

# **BACHELOR'S PAPER**

Degree Program Informatik/Computer Science

## **Conception and prototyping of a modern browser's user interface**

By: Thomas Greiner

Student Number: 0910257037

Supervisor: FH-Prof. DI Dr.techn. Robert Pucher

Gols, 28.05.2012



## Declaration

„I confirm that this paper is entirely my own work. All sources and quotations have been fully acknowledged in the appropriate places with adequate footnotes and citations. Quotations have been properly acknowledged and marked with appropriate punctuation. The works consulted are listed in the bibliography. This paper has not been submitted to another examination panel in the same or a similar form, and has not been published. I declare that the present paper is identical to the version uploaded.“

Gols, 28.05.2012

Place, Date

Signature

## Summary

Browsers have become an integral part of today's society. What they all have in common is that they tend to become less visible to the user and hide as much of their user interface as possible. But at the same time they experience a significant increase in the number of features.

This work discusses new ideas on both already proven and new features browser vendors came up with over the years. A concept based on those understandings is the foundation of a JavaScript-based prototype which interactively visualizes the ideas expressed in it. The objective of this work is to find out whether or not such a concept could potentially improve a browser's usability and user experience based on the results of an user acceptance evaluation undertaken in the course of this work.

**Keywords:** Browsers, Usability, User Interface, User Experience, Prototype, JavaScript

## **Abstract**

The purpose of this work is to create a concept for a modern browser's user interface based on current browser trends and to undertake an user acceptance evaluation to gather feedback on both strengths and weaknesses of that particular concept. The development and analysis of a JavaScript-based prototype which results from that concept answers the question whether or not such a concept can be realized and what the differences could be between the prototype and a real-world implementation. The results from the evaluation are the basis for concluding if such an implementation makes sense and whether or not it could potentially improve a browser's user experience.

# Table of Contents

1 Introduction.....	5
2 Basics.....	5
2.1 What is a browser?.....	6
2.2 What is an user interface?.....	6
2.3 What is user experience?.....	6
3 Conception.....	6
3.1 Ideas.....	7
3.1.1 The browser in the background.....	7
3.1.2 A different approach to bookmark management.....	8
3.1.3 Focus on navigation.....	9
3.2 Environment.....	11
3.2.1 Rendering engine.....	11
3.2.2 JavaScript interpreter & other components.....	12
3.2.3 Other technologies.....	12
3.3 Mockup.....	13
4 Prototyping.....	14
4.1 File structure.....	15
4.2 Basic functionality.....	17
4.3 Dynamic chrome.....	18
4.4 The Visualizer.....	21
4.5 Extendable, dynamic UI.....	22
4.6 Left out components.....	23
5 User acceptance.....	23
6 Conclusion.....	25
Bibliography.....	27
List of Figures.....	31
List of Abbreviations.....	32

A: Interview structure for user acceptance evaluation.....	33
B: Differentiation of UI elements (Participant A).....	36
C: Differentiation of UI elements (Participant B).....	37
D: Differentiation of UI elements (Participant C).....	38
E: Differentiation of UI elements (Participant D).....	39
F: Differentiation of UI elements (solution).....	40
G: Results of user acceptance evaluation.....	41
H: Hypercube displaying cnn.com.....	43
I: Hypercube displaying start page.....	44
J: Hypercube displaying ifttt.com.....	45
K: Visualizer with some nodes and connections.....	46
L: Hypercube running as a web application on Google Chrome Beta 0.18.449.2396 on Samsung Galaxy Nexus with Android 4.0.2.....	47

# 1 Introduction

This work is based on my previous bachelor thesis titled “Browsers and their influence on the evolution of the web”[GR12] in which I tried to answer the question whether or not and to what degree browsers influenced the evolution of the web. Those results provide the necessary information to think about what the look and feel of the next modern browser should be like. This work describes the process of creating a concept and later on a prototype of such a browser’s user interface (UI) by looking at current browsers and mixing in new ideas to try to improve a browser’s usability and user experience (UX).

The purpose of this work is to spread new ideas and to give the reader some insight in how future browsers may look like but also to find out if the ideas expressed in this work can be transformed into a working prototype with the potential to improve current browsers' usability and/or UX.

The structure of this work follows the creation process from initial concepts to an actual JavaScript-based prototype of a browser's user interface. The result is an assessment whether or not the concept can be realized in form of a prototype and used in a real-world implementation.

An user acceptance evaluation was undertaken to get feedback on both strengths and weaknesses of the concept and to find out whether or not it could potentially improve a browser's UX.

The prototype – including its source code – was published under the Artistic License 2.0 (<http://www.opensource.org/licenses/artistic-license-2.0>) and can be found at the following location:

<http://www.greinr.com/bachelorthesis/hypercube>

## 2 Basics

It is necessary to begin with a short overview over the terminology being used in this work before going into any more detail.

## 2.1 What is a browser?

According to Wiktionary a browser (or web browser) is *“a software component capable of rendering HTML [Hypertext Markup Language] pages and allowing for navigation of HTML links, for example on the Internet.”*[WIKT12] More generally speaking it is a window to the web which allows its user to view websites on the internet. Currently the most popular browser is Microsoft's Internet Explorer (IE) which has been the dominant browser since the late 90's.[WIKI12a]

## 2.2 What is an user interface?

An user interface is *“the part of a software application that a user sees and interacts with”*[WIKT12b] and has proven to be vital to a browser's success. In this context it will also be used as a synonym for a Graphical User Interface (GUI) which is *“a type of user interface which allows people to interact with a computer through a metaphor of direct manipulation of graphical images and widgets in addition to text.”*[WIKT12c]

A different kind of UI which does not belong to the group of GUIs is the command line UI. However, recent browser iterations have shown that it is possible to integrate a command line UI into a browser's GUI by extending the capabilities of the address bar:

## 2.3 What is user experience?

Lastly, it is important to define the term user experience: *“The desired, expected, or actual experience of a user interacting with a product, especially as it relates to the design of the product's user interface.”*[WIKT12d] UI and UX are highly dependent on each other. A great product has to have both a great UI and a great UX.

# 3 Conception

The concept for the prototype is based on a few number of ideas. Even small improvements can make a big difference in today's competitive browser market due to the need to diversify. Each browser has its own unique features and core values enabling it to differentiate itself from the competition.



While Mozilla's Firefox has been focusing on openness and the ability to add features to the browser through installing add-ons others have been focusing on different aspects of the browsing experience. Take for instance RockMelt: It has the same underlying rendering engine as Google's Chrome and Apple's Safari and was built on top of Chromium – Chrome's open source equivalent. However, it differentiates itself from all the others by focusing on integrating features which enable its users to better organize their social lives on Facebook, Twitter and Co.

## **3.1 Ideas**

The ideas for this particular prototype are partly based on current trends and/or problems with current browsers.

### **3.1.1 The browser in the background**

One of the results of the previous bachelor thesis was the understanding that browsers are continuously moving to the background when it comes to the actual browsing experience. Some browser vendors are currently trying to merge the browser with the underlying operating system. Examples are Mozilla's Boot2Gecko, Google's ChromeOS and Microsoft's Windows 8 approach which might not be as radical in this regard as its competitor's but it is a similar direction in which they are moving.

While those examples appear to be relatively obvious, others are not. It is not too far-fetched to imagine a browser that has no graphical user interface at all. If you take Apple's Siri – „*an intelligent assistant that helps you get things done just by asking*“[APPL11] – you see that such experiments are already underway and that they could easily be transformed into actual browsers being integrated into any device on the market – whether it has a screen or not.

Other approaches originated from the shift to mobile which is currently happening. Browsers on mobile devices are limited by the device's screen real estate. Due to that limitation browser vendors tend to hide most UI elements that a desktop browser usually shows.

Considering all of that it makes sense to design a browser around the promise of showing the user more of the website he or she is looking at and less of the browser's own UI elements.

### 3.1.2 A different approach to bookmark management

A bookmark (or “favorite” as they are called in Internet Explorer) generally consists of a title and an Uniform Resource Locator (URL) and is used to locally store an internet address. All major browsers follow the approach of managing bookmarks in an hierarchical structure similar to an operating system's directory structure by grouping them into directories and displaying them in a bookmark bar beneath the browser's address bar.

The basic idea of bookmark management has come up in 1993 when the National Center of Supercomputing Applications (NCSA) added this feature to their Mosaic browser.[DEJA93] Since then not much has changed even though there were efforts from companies like Delicious and Xmarks trying to make bookmarks “more social”. Both Delicious and Xmarks were almost discontinued until other companies saved those products by acquiring them. [ARCH12a][XMAR10]

The current approach to bookmark management does what it is supposed to do: giving the users control over how they want to organize their bookmarks. However, that promise fails as soon as a user reaches a certain amount of bookmarks. At that point the users are being forced to dedicate their time into maintaining their bookmarks which end up in not caring about structuring them anymore and saving them into a single folder. That behavior eventually leads to a bad UX, not only due to the lost structure but also due to the time it takes the browser to load all of them on its initial start.

Mozilla created a concept to tackle similar problems: losing interest in managing tabs and the consequence of losing overview over open tabs. They called it “Tab Candy” and it is now integrated into Firefox under the name “Tab Groups” (see Figure 1). On Mozilla's Wiki page its workings are described as follows: “*With one keystroke*

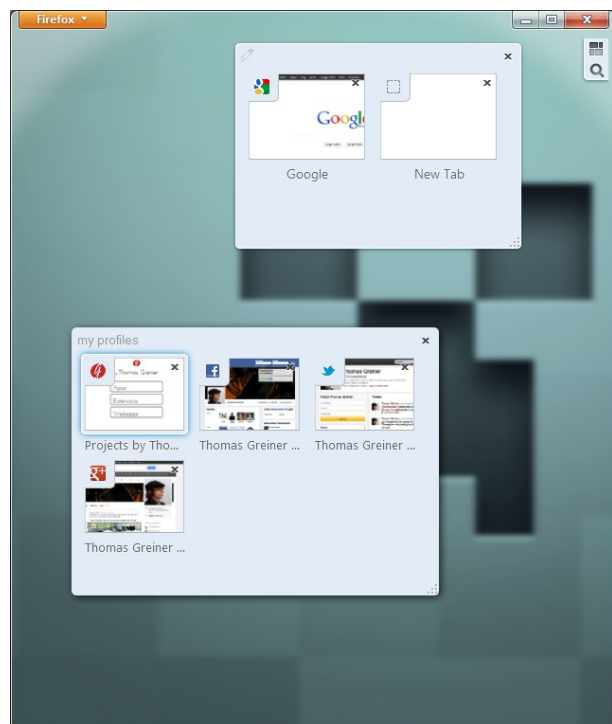


Figure 1: Tab Groups feature in Firefox 12.0

*Tab Candy shows an overview of all tabs to allow you to quickly locate and switch between them. Tab Candy also lets you group tabs to organize your work flow.* [MOZ12a]

Looking at a different market might help in this case. The market which might be interesting to look at in this particular problem is the search engine market. In the mid 90's Google tried to revolutionize the market and succeeded. With a market share of 91.7% [STC12a] it is now the most popular search engine worldwide. Their sudden success was due to the PageRank algorithm that Larry Page (current Chief Executive Officer (CEO) of Google) came up with to find the most relevant search results for a given query.

The theory behind it is simple: the more reputable sites link to you, the more reputable your site becomes. This relationship flow can be visualized by creating a node graph in which each node is a unique webpage and each edge represents the PageRank which flows from one node to another. The closer two nodes are together or the thicker the edge connecting those two nodes is the more PageRank flows between those nodes.

When reviewing the bookmark management problem we can use this visualization technique and the underlying algorithm to create a relevancy-based bookmark management system. This can be achieved by the fact that at the core of the PageRank algorithm every single webpage can be represented by a URL – or in this particular case: a bookmark.

### **3.1.3 Focus on navigation**

Depending on the time users want to invest in it they can literally modify Firefox anyway they want. It lets the user remove primary UI elements such as the bookmark bar, the address bar, the reload button and the back button or place new elements like a boss key by adding the proper extension (or “add-on” as it is called in Firefox).

The first version of Firefox showed the following UI elements by default: [EOTW12]

- title bar
- menu bar
- back button
- forward button
- reload button
- stop button
- “go to homepage” button
- address bar
- “go to page” button
- search box
- bookmark bar
- status bar

Those are very similar to the UI elements other browsers present the user with. However, what if someone were to take that approach of a plug-and-play UI but reducing the default to a bare minimum making even core features be nothing more than a simple extension? For example this could be achieved by combining similar or redundant UI elements. This approach has already been implemented to some extent by major browsers like Chrome, Firefox, Opera and Safari.

The reload button and the stop (stop loading page) button have been merged into a single button which is by default in the reload state unless when a page is currently loading. In that case it switches to the stop state and as soon as the site finished loading it will return to the reload state.

Another example are the address and search bar whereas on one users could go to a website and on the other users could search for websites. Chrome was the first browser to merge those two with their so-called Omnibox which automatically differentiates search terms from URLs and other queries by the user.

Another way to reduce UI elements to a bare minimum is to find different ways in which the features behind those could be used without taking up any more valuable space on the screen. Gestures are a popular way to accomplish that on mobile devices. Browsers like Fenrir's Sleipnir make it possible to reload, go back, go forward and more without the need to have any UI elements. However, they did not get rid of all the UI elements and only provide gestures as a supplemental feature. While gestures can be an excellent way to improve the UX they lack accessibility. To use a gesture you need to know/remember what the gesture for a specific action is and the software needs to be tolerant enough to detect and recognize those gestures but at the same time it also needs to be strict enough to differentiate between multiple different gestures to avoid unintentional behavior and confusion.

Not all UI elements are needed all the time. Therefore the browser could react to some specific behavior from the user and show only those UI elements which it thinks could be useful at a given moment. However, browser vendors would need to be very careful on one side not hide too much features from the user or at least give him more choice in this regard and on the other side keep the browser's overall UI as static as needed. Constantly hiding and showing UI elements might potentially lead to confusion for the user and a bad UX.

## 3.2 Environment

This work focuses on the narrow task of building a prototype for a modern browser's UI and therefore excludes other components a browser usually has. Therefore it is necessary to mention what possibilities there currently are and to find out which one might be the most appropriate for implementing this prototype.

### 3.2.1 Rendering engine

A rendering engine is a crucial component in every browser which displays the requested content from the internet in the browser.[H5R11] Over the years many different rendering engines arose including Internet Explorer's *"Triton"*, Firefox' *"Gecko"* and Chrome's and Safari's *"WebKit"* engine. The multiple years-long prevalence of IE 6 plus the fact that each rendering engine displays some pages on the web differently has led to the need for cross-browser optimization.

In recent years cross-browser optimization has been losing importance – mainly due to the rise of WebKit which stems from two sides: the introduction and popularity of Chrome and the rise of the mobile web with Apple's iOS and Google's Android operating system. Both operating systems' primary browser (Mobile Safari on iOS and Android's proprietary browser) use WebKit.

This has led to problems for non-WebKit browsers such as Opera and Firefox. Apart from losing market share[STC12b] they are having trouble with vendor-prefixes that came with the introduction of experimental features of HTML5 and CSS3 (Cascading Style Sheets) (e.g. *-webkit-border-radius*). *"[...] we have experienced that many authors of (especially mobile) sites only use -webkit- prefixed CSS, thereby ignoring other vendor prefixes and not even including an unprefixed equivalent."*, explains Opera's Bruce Lawson.[OP12]

Based on those facts the best choice for a modern browser's rendering engine is WebKit. Currently, it appears to be the best one too when it comes to both user and developer acceptance.

### 3.2.2 JavaScript interpreter & other components

At the moment the most popular open-source browsers are Mozilla's Firefox and Chromium which is the open-source version of Google's Chrome. Given those options and having chosen WebKit as the rendering engine it is a given that one should pick Chromium as the underlying basis for a modern browser having the positive side effect of being compatible with all existing Chrome extensions and apps.

Nevertheless, it is necessary to look at some of the most relevant pro's and con's of each of those.

Firefox was not the first browser to introduce extensions (or add-ons) but it popularized the idea of extending the browser's functionality without writing native code like plug-ins do. [WIKI12b] But since the arrival of Chrome the landscape changed to limiting the capabilities of extensions to a bare minimum and to extension development using web standards such as HTML, CSS and JavaScript.[WIKI12c] Later on other browsers such as Safari and Opera followed the trend.[APPL10][OP10] Additionally, Microsoft announced HTML5-based applications on devices running Windows 8.[MS11]

This leads to a different part of the browser which is essential for HTML5-based applications in the browser: the JavaScript engine (or JavaScript interpreter). When it was first introduced Chrome's V8 engine was superior to its competition[FAV09] but since then the competing browsers caught up and are now faster than ever[FAV12].

### 3.2.3 Other technologies

In addition to those components it is desirable to integrate new and upcoming web technologies which seem to be fitting more tightly into the UI to improve the UX. One of those technologies – as mentioned in the prior work – is the concept of Web Intents which allow web applications to communicate and share information with each other by using a standard way on how to do it.

In current implementations the browser simply acts as a hub which handles the communication between those applications.[CR12] However, one could leverage the usefulness of Web Intents by making the browser directly interact with those applications and act like a web application itself.

Possible use cases include authentication through single sign-on and accessing and searching resources across multiple different sources right from within the browser.[WI11]

### 3.3 Mockup

Prior to the actual development of the prototype for this concept I sat down, took a piece of paper and a pen and sketched out a rough draft on what it could look like. (see Figure 2)

Figure 2 shows the primary UI elements of the main view which are as follows:

- (a) a single navigation button to provide the most basic navigation features based on the current page and context
- (b) the address bar to enter a URL and to display additional information about the current page
- (c) optional UI elements that can be added via extensions
- (d) scrollable list containing all tabs that are currently open
- (e) currently displayed webpage
- (f) scrollable list containing webpages that are stored locally and can be accessed offline

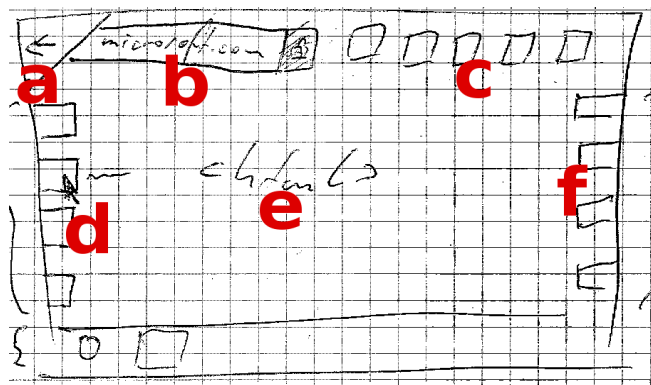


Figure 2: drawing of main layout

Not highlighted in this illustration is the bar at the bottom which represents the operating system's own task bar.

The next one shows my take on the bookmark management system (see Figure 3):

- (a) (see Figure 2/d) an element from this list can be slid out and dragged onto the bookmark management area
- (b) a cluster of similar/related webpages
- (c) a node in the graph which represents a specific webpage
- (d) a link between two different webpages (not necessarily a hyperlink)

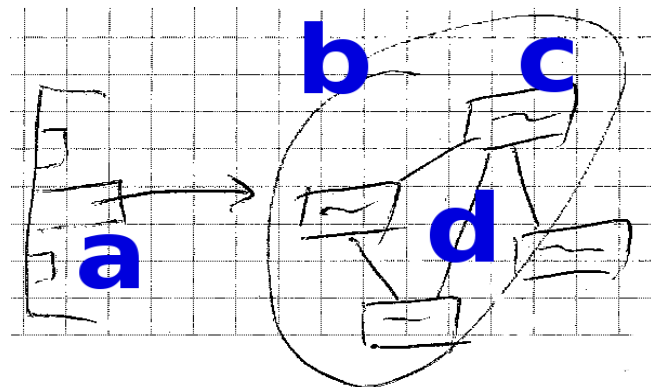


Figure 3: drawing of bookmark management concept

## 4 Prototyping

Based on the mockups I started working on the actual prototype. Due to my extensive experiences with writing extensions and apps for Chrome I decided to go with that to achieve quick and efficient development.

The second decision which had to be made prior to coding was which kind of extension/app it should be. There are three kinds of applications: extensions, packaged apps and hosted apps.

Hosted apps are webapps with some meta data attached required for the listing in the Chrome Web Store – Chrome's very own online directory for finding all kind of apps, extensions and themes for Chrome. Extensions are on the other side of the spectrum. They are stored and run locally but have access to Chrome's application programming interface (API) which enable them to do tasks like for instance manage bookmarks, search the browsing history and inject scripts into websites – provided that the user has given it the permission to do so. Unlike apps extensions run in the background most of the time.

For apps which require access to Chrome's API there is also a third category: Packaged apps are apps which – like webapps – deliver a full screen experience but additionally to that are able to make API calls.[GDEV12]



A packaged app seemed to be the best choice due to the prototype's potential needs for making cross-site requests, accessing bookmarks and the browsing history and making screenshots of webpages while at the same time delivering a full-screen experience.

Based on this decision the prototype had to be made using client-side web technology such as HTML, CSS and JavaScript. Therefore it was impossible to create an experience the way it would be when implemented in a lower-level language such as C or C++ like real-world browsers are.

When it comes to the decision which tools to use for this task, it became clear that there was no need for additional tools other than those I used prior to this work, which are:

- Inkscape (vector graphics editor; free)  
<http://inkscape.org/>
- Notepad++ (text editor; free)  
<http://notepad-plus-plus.org/>
- Sublime Text 2 (text editor; shareware)  
<http://www.sublimetext.com/>
- Google Chrome 18-19 (browser; free)  
including WebKit Web Inspector (developer tools; free)  
<https://www.google.com/intl/en/chrome/>

## 4.1 File structure

I decided to organize the individual scripts in a certain structure to provide a good overview:

- |   |   |
|---|---|
| <ul style="list-style-type: none"><li>• <code>/</code> (root directory)<ul style="list-style-type: none"><li>◦ <code>main.htm</code><br/>entry point for the application</li><li>◦ <code>manifest.json</code><br/>contains permissions and other meta information for Chrome</li><li>◦ <code>start.htm</code><br/>page that is shown on startup</li></ul></li></ul> | <ul style="list-style-type: none"><li>• <code>/dal</code> (data access)<ul style="list-style-type: none"><li>◦ <code>Database.js</code><br/>provides access to Chrome's Web SQL database</li></ul></li><li>• <code>/logic</code> (business logic)<ul style="list-style-type: none"><li>◦ <code>History.js</code><br/>used for back and forward navigation</li></ul></li></ul> |
|---|---|

- *Main.js*  
initializes application and provides basic methods
- *Navigator.js*  
used for navigation via URL
- *Renderer.js*  
provides access to certain color values of displayed page
- */res* (resources)
  - */res/fonts*  
contains fonts used by application
    - *lvnm.ttf*
    - *lvnmbd.ttf*
  - */res/images*  
contains images used by application
    - *logo.svg*
    - *logo16.png*
    - *logo64.png*
    - *logo128.png*
- */script* (generic scripts)
  - *HTMLElement.js*  
adds functions related to HTMLDivElement elements
- */ui* (user interface)
  - *Addressbar.js*  
renders address bar and defines event handlers
  - *BackgroundCreator*  
renders background of a HTMLDivElement depending on displayed page
  - *Cluster.js*  
renders a cluster which is a node representing a group of nodes
  - *Header.js*  
renders the header element
  - *Movable.js*  
provides the basic functionality for nodes and clusters to be movable
  - *Navigation.js*  
renders the main navigation element
  - *Node.js*  
renders a node
  - *Tabbar.js*  
renders a list of tabs
  - *Taskbar.js*  
renders a list of tasks
  - *Toolbar.js*  
renders a list of UI elements provided by extensions
  - *ToolbarButton.js*  
renders one UI element
  - *Visualizer.js*  
*renders the Visualizer and provides methods to interact with it*

## 4.2 Basic functionality

Hypercube – so the name of the prototype – focuses on simplicity and a clear UI. It has a modern look to it which is defined by its icon portraying a four-dimensional cube (also known as a tesseract or hypercube) that represents the interlinked nature of the web.

As soon as the application is installed, Chrome creates an app icon on its *New Tab Page* (*chrome://newtab*) from which it can be launched. Upon launch the *main.htm* file will be displayed and *Main.init* will be called and initializes all required UI elements such as the address bar and the Visualizer and listens to certain events to control the progress bar and the background rendering. (see Figure 4)

The prototype allows the user then to either enter a URL into the address bar, navigate to a predefined page by clicking on the tab in the tab bar or open the Visualizer by dragging or long-clicking any tab in the tab bar.

Navigating to a page in the prototype means setting the *src*-attribute of the *iFrame* (inline frame) to display the specified webpage. However, the absence of a rendering engine causes some problems which are listed below:

- The *HTMLIFrameElement* interface does not provide enough events to create a full experience and therefore the background can only be rendered after the *iFrame*'s onload event has been dispatched.

Because of that the header's and tab bar's background and the page appear to be out of sync while the page is being loaded.

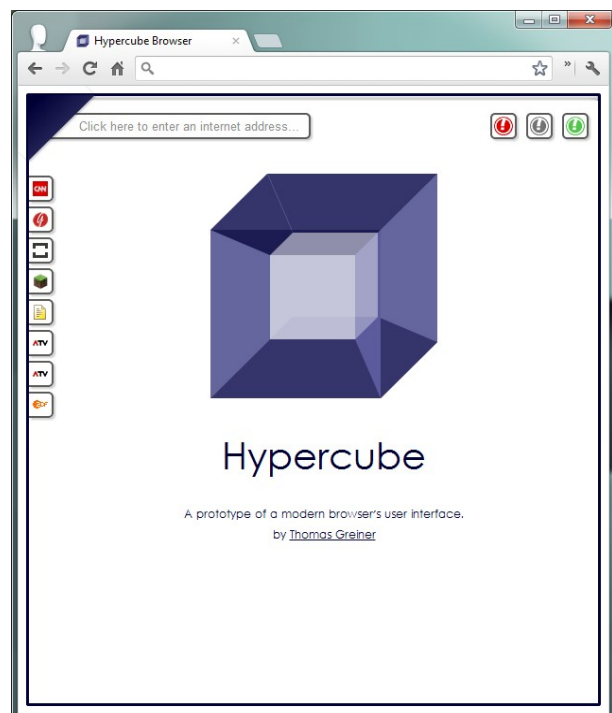


Figure 4: prototype running as a Chrome app displaying the start page

- Although the application has all permissions needed to access resources from the page it is unable to display pages in the iFrame from servers which send the *x-frame-options* HTTP (Hypertext Transfer Protocol) response header.[MOZ12b] Therefore the prototype is unable to display websites from domains such as *facebook.com*, *google.com* and *twitter.com*.

Coming back to the ideas expressed in chapter 3.1 of this work, it is important to look at how those ideas have been executed through the mockups that were made (see chapter 3.3) and in what way they have could be integrated into the Chrome app environment.

## 4.3 Dynamic chrome

As described in chapter 3.1.1 browsers tend to be more and more in the background while at the same time websites are being promoted into the foreground. However, as mentioned earlier, alternative user interfaces such as gestures have not yet proven to be utterly successful on desktop computers. Due to that browsers up to this day take up precious screen real estate by adding indispensable UI elements such as the address bar. They do that by placing an area at the top which contains all those elements.

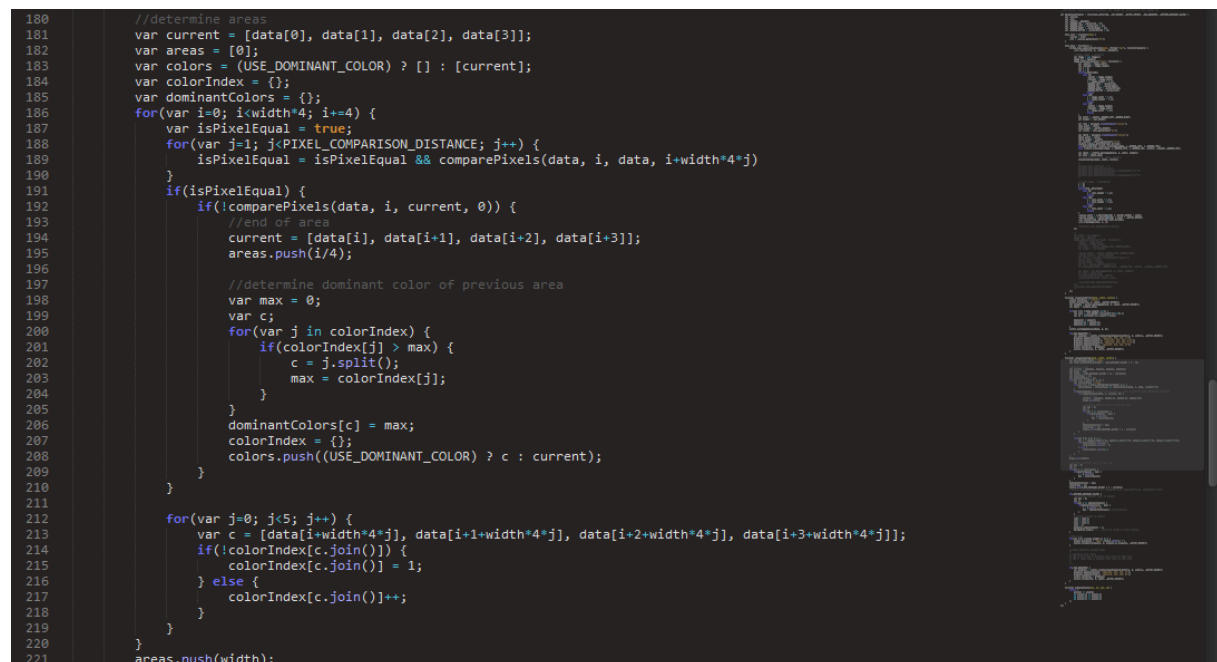


Figure 5: Hypercube's dynamic header as shown on *cnn.com* (top) and *ifttt.com* (bottom)

There is currently no generally accepted way of getting rid of that area which means that it needs to be hidden somehow to pretend to the user that it is a part of the webpage but without worsening its usability. This can be achieved through implementing an algorithm which scans the webpage and renders the header area's background in the exact same

pattern of the one of the actual page. (see Figure 5) In that case other UI elements need to be as minimalistic as possible such as the loading bar near the top edge of the browser window.

In the prototype this technique is also being used for the tab bar on the left edge of the screen to extend the page not only vertically but also horizontally. The implementation of that algorithm can be found in the *BackgroundCreator.js* file. It contains the *BackgroundCreator* function which provides a *draw* method that can be called from outside to paint on the specified *HTMLCanvasElement*. First, it makes an API call to Chrome to capture the visible tab and then crops the image to get the area in which the webpage is being displayed. This image is then passed on to the *visualizeAreas* method which scans the first line of that image and compares it to the pixel that lies *n* lines below it (where *n* is defined by the *PIXEL\_COMPARISON\_DISTANCE* variable). If they have the same color it is being saved in an array and its location is being saved in a different one. As soon as the scan encounters a pixel which does not have the same color value as the one encountered earlier it will check all colors that have appeared since then and find the one which occurred the most. This color represents the dominant color of that specific area and is being saved in a separate array. (see Figure 6)



```
180 //determine areas
181 var current = [data[0], data[1], data[2], data[3]];
182 var areas = [0];
183 var colors = (USE_DOMINANT_COLOR) ? [] : [current];
184 var colorIndex = {};
185 var dominantColors = {};
186 for(var i=0; i<width*4; i+=4) {
187     var isPixelEqual = true;
188     for(var j=1; j<PIXEL_COMPARISON_DISTANCE; j++) {
189         isPixelEqual = isPixelEqual && comparePixels(data, i, data, i+width*4*j)
190     }
191     if(isPixelEqual) {
192         if(!comparePixels(data, i, current, 0)) {
193             //end of area
194             current = [data[i], data[i+1], data[i+2], data[i+3]];
195             areas.push(i/4);
196
197             //determine dominant color of previous area
198             var max = 0;
199             var c;
200             for(var j in colorIndex) {
201                 if(colorIndex[j] > max) {
202                     c = j.split();
203                     max = colorIndex[j];
204                 }
205             }
206             dominantColors[c] = max;
207             colorIndex = {};
208             colors.push((USE_DOMINANT_COLOR) ? c : current);
209         }
210     }
211
212     for(var j=0; j<5; j++) {
213         var c = [data[i+width*4*j], data[i+1+width*4*j], data[i+2+width*4*j], data[i+3+width*4*j]];
214         if(!colorIndex[c.join()]) {
215             colorIndex[c.join()] = 1;
216         } else {
217             colorIndex[c.join()]++;
218         }
219     }
220 }
221 areas.push(width);
```

Figure 6: part of visualizeAreas method in BackgroundCreator.js showing the scan process

After the scan the algorithm loops through the array that contains the areas and draws a rectangle onto the specified *HTMLCanvasElement* from the start to the end location of the area. The dominant color of that specific area becomes the background color of the rectangle. It does that for each area in the array and overlays a gradient which goes from white to transparent afterwards.

Finally, it determines the overall dominant color by comparing the number of occurrences of all dominant colors that have been found earlier and sets the *Renderer's* *dominantColor* attribute. This allows the *Renderer* to calculate that specific color's brightness on a scale from 0 to 255 where 0 stands for dark and 255 stands for light. (see Figure 7)

```
8   this.__defineGetter__("dominantColor", function() {
9       return (_dominantColor) ? _dominantColor : [255, 255, 255, 1];
10  });
11  this.__defineSetter__("dominantColor", function(value) {
12      _dominantColor = value;
13  });
14
15  this.__defineGetter__("dominantColorBrightness", function() {
16      if(_dominantColorBrightness) {
17          return _dominantColorBrightness;
18      }
19
20      //calculate brightness
21      var color = this.dominantColor;
22      _dominantColorBrightness = Math.sqrt(0.241*Math.pow(color[0],2) + 0.691*Math.pow(color[1],2) + 0.068*Math.pow(color[2],
23          2));
24      return _dominantColorBrightness;
25  });
```

Figure 7: part of *Renderer.js* showing calculations for determining the brightness of the dominant color

Those values are then used by other components such as the *Navigation* object to render what else needs to be rendered.

Although this algorithm does the job of conveying the idea of making a webpage part of the browser UI, it is not perfect. A real world implementation could use more sophisticated techniques to extrapolate the background and to do the rendering. Another difference between the prototype and such an implementation comes to light when the page is being scrolled: The prototype keeps the background image unchanged and unmoved. Ideally, the canvases which extend the webpage's background should be prepended to the webpage making it an actual part of the webpage. By doing that it would be moving together with the webpage when the user scrolls.

Additionally, the canvases should be repainted as soon each time a relevant part of the webpage changes which can be caused by events like the loading of the page or visible HTML elements being added or removed from the DOM (Document Object Model) tree.

## 4.4 The Visualizer

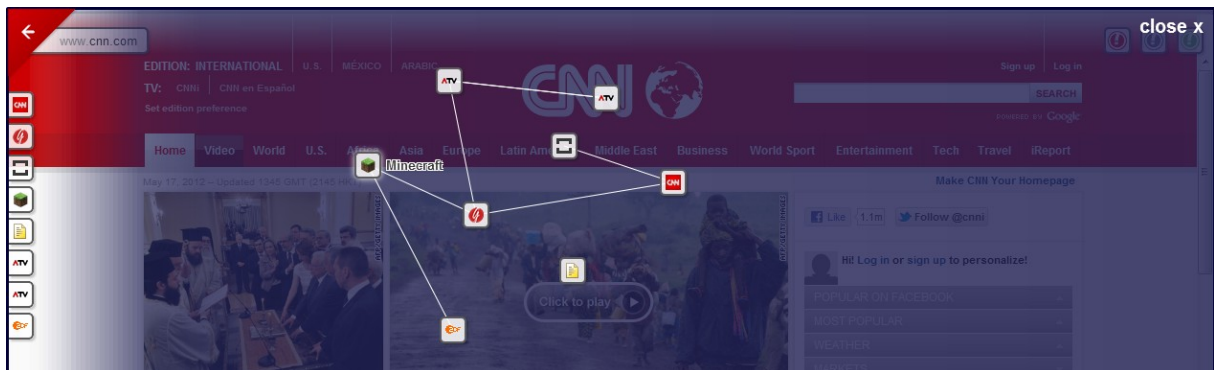


Figure 8: Visualizer UI with a simple node graph

The Visualizer (see Figure 8) is the core element of this prototype. Its main objective is to tackle the problems of current bookmark management solutions as mentioned in chapter 3.1.2 of this work. When the Visualizer is displayed it overlays itself over the current webpage and shows a graph of all nodes (similar to bookmarks) and clusters (similar to bookmark directories).

Each node is represented by a *HTMLDivElement* and an associated JavaScript object containing all relevant information about it. They are placed on top of a *HTMLCanvasElement* on which all connections between those nodes are being rendered. The connections are determined randomly and are added, removed or rerendered when the nodes change.

The nodes are stored locally in a Web SQL database because it turned out that the Indexed Database API (IndexedDB) – which is the official recommendation of the World Wide Web Consortium (W3C) – has shown that it was not yet ready for that task. However, it is only a matter of time until Chrome's implementation of IndexedDB stabilizes due to Web SQL no longer being actively maintained. [W3C10]

Due to the Visualizer being a highly visibly UI element there needed to be a quick and intuitive way of opening it. If the user starts dragging a tab from the tab bar the Visualizer shows up and the user can drag the tab onto the Visualizer to add it as a node to the graph. An alternative way of opening it in the prototype is by long-clicking a tab in the tab bar.

A more traditional way of doing that is to have an additional UI element. I decided not to add it to the prototype to emphasize the simplicity of the UI and I recommend to undertake further studies prior to real-world implementation.

Furthermore it is important to mention that the node presentation is for the purpose of this work only and should be seen as temporary. It should be improved in regard to integration with the tab bar due to it not being optimized for it.

Lastly, it requires a powerful algorithm which determines the relationship between nodes and which renders the graph that could potentially include many factors as mentioned in chapter 3.1.2.

## 4.5 Extendable, dynamic UI

Chapter 3.1.3 elaborated on making the UI dynamic and extendable. Those characteristics are crucial for the success of such a minimalistic design. Crafting a personalized UI through giving the user the ability to drag and drop an UI element is a key element to this approach.



Figure 9: prototype's header element with navigation element being a back button and three extension icons (placeholders)

Examples for such elements are the buttons in the upper-right corner of the prototype UI acting as mere placeholders for possible browser extensions. (see Figure 9) Implementations of this approach can be found among others in Mozilla Firefox and the VLC media player.

One UI element worth mentioning when talking about the dynamic nature of the UI is the navigation button located in the upper-left corner. It is an element which is meant to be very flexible and powerful. While in this prototype implementation its single purpose is to represent a back button its main purpose is to simplify overall navigation.

This led to it being implemented as an *HTMLCanvasElement* that is rotated 45° counter-clockwise allowing it to be very dynamic regarding its appearance and functionality. Not only does it change its color when navigating to a different page but it also reacts to mouse movements which can be seen in other software like Chrome when moving the mouse over a tab or Windows 7 when moving the mouse over an open application in the task bar.



## **4.6 Left out components**

Unfortunately some components mentioned in the concept had to be omitted when creating the prototype stemming from the limited time and the time consuming nature of UI and UX development.

The set of features of the Visualizer can be much greater than it is in the prototype by adding UI elements for common tasks such as removing nodes or adding menus for browser settings.

A feature that has been left out is the task bar which in most regards is very similar to the tab bar in the prototype. But while the tab bar is a temporary list of open tabs, the task bar contains tabs which are stored completely offline. Its main purpose is to provide users with an additional way of managing open tabs and to give them a way to read something at a later point in time.

Probably the most important feature of those not being in the actual prototype is the breadcrumb navigation. It is an automatically created list of webpages located between the loading bar and the address bar. Depending on the user's current session, his or her navigation behavior and on the Visualizer's graph it only shows the most relevant sites which the user might want to go to next.

Take for example a user who regularly visits facebook.com, twitter.com and plus.google.com. Upon visiting twitter.com Hypercube could suggest facebook.com and plus.google.com to him or her by adding them to the breadcrumb navigation which the user can then click on to navigate to those websites. The decision whether or not to click on one of those breadcrumbs helps the browser learn about how the user wants the browser to behave by either keep showing those breadcrumbs or not.

## **5 User acceptance**

After finishing the prototype it was necessary to gather some user feedback by undertaking an evaluation on how actual internet users react to the concept explained in this work. For this purpose I made appointments with each one of the four participants to talk about it either personally or via Skype.

In advance to those interviews I set up a questionnaire (see Appendix A). It includes demographic questions, questions to find out how experienced that person is in regard to browsers and the web and an interactive part after which I showcased the prototype and explained the concept behind it. At the end of the questionnaire there are some questions for comparing Hypercube with already existing browsers regarding categories such as “focus on website”, “bookmark management” and “extendability”.

The results (see Appendix G) show that all participants had different priorities and experiences when it comes to browsing the web which led to a wide variety of answers.

Answers to questions 3.1, 3.4, 4.2 and 4.3 make clear that the prototype's “dynamic chrome” does what it is supposed to do: it makes the user think that the website takes up the entire screen. However, three of the four participants were missing at least one UI element on the main UI as covered in question 3.3. The overall response to the dynamic nature of the browser's UI was positive.

The Visualizer was well received by three of the four participants with two participants expressing uncertainty regarding whether or not it could work as pictured (question 3.5). For one of them it seemed too mathematical and overwhelming.

Finally, the extendable, dynamic UI component of the concept was described as a “hygiene factor” in browsers with two of them requesting a way to organize extensions' UI elements by either grouping them, hiding them or marking them as important or unimportant. In comparison to current browsers the participants did not think that the concept is in any way superior due to some important UI elements not being shown right from the beginning such as the “forward button” and the “reload button”.

Apart from feedback to the ideas expressed in the concept they highlighted another important aspect of the browsing experience which has barely been touched in this work. As mentioned in a previous chapter the look and feel of a tab in the concept has not been fully specified yet. Therefore answers to question 3.5 in which the participants were asked about the disadvantages of the concept three of them criticized either tab representation or tab handling.

Questions 3.6, 3.7 and 3.7.1 focused on a possible real-world implementation of one or more ideas contained in the concept. Half of them think that it can only be done by creating a new browser while the other half can imagine them being integrated or based on either Firefox or Chromium. Being asked whether or not they would use a browser which adapts those ideas

half of them answered with a quick “yes” on condition that it works exactly as advertised while being resource-efficient and performant. The others are indecisive but willing to give it a chance.

Finally, I showed three of them a screenshot (see Appendix L) of Hypercube running as a web application in the browser of a mobile device which was exactly the same version as the one running in Chrome on the desktop without any additional small screen optimizations. The response was unanimously positive.

## 6 Conclusion

The browser market is a highly competitive one ever since Netscape has introduced its Navigator browser in 1994[ARCH12b] and browser vendors fight hard for their respective browser to stay relevant. It is especially difficult for small companies to get people to use the browser they created but history has shown that UI simplicity and the ambition and goodwill of the company behind it can lead to drastic changes within the browser market.[GR12]

This work's purpose is to spread ideas on how the next generation browser could look like. Over the course of creating the concept and developing Hypercube I analyzed state-of-the-art browsers to find similarities like address bars and back buttons and singularities such as Firefox' highly customizable UI trying to combine them to create a prototype which is as close to reality as feasible under the given circumstances and limitations.

This work proved that it is possible to create a prototype based on the concept mentioned in this work which – if done right – could potentially improve a browser's UX. Those results are backed by an user acceptance evaluation which provided feedback on both strengths and weaknesses of the concept and suggested that ideas such as a dynamic UI can theoretically improve the actual browsing experience. The success of any real-world implementations, however, does depend on other factors influencing the usefulness of the concept.

Though accessibility, usability and UX are important when thinking about integrating ideas such as those expressed in this work into an existing project, forcing it is not the right solution because it needs to be consistent with the rest of the UI, UX and with the overall message the project conveys.

Retrospectively, I recognized the potential of this concept when being ported over to a mobile environment due to it being highly visual and not making use of any non-mobile specific features e.g. the right mouse button (see Appendix L). The user acceptance evaluation that was undertaken in the course of this work hinted that it may be worth doing further research in this regard. Maximizing the accessibility appears to be an important factor when creating something as universal as a browser to navigate the web.

Steve Jobs' main demand when it came to producing the iPod was *"Simplify!"*[SJ11a] which is a vital part of Apple's product strategy that allowed it to become the world's most valuable company.[CHM12] *"Jobs had aimed for the simplicity that comes from conquering complexities, not ignoring them."*, wrote Walter Isaacson in his book about the life of Steve Jobs.[SJ11b] A minimalistic design proves to be a step in the right direction regarding accessibility and responsiveness of the UI.

Actually, the most valuable experience that I gained from working on this prototype is that simple ideas and small changes in the UI can have a big impact on the UX.

## Bibliography

[APPL10] Apple Inc., "Apple Releases Safari 5", Apple Inc. 2010, [online]. Available: <http://www.apple.com/pr/library/2010/06/07Apple-Releases-Safari-5.html> [Accessed: 18.05.2012]

[APPL11] Apple Inc., "Apple Launches iPhone 4S, iOS 5 & iCloud", Apple Inc. 2011, [online]. Available: <http://www.apple.com/pr/library/2011/10/04Apple-Launches-iPhone-4S-iOS-5-iCloud.html> [Accessed: 18.05.2012]

[ARCH12a] Delicious, "What's Next for Delicious?", Internet Archive, 2012, [online]. Available: <http://web.archive.org/web/20101217235131/http://blog.delicious.com/blog/2010/12/whats-next-for-delicious.html> [Accessed: 18.05.2012]

[ARCH12b] Netscape Communications Corporation, "Netscape Communications Now, Builds on Tradition of Freeware for the Net", Internet Archive, 2012, [online]. Available: <http://web.archive.org/web/20030621062544/http://wp.netscape.com/newsref/pr/newsrelease1.html> [Accessed: 18.05.2012]

[CR12] James Hawkins, "Connect with Web Intents", The Chromium Projects, 2012, [online]. Available: <http://blog.chromium.org/2012/05/connect-with-web-intents.html> [Accessed: 18.05.2012]

[DEJA93] Marc Andreessen, "New X-based information systems browser available.", National Center for Supercomputing Applications, 1993, [online]. Available: <http://www.dejavu.org/mosnew.html> [Accessed: 18.05.2012]

[EOTW12] Google, Hyperakt and Vizzuality, "The evolution of the web", [evolutionofweb.appspot.com](http://evolutionofweb.appspot.com), 2012, [online]. Available: [http://evolutionofweb.appspot.com/img/firefox/firefox\\_1.jpg](http://evolutionofweb.appspot.com/img/firefox/firefox_1.jpg) [Accessed: 18.05.2012]

[FAV09] Vygantas, "Chrome 2, Webkit 4 and Firefox 3.1 beats IE 8 and Opera 10", FavBrowser.com, 2009, [online]. Available: <http://www.favbrowser.com/chrome-2-webkit-4-and-firefox-31b1-beats-ie-8-and-opera-10/> [Accessed: 18.05.2012]

[FAV12] Vygantas, "IE8 vs. Google Chrome 18 vs. Firefox 11 vs. Opera 11.6 vs. Safari 5.1", FavBrowser.com, 2012, [online]. Available: <http://www.favbrowser.com/ie8-vs-google-chrome-18-vs-firefox-11-vs-opera-11-6-vs-safari-5-1/> [Accessed: 18.05.2012]

[GDEV12] Google Inc., "Choosing an App Type", Google Inc., 2012, [online]. Available: <https://developers.google.com/chrome/web-store/docs/choosing> [Accessed: 18.05.2012]

[GR12] Thomas Greiner, "Browsers and their influence on the evolution of the web", greinr.com, 2012, [online]. Available: <http://www.greinr.com/bachelorthesis/browsers-and-their-influence-on-the-evolution-of-the-web> [Accessed: 18.05.2012]

[H5R11] Tali Garsiel and Paul Irish, "How Browsers Work: Behind the scenes of modern web browsers – HTML5 Rocks", HTML5 Rocks, 2011, [online]. Available: <http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/> [Accessed: 18.05.2012]

[MOZ12a] Mozilla Foundation, "Firefox/Projects/TabCandy/FAQ", Mozilla Foundation, 2012, [online]. Available: <https://wiki.mozilla.org/Firefox/Projects/TabCandy/FAQ> [Accessed: 18.05.2012]

[MOZ12b] Mozilla Foundation, "The X-Frame-Options response header", Mozilla Foundation, 2012, [online]. Available: [https://developer.mozilla.org/en/The\\_X-FRAME-OPTIONS\\_response\\_header](https://developer.mozilla.org/en/The_X-FRAME-OPTIONS_response_header) [Accessed: 18.05.2012]

[MS11] Julie Larson-Green, "Previewing "Windows 8"", Microsoft Corporation, 2011, [online]. Available: <http://www.microsoft.com/en-us/news/features/2011/jun11/06-01corpnews.aspx> [Accessed: 18.05.2012]

[OP10] Opera Software ASA, "Are you ready for Opera 11?", Opera Software ASA, 2010, [online]. Available: <http://www.opera.com/press/releases/2010/12/16/> [Accessed: 18.05.2012]

[OP12] Opera Software ASA, "Opera Mobile Emulator build with experimental WebKit prefix support", Opera Software ASA, 2012, [online]. Available: <http://dev.opera.com/articles/view/opera-mobile-emulator-experimental-webkit-prefix-support/> [Accessed: 18.05.2012]

[SJ11a] Walter Isaacson, "Steve Jobs: A Biography", New York: Simon & Schuster, 2011, [E-Book]. Available: Amazon Kindle Store, page 388, ISBN: 9781451648553

[SJ11b] Walter Isaacson, "Steve Jobs: A Biography", New York: Simon & Schuster, 2011, [E-Book]. Available: Amazon Kindle Store, page 343, ISBN: 9781451648553

[STC12a] StatCounter, "Top 5 Search Engines from Apr 2011 to Apr 2012 | StatCounter Global Stats", StatCounter, 2012, [online]. Available: [http://gs.statcounter.com/#search\\_engine-ww-monthly-201104-201204](http://gs.statcounter.com/#search_engine-ww-monthly-201104-201204) [Accessed: 18.05.2012]

[STC12b] StatCounter, "Top 5 Browsers from Jul 2008 to May 2012 | StatCounter Global Stats", 2012, [online]. Available: <http://gs.statcounter.com/#browser-ww-monthly-200807-201205> [Accessed: 18.05.2012]

[W3C10] World Wide Web Consortium, "Web SQL Database", World Wide Web Consortium, 2010, [online]. Available: <http://www.w3.org/TR/webdatabase/> [Accessed: 18.05.2012]

[WI11] Paul Kinlan, "webintents", webintents.org, 2012, [online]. Available: <http://usecases.webintents.org/> [Accessed: 18.05.2012]

[WIKI12a] Wikipedia, "Usage share of web browsers", Wikipedia, 2012, [online]. Available: [http://en.wikipedia.org/wiki/Usage\\_share\\_of\\_web\\_browsers](http://en.wikipedia.org/wiki/Usage_share_of_web_browsers) [Accessed: 18.05.2012]

[WIKI12b] Wikipedia, "Add-on (Mozilla)", Wikipedia, 2012, [online]. Available: [http://en.wikipedia.org/wiki/Add-on\\_\(Mozilla\)](http://en.wikipedia.org/wiki/Add-on_(Mozilla)) [Accessed: 18.05.2012]

[WIKI12c] Wikipedia, "Google Chrome Extensions", Wikipedia, 2012, [online]. Available: [http://en.wikipedia.org/wiki/Google\\_Chrome\\_Extensions](http://en.wikipedia.org/wiki/Google_Chrome_Extensions) [Accessed: 18.05.2012]

[WIKT12a] Wiktionary, "browser", Wiktionary, 2012, [online]. Available: <http://en.wiktionary.org/wiki/browser> [Accessed: 18.05.2012]

[WIKT12b] Wiktionary, "user interface", Wiktionary, 2012, [online]. Available: [http://en.wiktionary.org/wiki/user\\_interface](http://en.wiktionary.org/wiki/user_interface) [Accessed: 18.05.2012]

[WIKT12c] Wiktionary, "graphical user interface", Wiktionary, 2012, [online]. Available: [http://en.wiktionary.org/wiki/graphical\\_user\\_interface](http://en.wiktionary.org/wiki/graphical_user_interface) [Accessed: 18.05.2012]

[WIKT12d] Wiktionary, "user experience", Wiktionary, 2012, [online]. Available: [http://en.wiktionary.org/wiki/user\\_experience](http://en.wiktionary.org/wiki/user_experience) [Accessed: 18.05.2012]

[XMAR10] Bob Billingslea, "Xmarks: LastPass Acquires Xmarks!", Xmarks, 2010, [online]. Available: [http://blog.xmarks.com/2010/12/lastpass-acquires-xmarks\\_3493.html](http://blog.xmarks.com/2010/12/lastpass-acquires-xmarks_3493.html) [Accessed: 18.05.2012]



## List of Figures

Figure 1: Tab Groups feature in Firefox 12.0.....	8
Figure 2: drawing of main layout.....	13
Figure 3: drawing of bookmark management concept.....	14
Figure 4: prototype running as a Chrome app displaying the start page.....	17
Figure 5: Hypercube's dynamic header as shown on cnn.com (top) and ifttt.com (bottom)...	18
Figure 6: part of visualizeAreas method in BackgroundCreator.js showing the scan process .....	19
Figure 7: part of Renderer.js showing calculations for determining the brightness of the dominant color.....	20
Figure 8: Visualizer UI with a simple node graph.....	21
Figure 9: prototype's header element with navigation element being a back button and three extension icons (placeholders).....	22

## List of Abbreviations

API	Application Programming Interface
CEO	Chief Executive Officer
CSS	Cascading Style Sheets
DOM	Document Object Model
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IE	Internet Explorer
iFrame	Inline frame
IndexedDB	Indexed Database API
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience
W3C	World Wide Web Consortium

## A: Interview structure for user acceptance evaluation

### 1 Demographic

- 1.1 age
- 1.2 occupation

### 2 Experience

- 2.1 number of days per week actively on the internet
- 2.2 average number of hours per day actively on the internet
- 2.3 knows what a browser is
  - possible answers:
    - yes (can explain how to use a browser and what it does in the background)
    - basics (knows how to use a browser and that it displays websites)
    - no (does not know how to use a browser)
- 2.4 known browsers (ordered: 1 – mentioned first, 5 – mentioned last)
  - Internet Explorer
  - Firefox
  - Chrome
  - Safari
  - Opera
  - others
- 2.5 regularly used browser(s) (ranking starting at 1 - most used)
  - Internet Explorer
  - Firefox
  - Chrome
  - Safari
  - Opera
  - others

### 3 Concept

#### ***show first screenshot (Appendix H)***

- 3.1 can differentiate between browser's and website's elements (see Appendix B, Appendix C, Appendix D and Appendix E) and recognizes certain elements
  - back button
  - address bar
  - tab bar

- extensions area
- background (left)
- background (top)
- content area

3.2 is uncertain about certain elements

- back button
- address bar
- tab bar
- extensions area
- background (left)
- background (top)
- content area

***show second screenshot (Appendix I)***

3.3 misses elements

***show third screenshot (Appendix J) and fourth screenshot (Appendix K)***

3.4 advantages of the concept

3.5 disadvantages of the concept

3.6 best way of doing a real-world implementation

- integrated with or based on an existing browser
- as a new browser

3.7 favor over other browsers

3.7.1 condition

## **4 Dynamic chrome**

4.1 personal priority

4.2 benchmark: highlighting of web content

- better
- even
- worse

4.3 benchmark: screen space usage

- better
- even
- worse

4.4 comments

## **5 The Visualizer**

5.1 personal priority

5.2 benchmark: clearness of structure

- better
- even
- worse

5.3 benchmark: assistance with bookmark management

- better
- even
- worse

5.4 comments

## **6 Extendable, dynamic UI**

6.1 personal priority

6.2 benchmark: integration of extensions in browser UI

- better
- even
- worse

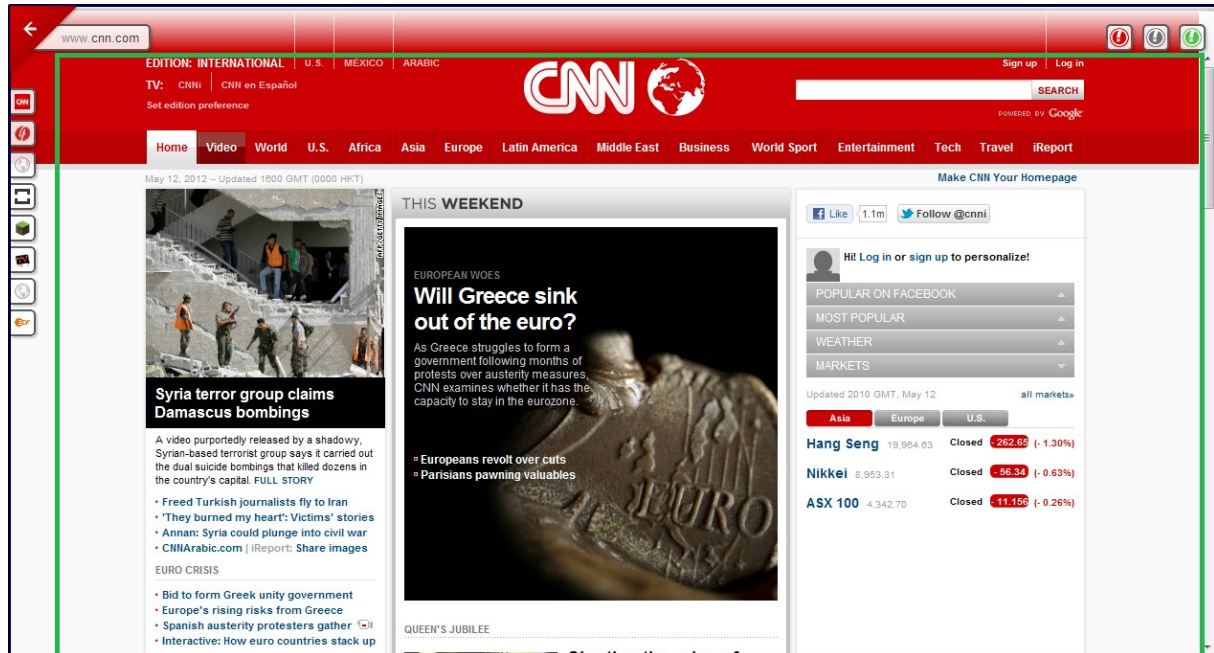
6.3 comments

## **7 Bonus**

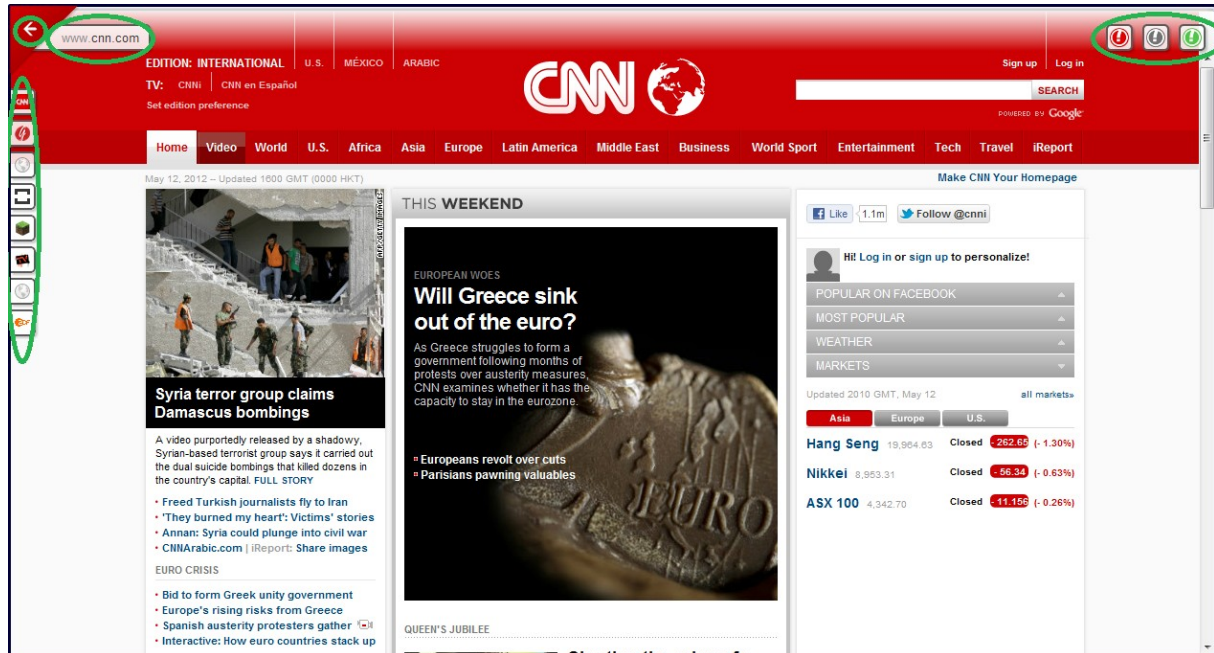
***show fifth screenshot (Appendix L)***

7.1 reaction to seeing Hypercube UI on a mobile device

## B: Differentiation of UI elements (Participant A)



## C: Differentiation of UI elements (Participant B)



## D: Differentiation of UI elements (Participant C)





## E: Differentiation of UI elements (Participant D)



## F: Differentiation of UI elements (solution)



## G: Results of user acceptance evaluation

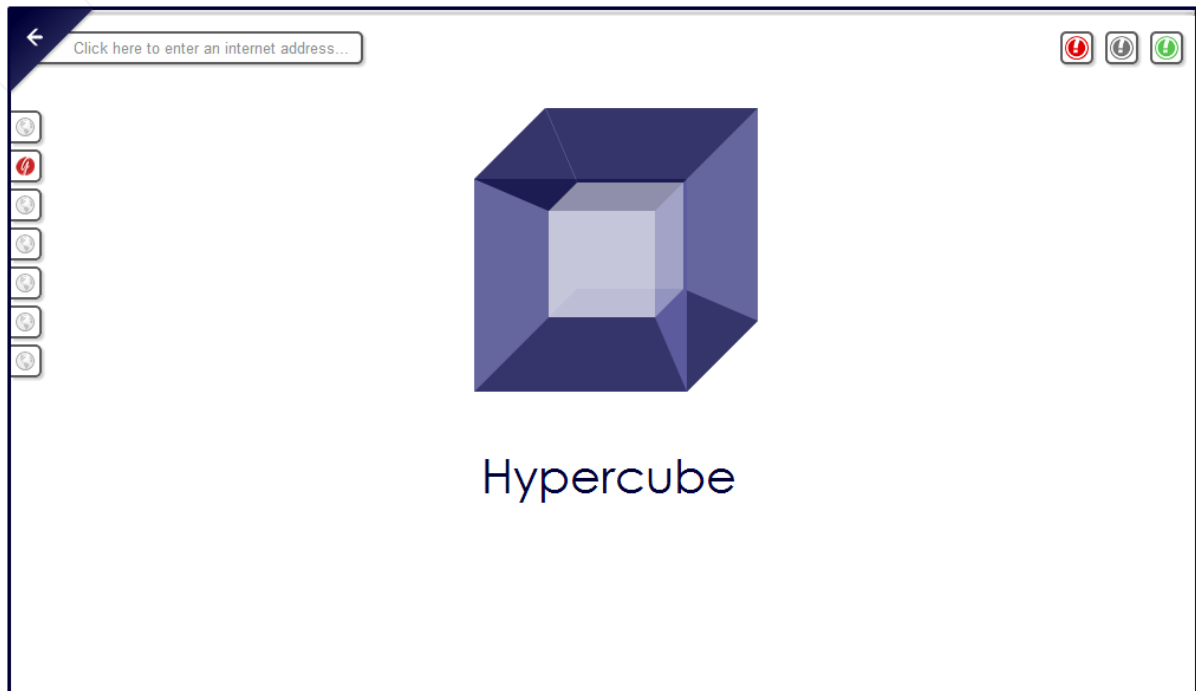
	A	B	C	D
1.1	20	22	46	24
1.2	student (history, english)	student (business information systems)	journalist	student (computer science)
2.1	7	7	7	7
2.2	5	4	5	8
2.3	yes	yes	basics	yes
2.4				
Internet Explorer	1	4	1	3
Firefox	2	5	3	2
Chrome	3	2	4	1
Safari	4	1	2	5
Opera	5	3	5	4
others	---	---	---	Chromium, Comodo Dragon, Lunascape, Maxthon
2.5				
Internet Explorer	---	---	---	2
Firefox	1	2	1	---
Chrome	2	1	3	1
Safari	---	---	2	---
Opera	---	---	---	---
others	---	---	---	---
3.1				
back button	x	x	x	x
address bar	x	x	---	x
tab bar	x	x	x	x
extensions area	x	x	x	x
background (left)	x	---	---	---
background (top)	x	---	---	---
content area	x	x	x	x
3.2				
back button	---	---	---	---
address bar	---	---	---	---
tab bar	either bookmark bar or tab bar	bookmark bar	bookmark bar	---
extensions area	extensions area (not sure)	---	---	---
background (left)	---	---	---	---
background (top)	---	---	---	---
content area	---	---	---	---
3.3				
forward button	x	---	---	---
search inputfield	---	x	---	---
stop button	---	---	---	x (poss. in combination with reload button)
element to show bookmarks	---	---	---	x
new tab button	---	---	---	x
settings button	---	---	---	x
element to show downloads	---	---	---	x

3.4	intuitivity personalizability goes with the trend regarding the browser being in the background screen space usage extendability visualization of bookmarks ease of finding bookmarks	x x x --- --- --- ---	--- --- --- x x x ---	--- --- --- --- --- --- ---	--- --- --- x x x x
3.5	uncertainty about whether or not bookmark management could work for them settling in period tab representation tab handling node size in Visualizer	x x --- --- x	--- --- x --- ---	x --- --- --- ---	--- --- --- x ---
3.6	as a new browser integrated with or based on Firefox integrated with or based on Chromium	x --- ---	x --- ---	--- x ---	--- x (as an add-on) x
3.7		yes	yes	indecisive	indecisive
3.7.1		feature complete, many extensions, resource-efficient	for personal use	---	feature complete, customizable, good performance, many extensions
4.1		9	9	8	7
4.2		---	better	better	better
4.3		---	better	even	better
4.4		---	---	---	provision of quick access to important elements
5.1		5	10	8	8
5.2		better	better	even	better
5.3		better	better	worse	better
5.4			high priority if concept is being realized; currently to limited	too mathematical	modern; findability is important
6.1		10	4	6	10
6.2		---	even	worse	even
6.3		ability to hide extensions; categorize extensions as important and unimportant	is a hygiene factor (according to Herzberg's two-factor theory)	show all needed elements without the need to search for them in the settings menu	needs ability to group extensions as seen on smartphone operating systems
7.1		---	positively surprised	would use it	optimize tabs representation (bigger icons and/or move elsewhere); needs gesture for hiding entire UI

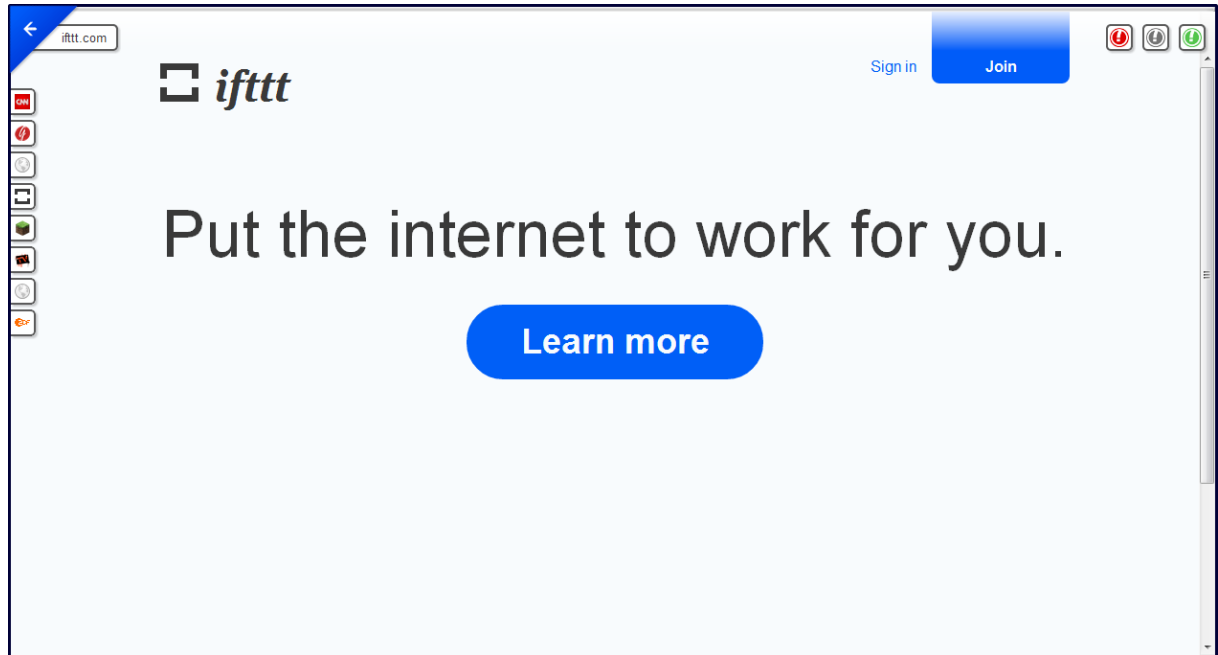
H: Hypercube displaying cnn.com



## I: Hypercube displaying start page



J: Hypercube displaying ifttt.com



## K: Visualizer with some nodes and connections

The image shows a screenshot of the CNN International website from May 17, 2012. A network visualization is overlaid on the page, featuring nodes and connections. The nodes include:

- Minecraft**: A green cube icon with the word "Minecraft" below it.
- ATV**: Two red square icons with the letters "ATV" in white.
- Google**: A red square icon with the word "Google" in white.
- Other icons**: Several small, less distinct icons, including one that looks like a red square with a white "C" and another that looks like a red square with a white "E".

The connections between these nodes are represented by thin white lines. The background of the image is the CNN International website, which includes a navigation bar with links to Home, Video, World, U.S., Africa, Asia, Europe, Latin Am, Middle East, Business, World Sport, Entertainment, Tech, Travel, and iReport. The main content area features several news stories, including "Greece's interim Cabinet sworn in as crisis deepens", "School kids flee bullets after exams", and "Indignados: 'We must fight for our rights'". The right sidebar contains a "Make CNN Your homepage" section, a "Log in or sign up to personalize!" section, and a "Popular on Facebook" section.



L: Hypercube running as a web application on Google Chrome Beta 0.18.449.2396 on Samsung Galaxy Nexus with Android 4.0.2

