšie rozlíšenie ako mobilné zariadenie.

---

<sup>1</sup>Tejto kapitole som sa už z časti venoval vo svojej bakalárskej práci Tvorba bohatých internetových aplikácií pre mobilné zariadenia [1]. V tomto dokumente sa nachádza rozšírená verzia doplnená o novo vzniknuté techniky adaptácie.

### **Výhody:**

- dosiahnutie nezávislosti zobrazenia pri rôznom rozlíšení zariadení, umožňuje prispôsobiť stránky pre mobilné zariadenia, tablety či stolové počítače
- umožňuje rýchlu tvorbu aplikácie

### **Nevýhody:**

- neumožňuje prispôsobenie obsahu, ale len jeho zobrazenie, stahujú sa všetky zdroje
- problémy pri zariadeniach s nepomerným rozlíšením displeja vzhľadom k fyzickej veľkosti

#### **2.2.2 Mobile First**

Postupný rozmach mobilných zariadení s menšími rozlíšeniami a veľkosťami displejov podnietil vznik novej metódy dizajnu - „Mobile First“. Základ tvorí metóda Responsive design, ale pôvodný návrh stránky sa nerobí pre desktopovú verziu ale na malé rozlíšenie mobilného zariadenia. Až pomocou media queries sa postupne prispôsobujú pravidlá elementov pre väčšie a väčšie rozlíšenie. Takto sa webová stránka prispôsobí pre všetky rozlíšenia od najmäňších až po tie najväčšie a zároveň je pripravená aj na ešte neexistujúce zariadenia, ktoré budú mať oveľa väčšie rozlíšenie ako majú tie súčasné.

Technika Mobile First však okrem samotného prispôsobenia zobrazenia stránok zahŕňa aj optimalizáciu webu pre používateľa či už z pohľadu UX alebo výkonu [39]. Predpokladá, že používateľ na web pristupuje najmä z mobilného zariadenia a tak sa mu snaží čo najviač tento pobyt spríjemniť. Minimalizuje sa množstvo údajov potrebných pre vyplnenie formulárov alebo sa pripraví zobrazenie správnej klávesnice pre zadávanie číselných vstupov, emailov či čistého textu bez toho, aby ju sám musel prepínať.

### **Výhody:**

- umožňuje zobrazenie obsahu pre rôzne rozlíšenia zariadení
- pokrýva zariadenia od najmenších až po tie najväčšie a zároveň webová stránka je pripravená aj na „zariadenia budúcnosti“
- vylepšenie UX

### **Nevýhody:**

- neumožňuje prispôsobenie obsahu, ale len úpravu vzhľadu
- pri existencii súčasného riešenia webu sa dizajn stránky musí vytvoriť od základu, čo však môže byť aj východa

### **2.2.3 Progressive enhancement**

S nástupom HTML5 sa v poslednom období sa stáva veľmi populárnu metódou Progressive Enhancement. Táto metóda je založená na princípe posielania jednotného počiatočného zdrojového kódu všetkým zariadeniam a umožňuje vytvoriť takzvanú „single page application”. Vykonávanie logiky aplikácie sa úplne presúva zo strany servera na klienta kde prebieha pomocou javascriptu. Tam sa ďalej presne špecifikuje čo, kedy a ako sa má vykonávať.

Každému zariadeniu je umožnený základný prístup k webu a jeho obsahu. Pokial však zariadenie, respektíve použitý webový prehliadač, podporuje pokročilejšie vlastnosti potrebné na beh aplikácie či prístup k senzorom, tak je mu ponúknutá obohatená verzia. Taktiež je možné doťahovať ďalšie zdroje zo servera len vtedy keď sú potrebné a tým sa zabraňuje zbytočným prenosom dát. Pomocou využívania pokričelejších možností HTML5 je dokonca možné tvoriť offline klientské webové aplikácie.

Pokial chceme pokryť celé spektrum zariadení tak je implementácia pomocou tejto metódy náročnejšia. V spojení s metódou responsive design je to ale výborné riešenie na prispôsobenie si obsahu a vzhľadu aplikácie.

#### **Výhody:**

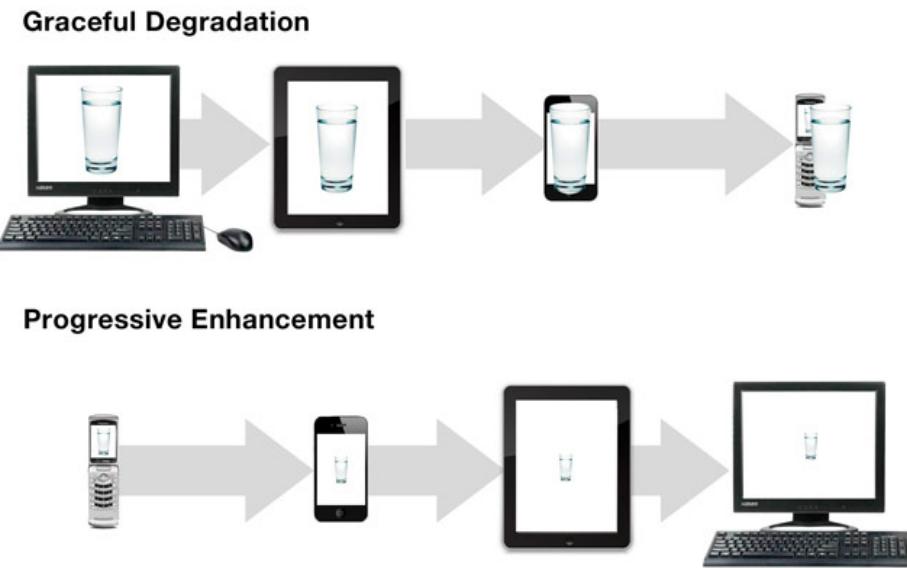
- úplne prispôsobenie obsahu a vzhľadu
- minimálne alebo žiadne dátové prenosy

#### **Nevýhody:**

- náročnejšia implementácia,
- rýchlosť vykonávania aplikácie závisí od výkonu zariadenia

### **2.2.4 Graceful degradation**

Vykonávanie aplikácie prebieha na zariadení, je však opačné ako v prípade metódy Progressive enhancement. Používateľovi sa predáva plne funkčná aplikácia nadizajnovaná na najlepšie zariadenia, kde sa na jednotlivé pokročilejšie funkcie postupne vypínajú ak nie sú dostupné pre použité zariadenie. Používa sa často na upravenie súčasnej už nasadenej verzie na potreby mobilných zariadení.



Obr. 8: Progressive enhancement vs Graceful degradation [13].  
Prevzaté z <http://bradfrostweb.com/blog/web/mobile-first-responsive-web-design/>

### Výhody:

- prispôsobenie obsahu a vzhľadu na jednoducšie zariadenia
- zachovanie súčasnej verzie webovej stránky

### Nevýhody:

- nezohľadňuje budúce zariadenia, už v súčasnosti majú niektoré tablety kvalitnejšie displeje ako stolové počítače

#### 2.2.5 Server-side Adaptation

Metóda Server-side Adaptation je používaná väčšinou webových stránok na detekovanie mobilného zariadenia a v súčasnosti patrí už medzi historické techniky. Pri prístupe na stránku sa pomocou servera detektuje zariadenie a je mu ponuknutá vhodná verzia stránky, väčšinou dochádza k presmerovaniu (na mobilnú verziu). Celá logika aplikácie sa nachádza na serveri. Stránka môže byť presne vytvorená pre dané zariadenie, takže nenastávajú problémy pri dizajne, je však potrebné mať na serveri nainštalovanú knižnicu na jeho detekovanie<sup>2</sup>. Detekcia zariadenia je pri priamej návštave stránky cez prehliadač úspešná, k problémom však prichádza pri návštave stránky cez iných klientov. Taktiež je dôležité mať databázu zariadení neustále aktualizovanú.

<sup>2</sup>napríklad DeviceAtlas alebo WURFL založené na detekovaní pola user agent

### **Výhody:**

- zobrazenie vhodnej stránky pre zariadenie, nestahuje sa nepotrebný obsah

### **Nevýhody:**

- potreba knižníc na detekovanie zariadenia, ktorú je nutné neustále aktualizovať
- dlhšie načítavanie stránky na pomalšom internetovom pripojení, pretože dochádza k presmerovaniu

V súčasnosti sa táto metóda nahradza pomocou metódy RESS alebo Progressive enhancement. Stále však potrebujeme informácie o type zariadenia a tak vznikajú mnohé knižnice ako napríklad WURFL.js<sup>3</sup> ktoré nám informácie o ňom prezradia aj na strane klienta.

### **2.2.6 RESS (Responsive Design + Server Side Components)**

Kombináciou techník Responsive Design a Server-side Adaptation vznikla novšia technika adaptácie. Pre každý typ zariadenia sa na serveri dynamicky vygenerujú pre neho špecifické komponenty webovej aplikácie, ktoré sa mu následne zobrazia a upravia sa na strane klienta pomocou responsive dizajnu. Pomocou detektie zariadenia sa však určí či sa samotný komponent má do výsledku pridať, respektívne či sa má nahradit nejakým iným.

### **Výhody:**

- jednoduchšia udržiavateľnosť, neexistujú rôzne verzie stránky ale len jedna z viacerými komponentami

### **Nevýhody:**

- potreba nainštalovaných knižníc na detekovanie zariadenia
- náročnejšia implementácia na strane servera

---

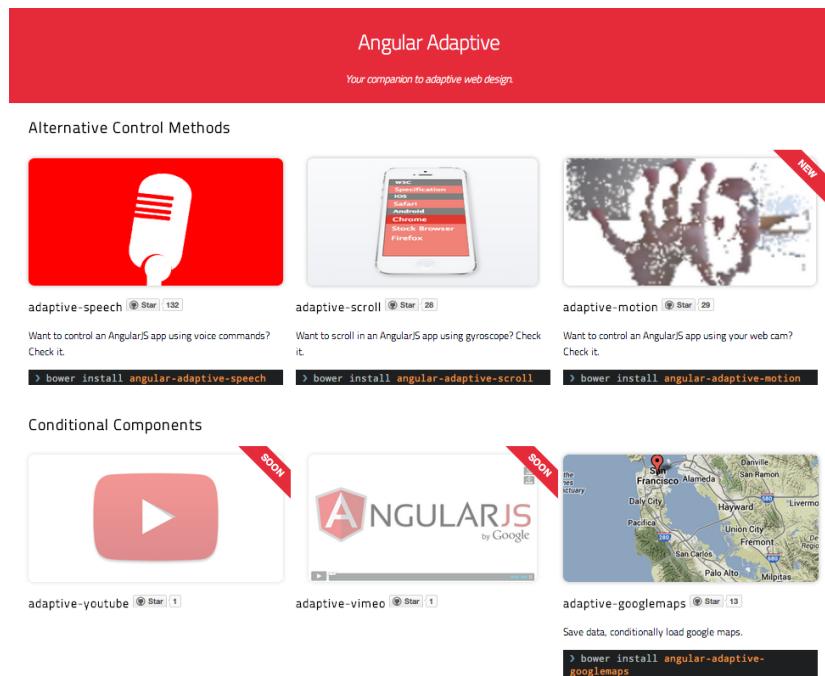
<sup>3</sup><http://wurfl.io/>

### 3 Návrh riešenia

V tejto časti práce sa venujem návrhu jednotlivých súčastí systému, ktoré umožnia adaptáciu webovej aplikácie na rôzne zariadenia. Vytvorenie jednotlivých súčastí knižnice vzišlo z potrieb komunity vývojárov a samotného trhu. Skúmali sa najmä možnosti alternatívnych spôsobov ovládania aplikácie a adaptácia používaných komponentov. Alternatívne možnosti ovládania aplikácie sú veľmi populárne a v prostredí webových aplikácií chýbal nástroj, ktorý by umožnil jednoduchú interakciu používateľa s aplikáciou. Potrebné bolo riešiť aj veľké množstvo zbytočne prenášaných dát pomocou internetu v prostrediach a na zariadeniach, ktoré ich všetky nepotrebuju.

Ako nástroj na overenie adaptácie sa vytvorila knižnica s názvom Angular-Adaptive<sup>4</sup> do populárneho JavaScriptového frameworku AngularJS, kde sa následne skúmali možnosti adaptácie komponentov pomocou metódy progressive enhancement.

Hlavným dôvodom výberu AngularJS bola možnosť vytvárania modulov, ktoré sa dajú následne jednoducho používať vo webových projektoch. Takáto podpora vytvárania externých modulov bude už o pár rokov natívna v prehliadačoch vďaka webovým komponentom [6]. Samotná distribúcia modulov prebieha pomocou balíčkovaciemu nástroja Bower<sup>5</sup>, ktorý je v súčasnosti štandardným distibučným miestom webových komponentov.



Obr. 9: Webová stránka vytvoreného projektu angular-adaptive

<sup>4</sup>Vytvorená knižnica Angular-Adaptive <http://angular-adaptive.github.io/> je pomocník vo svete adaptívneho web dizajnu.

<sup>5</sup>Bower <http://bower.io/> je balíčkovací manažér pre web.

Celý systém je možné rozdeliť do troch na sebe navzájom nezávislých logických celkov skladajúcich sa z nasledujúcich častí:

1. alternatívne spôsoby ovládania webovej aplikácie, kde sa vytvorili knižnice na ovládanie pomocou reči, pohybu a gyroskopu
2. adaptácie webových komponentov použitých v aplikáciach, kde sa skúmali hlavne ich možnosti prispôsobenia sa jednotlivým zariadeniam a internetovému pripojeniu používateľa
3. nástroje použité na implementáciu aplikácie

V nasledujúcich častiach kapitoly opisujem návrh implementácie jednotlivých súčasťí systému.

### **3.1 Alternatívne spôsoby ovládania**

Príchod mobilných zariadení priniesol a spopularizoval nový spôsob interakcie - ovládanie zariadenia pomocou dotyku. Ten však vždy nemusí byť ten najvhodnejší najmä v prostredí webových aplikácií. Súčasné moderné zariadenia obsahujú množstvo senzorov. Najmä mobilné zariadenia sú vybavené gyroskopom, ktorý môže uľahčiť ovládanie aplikácie len vďaka nakláňaniu zariadenia do strán. Podobne môže reagovať zariadenie sledovaním okolia vďaka vstupu z webovej kamery, ktorá je vstavaná v množstve zariadení. Typickým senzorom v zariadeniach je mikrofón umožňujúci zaznamenať zvuk, z ktorého sa analyzujú rečové povely a tie vedia zefektívniť ovládanie aplikácie.

#### **3.1.1 Reč**

Ovládanie mobilných zariadení pomocou rečových povelov sa v posledných rokoch teší narastajúcej popularite. Rozpoznávanie je ale špecifické pre jednotlivé mobilné platformy. JavaScriptové Web Speech API však umožňuje pridanie rozpoznania reči aj do webovej aplikácie.

Cieľom je umožniť ovládanie aplikácie zadávaním statických povelov, ale aj ovládať dynamicky vytvárané entity v aplikácii nezávislé od zvoleného jazyka používateľa. Ovládanie aplikácie by malo prebiehať kontinuálne bez akejkoľvek ďalšej potrebnej interakcie používateľa. Na ich základe bol vytvorený modul ktorý umožňuje pridať kompletné ovládanie webovej aplikácie. Používateľ však musí povoliť prístup aplikácie k mikrofónu. Po jeho potvrdení už prebieha kontinuálne rozpoznávanie reči.

Po pridaní modulu do webovej aplikácie je potrebné si ho nakonfigurovať. Musia sa vytvoriť úlohy, ktoré definujú správanie aplikácie a prípadne aj reč, v ktorej prebieha rozpoznávanie. Implementované rozpoznávanie reči v prehliadači Google Chrome podporuje 32 rôznych jazykov a ďalšie prízvuky vrátane slovenčiny.

Rozpoznanie úloh prebieha na základe regulárneho výrazu definovaného v konfigurácii danej úlohy. Vďaka tomu je zaručené, že ako rozpoznaný text sa získa dynamicky reťazec s ktorým je možné ďalej v aplikácii pracovať.

Nakoľko je v aplikácii dôležitá viacjazyčná podpora je potrebné ju správne definovať rozšírením kódu jazyka úlohy. Keď sa aktuálne nastavený jazyk rozpoznávania zhoduje s jazykom v úlohe a rozpoznaný text splňa podmienku regulárneho výrazu, tak sa zavolá asociovaná callback funkcia. Tá v parametri obsahuje rozpoznaný text v podobe textového reťazca, s ktorým sa dá ďalej pracovať. Príklad konfigurácie na pridanie novej úlohy má nasledujúci tvar:

---

```
1 'someTask': {
2   'regex': /^do .+/gi,
3   'lang': 'en-US',
4   'call': function(utterance){
5     // do something with utterance
6   }
7 }
```

---

Takýchto konfigurácií je možné pridať ľubovoľný počet. Takto navrhnutú konfiguráciu je možné zavolať globálne v rámci aplikácie alebo dynamicky vďaka podpore regulárneho výrazu a frameworku AngularJS pre dynamicky vytvárané elementy. V prípade globálneho volania stačí použiť samotnú konfiguráciu:

---

```
1 $speechRecognition.listenUtterance(
2   $scope.recognition['en-US']['someTask']
3 );
```

---

Pri dynamicky vytváraných elementoch je okrem samotnej konfigurácie s regulárnym výrazom potrebné pridať aj vstupný prvok elementu voči ktorému sa bude rozpoznávanie vykonávať. Takto sa vzájomne rozlíšia jednotlivé dynamické elementy a asociovaná úloha sa vykoná len pre jeden prvok.

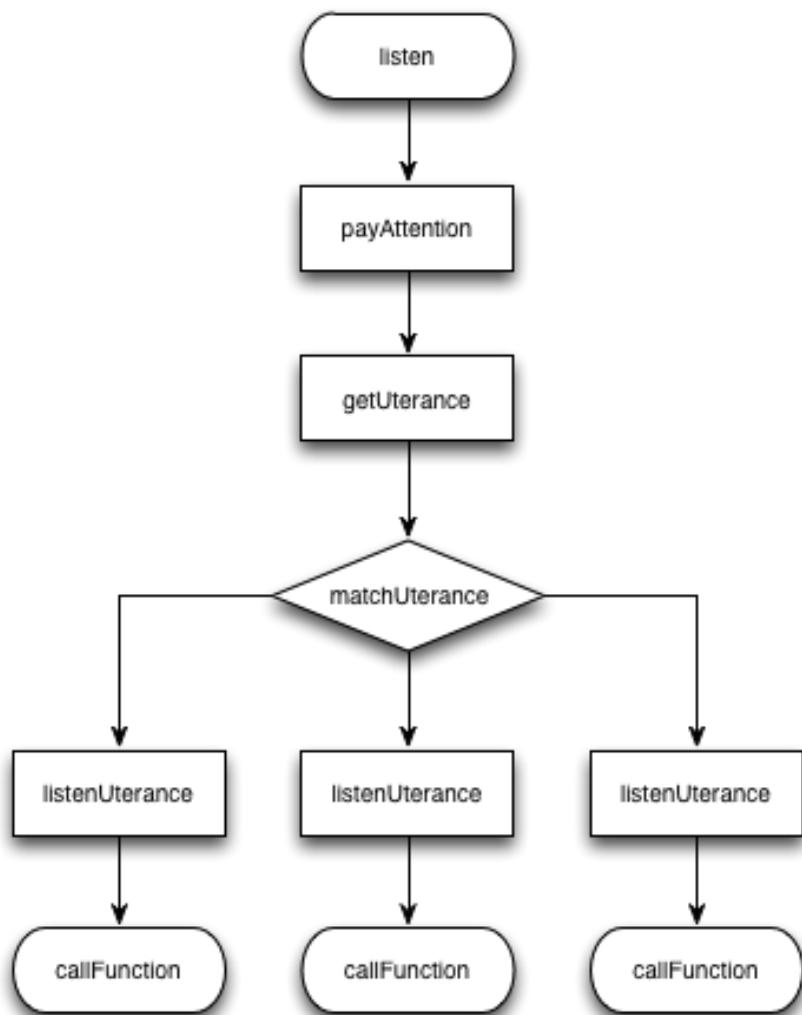
---

```
1 $scope.todos = ['buy milk', 'write essay'];
2
3 <li ng-repeat="todo in todos">
4   <div speechrecognition="{'tasks': recognition['en-US']['someTask'],
5     'input': todo}">{{todo}}</div>
6 </li>
```

---

Napríklad ak sa v aplikácii pracuje so zoznamom úloh, tak globálne je možné obsluhovať pridanie novej úlohy a dynamicky vytvárané elementy môžu mať vlastnú obsluhu, ktorá počúva na ich meno.

Niekedy je potrebné v aplikácii na istý čas ignorovať ovládanie pomocou reči. Kvôli takýmto prípadom sa pridala možnosť jeho nastavenia v podobe metódy payAttention. Diagram navrhnutého algoritmu ovládania webovej aplikácie pomoc reči sa nachádza na nasledujúcom obrázku:



Obr. 10: Algoritmus ovládania aplikácie pomocou reči

Použitie regulárneho výrazu pri konfigurácii úlohy má samozrejme aj nevýhodu. Najväčšou je potrebnosť presného určenia výrazu, nakoľko v opačnom prípade sa môžu detektovať aj nesprávne príkazy. Nevyhnutnosťou je aj stále pripojenie na internet kvôli vzdialenému rozpoznaniu reči, pričom samotné rozpoznanie nemusí byť vždy správne.

### 3.1.2 Pohyb

Ovládanie webovej aplikácie pohybom prebieha vďaka webovej kamere. Cielom riešenia je rozpoznať základné smery pohybov používateľa a umožniť na ne namapovať funkciu aplikácie, ktorá sa vykoná po ich správnom rozpoznaní.

Video sa sníma kontinuálne a zachytený video stream z kamery je s frekvenciou 60 fps. Každý jeden snímok je následne vykreslený do elementu canvas, v ktorom je možné s ním pracovať. Ten je tvorený množinou pixelov s rgba hodnotami.



Obr. 11: Vizualizácia videa zachyteného web kamerou

Získaný obrázok sa následne upraví pomocou hsv filtra aplikovaného na všetky pixely aby sa detekovali oblasti s kožou používateľa, pričom hsv filter je možné nakonfigurovať na vlastné hodnoty. Vďaka týmto oblastiam je rozpoznaný používateľ a je možné spracovať jednotlivé smery pohybov. Upravený snímok je tvorený dvomi farbami: tmavou znamenajúcou kožu a bielou pre ostatné oblasti. Rozpoznanie prebieha nasledovne:

---

```
1 if ( (
2     (hsv[0] > hsvFilter.huemin && hsv[0] < hsvFilter.huemax) ||
3     (hsv[0] > hsvFilter.losemin && hsv[0] < losemax)
4 ) &&
5     (hsv[1] > hsvFilter.satmin && hsv[1] < hsvFilter.satmax) &&
6     (hsv[2] > hsvFilter.valmin && hsv[2] < hsvFilter.valmax)
7 ) {
8     skinFilter[pix] = r;
9     skinFilter[pix+1] = g;
10    skinFilter[pix+2] = b;
11    skinFilter[pix+3] = a;
12 }
```

---



Obr. 12: Vizualizácia videa po aplikovaní filtra detekcie kože

Ako je možné vidieť, detekcia kože nie je úplne správna a na základe hsv sa detekuju aj iné oblasti. Následne prebieha ďalšie upravovanie pri ktorom sa detekuju len samotné hrany. Ich detekcia prebieha na základe rozdielov farebnosti pixelov posledných dvoch snímkov [2] s aplikovaným filtrom kože. Zistí sa celkový rozdiel v rgba hodnote každého pixelu a pokiaľ nastala výrazná zmena, tak sa detekuje ako hrana a zafarbí sa:

---

```
1 var rgbaDelta = Math.abs(draw.data[pix] - lastDraw.data[pix]) +
2     Math.abs(draw.data[pix+1] - lastDraw.data[pix+1]) +
3     Math.abs(draw.data[pix+2] - lastDraw.data[pix+2]);
4
5 if (rgbaDelta > threshold.rgb){
6     edge.data[pix] = 0;
7     edge.data[pix+1] = 0;
8     edge.data[pix+2] = 0;
9     edge.data[pix+3] = 255;
10 }
```

---

Opäť vznikne snímok tvorený dvomi farbami, ale tentokrát tmavá farba sa nachádza len na hranách a biela na ostatných častiach obrázku. Obrázok definuje miesta, na ktorých sa medzi snímkami vykonával pohyb. Na základe sekvencie takýchto obrázkov hrán vieme presnejšie určiť smer pohybu používateľa.



Obr. 13: Vizualizácia videa po aplikovaní filtra detekcie hrán

Ked' je obrázok v už takomto stave tak je možné začať rozpoznávať predvádzané gesto. Spočíta sa celkový počet zmenených bodov a pomer zmenených bodov v osiach x a y obrázka. Postupným hromadením snímkov a spočítavaním všetkých zmenených bodov sa určí celkový smer pohybu. Celkovo sa rozpoznávajú štyri základné gestá, kde po úspešnom rozpoznaní sa vykonajú priradené funkcie:

- onSwipeLeft - pohyb zprava vľavo
- onSwipeRight - pohyb zľava vpravo
- onSwipeUp - pohyb zdola hore
- onSwipeDown - pohyb zhora dole

Príklad nastavenia funkcie na odchytenie gesta má nasledujúci tvar, kde sa asociouje callback funkcia, ktorá sa má následne v programe vykonať.

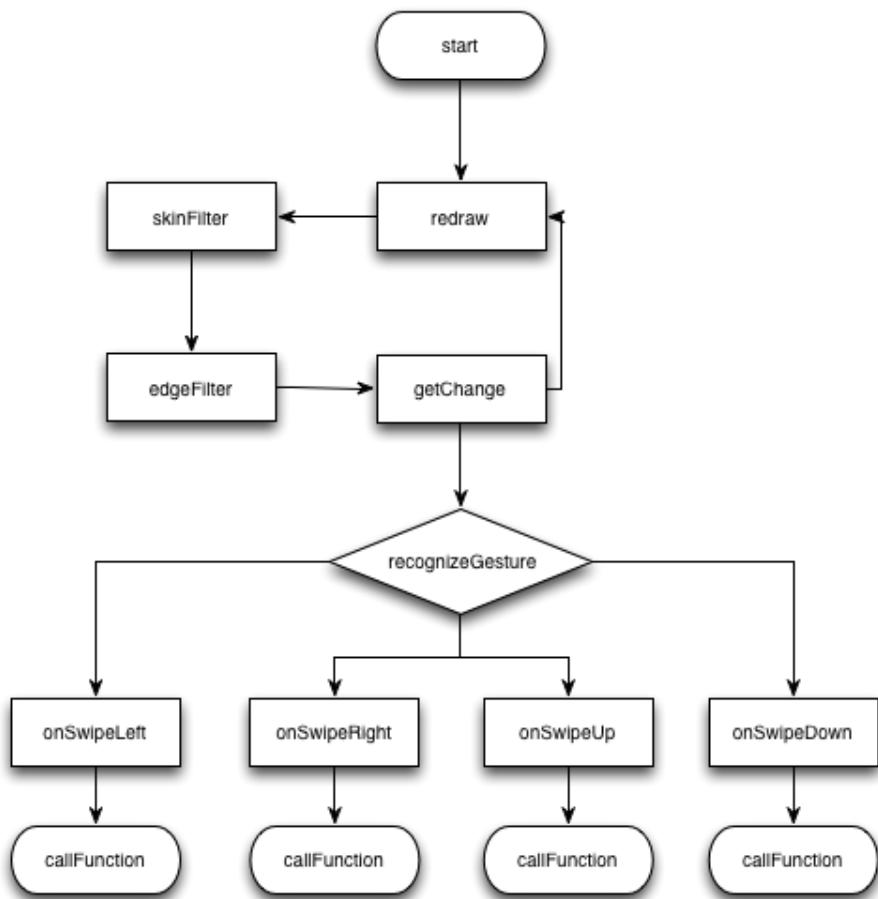
---

```
1 $motion.onSwipeRight(function(){  
2     // DO SOMETHING  
3 });
```

---

Citlivosť aplikovania filtrov a rozpoznávania gesta je samozrejme možné konfigurovať nastavením thresholdu a úpravou hsv filtra. Problémy nastávajú najmä pri extrémnych vonkajších podmienkach, ktorými sú nízke alebo príliš vysoké osvetlenie. Tie sa však dajú eliminovať jeho zmenou.

Vytvorený modul podporuje aj ďalšie udalosti na ktoré je možné viazať funkcie, akými sú začiatok a koniec snímania, pretože na prístup k web kamere je potrebné explicitné povolenie od používateľa. Diagram navrhnutého algoritmu ovládania webovej aplikácie pomoc pohybu sa nachádza na nasledujúcim obrázku:



Obr. 14: Algoritmus ovládania aplikácie pomocou pohybu

### Vizualizácia

V aplikácii je možné pridať aj zobrazenie aktuálneho náhľadu z webovej kamery a to použitím vytvorennej direktívy pridaním atribútu „adaptive-motion“ ku canvas elementu. Atribút môže nadobúdať nasledujúce hodnoty:

- video - zobrazí aktuálny náhľad z web kamery zariadenia, ktorého detail je zobrazený na obrázku 11
- skin - zobrazí video stream po aplikovaní filtra rozpoznania kože, ktorého detail je zobrazený na obrázku 12
- edge - zobrazí video stream po aplikovaní filtra rozpoznania hrán, ktorého detail je zobrazený na obrázku 13

### 3.1.3 Gyroskop

Jednou z možností ovládania webovej aplikácie je použitie gyroskopu. Na tento prípad som vytvoril modul pomocou ktorého je možné vo webovej aplikácií skrolovať len pomocou natáčania zariadenia do strán bez ďalšej potrebnej interakcie používateľa. Cieľom je zároveň umožniť ovplyvniť samotnú rýchlosť skrolovania na základe veľkosti natočenia oproti počiatocnej pozícii zariadenia.

Modul sa inicializuje pridaním atribútu „adaptivescroll“ k elementu, v ktorom má prebiehať skrolovanie. Tým môže byť napríklad element „body“, ktorý zohľadňuje skrolovanie celej webovej stránky, alebo aj nejaký iný vnorený element. V takom prípade však skrolovanie prebieha len v rámci daného elementu.

---

```
1 <body adaptivescroll> </body>
```

---

Následne stačí zavalať metódu so začatím sledovania polohy zariadenia, v ktorej sa môže upraviť ignorovaná hranica natočenia v uhloch a skrolovanie prebieha automaticky. V prípade potreby je toto sledovanie možné aj ukončiť.

Po začatí sledovania aktuálneho natočenia zariadenia sa uloží jeho počiatocná hodnota, ktorá sa používa ako relatívna hranica medzi stranami na ktoré prebieha skrolovanie. Postupným natáčaním zariadenia v smere hodinových ručičiek okolo osi beta od počiatocnej hodnoty k používateľovi sa vykonáva skrolovanie elementu smerom dolu, natáčaním na opačnú stranu od počiatocnej hodnoty prebieha skrolovanie hore.

Do úvahy pri zohľadňovaní rýchlosť skrolovania sa berie aj absolúttna hodnota natočenia zariadenia od počiatocnej hodnoty zariadenia. Čím je táto hodnota väčšia, tým rýchlejšie prebieha skrolovanie. Výpočet rozdielu natočenie zohľadňuje smer rotácie a je určený na základe počiatocnej rotácie, aktuálneho natočenia a celkovej veľkosti rotácie pre danú os:

---

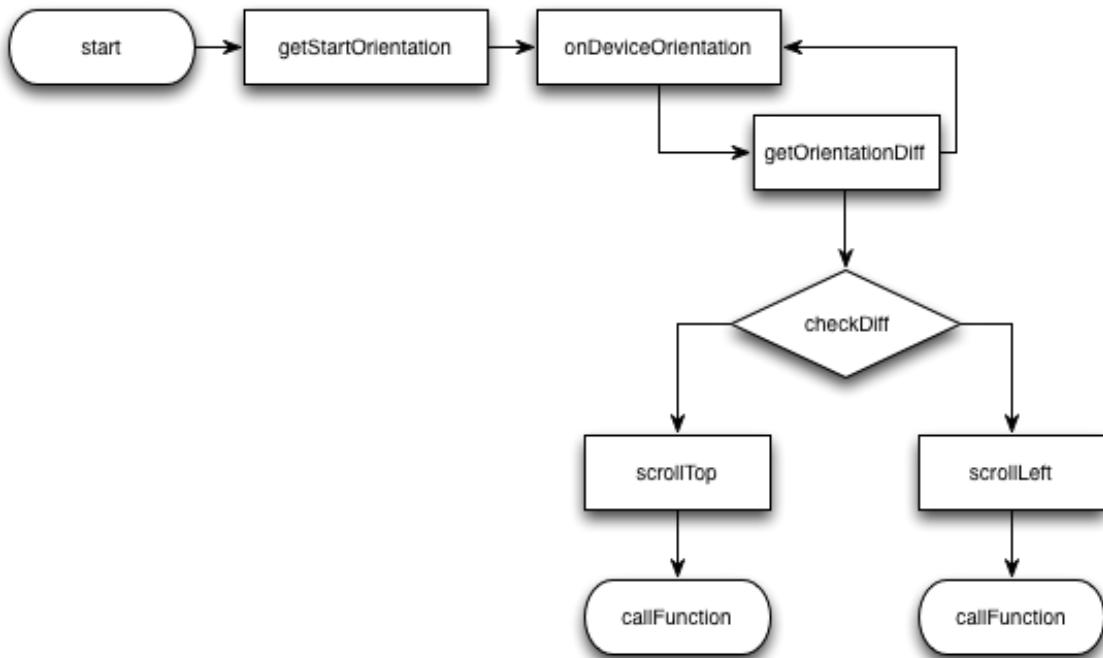
```
1 var getDiff = function(startOrientation, actualOrientation, interval){  
2     var d1 = startOrientation - actualOrientation;  
3     var d2 = startOrientation - actualOrientation - interval;  
4     return (Math.abs(d1) < Math.abs(d2)) ? d1 : d2;  
5};
```

---

Samotný proces skrolovania prebieha vďaka číselnej zmene offsetu elementu. Plynulosť skrolovania je zaručená použitím animácií pomocou requestAnimationFrame. Tá prebieha s frekvenciou 60 fps a je časovaná priamo prehliadačom. Tej však chýba podpora v straších verziách, v takom prípade musí byť použitý manuálny časovač aplikácie.

Okrem skrolovania modul obsahuje aj ďalšie metódy onalpha, onbeta a ongamma, ktorých návratová funkcia sa zavolá vždy po zmene rotácie zariadenia a je ich možné použiť

na iné potreby v aplikácii. Funkcia obsahuje parameter absolútne zmenu pozície v uhloch od počiatocnej k aktuálnej, ktorá sa používa aj pri skrolovaní v aplikácii. Diagram navrhnutého algoritmu ovládania webovej aplikácie pomoc gyroskopu sa nachádza na nasledujúcom obrázku:



Obr. 15: Algoritmus ovládania aplikácie pomocou gyroskopu

### 3.2 Komponenty

Jednou z najväčších kritík voči technike „responsive Web design“ je, že webové stránky sú príliš veľké a teda sa predlžuje čas ich načítania. Ale my vieme spraviť webové stránky s dobrým výkonom, len musíme vykonať prácu navyše [27].

Všetci by sme sa mali zaujímať o rýchlosť načítavania a mobilný výkon stránok, pretože ľudia potrebujú zážitok z rýchlosť stránok. Existuje mnoho štúdií ktoré zdôrazňujú dôležitosť rýchlosť. Viac ako polovica ľudí so zlou skúsenosťou s načítavania stránky na mobile sa nevráti späť [16]. 73% ľudí s mobilnym pripojením zažilo stránky, ktoré sú príliš pomalé, 1 sekundové spomalenie načítania vedie k poklesu o 7% pri konverziach [28]. Rýchlosť je základnou časťou našej stratégie a všetci by sa mali o ňu starat. Môžeme merať jej dopad na metriky a sledovať ju.

Zároveň existuje aj mnoho druhov internetových pripojení. Rýchlosť mobilných sietí sa sice zvýšili, ale stále nám to moc nepomôže pretože časové oneskorenia sú stále príliš dlhé. Latencia je jednou z najvýznamnejších častí pri stahovaní stránky. LTE štandard redukuje

latenciu k BTS stanicam o niekoľko milisekúnd ale my stále potrebujeme vykonať viacero požiadaviek na stiahnutie dát do zariadení [16].

Čo by sme teda mali urobiť? Môžme vytvoriť znovupoužiteľné webové komponenty a podmienene načítať ich zdroje. Webové komponenty nám umožnia vytvoriť vlastné HTML elementy v prehliadači. Pomocou nich môžme rozšíriť jednoduchý element alebo za nich schovať celú aplikačnú logiku [6]. Môžme vytvoriť webovú aplikáciu znovupoužiteľným a zapúzdreným spôsobom.

Dôležité sú však webové komponenty s ktorými používateľ webovej aplikácie priamo interaguje. Tieto boli skúmané najmä z hľadiska optimalizácie na rôzne zariadenia a rýchlosťi pripojenia.

### 3.2.1 Video

Populárnym doplnkom súčasných webových stránok je priložené video, ktoré sa často nachádza na serveroch YouTube alebo Vimeo.

Problémom je, že pri vložení videa pomocou iframe alebo embed API sa uskutočňuje množstvo dopytov na servery a prenášajú sa zbytočné dátá aj keď používateľ video nezaujíma a vôbec si ho neprehrá. Okrem zbytočne prenášaných dát sa aj znižuje výkon, nakoľko určitý čas trvá spracovanie požiadaviek.

Nasledujúce dátá sa prenesú pri zobrazení stránky, na ktorú bolo vložené video zo servera YouTube pomocou dostupného iframe API:

	<a href="http://www.youtube.com/embed/kxopViU98Xo?autoplay=0">kxopViU98Xo?autoplay=0</a>	GET	200 OK	text/html	Other	3.2 KB 5.2 KB	404 ms 400 ms
	<a href="http://s.ytimg.com/yts/cssbin/www-embed-vflKrJ4Q.css">www-embed-vflKrJ4Q.css</a>	GET	200 OK	text/css	<a href="#">kxopViU98Xo:7</a> Parser	26.7 KB 145 KB	124 ms 58 ms
	<a href="http://s.ytimg.com/yts/jsbin/www-embed-vflDpvIm5.js">www-embed-vflDpvIm5.js</a>	GET	200 OK	text/javascript	<a href="#">kxopViU98Xo:20</a> Parser	32.4 KB 84.5 KB	156 ms 81 ms
	<a href="http://s.ytimg.com/yts/swfbin/watch_as3-vflZFBBBo.swf">watch_as3-vflZFBBBo.swf</a>	GET	200 OK	application...	Other	282 KB 282 KB	530 ms 56 ms
	<a href="http://i4.ytimg.com/crossdomain.xml">crossdomain.xml</a>	GET	200 OK	text/x-crossdomainxml	<a href="#">www-embed-vflDpvIm5</a> Script	478 B 102 B	90 ms 89 ms
	<a href="http://i4.ytimg.com/vi/kxopViU98Xo/hqdefault.jpg">hqdefault.jpg</a>	GET	200 OK	image/jpeg	Other	10.8 KB 10.5 KB	68 ms 52 ms

Obr. 16: Prenášané dátá pri požiadavke na video zo servera YouTube

Celkovo sa vykoná 6 požiadaviek a prenesie sa 350 kB dát bez toho, aby používateľ spustil video. Podobná situácia sa opakuje aj pri požiadavke na video zo servera Vimeo pomocou iframe API.

 <a href="#">57625268?autoplay=0</a> player.vimeo.com/video	GET	200 OK	text/html	Other	3.9 KB 10.0 KB	405 ms 401 ms
 <a href="#">player.core.opt.css</a> a.vimeocdn.com/p/1.4.31/css	GET	200 OK	text/css	<a href="#">57625268:1</a> Parser	2.3 KB 7.5 KB	54 ms 53 ms
 <a href="#">player.core.opt.js</a> a.vimeocdn.com/p/1.4.31/js	GET	200 OK	application...	<a href="#">57625268:1</a> Parser	14.7 KB 39.1 KB	152 ms 57 ms
 <a href="#">399544952_295.jpg</a> b.vimeocdn.com/ts/399/544	GET	200 OK	image/jpeg	<a href="#">57625268:1</a> Parser	10.0 KB 9.6 KB	571 ms 359 ms
 <a href="#">gajs</a> www.google-analytics.com	GET	200 OK	text/javascript	<a href="#">57625268:1</a> Script	15.7 KB 38.5 KB	124 ms 64 ms
 <a href="#">3591052_75.jpg</a> b.vimeocdn.com/ps/359/105	GET	200 OK	image/jpeg	<a href="#">57625268:1</a> Parser	1.8 KB 1.4 KB	49 ms 48 ms
 <a href="#">swfobject.v2.2.js</a> a.vimeocdn.com/p/1.4.31/js	GET	200 OK	application...	<a href="#">player.core.opt.js:6</a> Script	4.1 KB 9.8 KB	86 ms 81 ms
 <a href="#">_utm.gif?utmwv=5.4.0&amp;utms=1&amp;utmn=</a> www.google-analytics.com	GET	200 OK	image/gif	<a href="#">ga.js:60</a> Script	376 B 35 B	68 ms 68 ms
 <a href="#">_utm.gif?utmwv=5.4.0&amp;utms=2&amp;utmn=</a> www.google-analytics.com	GET	200 OK	image/gif	<a href="#">ga.js:60</a> Script	376 B 35 B	123 ms 122 ms
 <a href="#">moogalover.swf?v=1.0.0</a> a.vimeocdn.com/p/flash/moogalover/1.1.	GET	200 OK	application...	<a href="#">swfobject.v2.2.js:1</a> Script	33.5 KB 33.2 KB	127 ms 50 ms
 <a href="#">plus_icon.gif</a> a.vimeocdn.com/images_v6	GET	200 OK	image/gif	Other	396 B 89 B	49 ms 48 ms
 <a href="#">crossdomain.xml</a> b.vimeocdn.com	GET	200 OK	application...	Other	582 B 342 B	42 ms 41 ms
 <a href="#">crossdomain.xml</a> player.vimeo.com	GET	200 OK	application...	Other	714 B 342 B	291 ms 291 ms
 <a href="#">3591052_75.jpg</a> b.vimeocdn.com/ps/359/105	GET	200 OK	image/jpeg	Other	1.8 KB 1.4 KB	41 ms 41 ms
 <a href="#">399544952_640.jpg</a> b.vimeocdn.com/ts/399/544	GET	200 OK	image/jpeg	Other	29.9 KB 29.5 KB	149 ms 48 ms

Obr. 17: Prenášané dátá pri požiadavke na video zo servera Vimeo

V tomto prípade sa dokonca vykoná 15 požiadaviek a prenesie sa 120 kB dát. V prípade použitia javascriptového API s HTML 5 prehrávačom videa sa síce nestiahnu flashové komponenty, ale nahradia ich rovnakoveľké javascriptové súbory.

Lepšie riešenie je použiť podmienené načítavanie. Najskôr sa načíta len obrázok videa a pridajú sa jednoduché štylistické prvky aby vytvorený element pripomínal video súbor a až po kliknutí sa urobia dopyty na vzdialenosť servera s automatickým prehratím videa. Výhodou takého riešenia je zmenšenie veľkosti dopytov a s tým spojené šetrenie dát používateľov.

Čo sa týka získania obrázkového náhľadu videa, tak YouTube túto možnosť priamo poskytuje a záleží len od identifikátora videa. K obrázku je tak možné pristúpiť priamo. Navyše po vykonaní požiadaviek na server po kliknutí na obrázok a následnom spustení videa pomocou API sa už daný obrázok nachádza v pamäti, tak sa nemusia robiť ďalšie požiadavky.

Samotný proces automatického spustenia YouTube videa nie je úplne jednoduchý, nakoľko nastavenie hodnoty „autoplay“ pri vložení elementu iframe s videom na stránku nefunguje na mobilnom prehliadači Safari, kde je táto možnosť zakázaná. V takomto prípade by sa po kliknutí na obrázok videa načítal len element s videom a aby sa samotné video začalo prehrávať očakáva sa ďalšie kliknutie. Takéto správanie je neželené a je proti používateľskému zážitku.

Preto bolo potrebné vymyslieť lepšie riešenie. To spočíva v načítaní scriptu s javascriptom

tovým youtube API a následným vytvorením iframe elementu s videom až pomocou neho. Takto máme prístup k programátorskému ovládaniu prehravača videa a môžme ho spustiť keď potrebujeme, teda po kliknutí na obrázok s náhľadom videa. Takéto riešenie už funguje aj na iOS s mobilným prehliadačom Safari.

Stále však máme problém pri staršej verzii iOS 5 a menej, tam nefunguje ani programátorske spustenie videa. V tejto verzii iOS sa však ešte distribuovala predinštalovaná aplikácia na prehrávanie Youtube videí, ktorá bola v neskorších verziach odstránená a je ju možné stiahnuť z iTunes. Výhodou je, že video môžme otvoriť priamo v nej pomocou youtube url schémy. Musíme však používaný prehliadač správne detektovať a následne sa rozhodnúť, aké prehrávanie zvolíme.

Na detekovanie funkcionality otvorenia youtube videa v aplikácii pomocou systému iOS a jej zistovaním len z pola „user agent“ nie je správne, nakolko by bolo potrebné získať úplne všetky verzie systému, ktorých je mnoho, a následne porovnať s verziou zariadenia.

Lepším riešením je vytvorenie testu funkčnosti prehrávania videa, ktoré je riešené pomocou nástroja Modernizr<sup>6</sup> umožňujúcim detekciu základných HTML5 a CSS3 vlastností prehliadača a vytváranie vlastných testov. Následne sa na základe výsledku testu rozhodnemu akú akciu vykonáť. Problémom je, že testy sa vykonávajú po načítaní stránky a nechceme používateľovi automaticky otvoriť natívnu aplikáciu alebo ho presmerovať na stránky youtube. Preto je zvolená iná varianta a detekuje sa podpora vlastnosti, ktorá bola pridaná až v novšej verzii iOS 6. Konkrétnie sa jedná o podporu jednotiek „vh a vw“ (viewport height a viewport width) slúžiacich na nastavenie veľkosti DOM elementu. Detekcia či je zariadenie iOS vychádza z pola user agent, ale už sa nezisťuje jej verzia.

Výsledok je taký, že po kliknutí na obrázok s náhľadom videa a vyhodnotení testu prehrávania sa začne prehrávať v prehliadači alebo v natívnej aplikácii.

V prípade Vimea je situácia trochu komplikovanejšia pretože k náhľadovému obrázku sa priamo nevieme dostať a je potrebné urobiť jednu požiadavku na API, ktorá vráti informácie o videu. Vykonanie požiadavky síca nejaký čas trvá, ale aj tak je výsledok z pohľadu prenášaný údajov lepší ako robiť požiadavku priamo na samotné video.

S prehrávaním videa zo služby vimeo je situácia podobná, neexistuje však natívna aplikácia a po klinutí na náhľad videa je používateľ v starších verziách iOS presmerovaný na stránky vimeo, v novších verziách iOS a v Androide sa mu automaticky prehrá.

### 3.2.2 Mapy

Mapy podobne ako videá vytvárajú nechcené dátové prenosy, dokonca ich ešte aj prevyšujú pretože sa neprenášajú len údaje o aktuálne zobrazenej časti mapy ale aj jej okolie. Okrem nich navyše na webe neposkytujú taký plnohodnotný zážitok z prezerania ako v natívnej

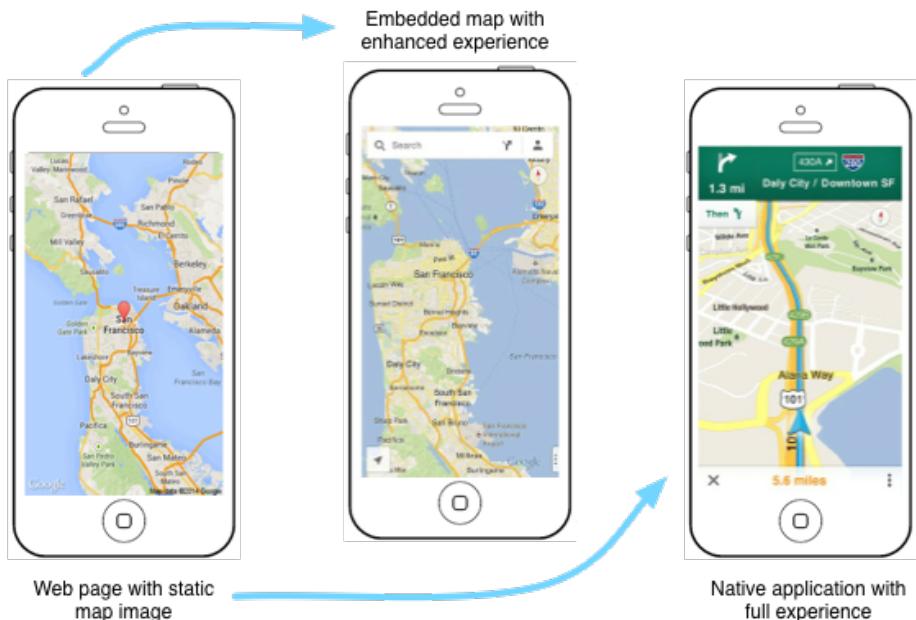
---

<sup>6</sup>Modernizr <http://modernizr.com/> je JavaScriptová knižnica, ktorá detekuje dostupnosť natívnej implementácie nových technológií v prehliadači.

aplikácií. Nevýhodou je aj nemožnosť posúvania webovej stránky pokial je element s mapou väčší ako je rozlíšenie displeja mobilného zariadenia, lebo eventy sú zachytávané mapou a posúva sa tá.

Používateľ nemusí chcieť okamžite interagovať s mapou a v takomto prípade ho zbytočne zatažuje. Výhodnejšie je tak zobraziť len náhľad mapy, ktorý sa načíta rýchlejšie pretože šetrí prenášané dátá, a pokial sa používateľ chce dozvedieť viac, tak len jednoducho na mapu klikne. V takomto prípade sa mu automaticky načíta plne interaktívna mapa. Na zobrazenie náhľadu mapy existuje API v službe google maps<sup>7</sup>, takže je možné využiť priamo to. Nevýhodou je, že počet požiadaviek za deň, ktoré sú zadarmo, je obmedzený, po získaní API klúča je možné ich spraviť 25000, ďalšie sú spoplatňované.

Rozšírením na mobilných zariadeniach iOS je možnosť otvorenia mapy priamo v natívnej aplikácii, ktorá prináša ešte väčší používateľský zážitok ako zobrazenie na webe. To je uskutočňované pomocou url schémy. V starších verziách iOS sa nachádzala natívna aplikácia na Google API a bola vyvolaná otvorením odkazu <http://maps.google.com/>, kde bolo možné zadávať parametre zobrazenia.



Obr. 18: Podmienené otváranie máp

S príchodom iOS 6 bola aplikácia nahradená vlastnom Apple aplikáciou, na ktorej spustenie sa zmenila aj url schéma a pôvodná otvorí mapu len v prehliadači. Výhodou je, že na starších, respektíve nepodporovaných zariadeniach nastáva presmerovanie na stránky Google a tak na rovnakom odkaze funguje spúšťanie aj pôvodnej aplikácie. Nová url schéma má tvar <http://maps.apple.com/>.

<sup>7</sup><https://developers.google.com/maps/documentation/staticmaps/>

### 3.2.3 Lightboxy

Lightbox je technika umožňujúca zobrazovať webový obsah v modálnych oknách nachádzajúcich sa nad úrovňou pôvodnej stránky.

Hlavným problémom použitia „lightboxov“ na mobilných zariadeniach je nesprávne zobrazovanie stránok pokiaľ nové okno je väčšie ako displej. V takomto prípade by bolo vhodnejšie používateľa presmerovať priamo na novú stránku.

### 3.2.4 Otvorenie v aplikácii / Stiahnutie aplikácie

V súčasnosti sme opklopaný množstvom informačných zdrojov, ktoré sa zväčša nachádzajú na internete dostupné na konkrétnej webovej adrese. Na tieto zdroje existuje množstvo odkazov z rôznych webových stránok, sociálnych sietí či mobilných aplikácií, ktoré zobrazia ich obsah v prehliadači.

„Links don't open apps.“ Jason Grigsby [19]

Webové odkazy neotvárajú natívne aplikácie, čo je technicky pravda, no realita je trochu odlišná. Prepojenie webovej časti aplikácie s natívnou umožňuje otvárať komplexnejšie úlohy, na ktoré je vo webe nedostatočny výkon, priamo v natívnom kóde. To umožní plynulejší chod aplikácie a lepší používateľský zážitok.

Prepájanie aplikácií sa uskutočňuje pomocou „custom url schémy“ [23], ktorá je špecifická pre jednotlivé operačné systémy zariadení. Nevýhodou vlastných odkazov je, že pokiaľ používateľ nemá nainštalovanú aplikáciu, tak po kliknutí sa objaví celkom nepekná chybová hláška.



Obr. 19: Chybová hláška po otvorení custom URL schémy, pokiaľ aplikácia neexistuje [38].  
Prevzaté z <http://www.lukew.com/ff/entry.asp?1654>

Lepším riešením je detekovanie, či si používateľ stiahol natívnu aplikáciu a odkaz na otvorenie v nej vytvorí až potom, čo je overenie úspešné. Pokiaľ by nebolo, tak by sa mohlo zobraziť tlačítko na stiahnutie natívnej aplikácie, ktoré opäť musí byť prispôsobené platforme na ktorej na stránku prispôsobujeme, pokiaľ nechceme používateľa zahľatíť všetkými platformami ktoré podporujeme.

Samotný proces detekcie či má používateľ nainštalovanú našu aplikáciu vôbec nie je triviálny, pretože na webe neexistuje žiadne dostupné API, ktoré by zahrňovalo všetky platormy. Apple sice vydal možnosť otvorenia aplikácie pomocou „Smart Banners”, ale až v iOS 6 a tak táto možnosť nie je úplne použiteľná.



Obr. 20: Otvorenie natívnej aplikácie v iOS 6 [22].

Prevzaté z <http://developer.apple.com/>

Naďve „smart banner” je priamo definovaný v html meta tagu aplikácie. Na stránke existuje len jeden ktorý volá url schému aplikácie a aj ten má prednastevený vzhľad v podobne okna v hornej časti obrazovky. Keby chceme vlastnú natívnu aplikáciu zavolať z viacerých častí webovej, prípadné volať viacero natívnych aplikácií, tak toto riešenie je nepostačujúce. Nemôže byť upravané vlastným potrebám.

Jediné možné riešenie je len v spolupráci s natívou aplikáciou, aj tak sa však musí vyvolať otvorenie stránky v prehliadači, kde sa dá už do cookies alebo lokálneho úložiska programátorksy zapísat existencia aplikácie. Vytvorenie samotného „webView” komponentu s webovou stránkou vrámcí aplikácie a následné zatvorenie nestačí. Prehliadač má oddelené úložiská stránok pre rôzne typy prístupov ako sú s internetový prehliadač, aplikácia a v iOS aj pre otvorenie internetovaj stránky z plochy. Možnosť ako tento proces zamaskovať je skrytá v procese registrácie, keď po otvorení aplikácie a zaregistrovaní pošleme používateľovi e-mail s potvrdzujúcim odkazom, ktorý otvorí webovú stránku v prehliadači.

### 3.3 Nástroje

#### 3.3.1 Detekcia

Pre potreby rozpoznávania platformy v aplikácii bol vytvorený modul detektie podľa jedinečného „user agent” textu. Modul umožňuje detektovať iOS a Android zariadenia. Rozpoznávanie prebieha na základe regulárneho výrazu, kde pri metóde detektie iOS zariadenia sa kontroluje výsky slov „iPhone”, „iPad” alebo „iPod”, ktoré jedinečne definujú platformu. Pri Android zariadeniach je detektia jednoduchšia, pretože stačí ak sa v identifikačnom teste nachádza slovo „Android”. Príklad metódy detektie má nasledujúci tvar:

---

```
1 var isAndroid = function(userAgent){  
2     return (/Android/gi).test(userAgent);  
3 }
```

---

Pri konfigurácii webovej aplikácie je možné aj aktálny „user agent” text nahradieť vlastnou hodnotou, takže prehliadač sa vždy môže správať ako požadované zariadenie. Problém pri detekcii môže nastať, pokiaľ by user agent už bol raz nahradený používateľom, respektíve by sa prestali používať súčasné identifikátory. Potom by sa detekovala nesprávna platforma.

## 4 Overenie

Detekovali sme niekoľko vstupných metód, webových komponentov a overovali ktoré z nich poskytujú lepší používateľský zážitok pre návštevníkov webových stránok ako aj pre samotných webových vývojárov. Na obe adaptívne vstupné metódy aj webové komponenty boli vytvorené série experimentov. Na preukázanie užitočnosti navrhnutého konceptu sme navrhli a implementovali niekoľko opakovane použiteľných modulov a komponentov. Overenie jednotlivých častí projektu je riešené vytvorením prototypových aplikácií ku každému vytvorenému modulu a pomocou získanej späťnej väzby z komunity. Všetky súčasti boli vytvorené pod open source licenciou a sú dostupné na GitHube. Implementované sú nasledujúce súčasti systému:

- **angular-adaptive/adaptive-speech**  
modul na ovládanie aplikácie pomocou reči
- **angular-adaptive/adaptive-scroll**  
modul na ovládanie aplikácie pomocou gyroscopu
- **angular-adaptive/adaptive-motion**  
modul na ovládanie aplikácie pomocou pohybu
- **angular-adaptive/adaptive-detection**  
modul na detektovanie zariadenia
- **angular-adaptive/adaptive-googlemaps**  
webový komponent google máp
- **angular-adaptive/adaptive-youtube**  
webový komponent youtube videí
- **Gyrocopter**  
rozšírenie do Chrome Developer Tools na emulovanie gyroskopu
- **speech-synthesis**  
polyfill rečovej syntézy

Pre vytvorené webové komponenty sme sledovali množstvo ušetrených dát, merali zrýchlenia načítavania webových stránok či počty vykonaných dopytorov na server pre rôzne druhy internetových pripojení. Pre adaptívne vstupné metódy sme zisťovali najmä záujem o jednotlivé druhy alternatívnych ovládaní webových stránok. Z komunity sme získali množstvo späťnej väzby, ktorá nám taktiež pomohla overiť navrhnutý koncept a ďalej prispôsobiť riešenia. Podľa analytických nástrojov sa experimentov zúčastnili desaťtisíce používateľov a vývojárov.

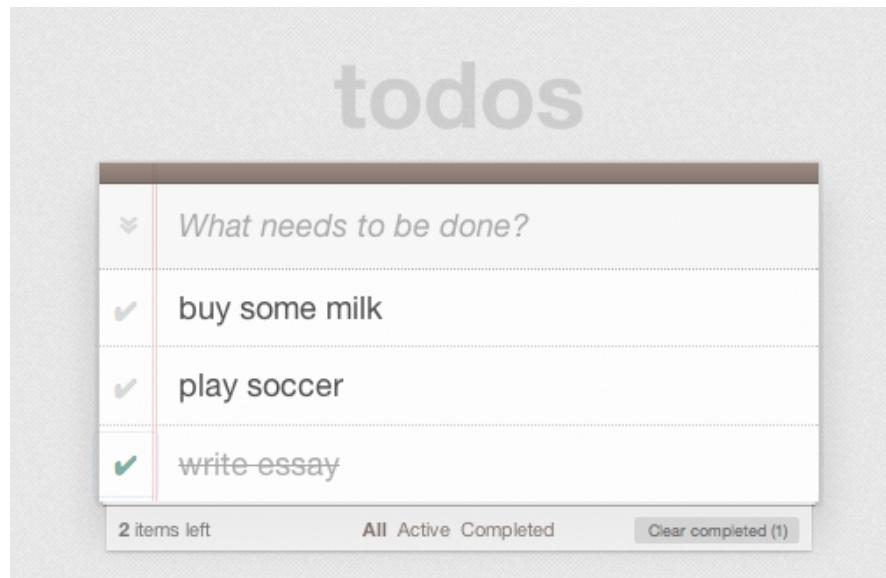
## 4.1 Adaptívne vstupné metódy

Adaptívne vstupné metódy poskytujú alternatívne spôsoby ovládania webových aplikácií. Experimentovali sme ovládaním webových aplikácií pomocou rečových povelov, pohybu detekovaného pomocou gyroskopu a video kamery.

### 4.1.1 Ovládanie pomocou reči

Ovládanie pomocou reči bolo demonštrované vytvorením modulu adaptive-speech<sup>8</sup> a prototypovej aplikácie nad existujúcim systémom TodoMVC<sup>9</sup>, ktorý demonštruje vytvorenie todo aplikácie v rôznych javascriptových frameworkoch.

Vytvorený experiment založený na našom module poskytoval ovládanie aplikácie todo listu. Po spustení aplikácie a potvrdení prístupových povolení od používateľa k používaniu mikrofónu nie je potrebné ďalšia interakcia okrem zadávania rečových príkazov, nakoľko prebieha kontinuálne rozpoznávanie reči. Celkovo sa v ňom rozpoznávalo 7 presných rečových povelov a 9 rečových povelov založených na použití regulárnych výrazov. V aplikácii je tak umožnené pridávanie novej úlohy, jej označenie za splnenú alebo úplne odstránenie zo zoznamu. Taktiež je umožnené prechádzanie medzi jej jednotlivými časťami a prepínaním zobrazenia všetkých úloh, aktívnych úloh a splnených úloh. Rovnako je umožnené pomocou rečového príkazu aj vymazať všetky splnené úlohy. Vytvorená demo aplikácia je znázornená na nasledujúcom obrázku:



Obr. 21: Prototypová aplikácia vytvoreného modulu adaptive-speech

K samotnému modulu som získal veľké množstvo spätej väzby a je v komunite najpo-

<sup>8</sup><https://github.com/angular-adaptive/adaptive-speech>

<sup>9</sup><http://todomvc.com/>

pulárnejší spomedzi ostatných alternatívnych metód ovládania aplikácie. Samotného experimentu sa zúčastnili tisíce používateľov. Pomocou hlasového ovládania sa dá v aplikácii dosiahnuť množstvo vecí a pomocou neho vieme ovládať kompletnú aplikáciu bez použitia ostatných vstupných metód. Dokonca vieme aplikáciu ovládať vo viacerých jazykoch. Najväčším problémom v súčasnosti je však nie úplne správne rozpoznanie hovorených príkazov. Na jeho zmiernenie ale vieme použiť regulárne výrazy alebo nápravu výrazov v slovníku od používateľa.

Podpora ovládania pomocou reči v prehliadačoch v súčasnosti ešte nie je veľmi rozšírená a vyzerá nasledujúco:

IE	Firefox	Chrome	Safari	Opera	iOS Safari	Android	Chrome for Android
nie	nie	25.0	nie	nie	nie	nie	31.0

Tabuľka 1: Podpora rozpoznávania reči

S použitím rozpoznania reči súvisí aj rečová syntáza, vďaka ktorej vieme interagovať pomocou reči a ešte viac zvýšiť používateľský zážitok. Nativná podpora však tiež nemá širokú podporu a pre tieto potreby bol vytvorený polyfill vďaka ktorému vieme rečovú syntézu nahradit pomocou Audio elementu a prehravaním audia zo služby Google Translate. Porovnanie podpory v prehliadačoch vyzerá nasledujúco:

IE	Firefox	Chrome	Safari	Opera	iOS Safari	Android	Chrome for Android
nie	nie	33.0	7.0	nie	7.0	nie	33.0

Tabuľka 2: Natívna podpora rečovej syntézy

IE	Firefox	Chrome	Safari	Opera	iOS Safari	Android	Chrome for Android
9	áno	áno	áno	áno	4.0	2.3	áno

Tabuľka 3: Podpora polyfillu rečovej syntézy

#### 4.1.2 Ovládanie pomocou gyroskopu

Ovládanie pomocou gyroskopu bolo demonštrované vytvorením modulu adaptive-scroll<sup>10</sup> a prototypovej aplikácie. V aplikácii je možné po spustení skrolovať pomocou natáčania zariadenia do strán. Skrolovanie prebieha na základe veľkosti uhlu natočenia zariadenia oproti počiatočnej polohe, čím je rozdiel väčší, tým sa vo webovej aplikácii skroluje rýchlejšie. Na skrolovanie v aplikácii nie je potrebná ďalšia interakcia používateľa, napríklad pomocou dotyku.

<sup>10</sup><https://github.com/angular-adaptive/adaptive-scroll>

Experimentu sa zúčasnili tisíce používateľov a je podporovaný vo veľkom počte prehliadačoch, konkrétnie v nasledujúcich:

IE	Firefox	Chrome	Safari	Opera	iOS Safari	Android	Chrome for Android
11	áno	áno	nie	áno	4.2	3.0	áno

Tabuľka 4: Podpora získavania orientácie v prehliadačoch

Pomocou modulu vieme zvýšiť používateľský zážitok, nevieme však ovládať kompletnejšiu webovú aplikáciu ako v prípade rečových povelov. Po vytvorení modulu a testovaní na zariadeniach sa navyše zistili problémy v skutočnej implementácii udalostí natáčania zariadenia v rôznych prehliadačoch. W3C špecifikácia orientácie zariadenia uvádza nasledovné:

- os X je rovnobežná na rovinu zeme, kladné hodnoty má v smere na východ, záporné na západ
- os Y je rovnobežná na rovinu zeme, kladné hodnoty má v smere na sever, záporné na juh
- os Z je kolmá na rovinu zeme, kladné hodnoty má v smere od zeme, záporné smerom k zemi

Rotácia by mala byť popísaná pravidlom pravej ruky, teda kladné hodnoty rotácie sú v smere hodinových ručičiek okolo osí.

Nasledujúce tabuľky ukazujú, kde sa nachádzajú nulové body pre jednotlivé osi, aké hodnoty nadobúda rozsah rotácie a či je dodržané pravidlo pravej ruky.

Pravidlo pravej ruky (PPR) hovorí:

„Kladné hodnoty rotácie sa zvyšujú pri rotovaní okolo osi v smere hodinových ručičiek pri ukazovaní smerom na narastajúcu kladnú hodnotu smerovej osi.” W3C [30]

Táto definícia je však protichodná s hodnotami kompasu. Pri rotácii okolo osi Z v smere rotácie kompasu sa hodnoty rotácie znižujú a nie narastajú. To je však spôsobené pohľadom do záporných hodnôt osi Z.

## Alpha

Rotácia okolo osi Z nadobúda hodnoty rotácie alpha. V špecifikácii je nejasne definové aké by mali byť východiskové hodnoty, čo vo výsledku prináša rôzne implementácie rotácie v prehliadačoch. To spôsobuje zmätok nakoľko zo špecifikácie nie je jasné, na akú stranu by mal ukazovať nulový bod. Dá sa odpozorovať podľa pravidla pravej ruky, že 0 stupňov ukazuje smerom na sever, pretože implementovaných 90 stupňov ukazuje na východ.

Rozsah hodnôt bol testovaný pri držaní zariadenia v horizontálnej polohe a vycentrovaní smerom k nulovému bodu, keď hodnota rotácie bola 360 stupňov. Pozorované boli nasledujúce hodnoty:

	Nulový bod	PPR	Rozsah
Reference	Sever (0)	A	[0, 360]
iOS Chome	Východ (90)	A	[0, 360]
iOS Safari	Východ (90)	A	[0, 360]
Android			
Chrome	Sever (0)	A	[0, 360]
Stock	Západ (270)	A	[0, 360]
Firefox	Sever (0)	N	[0, 360]

Tabuľka 5: Alpha rotácia gyroskopu

### Beta

Rotácia okolo osi X nadobúda hodnoty rotácie beta. Špecifikácia definuje nulový bod ako horizontálnu polohu. Všetky testované prehliadače majú implementovaný smer osi rotácie správne, problémy však nastávajú pri rozsahu rotačných hodnôt.

Rozsah bol testovaný pri držaní zariadenia v horizontálnej polohe, čo je nulový bod. Zariadenie bolo potom otáčané okolo osi X o 90 stupňov tak, že displej zariadenia smeroval k pozorovateľovi. Potom nasledovalo otáčanie o ďalších 90 stupňov tak, že displej bol smerom k zemi. Na koniec nasledovala rotácia zvyšných 180 stupňov smerom do počiatočnej polohy, čím sa rotácia dokončila.

	Nulový bod	PPR	Rozsah	Poznámky
Reference	Horizontálne	A	[0, -180 180]	
iOS Chome	Horizontálne	A	[-90, 90]	Plný rozsah rotácie nie je podporovaný.
iOS Safari	Horizontálne	A	[-90, 90]	Plný rozsah rotácie nie je podporovaný.
Android				
Chrome	Horizontálne	A	[-90, 90]	Plný rozsah rotácie nie je podporovaný.
Stock	Horizontálne	A	[-90, 90]	Plný rozsah rotácie nie je podporovaný.
Firefox	Horizontálne	N	[0, 180 -180]	Rotácie pokračuje naspäť na začiatok

Tabuľka 6: Beta rotácia gyroskopu

V iOS zariadeniach rovako ako v Android prehliadači a Chrome pre Android hodnoty rotácie od počiatku stúpajú správne. Ale keď rotácia nadobudne svoje maximum, čo je 90 a -90 stupňov, tak jej hodnoty začnú postupne klesať. Nie je tak podporovaný celý rozsah rotácie.

Vo Firefoxe pre Android je rotácia implementovaná opačným smerom a nie je dodržané špecifikované pravidlo pravej ruky. Je však podporovaný celý rozsah rotácie. Najskôr klesne k hodnote -180 stupňov, čo sa pri pokračujúcej rotácii zmení na kladnú hodnotu a potom klesá k nule.

## Gamma

Rotácia okolo osi Y nadobúda hodnoty rotácie gamma. Špecifikácia definuje nulový bod v horizontálnej polohe. Ale aj v tomto smere rotácie sú problémy s rozsahom rotácie, ale to je spôsobené najmä samotnou špecifikáciou zo strany W3C, napäť v nej sa nachádza rozsah len od -90 po 90 stupňov, čo je celkovo 180 a nie je tak pokryté celé otočenie.

Rozsah hodnôt bol testovaný pri držaní zariadenia v horizontálnej polohe a natočením na nulový bod. Zariadenie bolo potom otáčané okolo osi Y o 90 stupňov v smere hodinových ručičiek tak, že jeho displej smeroval vpravo. Potom nasledovalo otočenie o ďalších 90 stupňov tak, že zariadenie smerovalo dolu. Na koniec sa dokočilo zostávajúcich 180 stupňov rotácie do pôvodného stavu.

	Nulový bod	PPR	Rozsah	Poznámky
Reference	Horizontálne	A	[0, 90]-90]	Zlá definícia od W3C
iOS Chome	Horizontálne	A	[0, 180]-180]	Plný rozsah rotácie nie je podporovaný
iOS Safari	Horizontálne	A	[0, 180]-180]	Plný rozsah rotácie nie je podporovaný.
Android				
Chrome	Horizontálne	A	[0, 270]-90]	Upravený rozsah rotácie definuje natočenie
Stock	Horizontálne	A	[0, 270]-90]	Upravený rozsah rotácie definuje natočenie
Firefox	Horizontálne	N	[0, -90]90]	Rozsah rotácie pokračuje naspäť na začiatok

Tabuľka 7: Gamma rotácia gyroskopu

Zlá definícia rotácie od W3C špecifikuje rotáciu len po hodnote 90 stupňov z horizontálnej polohy, čo spôsobuje problémy pri väčšom natočení zariadenia.

V iOS rotácia nadobúda nulový bod v horizontálnej polohe s displejom smerujúcim hore. Otočením zariadenia okolo osi Y tak, že obrazovka bude smerovať smerom dole bude aktuálne gamma natočenie nadobúdať hodnotu 180, respektívne -180 stupňov. Ak sa rotácia uskutočňuje v smere hodinových ručičiek, tak hodnoty rotácie sa postupne zvyšujú v kladných číslach. Ak rotácia ide opačným smerom proti smeru hodinových ručičiek, tak hodnoty rotácie sa zmenšujú. Pri natočení zariadenia smerom na bok lavou stranou hore je hodnota rotácia 90 stupňov, na opačnej strane pri smerovaní pravej hrany smerom hore je to -90 stupňov.

Chrome pre Android a vstavaný Android prehliadač majú spravné hodnoty rotácie od -90 po +90 stupňov. Ale po natočení zariadenia o viac ako 90 stupňov v smere hodinových ručičiek hodnota rotácie naďalej stúpa až po hodnotu 270 stupňov, čo je zároveň aj hodnota

-90. Toto poskytuje možnosť presne vedieť veľkosť natočenia zariadenia.

Firefox má aj pri rotácii okolo osi Y podobný problém, a to je že hodnoty sú opačné a nedodržiava špecifikované pravidlo pravej ruky. Rozsah je správny, ale rotácie v smere hodinových ručičiek nadobúdajú záporné hodnoty a naopak.

### Gyrocopter

Na vyriešenie problémov pri vývoji webových aplikácií spôsobených rôznymi implementáciami rotácií v prehliadačoch som vytvoril rozšírenie Gyrocopter pre prehliadač Chrome<sup>11</sup>. Hlavným cieľom rozšírenia je umožniť zvoliť si simuláciu konkrétneho prehliadača a dynamicky meniť rotáciu zariadenia bez potreby testovania funkcionality na reálnom zariadení. Podobná funkcionalita sa už nachádza v developerských nástrojoch prehliadača Chrome, ale tam je možné len manuálne zadať hodnoty rotácie bez znalosti skutočných rozsahov a náhľadu natočenia zariadenia.

Toto rozšírenie umožňuje simulať udalosti ktoré nastanú pri rotácii zariadenia, pričom používateľ môže natáčať zariadenie vo všetkých troch osiach. Rozšírenie je implementované pre vývojárov priamo do prostredia „Chrome Developer Tools“.

Nakoľko hodnoty rotácií sú implementované v prehliadačoch rôzne, tak používateľ má na výber zvoliť si podľa akej implementácie chce vyvolávať udalosti rotácie. Simulovanými prehliadačmi na výber sú:

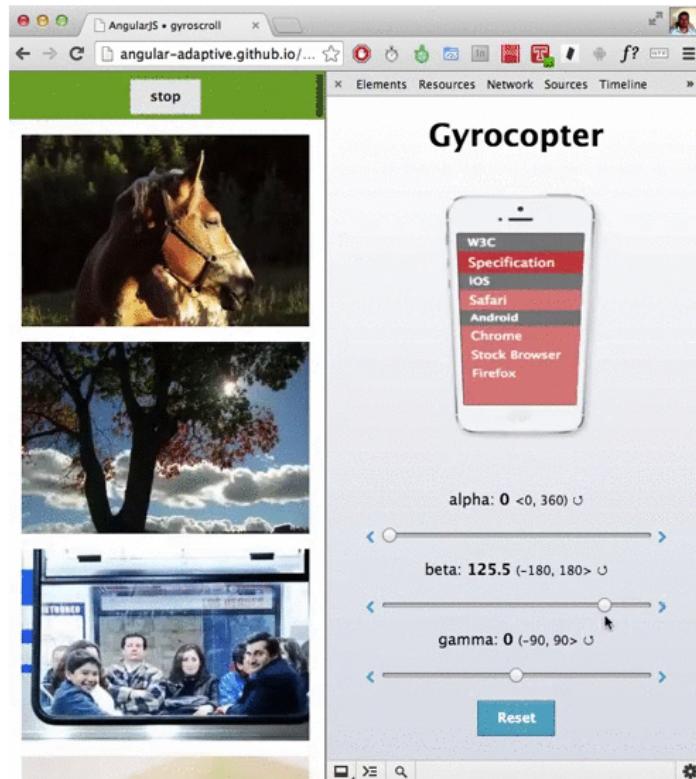
- W3C špecifikácia
- iOS Safari (Chrome používa web view)
- Android Chrome
- Android prehliadač
- Firefox pre Android

Samotné používateľské rozhranie rozšírenia obsahuje náhľad aktuálnej rotácie zariadenia na ktorom je možné zmeniť aktuálne simulovaný prehliadač. Pod ním sa nachádzajú uhly natočenia alpha, beta a gamma, ktoré je možné meniť. Tieto obsahujú pre jednotlivé prehliadače rozsahy a smer natočenia. Pri zmene rotácie na jednotlivých osiach sa zmena automaticky prejaví aj na náhľade zariadenia.

---

<sup>11</sup>Dostupné na stiahnutie z Chrome Web Store <https://chrome.google.com/webstore/detail/gyrocopter/oooalfgemajfclliinfcdkifafmcfjop>

Vytvorená prototypová aplikácia na skrolovanie spolu s rozšírením Gyrocopter sú znázornené na nasledujúcim obrázku:



Obr. 22: Prototypová aplikácia vytvoreného modulu adaptive-scroll a rozšírenie Gyrocopter

Po výbere, respektíve zmene prehliadača, ktorý sa má simulaovať, sa automaticky prepočítajú rozsahy pre jednotlivé uhly otáčania. Zároveň sa upravia na nich aj aktuálne hodnoty natočenia bez toho, aby sa musela znázornená rotácia zariadenia meniť.

#### 4.1.3 Ovládanie pomocou webkamery

Ovládanie pomocou pohybu bolo demonštrované vytvorením modulu adaptive-motion<sup>12</sup> a prototypovej aplikácie. V nej je umožnené sledovanie používateľa a možnosť výberu jeho vizualizácie, či už klasický obraz alebo upravený zobrazením len detekovaním jeho kože alebo detekciou hrán. V aplikácii prebieha kontinuálne rozpoznávanie používateľa a postupne sa vyplisujú rozpoznané gestá, pričom posledné rozpoznané gesto sa zakaždým zvýrazní. Vytvorená prototypová aplikácia je znázornená na nasledujúcim obrázku:

<sup>12</sup><https://github.com/angular-adaptive/adaptive-motion>



Visualization:

video  skin  edge  none



Recent gestures:

Swipe Down

Swipe Right

Swipe Left

Swipe Down

Swipe Right

Obr. 23: Prototypová aplikácia vytvoreného modulu adaptive-motion

Experimentu sa opäť zúčasnili tisíce používateľov a vytvorený modul je podporovaný v týchto konkrétnych prehliadačoch:

IE	Firefox	Chrome	Safari	Opera	iOS Safari	Android	Chrome for Android
nie	17	21	nie	18	nie	nie	33

Tabuľka 8: Podpora prístupu ku kamere v prehliadačoch

Použitím modulu vieme zvýšiť používateľský zážitok, nedokážeme však len pomocou neho ovládať celú webovú aplikáciu. Preto sa dá použiť len ako doplnková metóda interakcie čomu nasvädčuje aj získaný ohlas z komunity. Navyše vo svetlých a tmavých scénach môžu nastáť problémy pri detekcii.

## 4.2 Adaptívne webové komponenty

Pri adaptívnych webových komponentoch sme sledovali rýchlosť načítavania stránok, počty uskutočnených dopytov na server, množstvo prenášaných dát a rýchlosť zostovenia DOM stromu. Rýchlosť sú porovnávali s použitím vytvorených modulov a bez použitia pri vývoji klasickým spôsobom. Testovanie prebiehalo v rôznych podmienkach s použitím internetových pripojení s nasledujúcimi konfiguráciami:

Pripojenie	Dátový tok	Latencia	Stratené pakety
2G	240 kbps	400 ms	5 %
3G	780 kbps	100 ms	2 %
4G	20 Mbps	50 ms	1 %
Broadband	100 Mbps	5 ms	0 %

Tabuľka 9: Konfigurácie internetových pripojení

Emulácia internetového pripojenia prebiehala pomocou programu „Network Lint Conditioner“ a ďalej sa uvádzajú priemerné namerané hodnoty z 10 experimentov spolu so smerodajnými odchýlkami pre jednotlivé druhy internetového pripojenia.

### 4.2.1 Modul youtube videa

Webový komponent youtube videa bol demonštrovaný vytvorením modulu adaptive-youtube<sup>13</sup> a demo aplikácie. V aplikácii je umožnené meniť automatické načítavanie videa. Pokiaľ sa pristupuje z mobilného zariadenia a pomalého pripojenia na internet, tak sa zobrazí len náhľad videa v podobe obrázku. Až po kliknutí na video sa dotiahne samotné video a začne sa prehrávať, respektíve video sa otvorí v natívnej mobilnej aplikácii.

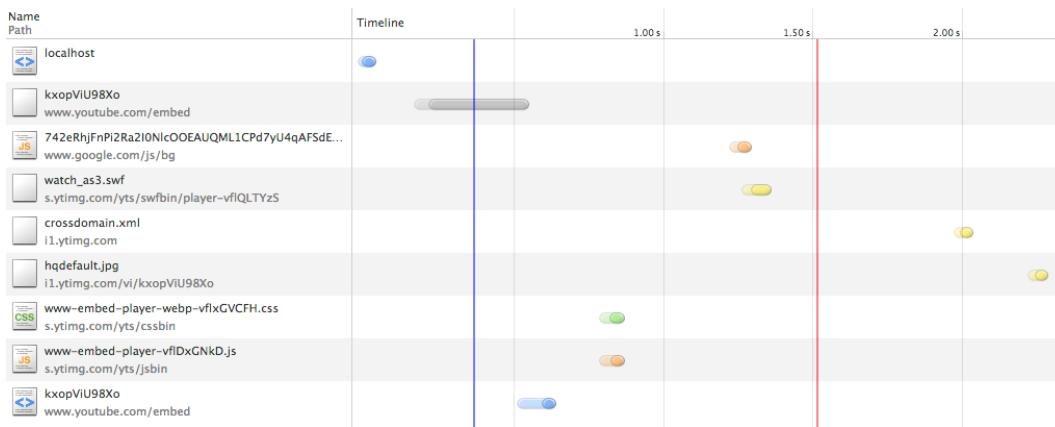
Použitím vytvoreného modulu vieme ušetriť mnoho dát a zároveň aj znížiť množstvo dopytov na servery. Porovnanie priemerného množstva spotrebovaných dát a vytvorených dopytov pre jedno youtube video je zobrazené v nasledujúcej tabuľke:

	Embed video	Vytvorený modul
Počet požiadaviek na server	9	6
Množstvo prenesených dát	1517 KB	389 KB

Tabuľka 10: Množstvo požiadaviek a prenesených dát pre youtube video

Z toho je zrejme, že použitím vytvoreného modulu sa zrýchli aj načítavanie webovej stránky. Porovnanie načítavania stránky s použitím klasického prístupu vloženia videa a použitím nášho modulu spolu s jednotlivými uskutočnenými požiadavkami sa nachádza na nasledujúcich obrázkoch:

<sup>13</sup><https://github.com/angular-adaptive/adaptive-youtube>

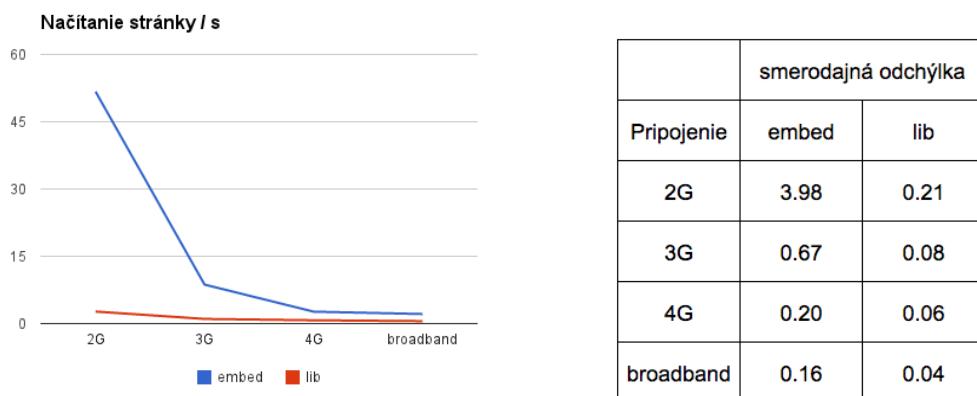


Obr. 24: Časová os načítavania stránky s použitím youtube embed videa



Obr. 25: Časová os načítavania stránky s použitím youtube modulu

Časové osy načítania stránky sú znázornené pre broadband pripojenie. Ako sa prejaví zmena rýchlosťi na iných druhoch najmä mobilných pripojení je znázornené na nasledujúcim grafe:

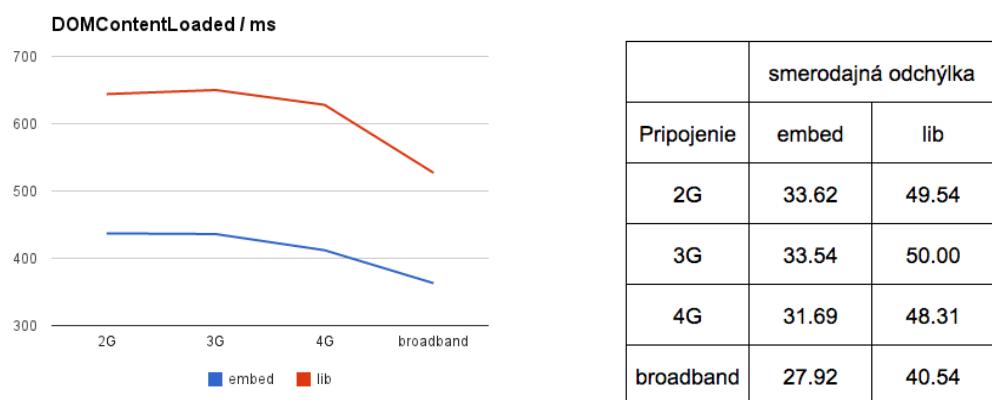


Obr. 26: Porovnanie rýchlosťi plného načítania stránky s youtube komponentom

Rozdiel v rýchlosti načítania kompletnej stránky aj s externými zdrojmi je veľmi výrazný

a pri pomalšom pripojení narastá. Použitím nášho modulu vieme zrýchliť načítanie stránky o sekundy a rozdiel sa prejaví aj pri rýchlejších pripojeniach. Kedže v súčasnosti je na mobilných platformách rozšírené 3G pripojenie, tak sa podstatne zvýši aj používateľský zážitok, keďže tým ubúda podstatná doba čakania a zároveň používateľom ušetríme aj dátu z predpladtných internetových balíkov.

Použitím nášho modulu sa však zvýši aj čas zostavenia DOM stromu stránky. Nasledujúci graf nie je prekvapením keďže oproti vloženiu videa do stránky potrebujeme ešte načítať a spúštať naše skripty. Napriek tejto počiatočnej nevýhode však umožníme výrazné zrýchlenie načítania stránky s youtube videom.



Obr. 27: Porovnanie rýchlosťi zostavenia dom stromu s youtube komponentom

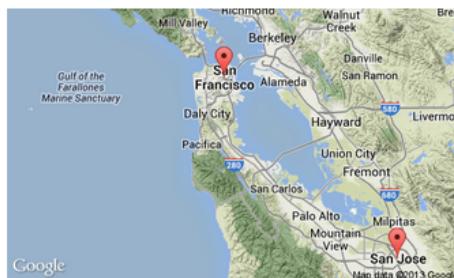
#### 4.2.2 Modul google maps

Webový komponent google máp bol demonštrovaný vytvorením modulu adaptive-googlemaps<sup>14</sup> a demo aplikácie. V aplikácii je umožnené meniť zvýraznené body, centrovanie, priblíženie a zobrazený terén a automaticky sa prispôsobí používateľovmu zariadeniu. Pokial sa pristupuje z mobilného zariadenia a pomalého pripojenia na internet, tak sa zobrazí len náhľad mapy v podobe obrázku. Až po kliknutí na mapu sa dotiahne interaktívna mapa, respektíve mapa sa otvorí v natívnej mobilnej aplikácii. Vytvorená demo aplikácia je znázornená na nasledujúcim obrázku:

<sup>14</sup><https://github.com/angular-adaptive/adaptive-googlemaps>

```
mapevents: { "redirect": false, "loadmap": true }
```

Loads dynamic google map.



```
{ "redirect": true, "loadmap": false }
```

Opens google map in a new tab / a native mobile application.



```
mapevents: no mapevents
```

Nothing happens.



Obr. 28: Prototypová aplikácia vytvoreného modulu adaptive-googlemaps

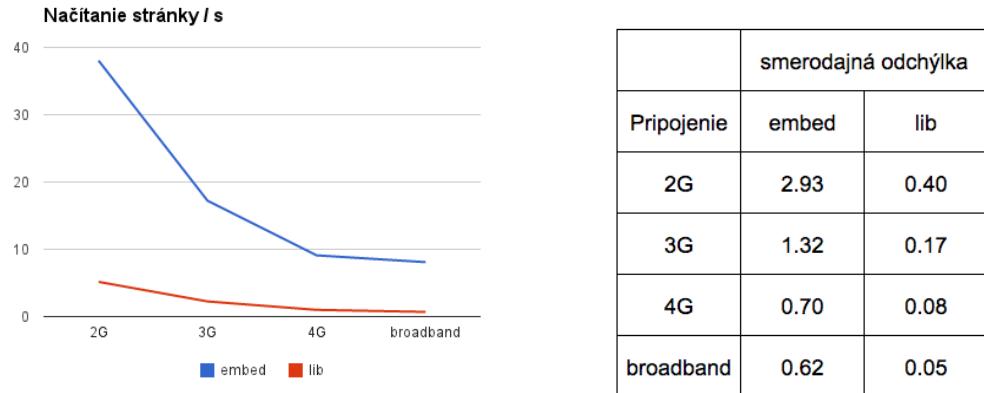
Použitím vytvoreného modulu vieme opäť ušetriť mnoho dát a znížiť množstvo dopytov na servery. Porovnanie priemerného množstva spotrebovaných dát a vytvorených dopytov pre jeden element google mapy je zobrazené v nasledujúcej tabuľke:

	Embed mapa	Vytvorený modul
Počet požiadaviek na server	68	6
Množstvo prenesených dát	5540 KB	476 KB

Tabuľka 11: Množstvo požiadaviek a prenesených dát pre google mapy

Z toho je zrejme, že použitím vytvoreného modulu sa opäť zrýchli aj načítavanie webovej stránky. Porovnanie načítavania stránky s použitím klasického prístupu vloženia mapy a

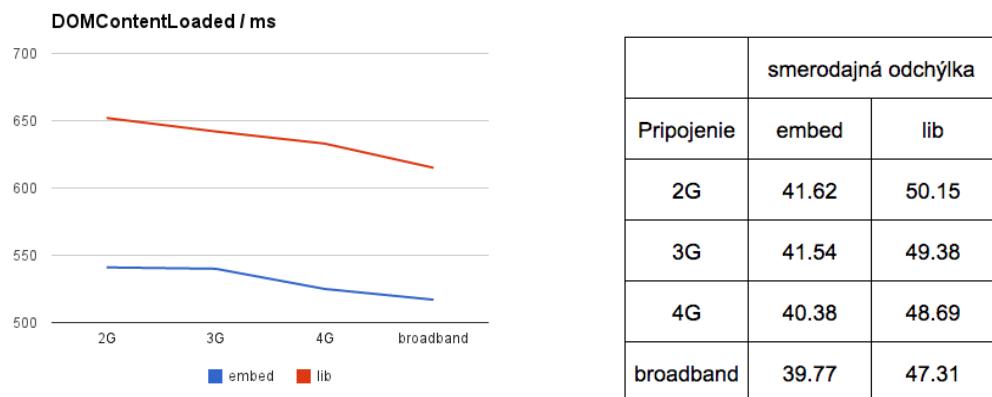
použitím nášho modulu a ako sa prejaví zmena rýchlosťi na iných druhoch najmä mobilných pripojení je znázornené na nasledujúcom grafe:



Obr. 29: Porovnanie rýchlosťi plného načítania stránky s google maps komponentom

Rozdiel v rýchlosti načítania kompletnej stránky aj s externými zdrojmi je v prípade google mapy ešte výraznejší ako pre youtube video. Použitím nášho modulu vieme zrýchliť načítanie stránky o sekundu a rozdiel sa prejaví aj pri rýchlejších pripojeniach. Najmä na mobilných pripojeniach sa tým podstatne zvýši aj používateľský zážitok, keďže tým ubúda podstatná doba čakania a opäť používateľom ušetríme aj dátu z predpladtných internetových balíkov.

Použitím nášho modulu sa opäť zvýši aj čas zostavenia DOM stromu stránky ako v prípade youtube videa, keďže oproti vloženiu mapy do stránky potrebujeme načítať a spúštať naše skripty. Napriek tejto počiatočnej nevýhode však umožníme výrazné zrýchlenie načítania stránky s google mapou.



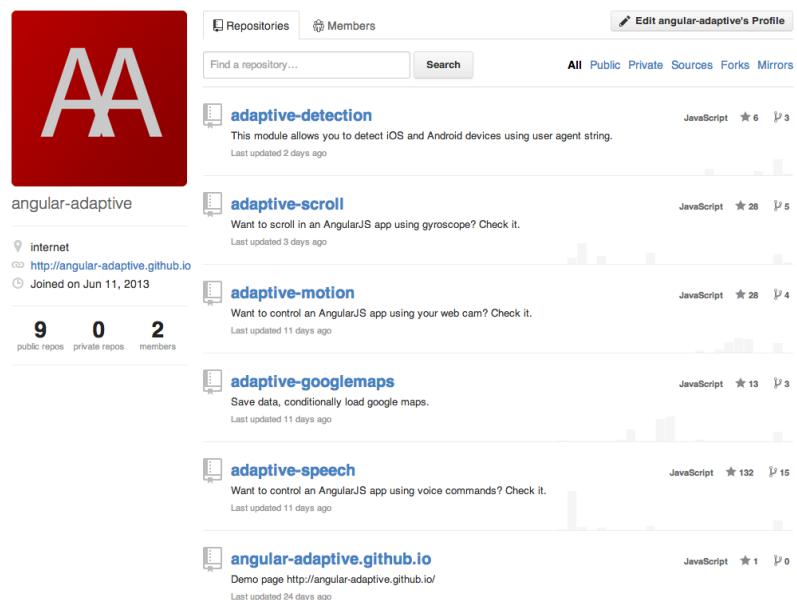
Obr. 30: Porovnanie rýchlosťi zostavenia dom stromu s google maps komponentom

## 5 Záver

V práci som navrhol a vytvoril možnosti adaptácie webových aplikácií, či už pomocou alternatívnych možností ovládania rečou, pohybom alebo gyroskopom, prispôsobovaním sa webových komponentov podmienkam používateľa alebo samotných nástrojov použitých pri vývoji adaptívnej webovej aplikácie.

Vytvorené nástroje sú umiestnené na serveri github pod open source licenciou, kde komunita o ne prejavila veľký záujem. Dostal som k nim množstvo pozitívnej späťnej väzby od kvalifikovaných ľudí, pull requestov na drobné úpravy funkcionality či dokumentácie a ďalej sa tak rozvíjajú aj za pomoci ostatných ľudí, za čo by som sa im chcel ešte raz podakovať. Vďaka tomu som mal možnosť naučiť sa množstvo nových vecí z tvorby open source aplikácií a následne nadobudné poznatky ďalej prezentovať ostatným vývojárom. Zároveň sa referencie na našu prácu objavili v množstve rôznych vydaní newsletterov, konferencií či iných prezentácií:

- **JavaScript Weekly #135**
- **AngularJS Weekly**
- **deSymfony 2013**, 20-22 jún 2013, Madrid
- **Lone StarPHP 2013**, 28-29 jún 2013, Dallas, TX
- **WebElement #25**, 5 december 2013, Bratislava



Obr. 31: Umiestnenie projektu na serveri github  
<https://github.com/angular-adaptive>

Overením ich používania sa zistilo, že alternatívne spôsoby ovládania sú celkom dobrý doplnok v súčasnosti ku klasickej interakcii so zariadeniami. Tak ako sme očakávali, najpopulárnejším alternatívnym spôsobom ovládania je ovládanie pomocou reči, nakoľko pomocou neho sa dá kontrolovať celá webová aplikácia len za pomoci rečových príkazov. Ovládania pomocou detakcie pohybu z gyroskopu a video kamery sa tiež presadili, neboli však až na takej úrovni ako ovládanie pomocou reči. Hlavným dôvodom sú výrazne limitované možnosti kde sa dá takéto ovládanie využiť a obe slúžia len ako doplnkový spôsob interakcie. Použitie alternatívneho ovládania je ale aj tak dobrou príležitosťou na zvýšenie používateľského zážitku.

Adaptívne webové komponenty sú dobrý spôsob ako vieme ušetriť množstvo prenášaných dát. Ich použitím sa zrýchliло načítavanie aplikácie a s tým aj používateľský zážitok. Avšak aj tu existujú limity, nakoľko množstvo webových stránok si potrebuje prispôsobiť webové komponenty vlastným požiadavkam. To preukázal aj nižší záujem z komunity oproti adaptívnym vstupným metódam. Sú však veľmi dôležitou súčasťou webu a verím, že v budúcnosti sa budú takéto spôsoby ešte viac presadzovať.

## Literatúra

- [1] ANTALA, J. Tvorba bohatých internetových aplikácií pre mobilné zariadenia. Bakalár-ska práca, FIIT STU, Bratislava, 2012.
- [2] AVGERINAKIS, K. – BRIASSOULI, A. – KOMPATSIARIS, I. Real Time Illumination Invariant Motion Change Detection. In *Proceedings of the First ACM International Workshop on Analysis and Retrieval of Tracked Events and Motion in Imagery Streams*, ARTEMIS '10, pp. 75–80, New York, NY, USA, 2010. ACM. doi: 10.1145/1877868.1877887. Dostupné z: <http://doi.acm.org/10.1145/1877868.1877887>. ISBN 978-1-4503-0163-3.
- [3] BARKHUUS, L. – POLICHAR, V. E. Empowerment through seamfulness: Smart phones in everyday life. In *Personal and Ubiquitous Computing* 15, pp. 629–639. Springer, 2011. ISSN 1617-4909.
- [4] BASSON, S. – FAIRWEATHER, P. G. – HANSON, V. L. Speech Recognition and Alternative Interfaces for Older Users. *interactions*. Júl 2007, 14, 4, pp. 26–29. ISSN 1072-5520. doi: 10.1145/1273961.1273980. Dostupné z: <http://doi.acm.org/10.1145/1273961.1273980>.
- [5] BIDELMAN, E. A More Awesome Web. In *Google IO 2013 conference*, Moscone Center, San Francisco, CA, USA, 2013.
- [6] BIDELMAN, E. Web Components: A Tectonic Shift for Web Development. In *Google IO 2013 conference*, Moscone Center, San Francisco, CA, USA, 2013.
- [7] CHUA, A. Y. K. – BALKUNJE, R. S. – GOH, D. H.-L. Fulfilling mobile information needs: a study on the use of mobile phones. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '11, pp. 92:1–92:7, New York, NY, USA, 2011. ACM. doi: 10.1145/1968613.1968721. Dostupné z: <http://doi.acm.org/10.1145/1968613.1968721>. ISBN 978-1-4503-0571-6.
- [8] CLARK, J. Designing for Touch. In *An Event Apart*, Seattle, WA, USA, April 1-3, 2013.
- [9] CLARK, J. Designing For Touch. In *The Mobile Book*, Freiburg, Germany, 2012. Smashing Magazine. ISBN 9783943075502.
- [10] CREMIN, R. Mobile web content adaptation techniques. Technical report, mobiForge, November 2, 2011. Dostupné z: <http://mobiforge.com/startling/story/mobile-web-content-adaptation-techniques>.
- [11] CREMIN, R. Performance is money, part 1: the end-user's wallet. Technical report, mobiForge, March 12, 2013. Dostupné z: <http://mobiforge.com/designing/blog/performance-money-part-1-end-users-wallet>.

- [12] FROST, B. Beyond Media Queries: Anatomy of an Adaptive Web Design. In *An Event Apart conference*, Washington DC, USA, August 6, 2012.
- [13] FROST, B. Beyond Squishy: The Principles of Adaptive Design. In *SXSW conference*, Austin, TX, USA, March 9, 2013.
- [14] FROST, B. Responsive Design Patterns. In *The Mobile Book*, Freiburg, Germany, 2012. Smashing Magazine. ISBN 9783943075502.
- [15] GRIGORIK, I. Breaking the 1000 ms Time to Glass Mobile Barrier. In *SF HTML5*, San Francisco, CA, USA, Mar 22, 2013.
- [16] GRIGORIK, I. Optimizing the Critical Rendering Path. In *Breaking Development*, San Diego, CA, USA, July 22-24, 2013.
- [17] GRIGORIK, I. High Performance Browser Networking. O'Reilly Media, Incorporated, 2013. ISBN 9781449344764.
- [18] GRIGSBY, J. Adaptive Input. In *Breaking Development*, San Diego, CA, USA, July 22-24, 2013.
- [19] GRIGSBY, J. Links Don't Open Apps. Technical report, Cloud Four, Portland, OR, USA, Mar 16, 2011. Dostupné z: <http://blog.cloudfour.com/links-do-not-open-apps/>.
- [20] GUSTAFSON, A. Building Adaptive Designs Now. In *User Interface 17 Conference*, Boston, MA, USA, November 5-7, 2012.
- [21] HAY, S. There is no Mobile Web, Jan 07, 2011. Dostupné z: <http://www.the-haystack.com/2011/01/07/there-is-no-mobile-web/>.
- [22] iOS Developer Library. Promoting Apps with Smart App Banners. Technical report, Safari Web Content Guide, . Dostupné z: <http://developer.apple.com/library/ios/#documentation/AppleApplications/Reference/SafariWebContent/PromotingAppswithAppBanners/PromotingAppswithAppBanners.html>.
- [23] iOS Developer Library. Implementing Custom URL Schemes. Technical report, Safari Web Content Guide, . Dostupné z: [http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphoneosprogrammingguide/AdvancedAppTricks/AdvancedAppTricks.html#/apple\\_ref/doc/uid/TP40007072-CH7-SW50](http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphoneosprogrammingguide/AdvancedAppTricks/AdvancedAppTricks.html#/apple_ref/doc/uid/TP40007072-CH7-SW50).
- [24] KOCH, P.-P. A pixel is not a pixel. In *Fronteers conference*, Amsterdam, Netherlands, Oct 4, 2012.

- [25] KRATZ, S. – ROHS, M. – ESSL, G. Combining Acceleration and Gyroscope Data for Motion Gesture Recognition Using Classifiers with Dimensionality Constraints. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, IUI '13, pp. 173–178, New York, NY, USA, 2013. ACM. doi: 10.1145/2449396.2449419. Dostupné z: <http://doi.acm.org/10.1145/2449396.2449419>. ISBN 978-1-4503-1965-2.
- [26] MARCOTTE, E. Responsive Web Design. In *A List Apart Magazine: Issue 306*, New York, New York, USA, May 25, 2010. A List Apart. Dostupné z: <http://www.alistapart.com/articles/responsive-web-design>. ISSN 1534-0295.
- [27] MARQUIS, M. 20MB Responsive Websites. In *Breaking Development*, Orlando, FL, USA, April 8-10, 2013.
- [28] MCLACHLAN, P. Page Speed is Only the Beginning. In *Breaking Development*, San Diego, CA, USA, July 22-24, 2013.
- [29] MUELLER, H. – GOVE, J. L. – WEBB, J. S. Understanding Tablet Use: A Multi-Method Exploration. In *Proceedings of the 14th Conference on Human-Computer Interaction with Mobile Devices and Services (Mobile HCI 2012)*, 2012.
- [30] W3C. DeviceOrientation Event Specification. In *Editor's Draft*, 13 June 2012. Dostupné z: <http://dev.w3.org/geo/api/spec-source-orientation.html>.
- [31] W3C. Media Queries. In *W3C Recommendation*, 19 June 2012. Dostupné z: <http://www.w3.org/TR/css3-mediaqueries/>.
- [32] W3C. The Network Information API. In *W3C Working Draft*, 29 November 2012. Dostupné z: <http://www.w3.org/TR/netinfo-api/>.
- [33] W3C. Navigation Timing. In *W3C Recommendation*, 17 December 2012. Dostupné z: <http://www.w3.org/TR/navigation-timing/>.
- [34] W3C. CSS Values and Units Module Level 3. In *W3C Candidate Recommendation*, 4 April 2013. Dostupné z: <http://www.w3.org/TR/css3-values/>.
- [35] W3C. Web Speech API Specification. In *Specification Draft*, 19 June 2012. Dostupné z: <https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html>.
- [36] WROBLEWSKI, L. Data Monday: Big Screen Smartphones. Technical report, LukeW Ideation + Design, Silicon Valley, CA, USA, October 15, 2012. Dostupné z: <http://www.lukew.com/ff/entry.asp?1644>.
- [37] WROBLEWSKI, L. Data Monday: Can Smartphones Keep Growing? Technical report, LukeW Ideation + Design, Silicon Valley, CA, USA, October 21, 2012. Dostupné z: <http://www.lukew.com/ff/entry.asp?1644>.

- [38] WROBLEWSKI, L. Linking Mobile Web and Native App Experiences. Technical report, LukeW Ideation + Design, Silicon Valley, CA, USA, November 14, 2012. Dostupné z: <http://www.lukew.com/ff/entry.asp?1654>.
- [39] WROBLEWSKI, L. Mobile First. New York, New York, USA : A Book Apart, 1st edition, 2011. Dostupné z: <http://www.abookapart.com/products/mobile-first>. ISBN 9781937557027.
- [40] WROBLEWSKI, L. Responsive Navigation: Optimizing for Touch Across Devices. Technical report, LukeW Ideation + Design, Silicon Valley, CA, USA, November 2, 2012. Dostupné z: <http://www.lukew.com/ff/entry.asp?1649>.
- [41] WROBLEWSKI, L. Device Motion. In *Re-imaging Apps for Ultrabook*. Intel Software, 2013.

## A Obsah CD nosiča

K práci je priložené CD, na ktorom sa nachádzajú implementované moduly spolu s demonštračnými aplikáciami a dokumentácia. Jeho súborová štruktúra je nasledujúca:

- **/Dokument/**  
dokument a anotácie v slovenskom a anglickom jazyku  
<https://github.com/janantala/adaptive-web-design>
- **/IIT.SRC2014/**  
článok a poster prezentovaný na IIT.SRC2014  
<https://github.com/janantala/beyond-adaptive-web-design>
- **/source/angular-adaptive/**  
implementované moduly spolu s technickou dokumentáciou a inštalačnou príručkou  
<https://github.com/angular-adaptive>
- **/source/Gyrocopter/**  
rozšírenie do Chrome Developer Tools na emulovanie gyroskopu  
<https://github.com/janantala/Gyrocopter>
- **/source/speech-synthesis/**  
polyfill na syntézu reči spolu s technickou dokumentáciou a inštalačnou príručkou  
<https://github.com/janantala/speech-synthesis>

## B Technická dokumentácia a inštalačná príručka

- adaptive-speech
- adaptive-scroll
- adaptive-motion
- adaptive-youtube
- adaptive-googlemaps
- adaptive-detection
- Gyrocopter
- speech-synthesis

# adaptive-speech v0.3.0 build passing

This module allows you to control web app using voice commands. It's based on Chrome's speech recognition API.

## Demo

Check out <http://angular-adaptive.github.io/adaptive-speech/demo/>

## References

We recomend you to read: - [A More Awesome Web](#) presentation from Google IO 2013 by Eric Bidelman - [Voice Driven Web Apps](#) article from HTML5 ROCKS by Glen Shires

## Requirements

- AngularJS v ~1.2.x

## Usage

We use [bower](#) for dependency management. Add

```
dependencies: {  
    "angular-adaptive-speech": "latest"  
}
```

To your `bower.json` file. Then run

```
bower install
```

This will copy the speech recognition files into your `bower_components` folder, along with its dependencies. Load the script files in your application:

```
<script type="text/javascript" src="bower_components/angular/angular.js"></script>  
<script type="text/javascript"  
    src="bower_components/angular-adaptive-speech/angular-adaptive-speech.min.js">  
</script>
```

Add the adaptive.speech module as a dependency to your application module:

```
var myAppModule = angular.module('MyApp', ['adaptive.speech']);
```

and include \$speechRecognition, \$speechSynthetis, \$speechCorrection service as a dependency to your controller:

```
angular.module('MyApp').controller('MainCtrl',
  function ['$scope', '$speechRecognition', '$speechSynthetis',
  ($scope, $speechRecognition, $speechSynthetis) {
}]);
```

To start speech recognition run from controller:

```
$speechRecognition.onStart(function(){
  $speechSynthetis.speak('Yes? How can I help you?', 'en-UK');
});
$speechRecognition.setLang('en-UK'); // Default value is en-US
$speechRecognition.listen();
```

Apply the directive to your elements where *reference* is keyword reference:

```
<ul>
<li
  ng-repeat="todo in todos | filter:statusFilter track by $index"
  speechrecognition="{ 'tasks': recognition['en-US']['listTasks'], 'reference': todo}"
  {{todo}}>
</li>
</ul>
```

Or run recognition directly from controller:

```
$speechRecognition.listenUtterance($scope.recognition['en-US']['addToList']);
```

## Options

All the speechRecognition options can be set up in your controller.

```
myAppModule.controller('MyController', function($scope) {
  $scope.recognition = {};
  $scope.recognition['en-US'] = {
    'addToList': {
      'regex': '^to do .+/gi',
      'lang': 'en-US',
      'call': function(e){
        $scope.addToList(e);
      }
    },
    'listTasks': [
      {
        'regex': '^complete .+/gi',
        'lang': 'en-US',
      }
    ]
  }
});
```

```
        'call': function(e){
            $scope.completeTask(e);
        }
    },{
        'regex': /^remove .+/gi,
        'lang': 'en-US',
        'call': function(e){
            $scope.removeTask(e);
        }
    }]
};

});
```

# API

## \$speechRecognition

---

### onstart(fn)

On start event.

```
$speechRecognition.onstart(function(e){
    // onstart
});
```

### onerror(fn)

On error event.

```
$speechRecognition.error(function(e){
    // onerror
});
```

### onUtterance(cb)

On recognised utterance callback.

```
$speechRecognition.onUtterance(function(utterance){
    console.log(utterance); // buy a milk
});
```

### setLang(lang)

Set recognition language.

```
$speechRecognition.setLang('en-US');
```

To change language when recognition is already running you need to also restart recognizer:

```
$speechRecognition.stopListening();  
$speechRecognition.listen();
```

## getLang()

Get recognition language.

```
$speechRecognition.getLang(); // 'en-US'
```

## payAttention()

Continue speech recognition after pause caused by `ignore()`. You don't need user permission again.

```
$speechRecognition.payAttention();
```

## ignore()

Pause speech recognition.

```
$speechRecognition.ignore();
```

## listen()

Start speech recognition. User permission is required.

```
$speechRecognition.listen();
```

## stopListening()

Stop speech recognition.

```
$speechRecognition.stopListening();
```

## command(utterance)

Call utterance manually.

```
$speechRecognition.command('do something');
```

## **listenUtterance(tasks)**

Add listener for task

```
var task = {  
    'regex': '^do .+/gi,  
    'lang': 'en-US',  
    'call': function(utterance){  
        // do something with utterance 'do something'  
    }  
};  
$speechRecognition.listenUtterance(task);
```

## **\$speechSynthetis**

### **speak(text, lang)**

Speak utterance.

```
$speechSynthetis.speak('Hello there!', 'en-US');
```

### **justSpoke()**

Return true after `speak()` has been called.

```
$speechSynthetis.justSpoke(); // true or false
```

### **recognised()**

Manually mark speechSynthetis voice as recognised. justSpoke will be `true`.

```
$speechSynthetis.recognised();
```

## **\$speechCorrection**

Correct speech recognition. After incorrect recognition utterance will be corrected.

### **addUtterance(utterance, correction, lang)**

Create a key - value pair with incorret recognition, correction and language.

```
$speechCorrection.addUtterance('to something', 'do something', 'en-US');
```

### **removeUtterance(utterance, lang)**

Remove utterance correction.

```
$speechCorrection.removeUtterance('to something', 'en-US');
```

## **addLangMap(lang, map)**

Add complete correction map for a language.

```
var map = {
    'to something': 'do something',
    'pseudo make me a sandwich': 'sudo make me a sandwich'
};
$speechCorrection.addUtterance('en-US', map);
```

## **clearLangMap(lang)**

Remove language map.

```
$speechCorrection.clearLangMap('en-US');
```

## **getCorrectionMap()**

Get correction map for all languages.

```
$speechCorrection.getCorrectionMap();
// {
//     'en-US: {
//         'to something': 'do something',
//         'pseudo make me a sandwich': 'sudo make me a sandwich'
//     }
// }
```

## **getLangMap(lang)**

Get correction map for a language.

```
$speechCorrection.getCorrectionMap('en-US');
// {
//     'to something': 'do something',
//     'pseudo make me a sandwich': 'sudo make me a sandwich'
// }
```

## **getCorrection(utterance, lang)**

Get a single utterance correction.

```
$speechCorrection.getCorrection('pseudo make me a sandwich', 'en-US');
// 'sudo make me a sandwich'
```

## speechrecognition directive

---

Add listener to html element. - tasks: configuration object (*remove something*) - reference: element reference name (*something*)

```
<li
  ng-repeat="todo in todos | filter:statusFilter track by $index"
  speechrecognition="{'tasks': recognition['en-US']['listTasks'], 'reference': todo}"
  {{todo}}
</li>
```

## Testing

We use karma and jshint to ensure the quality of the code. The easiest way to run these checks is to use grunt:

```
npm install -g grunt-cli
npm install
bower install
grunt
```

The karma task will try to open Chrome as a browser in which to run the tests. Make sure this is available or change the configuration in `test/test.config.js`

## Contributing

Pull requests are welcome.

Make a PR against canary branch and don't bump any versions.

Please respect the code style in place.

# adaptive-scroll v0.2.0

[build](#) [passing](#)

This module allows you to scroll an AngularJS app using gyroscope.

**Note:** You need a device with a gyroscope and a browser with requestAnimationFrame support for smooth scrolling

We recommend [Gyrocopter extension for Chrome](#).

## Demo

Check out <http://angular-adaptive.github.io/adaptive-scroll/demo/> Demo image with Gyrocopter extension.



## Requirements

- AngularJS v 1.0+
- device with a gyroscope or a [simulator](#)
- browser with requestAnimationFrame support for smooth scrolling
  - iOS Safari 6+,
  - Chrome for Android 25+,
  - Opera Mobile 14+
  - Firefox for Android 19+

# Usage

We use [bower](#) for dependency management. Add

```
dependencies: {  
    "angular-adaptive-scroll": "latest"  
}
```

To your `bower.json` file. Then run

```
bower install
```

This will copy the scroll files into your `bower_components` folder, along with its dependencies. Load the script files in your application:

```
<script type="text/javascript" src="bower_components/angular/angular.js"></script>  
<script type="text/javascript"  
src="bower_components/angular-adaptive-scroll/angular-adaptive-scroll.min.js"></script>
```

Add the **adaptive.scroll** module as a dependency to your application module:

```
var myAppModule = angular.module('MyApp', ['adaptive.scroll']);
```

and **include \$gyroscope service** as a dependency to your controller:

```
angular.module('MyApp').controller('MainCtrl',  
    function ['$scope', '$gyroscope', ($scope, $gyroscope) {  
  
}]);
```

Append a directive **adaptivescroll** to the scrollable element.

```
<body adaptivescroll>  
<!-- --&gt;<br/></body>  
  
<textarea adaptivescroll></textarea>
```

To start scrolling run method **watchPosition()** and pass trashold argument (in degrees):

```
$gyroscope.watchPosition(10);
```

To stop scrolling run method **ignorePosition()**:

```
$gyroscope.ignorePosition();
```

There are also direction events returning direction diff you can handle

```
$gyroscope.onalpha(function(alphaDiff){  
    console.log(alphaDiff);  
});  
$gyroscope.onbeta(function(betaDiff){  
    console.log(betaDiff);  
});  
$gyroscope.ongamma(function(gammaDiff){  
    console.log(gammaDiff);  
});
```

## Testing

We use karma and jshint to ensure the quality of the code. The easiest way to run these checks is to use grunt:

```
npm install -g grunt-cli  
npm install  
bower install  
grunt
```

The karma task will try to open Chrome as a browser in which to run the tests. Make sure this is available or change the configuration in `test/test.config.js`

## Contributing

Pull requests are welcome.

Make a PR against canary branch and don't bump any versions.

Please respect the code style in place.

# adaptive-motion v0.2.0

[build](#) [passing](#)

This module allows you to control an AngularJS app using web camera.

## Demo

Check out <http://angular-adaptive.github.io/adaptive-motion/demo/>

## Requirements

- AngularJS v 1.0+
- [getUserMedia Stream support](#)

## Usage

We use [bower](#) for dependency management. Add

```
dependencies: {
  "angular-adaptive-motion": "latest"
}
```

To your `bower.json` file. Then run

```
bower install
```

This will copy the angular-adaptive-motion files into your `bower_components` folder, along with its dependencies. Load the script files in your application:

```
<script type="text/javascript" src="bower_components/angular/angular.js"></script>
<script type="text/javascript"
  src="bower_components/angular-adaptive-motion/angular-adaptive-motion.js"></script>
```

Add the **adaptive.motion** module as a dependency to your application module:

```
var myAppModule = angular.module('MyApp', ['adaptive.motion']);
```

and include `$motion` service as a dependency to your controller:

```
angular.module('MyApp').controller('MainCtrl',
  function ['$scope', '$motion', ($scope, $motion) {
}]);
```

## Public methods

**\$motion.start();**

Starts gesture recognition.

**\$motion.stop();**

Stops gesture recognition.

**\$motion.onStart(cb);**

On start callback.

**\$motion.onStop(cb);**

On stop callback.

**\$motion.onError(cb);**

On error callback.

**\$motion.onSwipeLeft(cb);**

On swipe left gesture.

```
$motion.onSwipeLeft(function(data){  
    $scope.$apply(function(){  
        console.log('onSwipeLeft');  
    });  
});
```

**\$motion.onSwipeRight(cb);**

On swipe right gesture.

```
$motion.onSwipeRight(function(data){  
    $scope.$apply(function(){  
        console.log('onSwipeRight');  
    });  
});
```

**\$motion.onSwipeUp(cb);**

On swipe up gesture.

```
$motion.onSwipeUp(function(data){
```

```
$scope.$apply(function(){
    console.log('onSwipeUp');
});
```

## \$motion.onSwipeDown(cb);

On swipe down gesture.

```
$motion.onSwipeDown(function(data){
    $scope.$apply(function(){
        console.log('onSwipeDown');
    });
});
```

## Configuration

You can configure `$motionProvider` to a custom threshold options in app configuration.

```
$motionProvider.setThreshold({
    'rgb': 150,
    'move': 3,
    'bright': 300
});
```

You can also set custom hsv filter.

```
$motionProvider.setHsvFilter({
    'hueMin': 0.0,
    'hueMax': 0.1,
    'satMin': 0.0,
    'satMax': 1.0,
    'valMin': 0.4,
    'valMax': 1.0
});
```

## Visualization

If you want to visualize you can add `adaptive-motion` attribute into your canvas element. You can choose from following styles:

### Video

```
<canvas adaptive-motion="video"></canvas>
```



## Skin

```
<canvas adaptive-motion="skin"></canvas>
```



## Edge

```
<canvas adaptive-motion="edge"></canvas>
```



# Contributing

Contributions are welcome. Please make a pull request against canary branch and do not bump versions. Also include tests.

# Testing

We use karma and jshint to ensure the quality of the code. The easiest way to run these checks is to use grunt:

```
npm install -g grunt-cli  
npm install  
bower install  
grunt
```

The karma task will try to open Chrome as a browser in which to run the tests. Make sure this is available or change the configuration in `test/test.config.js`

# angular-adaptive-youtube v0.1.0

Save data, conditionally load youtube videos.

## Demo

Check out <http://angular-adaptive.github.io/adaptive-youtube/demo/>

## Requirements

- AngularJS v ~1.2.x

## Usage

We use [bower](#) for dependency management. Add

```
dependencies: {
  "angular-adaptive-youtube": "latest"
}
```

To your `bower.json` file. Then run

```
bower install
```

This will copy the youtube module files into your `bower_components` folder, along with its dependencies. Load the script files in your application:

```
<script type="text/javascript" src="bower_components/angular/angular.js"></script>
<script type="text/javascript"
  src="bower_components/angular-adaptive-youtube/angular-adaptive-youtube.min.js"></script>
```

Add the adaptive.youtube module as a dependency to your application module:

```
var myAppModule = angular.module('MyApp', ['adaptive.youtube']);
```

Set up styles and a video id

```
.youtube { width: 100%; height: 500px; }
```

```
$scope.youtubeId = 'kxopViU98Xo';
```

And use it

```
<youtube video="youtubeId" class="youtube"></youtube>
```

## Contributing

Pull requests are welcome.

Make a PR against canary branch and don't bump any versions.

Please respect the code style in place.

# adaptive-googlemaps v0.3.0

[build](#) [passing](#)

This module allows you to adapt googlemaps component for different occasions. - static google map - open map in a new tab / a native mobile application - load dynamic google map

## Demo

Check out <http://angular-adaptive.github.io/adaptive-googlemaps/demo/>

## Requirements

- AngularJS v 1.0+
- Googlemaps JS API v3 (only for dynamic google map)

## Usage

We use [bower](#) for dependency management. Add

```
dependencies: {
  "angular-adaptive-googlemaps": "latest"
}
```

To your `bower.json` file. Then run

```
bower install
```

This will copy the angular-adaptive-googlemaps files into your `bower_components` folder, along with its dependencies. Load the script files in your application:

```
<script src="https://maps.googleapis.com/maps/api/js?sensor=false"></script>
<script type="text/javascript" src="bower_components/angular/angular.js"></script>
<script type="text/javascript"
  src="bower_components/angular-adaptive-googlemaps/angular-adaptive-googlemaps.min.js"
></script>
```

Add the **adaptive.googlemaps** module as a dependency to your application module:

```
var myAppModule = angular.module('MyApp', ['adaptive.googlemaps']);
```

Add **googlemaps** element into your template

```
<googlemaps class="google-maps" options="map1"></googlemaps>
```

## Directive attributes

```
$scope.map1 = {  
    sensor: false,  
    size: '500x300',  
    zoom: 9,  
    center: 'San Francisco International Airport',  
    markers: ['San Francisco', 'San Jose'],  
    maptype: 'terrain',  
    mapevents: {redirect: false, loadmap: true},  
    listen: true  
};
```

### Required:

- sensor: false // true, false
- size: '500x300' // width x height
- zoom: 6
- center: 'California'

### Optional:

- markers: ['San Francisco', 'San Jose']
- maptype: 'roadmap' // roadmap, satellite, terrain, hybrid
- mapevents: {redirect: true, loadmap: false}
- listen: true // watch for attributes change

**mapevents: { "redirect": false, "loadmap": true }**

Loads dynamic google map.



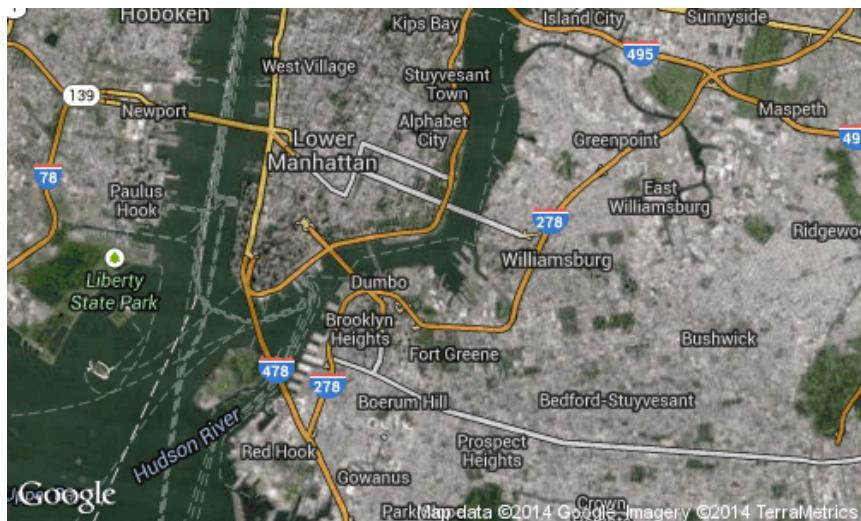
```
{ "redirect": true, "loadmap": false }
```

Opens google map in a new tab / a native mobile application.



```
mapevents: no mapevents
```

Nothing happens.



## Contributing

Contributions are welcome. Please make a pull request against canary branch and do not bump versions. Also include tests.

### Todo

- Let the device/browser decide what kinds of map events will be used. *Example: mobile device will open native maps application, browser with fast internet connection will auto load dynamic map...*

## Testing

More tests will be added...

We use karma and jshint to ensure the quality of the code. The easiest way to run these checks is to use grunt:

```
npm install -g grunt-cli  
npm install  
bower install  
grunt
```

The karma task will try to open Chrome as a browser in which to run the tests. Make sure this is available or change the configuration in `test/test.config.js`

# adaptive-detection v0.3.0 build passing

This module allows you to detect iOS, Android and Windows Phone devices using user agent string

## Demo

Check out <http://www.janantala.com/slides/how-to-build-an-open-source-angularjs-module/#/7/4>

## Requirements

- AngularJS v 1.2.x+

## Usage

We use [bower](#) for dependency management. Add

```
"dependencies": {  
    "angular-adaptive-detection": "latest"  
}
```

To your `bower.json` file. Then run

```
bower install
```

This will copy the angular-adaptive-detection files into your `bower_components` folder, along with its dependencies. Load the script files in your application:

```
<script type="text/javascript" src="bower_components/angular/angular.js"></script>  
<script type="text/javascript"  
src="bower_components/angular-adaptive-detection/angular-adaptive-detection.min.js">  
</script>
```

Add the **adaptive.detection** module as a dependency to your application module:

```
var myAppModule = angular.module('MyApp', ['adaptive.detection']);
```

and include \$detection provider as a dependency to your controller:

```
angular.module('MyApp').controller('MainCtrl',  
    function ['$scope', '$detection', ($scope, $detection) {  
}]);
```

## Configuration

You can configure provider to a custom User Agent string in app configuration.

```
$detectionProvider.setUserAgent('angular browser');
```

## Public methods

You can detect Android and iOS devices using:

```
$detection.isAndroid();
$detection.isiOS();
$detection.isWindowsPhone();
```

## Contributing

Contributions are welcome. Please make a pull request against canary branch and do not bump versions. Also include tests.

## Testing

We use karma and jshint to ensure the quality of the code. The easiest way to run these checks is to use grunt:

```
npm install -g grunt-cli
npm install
bower install
grunt
```

The karma task will try to open Chrome as a browser in which to run the tests. Make sure this is available or change the configuration in `test/test.config.js`

# Gyrocopter v0.3.0

Gyroscope Simulator - Extension for Chrome Developer Tools



[Download from Chrome Web Store](#)

Do you develop for mobile? Then you need a proper gyroscope simulator.

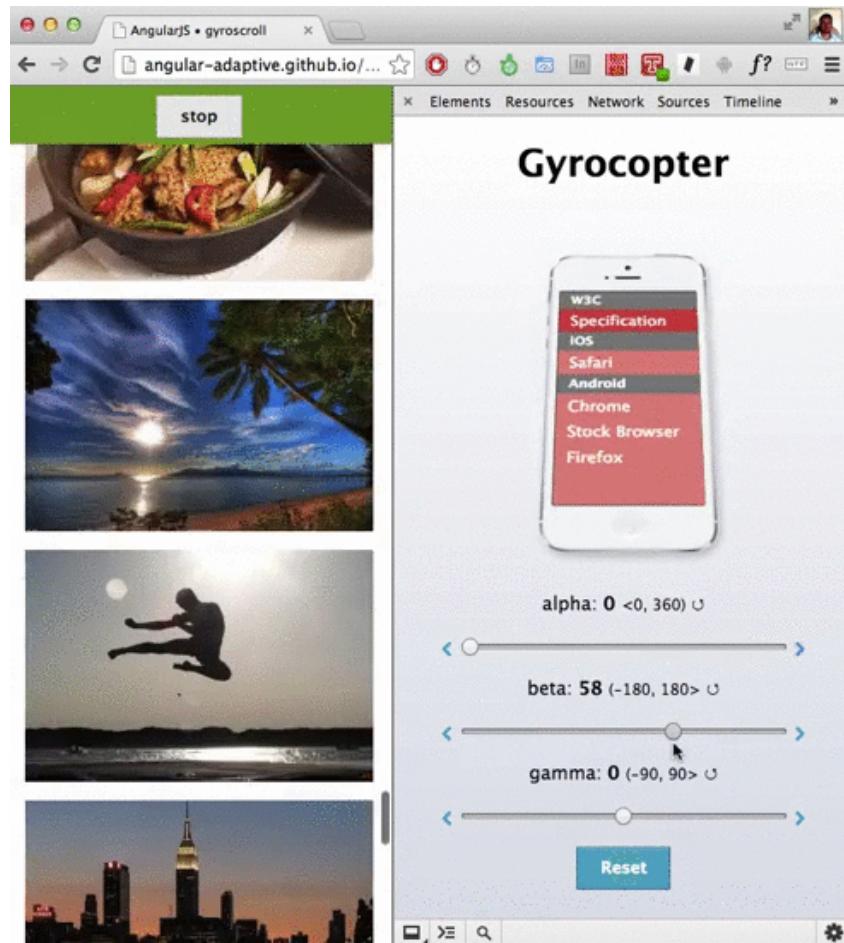
This simulator allows you to dynamically change rotation of a mobile device and also sends deviceorientation events into your web app.

You can also select browser in which you want to perform device rotation. If you want to know how browsers handle device orientation events read [test results](#).

## Demo

---

Check out [Angular Adaptive Scroll project](#)



## Why?

Because Chrome developer tools override is not enough.

Override Device Orientation

α:  β:  γ:

You want to dynamically change rotation of a mobile device and also see live preview when you develop.

# speech-synthesis v0.3.3

Speech Synthesis polyfill based on Google Translate service. Polyfill downloads audio from Google Translate server using [CORS](#) and plays it using [audio](#) element.

## Demo

Check out <http://janantala.github.io/speech-synthesis/>

## References

- [W3C: Web Speech API Specification](#)
- [HTML5 rocks: Web apps that talk - Introduction to the Speech Synthesis API](#)
- Speech recognition is more tricky. [Control an AngularJS app using voice commands](#)

## Usage

We use [bower](#) for dependency management. Add

```
dependencies: {  
  "speech-synthesis": "latest"  
}
```

To your `bower.json` file. Then run

```
bower install
```

This will copy the files into your `bower_components` folder, along with its dependencies. Load the script files in your application:

```
<script type="text/javascript"  
src="bower_components/speech-synthesis/polyfill.min.js"></script>
```

And finally use speech synthesis:

```
// Initialize polyfill  
var fallbackSpeechSynthesis =  
  window.speechSynthesis || window.speechSynthesisPolyfill;  
var fallbackSpeechSynthesisUtterance =  
  window.SpeechSynthesisUtterance || window.SpeechSynthesisUtterancePolyfill;  
  
var u = new fallbackSpeechSynthesisUtterance('Hello World');  
u.lang = 'en-US';
```

```
u.volume = 1.0;
u.rate = 1.0;
u.onend = function(event) {
  console.log('Finished in ' + event.elapsedTime + ' seconds.');
};
fallbackSpeechSynthesis.speak(u);
```

## CORS proxy server

CORS proxy server is required to download audio from google translate service into your web application. Default value is set to `http://www.corsproxy.com/` but we would recommend you to use your own server. To set up your own change `corsProxyServer` attribute in `SpeechSynthesisUtterance` instance.

```
u.corsProxyServer = 'http://www.corsproxy.com/';
```

## Identify the polyfill usage

To identify the polyfill usage you can use `isPolyfill` attributes.

```
var u = new window.SpeechSynthesisUtterancePolyfill('Hello World');
u.isPolyfill; // true

window.speechSynthesisPolyfill.isPolyfill; // true
```

# Supported attributes and methods

## SpeechSynthesis Attributes

- pending
- speaking
- paused

## SpeechSynthesis Methods

- speak()
- cancel()
- pause()
- resume()
- getVoices()

## SpeechSynthesisUtterance Attributes

- `text`
- `lang`
- `voiceURI`
- `volume`
- `rate`
- `pitch`

## **SpeechSynthesisUtterance Events**

- `onstart`
- `onend`
- `onerror`
- `onpause`
- `onresume`
- `onmark`
- `onboundary`

## **SpeechSynthesisEvent Attributes**

- `charIndex`
- `elapsedTime`
- `name`

## **SpeechSynthesisVoice**

- `voiceURI`
- `name`
- `lang`
- `localService`
- `default`

*Voice depends on google translate service.*

## **SpeechSynthesisVoiceList**

- `length`
- `item`

## **C Článok prezentovaný na IIT.SRC2014**

Na základe našej práce sme zároveň vytvorili odborný článok a prezentovali ho na konferencii IIT.SRC 2014 pod názvom „Beyond Adaptive Web Design”. V prílohe prinášame jeho plnú verziu.

# Beyond Adaptive Web Design

Ján ANTALA\*

*Slovak University of Technology in Bratislava  
Faculty of Informatics and Information Technologies  
Ilkovičova 2, 842 16 Bratislava, Slovakia  
hello@janantala.com*

**Abstract.** Mobile device penetration grows rapidly and it brings not only different display sizes with new human interaction methods, but also high-latency networks. But there are not just smartphones and desktop computers. We now have Web-enabled smartphones, tablets, e-readers, netbooks, watches, TVs, phablets, notebooks, game consoles, cars and more. So it is necessary to adapt web pages for different devices and new interaction methods. Several new methods for designing web are rapidly evolving. We present a new way of web components adaptation based on a device features and external conditions in our work. We introduce new possibilities of interaction with the web applications based on speech recognition, motion or device rotation.

## 1 Introduction

The Web is continually evolving and we need to evolve with it. It used to be easier to manage browsers when there were just a few of them on the desktop. Today we not only have to deal with a wide range of desktop browsers but mobile devices, tablets, televisions, weareable devices and more. Even for the average web site things have changed a lot over four years: browser share, operating systems, screen resolutions, and more [7].

The basic approach how to provide an optimal viewing experience is to use “Responsive web design” which comes with fluid grids, flexible images and CSS3 media queries [9]. While creating flexible layouts is important, there is a lot more that requires remarkable adaptive web experiences. The principles of Adaptive Design are: ubiquity, flexibility, performance, enhancement and future-friendliness [4].

The power of the web is its ubiquity. No one knows what the landscape is going to appear in a couple years, but there is a reasonable probability the new devices in a few years from now will have access to the web. Because of this, we need to preserve and embrace the web’s ubiquity. This requires us to provide full web experiences regardless of how and where people access the web. It is important to remain in creating flexible interfaces that can adapt to any screen size. Performance is often less important to everything else. “Progressive enhancement”, feature detection and many other techniques allow us to increase experience that allow us to support more devices while still

---

\* Master study programme in field: Information Systems

Supervisor: Assoc. Professor Michal Čerňanský, Institute of Applied Informatics, Faculty of Informatics and Information Technologies STU in Bratislava

optimizing for the finest. The key aspect of “Future Friendly” thinking is to acknowledge and embrace unpredictability. Nobody knows where things are going but the one thing we can trust is the evolution.

Adaptive web design is fundamentally progressive enhancement, but it is being applied to a much larger and more diverse landscape. We now have Web-enabled smartphones, tablets, e-readers, netbooks, watches, TVs, phablets, notebooks, game consoles, cars and more. We also have many types of internet networks with different speed, latency and quality. Responsive design is also one technique in an adaptive web design strategy. Creating flexible layouts is important, but there are many more factors we need to think about. It is also important to consider as well ergonomics, touch capability, other input methods, internet connections and many other features that can be detected.

We consider “adaptive web design” as an equal with creating a single Web experience. We can adjust it based on the capabilities of the device and browser. Website can access sensors in devices and use them to enhance user experience.

## 2 Adaptive Input methods

As we have many types of devices with web access we also have to adapt input methods for them. It is not just about screen sizes, but many displays also have touch capabilities, are equipped with microphone, accelerometer, gyroscope or camera.

Web browsers on TVs are actually really good but the input is terrible so no one uses them. Input is much more important to interface design than screen size. The input defines what a design needs to do in order to accomplish a task [6]. We also have wearable computers. The current Google Glass interface is also limited to several input methods.

There is no universal input for the Web. We have to contend with fingers, mice, keyboards, voice and more. The challenge for us over the next few years is not designing for one input but many.

### **Touch**

Touch is no longer just isolated to smartphones and tablets. Nowadays many large screens on laptops or desktops are touch capable. Thus every desktop design has to be touch-friendly now.

The optimal touch target size is 7mm, based on the average size of human finger tips and pads. CSS2.1 defines a pixel as 1/96 of an inch. So 7mm should be 30pixels [3]. However, things are not so easy because of dynamic viewports. It means that input elements in website have to be usually bigger.

The best interface can be sometimes controlled without touching at all. We can use sensors and speech input which is the next frontier for interaction design.

### **Speech**

The speech input aims to provide an alternative input method for web applications, without using a keyboard or other physical device. This is no longer science fiction but reality. Many companies rely on it and deploy it to the modern devices and software. It could be also beneficial for older or disabled users [1] and can easily become the next big thing.

We can now use speech recognition to take a full control of the web site as we have access to continuous recognition stream. We do not need another input methods at all. A speech feedback for user interaction is also important and we can use speech synthesis feature for it.

### **Device motion**

We can use a lot of sensors in the devices to detect the device motion. It is made possible by a combination of always-on sensors as are an accelerometer a gyroscope, and a magnetometer, that

tell us a lot about how a device is moving through the space around it. The ability of these sensors to provide precise information about the movement of a device opens up new design possibilities for web applications [12].

We can control the user interface based on three dimensional motion as input or we can combine it with another sensors as are location detection or video cameras [8]. There are a lot of interesting interface designs made possible by device motion. It depends on us how we use them.

### **Video motion detection**

Many modern devices contains built in video camera sensor which we can use as alternative method to device motion. We can access video stream and analyse landscape around the device. Video motion detection is a way of defining user activity in a scene by analyzing image data and differences in a series of images.

We can detect user motion and use it as an alternatove part of input method. We can build web games and another real time interfaces. There is again a lot of interesting interface designs made possible.

## **3 Adaptive Web Components**

One of the most common critiques against responsive Web design is that Web sites have large file size and so they are slowing Web pages down. But we can make responsive sites perform well, we just have to do the work to make it possible [10].

We all should be interested in page speed and mobile performance because people need fast experiences on the Web. There is lot of studies that highlight the importance of speed. More than half of people with a bad loading experience on mobile, will not come back [5]. 73% of mobile internet users say they have encountered Web pages that are too slow, a 1 second delay can result in a 7% reduction in conversions [11]. Speed is an esencial part of our strategy and we should care about it. We can quantify its impact on the metrics and track it.

There are many different types of internet connections. Mobile network speeds have rose but this does not help much as page load times are still high as bandwidth increases. Latency is a much bigger fragment to download times. LTE standart reduces tower latency by several milliseconds but we still need to make a number of connections to download data to our devices [5].

So what we should do about it? We can create reusable components and conditionally load resources. Web components allow us to use custom HTML elements in the browser. We can enclose a simple element or an entire application logic within an HTML element [2]. We can build Web applications in a reusable, commutable and encapsulated way.

## **4 Evaluation and experiments**

We have detected several web components and input methods and tried to verify which of them provide better experience for the website users and are also useful for the web developers. There have been made series of experiments on both adaptive input methods and adaptive web components. To prove the usefulness of the proposed concept we have designed and implemented several reusable modules and components and we have used them in propotype web projects. All of them have been released as open source and are available on GitHub<sup>1</sup>.

We have tracked amount of saved web traffic and web requests, webpage rendering time and interest in alternative input methods. We have also received a lot of feedback from the community

---

<sup>1</sup> The modules and components we used to verify the usefulness of the proposed concept are available on GitHub <https://github.com/angular-adaptive>

which helped us to verify the proposed concept. According to analytics the experiment has been attended by thousands of users and still has the weekly traffic one hundred of unique visitors.

#### **4.1 Input methods**

Adaptive input methods provide alternative way of web application control and extend current approach. We have been experimenting to control web applications using voice commands, motion detected by a gyroscope or a camera.

##### **Voice commands**

Speech input provides a great opportunity to take a complete control over web application using only voice commands. For this purpose we have used web browser's Speech API for continuous speech recognition and built a configurable and reusable module. Speech recognition is however currently an experimental feature and is possible online only.

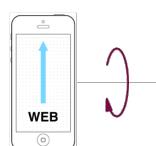
Using speech recognition we get an utterance string which can be compared to voice commands from our configuration. We use regular expressions for this purpose with attached callback methods. These configurations are bound to utterance watchers and the callback is called when voice command is recognised. Regular expressions are beneficial since we can set up a single watcher for dynamical commands where one command can be used for multiple subjects.

We have created an experiment based on our module which provides control over to-do list. There were 7 voice commands using exact expressions and 9 using regular expressions. This has been the most popular input method and the experiment has been attended by thousands of users. But there are much more interesting things we can achieve using speech input. We can even use multiple languages for speech recognition. One of the biggest problems of the voice commands is however incorrect speech recognition. To solve this issue we can conditionalize regular expressions or use utterance error correction.

##### **Gyroscope rotation**

We have used device orientation API as alternative control method to web application and built a module which can scroll in website detecting rotation using a gyroscope. Developers can use this module as replacement of the global application scroll instead of touch events or mice wheel and also a local element scroll too.

We have access to 3 axis rotation information which are recalculated every time on device orientation change. We remember starting device orientation and compare it to the current one. Based on this subtraction we can make a decision which direction we scroll and how fast. The bigger the orientation difference is, the faster we scroll.



*Figure 1. Scrolling in the webpage by gyroscope*

The experiment has been attended by thousands of users. However some problems have been detected. The biggest one is that browser vendors do not use specification correctly and use own orientation ranges and directions. This causes some scroll issues.

#### 4.1.1 Video motion detection

Almost every modern device contains a video camera. This is a great opportunity to introduce video motion to web applications. We can control web sites using motion gestures. We use a video camera stream and divide it into image frames. Then we can detect a website user in each frame using a HSV filter. When we compare all pixels in two adjacent frames we get difference points. When we apply this comparison to multiple frames we can calculate a motion vector.



*Figure 2. Converting video stream into motion vector*

We support 4 direction motion events: swipe from the left to the right, right to left, bottom to top and top to bottom. The experiment has been attended by hundreds of users. Using video motion detection we can enhance experience in websites and games. We can also detect device motion and orientation changes so it can be used as a supplementary input method to the accelerometer and gyroscope rotation. However we cannot take a full control over the webapp and need a proper ambient lighting. There can be detection issues in too dark and too bright scenes.

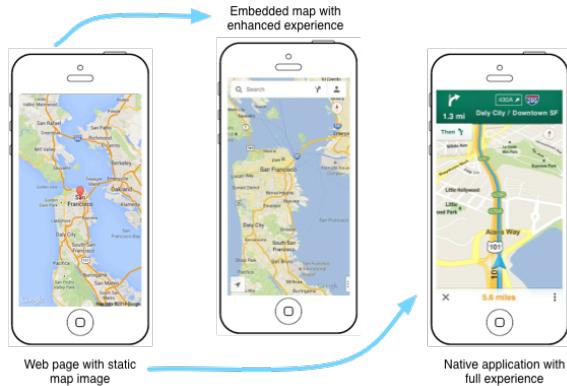
## 4.2 Web Components

There are many commonly used web services that produce unnecessary traffic. Elements like videos or maps are a common part of the most web sites. All of them produce many requests and the web traffic so the website user does not want to watch the video or browse the map. There are also native mobile application for those services with better performance which provide full user experience. So what can we do? This is a great opportunity to utilize conditional loading to serve the best experience for the right context.

We have used the Mobile First principle to develop Google Maps and YouTube videos web components. So we make a default element static and pull in a static image. This approach saves a lot of web traffic and because of limited mobile data plans also user's money. We can then detect when it is appropriate to download the embedded map or to launch the native mobile application.

By default we set an anchor to the location of the service website. When a user taps the component, the browser will try to open the native application where they can have a better experience. If the native experience is unavailable, the user is redirected to the service website, where they receive a full-screen experience. However, we do not always want to open the native application. We can download an embedded map or play the video directly after the tap. The user can interact with them directly in the web application so developers have to make a decision between these choices.

Using adaptive web components we reduce initial traffic by 400 kB and a page load time by 350 ms for a YouTube video on average. For the map elements the amount of saved traffic and load times are various and depend on element size. This approach works quite well for simple use cases. We can show multiple map types, markers or videos. However, more interactive elements require additional consideration. Even then use of an embedded element directly still might not make sense because of unnecessary traffic. We can still use adaptive techniques to reduce it and replace basic static image with richer elements.

*Figure 3. Maps component*

## 5 Conclusion

We have received a lot feedback and have usage and popularity results of adaptive input methods. As we have expected, the most popular input method is speech input since we can control an entire web application using only voice commands. Gyroscope and video motion are also useful input methods, but there are limited possibilities where to use them and can be used as supplementary input method only. All of them however provide a great opportunity to enhance user experience.

Adaptive web components are a great to save an unwanted web traffic. They also increase user experience because the web page produces less requests and loads faster. However the limitations also exist because many websites need custom elements and design. As a result the adaptive web components produce less interest than adaptive input methods, but are the necessary part of the Web.

More changes are coming, new devices, web APIs and beyond. How it will look in a few years? The devices will be more diversified and have the web access. So we have to prepare the Web to this evolution and provide the best user experience, design for many inputs and save the inessential web traffic.

## References

- [1] Basson, S., Fairweather, P.G., Hanson, V.L.: Speech Recognition and Alternative Interfaces for Older Users. *interactions*, 2007, vol. 14, no. 4, pp. 26–29.
- [2] Bidelman, E.: Web Components: A Tectonic Shift for Web Development. In: *Google IO*, San Francisco, CA, USA, 2013.
- [3] Clark, J.: Designing for Touch. In: *An Event Apart*, Seattle, WA, USA, April 1-3, 2013.
- [4] Frost, B.: Beyond Squishy: The Principles of Adaptive Design. In: *SXSW*, Austin, TX, USA, March 9, 2013.
- [5] Grigorik, I.: Optimizing the Critical Rendering Path. In: *Breaking Development*, San Diego, CA, USA, July 22-24, 2013.
- [6] Grigsby, J.: Adaptive Input. In: *Breaking Development*, San Diego, CA, USA, July 22-24, 2013.
- [7] Gustafson, A.: Building Adaptive Designs Now. In: *User Interface 17 Conference*, Boston, MA, USA, November 5-7, 2012.
- [8] Kratz, S., Rohs, M., Essl, G.: Combining Acceleration and Gyroscope Data for Motion Gesture Recognition Using Classifiers with Dimensionality Constraints. In: *Proceedings of the 2013*

*International Conference on Intelligent User Interfaces.* IUI '13, New York, NY, USA, ACM, 2013, pp. 173–178.

- [9] Marcotte, E.: Responsive Web Design. In: *A List Apart Magazine: Issue 306*, New York, NY, USA, A List Apart, May 25, 2010.
- [10] Marquis, M.: 20MB Responsive Websites. In: *Breaking Development*, Orlando, FL, USA, April 8-10, 2013.
- [11] McLachlan, P.: Page Speed is Only the Beginning. In: *Breaking Development*, San Diego, CA, USA, July 22-24, 2013.
- [12] Wroblewski, L.: Device Motion. In: *Re-imagining Apps for Ultrabook*, Intel Software, 2013.