FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO


FEUP

# Storytelling for Older Adults in Online Social Networks with Novel Web Technologies

**Tiago Boldt Sousa**

Master in Informatics and Computing Engineering

Supervisor: Eduarda Mendes Rodrigues (PhD)

Second Supervisor: Paula Alexandra Silva (PhD)

21$^{st}$ July, 2011

# Storytelling for Older Adults in Online Social Networks with Novel Web Technologies

**Tiago Boldt Sousa**

Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: João Correia Lopes (PhD)

External Examiner: Frutuoso Silva (PhD)

Supervisor: Eduarda Mendes Rodrigues (PhD)

---

$19^{th}$ July, 2011

# Abstract

This thesis describes the research and implementation details of an online storytelling application using novel web technologies, specifically designed for older adults. It integrates into an existing social network, with the goal of making it easier while still fun for older adults to remotely share life stories or general thoughts with family and friends.

The Web already exists for more than two decades. Still, it's growth has since been continuous, expanding both in adopters and services. The Hypertext Markup Language (HTML) is the markup language used to create content for the web; recently the fifth version of the specification has been drafted and is already being implemented by most browsers. The term *HTML 5*, further than referring to this latest version of the specification, is indeed used as an umbrella term for a wider set of web-related specifications. Together, those bring the possibility to create complex applications that run inside any recent browser, taking web development a step further.

With the dissemination of smart phones and tablets, we believe that web technologies might pose as an alternative to native software development, reducing the costs associated to developing and maintaining applications in multiple platforms. Additionally, these devices are usually easier to adopt due to their smaller size, reduced need for configurations and high interactivity, making them ideal for older adults.

Despite the increasing rate of adoption of the Internet, older adults still generally resist to adopt it due to lack of motivation or not understanding how it can be used to improve their well-being, which has been observed by the satisfaction in those that do adopt it. Literature review show us that most people in their later life tend to be exposed to physical isolation or the loss of connections to friends and families for several reasons, such as relocation or mobility impairments. The internet can be an asset in fighting those situations by providing a new channel of communication between them and those who they like on a more frequent basis. We believe that, if provided with the appropriate design and technology, most older adults are able and motivated to create content in online social networks in order to communicate with their loved ones.

This thesis verifies the readiness of HTML 5 for mainstream adoption, while using it to develop a prototype for an online storytelling application for older adults, integrated with an existing online social network. We attempt to facilitate their internet adoption and increase their well-being, by creating a new communication channel to their physically distant loved ones. We conclude that HTML 5 is a viable adoption for cross-platform development, as long as only devices with recent browsers are target. Also, we verify that with the right design and devices, the lack of familiarity between some older adults and technology can be overcome.

# Resumo

Esta tese descreve o estudo e detalhes de implementação de uma aplicação online para storytelling, recorrendo às mais recentes tecnologias Web, especificamente desenhada para idosos. A aplicação integra numa rede social, com o objectivo de permitir a partilha de estórias de vida e pensamentos de uma forma simples e divertida pelos idosos.

A Web existe há mais de duas décadas. No entanto, tem vindo a crescer desde então, expandindo tanto em utilizadores como em serviços. O Hypertext Markup Language (HTML) é a linguagem usada para criar conteúdos para a Web; recentemente apareceu a quinta versão da especificação do HTML e está já implementada na maioria dos browsers. O termo *HTML 5*, mais do que referir-se a esta especificação, é na verdade usada para agrupar um conjunto mais vasto de especificações para a Web. Juntas, estas trazem novas possibilidades aos browsers, permitindo a criação de aplicações de maior complexidade, levando as capacidades Web mais longe.

Com a disseminação dos smart phones e tablets, acreditamos que as tecnologias Web podem ser uma alternativa ao desenvolvimento nativo de software, reduzindo os custos associados ao desenvolvimento e evolução de aplicações em diversas plataformas. Ainda, estes dispositivos são normalmente mais fáceis de adoptar devido ao seu tamanho, diminuída necessidade de configuração e alta interactividade, fazendo deles ideais para idosos.

Apesar do elevado ritmo de adopção da internet, a população idosa tem tendência a resistir a sua adopção devido à falta de motivação ou não entender como estas tecnologias podem melhorar o seu bem-estar, algo que é observado naqueles que a adoptam. Pela revisão de literatura aprendemos que a maioria das pessoas na sua idade avançada tendem a estar expostas a isolamento e perda de ligação com amigos e família por diversas razões, como mudança de habitação ou dificuldades de movimento. A internet pode ser um bem essencial para ajudar a combater essas situações, disponibilizando um novo canal de comunicação entre eles e aqueles de quem gostam e com quem gostariam de comunicar mais frequentemente. Acreditamos que, caso lhes seja disponibilizada uma aplicação com o design e tecnologias certas, a maioria dos idosos estão aptos e motivados para partilhar conteúdos online com as suas ligações.

Esta tese verifica se o HTML 5 está pronto para ser adoptado, usando-o no desenvolvimento de um protótipo para uma aplicação de storytelling online para idosos, integrada numa rede social. Temos como objectivo facilitar a adopção da internet e o aumento do bem-estar pelos idosos, criando um novo canal de comunicação entre estes e aqueles de quem eles gostam. Concluímos que o HTML 5 é uma opção viável para o desenvolvimento de aplicações multi-plataforma desde que os dispositivos em vista possuam browsers recentes. Verificamos ainda que com o desenho e tecnologias certas, o desconforto de alguns idosos para com a tecnologia pode ser ultrapassado.

# Contents

CONTENTS

# List of Figures

# List of Tables

LIST OF TABLES

# List of Sources

LIST OF SOURCES

# Abbreviations

| | |
|---|---|
| AJAX | Asynchronous JavaScript and XML |
| API | Application Programming Interface |
| app | Application |
| apps | Applications |
| BBC | British Broadcasting Corporation |
| CERN | Conseil Européen pour la Recherche Nucléaire |
| CMC | Computer-Mediated Communication |
| CSS | Cascade Style Sheets |
| DOM | Document Object Model |
| FhP | Fraunhofer Portugal |
| FQL | Facebook Query Language |
| GPS | Global Positioning System |
| GPU | Graphics Processing Unit |
| GSM | Groupe Spécial Mobile |
| HCI | Human-Computer Interaction |
| HP | Hewlett Packard |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| IP | Internet Protocol |
| JS | JavaScript |
| JSON | JavaScript Object Notation |
| MIT | Massachusetts Institute of Technology |
| OAuth | Open Authorization |
| OS | Operative System |
| REST | Representational State Transfer |
| SDK | Software Development Kit |
| SQL | Structured Query Language |
| SVG | Scalable Vector Graphics |
| URL | Uniform Resource Locator |
| W3C | World Wide Web Consortium |
| WSGI | Web Server Gateway Interface |
| WWW | World Wide Web |
| XHTML | Extensible Hypertext Markup Language |

# Chapter 1

# Introduction

This thesis describes the research and implementation process of an online storytelling application for older adults that integrates into an existing social network. With novel web technologies and specially crafted design, we intend to make it easier, while still fun, to remotely share life stories or general thoughts with family and friends.

We believe the Web is growing into a powerful development platform. HTML 5 and related specifications were a crucial part of this study as it improves previous versions in elements, features and more features, required for us to build a truly cross-platform application in the web. This means it should integrate seamlessly to any device, being it a desktop computer, a tablet or a smart phone, as long as it has a recent browser.

This thesis was developed in a collaboration between Faculdade de Engenharia da Universidade do Porto, represented by Eduarda Mendes Rodrigues, PhD, and Fraunhofer Portugal (FhP), under supervision of Paula Alexandra Silva, PhD. It is the work of Tiago Boldt Sousa in fulfilment of the requirements to achieve his Master's degree.

## 1.1 Rising of the Web

The proliferation of the World Wide Web has been happening for more than a decade now and it's still expanding (fig. 1.1). Today, the web is not only for the technological savvy anymore.

Web applications, as well as internet connected devices are multiplying around us. Way beyond computers, these are now available on our smart phones or tablets and even in more unusual devices such as set top boxes, e-book readers, gaming consoles and so on.

That fast adoption of the web, along the different pace of evolution by browsers led to a non-standard implementation of Web specifications. Web development suffered from the fact that the developer had to support how different browsers implement the same feature, due to the lack of standards compliance in them. Also, it is common to adopt

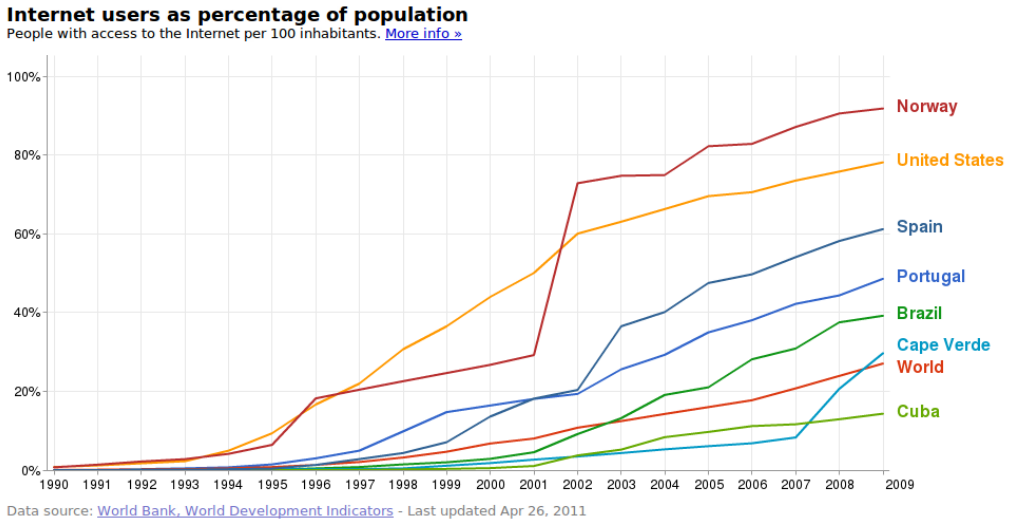



Figure 1.1: World internet access growth. Copyright to Google.

external browser plug-ins, such as Flash or Silverlight, to perform more complex tasks that were previously impossible to do by only using the browser's possibilities, such as displaying video or playing a game, resulting in a fragmentation of the web — plug-in vendors have the option to choose which platforms and browsers to support.

With the introduction of HTML 5, these issues are claimed to be coming to their end, allowing developers to focus on standard web specifications, with the guarantee that browsers will properly support them. This growing interest of web browsers in supporting the specification has even made them compete on how well they implement it.

## 1.2 Storytelling for the Elderly

Storytelling and chatting is something that we see older adults eager to do on a daily basis. From those we interacted with, we observed happiness in them just from having someone new to talk to. Still, isolation and ageing might prevent them from doing this on a daily basis. Online social networks could provide them with an alternative way to increase their social inclusion and sense of well-being [GMF+10] , but studies claim that they commonly have technical difficulties in engaging with those [Che09] .

We believe that providing them with an application, specially designed for them, for providing a method of socialization were they could find friends and family would be the needed motivation to trigger the will to embrace new technology, resulting in the restored connections with those who are physically distant from them.

## 1.3 Goals

With this thesis, we intent to answer the questions posed in each of the goals presented bellow:

1. **Review digital storytelling concepts and implementations.** What is storytelling? Can it be used by older adults? Are there any online approaches to it?

2. **Evaluate internet adoption by older adults and design methodologies to work with them.** Is there reluctance in adopting the internet and new technologies by older adults? How can we help them in that adoption process?

3. **Study why HTML 5 is claimed to be revolutionizing the web.** HTML 5 has been a buzz-word in the internet for the last months. How is it any different from previous versions? Who is maintaining/supporting it?

4. **Understand the current state of HTML 5 as a tool for cross-platform development.** Is HTML 5 ready for mainstream adoption? Can developers rely on it without worrying if browsers will comply with the standards? Are browsers from mobile devices ready to adopt this technology?

5. **Design, implement and evaluate an online storytelling application for older adults.** The application should integrate with an online social network for sharing and obtaining content, as well as have special design cares to provide an better user experience for the older adult.

## 1.4 Thesis Contributions

With this thesis, we want to provide developers who have the web as a target platform or anyone who wants to understand how web applications can be an alternative to native development with a solid starting ground of information. We provide developers with a set of guidelines on how to develop for the web, highlighting common issues and proposing solutions for them. If used with the thesis by Pedro Tenreiro, fellow in this project, it can also give some detailed guidelines of how to design for older adults. Tenreiro was responsible for the system requirements and HCI evaluation which resulting in our application's design [Ten11] .

This thesis resulted in two publications, one for Interact 2011 and another for ICEC 2011, as described in section 7.3.1.

Furthermore, the resulting application can benefice older adults by providing them with a new channel of communication to their friends and family.

## 1.5 Outline

This thesis is organized into three parts, with the following structure:

**Part 1: State of the Art.** This first part has a literature review on older adults and their adoption of storytelling applications in particular and the web and technology in general. It is composed by two chapters:

- Chapter 2, *Storytelling and Older Adults*, describes the willingness of older adults to adopt new technologies and reviews studies of storytelling applications designed for them.
- Chapter 3, *The Web, Revisited*, concisely tells the story of the World Wide Web (WWW), from it's creation to the dawn of it's expansion. Later in the chapter we jump to our current year and review novel web technologies, such as HTML 5 and related specifications — actually generally grouped under the umbrella term of HTML 5.

**Part 2: Requirements and Implementation.** The second part of this thesis describes our approach to the project. We researched how older adults would receive our idea, gathered the system requirements and how we prototyped it using initially paper prototypes and later with novel web technologies.

- Chapter 4, *Designing a Storytelling Application for Older Adults*, explains the methodology used while working with older adults to provide the application's design. It also lists the system requirements, obtained from discussions between the involved members in the project and informal interviews with the older adults.
- Chapter 5, *Implementation*, describes the system's technical design and implementation decisions. It also overviews tools and crafts during the implementation.

**Part 3: Results and Conclusions.** The last chapter in this thesis lists our evaluation of our implementation, according to our usability tests and observations. We finish with the project conclusions.

- Chapter 6, *Results and Discussion*, present our system evaluation, according to the results from the usability tests we have applied and describe in the chapter.
- Chapter 7, *Conclusions*, is the last of this thesis and sums up our work by reviewing our goals and achievements, explaining our approach to the dissemination or our work and our future work.

# Chapter 2

# Storytelling and Older Adults

## 2.1 Social Inclusion of Older Adults

Unlike younger adults, who commonly have bigger social networks, older adults tend to invest in the most satisfying and humanly rich relationships for them [CH06] . As age goes by, with a natural decay of their social network, there's a tendency for elder people to feel lonelier. This happens because the social network size of the older adult is directly influenced by the restrictions imposed by ageing, such as limited mobility, income, and loss of friendships due to death or relocation [Wri00] . These conditions make it hard for them to keep in contact with friends and family resulting in a decreased possibility to share their lives's legacy.

Quoting Joseph F. Coughlin, director of the AgeLab at MIT on an interview to the New York Times [Cli09] :

> *"One of the greatest challenges or losses that we face as older adults, frankly, is not about our health, but it is actually about our social network deteriorating on us, because our friends get sick, our spouse passes away, friends pass away, or we move."*

According to that same article, about one third of people aged 75 or older live by themselves. This might have an impact on the psychological well being of the person.

**Receptiveness to go Online** Unlike a few decades ago, today's households no longer have more than two generations [WSW05] , resulting in older adults living by themselves. Distance from family can result in feelings of loneliness and can limit their ability for sharing their legacy, something which is paramount for them. Furthermore, transportation and communication problems are known constraints due to the decline of their physical and mental state, resulting in a general decline of their well-being and happiness [Che09] . While they can be positively influenced by the use of online social networks [GMF+10]

, younger people are the mainstream audience of these services [Pin10] . Feature-rich services and web pages with large amounts of information are common and difficult to use by those lacking computer skills; this results in a general inhibition towards the exploration of technology from the elderly due to their reduced short term memory. Still, there are those who are highly motivated and embrace these new technologies, understanding how they can be helpful for them. Pereira [Per11] reports the case of Isabel Moreira, aged 74, and Maria do Céu, aged 84, who daily use their Facebook accounts by themselves and feel closer to their families by doing so. The same author claims that there are in Portugal about 33 thousand Facebook accounts by users older than 65.

While we acknowledge that not all older adults might be receptive for adopting new technologies, if we are able to show them how advantageous for them using such application might be, many may become receptive to it [Gre09] . To those, we intended to bring an easy to use application that might improve their well-being, similarly to what happens with those who already use such social networks.

## 2.2 Digital Storytelling

### 2.2.1 Computer-Mediated Communication

Computer-Mediated Communication (CMC) is a term defining communications though networked devices [TLT04] . The first digital communication is put somewhere between the World War II and the early sixties when the first exchange was made on the prototyping service to email. The principle follows the same idea of in person communications where a sender sends a message to a receiver, as seen in figure 2.1.

### 2.2.2 Studies and Implementations

Digital Storytelling is a mature subject and many studies and implementations have been conducted. Below some are presented.

**Facebook Stories**

Integrated in the Facebook social engine, it allows users to write their stories. Each story is not only published to the user's profile but also tagged with the location. All places with stories associated to them are shown on a map and the users can click on that point to filter and read those stories[1]

**TellTable**

Created by Microsoft's Research, TellTable is and interactive storytelling application for children similar to how children tell their stories using physical toys. Using

[1] Read more at http://stories.facebook.com/

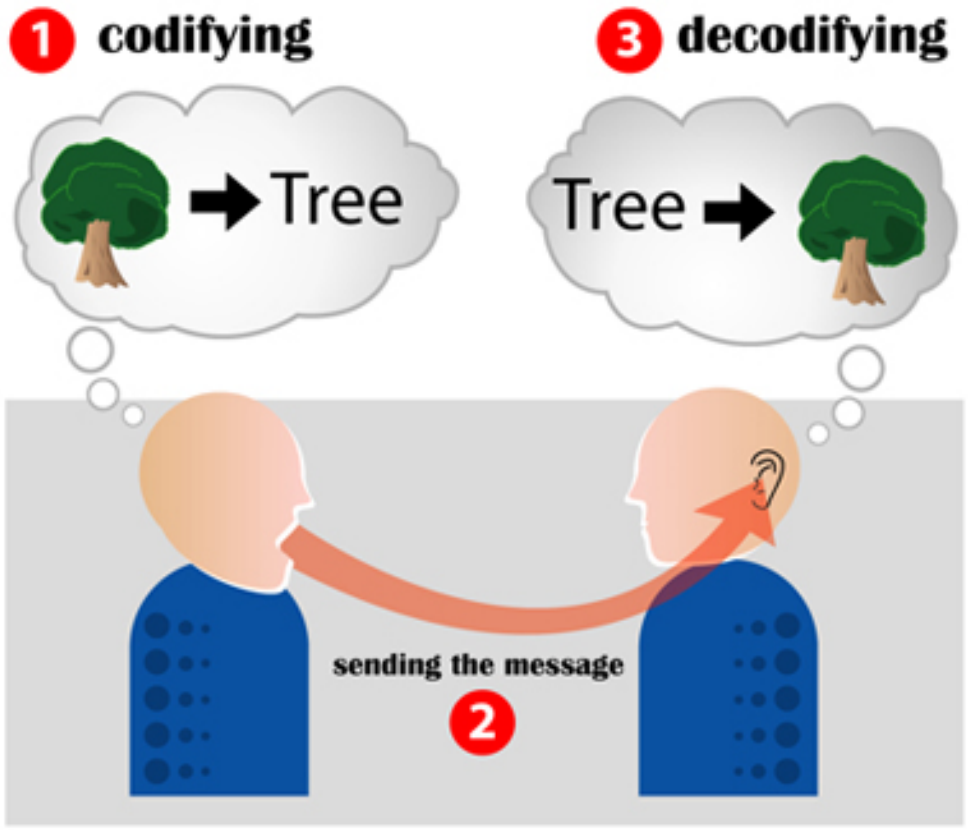



Figure 2.1: Sending a message. Emitter on the left encodes the message using his voice and the decoder on the right decodes it with his ears. The communication channel used is the air

the Microsoft's tabletop device (Surface), it allows them to create characters and sceneries, recording the narration, so that they can latter use the device to show that story to their friends. [HCLS09]

**Capture Wales**

An initiative by BBC that started in 2001 with the objective of connecting BBC with the communities from Wales. They've contacted directly with the communities with workshops to teach them how to use the platform and share their stories. The project only required low-budgeted digital photography cameras, video-editing software and laptops in order to create multimedia stories. This technology allows users to create videos from a compilation of pictures and an audio stream. The online site[2] is still available and it shares stories mainly from older adults about their thoughts and lives [Mea03] .

**ICDL for iPhone/iPad**

The International Children's Digital Library (ICDL) is a foundation that aims at

[2]http://www.bbc.co.uk/capturewales/

creating a digital library from children's books from all over the world, having the main goal of covering every language and culture [ICD] . They have an Ipad/Iphone application with some of these stories available and in a recent study, they have tried to improve their usability so that it would be easy for an older adult to use the device to tell the story to a child [DBQ09] .

This study allowed them to take better advantage of the device capabilities. Between their conclusions, they have decided to rotate the image of the book to match the orientation of the device, showing two pages in landscape mode and only one when the device is vertical. Tapping on the cover of the book shows thumbnails of all pages from the story and tapping in a page zooms in to it. Pages are switched by swapping the finger in the screen. Taping on the text will also enlarge it to facilitate reading, as suggested by an older adult who was test subject in this study.

The books are also editable, enabling users to not only browse existing stories, but also to edit them or create new ones. The application allows to choose from a set of colors and draw on top of the pages with the touch input. Text boxes can also be edited.

**Family Storytelling for Grandparents and Grandchildren living apart**

A study [VKPV10] conducted to verify how synchronous communication between grandparents and grandchildren could be achieved. It is known that children have a hard time to keep a phone conversation [BKA+09] , so this study tried out new and more interactive approaches to provide a more enjoyable experience for both parties during communication. A digital storytelling application was developed to be experimented without any help from researchers to see how enjoyable would the experience be.

The application consisted in a shared display that allowed to share virtual objects, such as images or text. An audio stream keeps both sides communicating while interacting with the display. A top toolbar allows any user to choose one of the available stories, as well as choosing a tool, from pencil, rubber or normal cursor, to interact with the shared space. A bottom toolbar showed pictures available locally that would go to the shared space upon selection. These could be used to share a personal life-story, like telling how a field trip went.

The study concluded that grandparents and grandchildren are keen to keep contact using story-telling technologies, both for reading stories and for sharing life-stories using voice, text and personal photos.

## 2.3 Designing for Older Adults

The process of designing an interactive environment for older adults is a complex task in this project. There is a wealth of evidence suggesting that older adults have an increased difficulty in learning new information, exhibit less efficient reasoning skills, are slower to respond on all types of cognitive tasks, and are more susceptible to disruption from interfering information than younger adults [PPM+01] . Designing interfaces for older adults demand special care in order to overcome common difficulties those common obstacles.

The process of designing for older adults is a commonly studied subject within the HCI community. Designers have scientific material to guide their work using existing guidelines/heuristics. These are principles that can help identify and overcome design problems resulting from previous studies. In other words, designers can use existing studies' results to improve their work. These can be obtained both from industries or academia [KSZP05] .

The design of this project was pursued by Pedro Tenreiro, collaborator in this project. His thesis contains all the details related to our project's design [Ten11] .

## 2.4 Summary

From the literature review presented in this chapter we inferred that the internet is still a largely expanding platform. Even though it's adopters are mainly users of more younger age, many older adults are already learning how this technology might be of use in their daily activities and well-being. Also, we have observed that there are in fact many studying how to help older adults embrace new technologies in general and the internet in particular. Amongst those studies, storytelling is a subject that they are found receptive to.

We believe that an online storytelling application that integrates with a social network where they can easily share content with friends and family would have a good acceptance by the elderly.

# Chapter 3

# The Web, Revisited

The central focus of this thesis is on studying novel web technologies and how these can be applied to real world problems such as creating adaptive interfaces. With the high amount of devices available to access the web, online applications should always consider browser capabilities and screen resolutions, hence that applications should always be adaptive to the hosting device.

This chapter briefly reviews the history of the WWW, from it's creation to more recent times where new emerging technologies are revolutionizing how and what for we use our browsers.

## 3.1 History of the WWW

### 3.1.1 The Hypertext

During the year of 1989, Tim Berners-Lee, at the time a physicist at CERN, troubled by the possibility of loss of information in the institution started to work on a better way to organize it. Berners-Lee wrote an internal memo justifying his concern and suggesting to have that information stored in a centralized server in the CERN's network, allowing it to be accessed in any remote terminal. He called this proposal Hypertext [BL89] .

"We should work toward a universal linked information system" — Tim Berners-Lee [BL89] .

By 1990 he had wrote both the first Web server and browser (fig. 3.1), as well as the initial specifications for URI's, HTTP and HTML [W3Cb] . He named *WorldWideWeb* to the his web browser, which was already more than just a browser, being able to browse *http:*, *ftp:*, *news:* and the local *file:* namespaces. It also had the capability to edit local files [BL] .

In 1992, inspired by the ISO certified SGML, Tim Berners-Lee published the first public HTML specification draft, with 20 elements [BL92] and with it began the WWW movement.

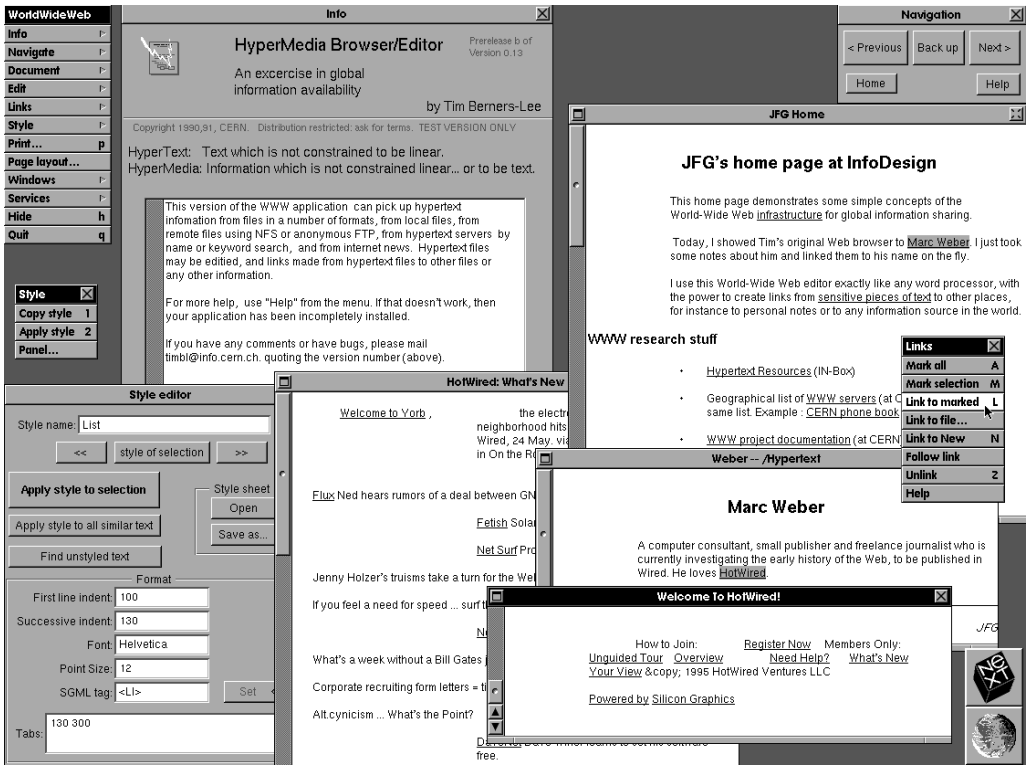



Figure 3.1: *WorldWideWeb* — the first web browser, written by Tim Berners-Lee, running on a NeXT computer.

### 3.1.2 World Wide Web Consortium

To provide a solid ground for evolving the web, Berners-Lee created the World Wide Web Consortium (W3C) in October 1994 [W3Cb] , an international community that works on Web standards. It is still led by the Web inventor, Tim Berners-Lee and the CEO Jeffrey Jaffe. Their mission is to lead the Web to its full potential [W3Ca] . Detailed history is graphically described in figure 3.2.

Their work is done in collaboration with the consortium members and the general public. Members of the consortium include huge IT companies, such as Adobe, Apple, BT, CAIXA, CERN, Cisco, ERICSSON, Google, HP, Microsoft, Samsung, Sony, Walt Disney, Xerox, many universities and many others[1] [W3Cc] .

[1] As of 8 February 2011, the World Wide Web Consortium (W3C) has 324 Members.


Figure 3.2: The history of the web according to W3C — Image available at http://www.w3.org/2005/01/timelines/timeline-thumbnail.png

W3C is responsible for a large number of standards in areas such as:

- Web Design and Applications;
- Web Architecture;
- Semantic Web;
- XML Technology;
- Web of Services;
- Web of Devices;
- Browsers and Authoring Tools.

This group produces the specifications for CSS, DOM, HTML, RDF, SVG, SOAP, XHTML and XML, among many others.

### 3.1.3 Hypertext Markup Language

#### 3.1.3.1 Overview

HTML is a plain-text markup language used to describe the contents of a web page [Rag05] . Browsers are used to interpret HTML files, downloading them from web servers and interpreting their source to display a web page to the user [Rag05] .

HTML is composed by elements which are configured using attributes. An HTML document describes the hierarchy of elements contained in a web page.

In addition to HTML, Cascade Style Sheets can be loaded to configure the looks of the HTML elements. JavaScript can be used to interact with the HTML in the client side.

**Elements** An HTML element is the smallest component of an web page. HTML includes element types that represent, between many others, paragraphs, Hypertext links, lists, tables or images. [RHJ99] .

Elements are generally enclosed by their opening and closing tags. Whenever an element has attributes in it, these are described in the starting tag. It is possible to include other elements between these tags, named as child elements. An example is available in source example 3.1.

**Source 3.1** HTML for a paragraph element with bold normal text and a bold element with additional text

```
<p>This is a paragraph <b>with bold text</b></p>
```

**Attributes** Elements can be configured by using attributes. These appear on the opening tag of the element, setting it's properties. Older techniques on styling web pages were based on attributes, setting fonts, alignments, text sizes and other directly in the HTML file. Nowadays, using in-line styling is not recommended and should be replaced by the use of CSS files.

Using attributes with HTML elements is exemplified in source example 3.2.

**Source 3.2** Setting the ID attribute for the paragraph element

```
<p id="par1">This is a paragraph <b>with bold text</b></p>
```

#### 3.1.3.2 Versions

As technologies evolved, so did the HTML specification. It started with 20 elements in the first version [BL92] and has since grown in both elements and attributes.

With the appearance of the first web browsers, a mis-interpretation on the specification was also happening, having the same code resulting in different results according to the browser being used; the same code generated different results according to the used browser. In addition, each browser introduced support for features not covered in the specification, making it ill-defined [RLAK88] . To avoid this fragmentation, a team, led by Dan Connolly and Berners-Lee, summed up all the widely used tags into what was called the HTML 2 draft and circulated it around the web for comments. To guarantee it's correct use, Dan also wrote the Document Type Definition (DTD), a formal description of the language [BLC95] with which developers could validate their pages.

**HTML vs XHTML** XHTML (Extensible Hypertext Markup Language) is a family of document types that reproduce, subset and extend the original HTML [Gro00] . The benefits behind XHTML are:

- XHTML documents are XML conforming. As such, they are readily viewed, edited, and validated with standard XML tools.
- XHTML documents can be written to operate as well or better than they do with simple HTML.
- Any application that can process or edit XML files would also be usable with XHTML as the concepts are the same, making the XML Document Object Model (DOM) available for developers.
- As the XHTML family evolves, documents conforming to XHTML will be more likely to interoperate within and among various XHTML environments.

Distinguishing XHTML from HTML is the need for well-formedness [Gro00] . This concept is introduced by XML and it means that all elements must either have closing tags or be written in a special form. For these reasons, all elements must be properly nested. XHTML can be considered a more strict version of HTML.

#### 3.1.3.3 Document Object Model

The DOM is a platform and language-neutral interface that allows programs and scripts do dynamically interact with the content of a structured document such as HTML or XML. With JavaScript, developers can navigate or query the DOM of the web page for creating, deleting or updating individual elements such as forms, text, links, images or any other element of the HTML document.

### 3.1.4 CSS

Cascade Style Sheets are used to style documents written in markup languages, such as HTML. The specifications are maintained by the W3C and the file type has it's own MIME[2] type. The first formal specification was published by the end of 1996 [LB08] . CSS can be embedded into HTML documents or linked to external files.

#### 3.1.4.1 Selectors

While using CSS documents, the programmer can use selectors to decide where rules apply. Selectors are patterns that match against elements in a tree, and as such form one of several technologies that can be used to select nodes in an XML document. Selectors have been optimized for use with HTML and XML, and are designed to be usable in performance-critical code [ÇEG+09] .

Using Selectors, designers are able to match a single HTML or XML element, based on id's or hierarchy description, join elements using classes, choose all elements of one kind and so on [ÇEG+09] .

#### 3.1.4.2 Rules

CSS rules are what style the HTML elements. The property is separated from the value by a colon (:) and the rule is ended by a semi-colon (;), as you can see in source 3.3. A CSS file can have an unlimited number of rules described in it.

#### 3.1.4.3 Browser Support

CSS compliance is one of the competing aspects of web browsers. A number of online tests are available to verify the quality of the layout engine rendering capabilities, such

[2]Multipurpose Internet Mail Extensions - internet standard used to describe the content of a data stream

**Source 3.3** CSS example code, aligning all tables to the middle

```
table{
    margin-left: auto;
    margin-right: auto;
}
```

as the Acid test. These tests use standard CSS code to display a web page, covering as much of the standard as possible. Only CSS compliant browsers are able to render the page as expected and there's usually a figure exemplifying the expected result. Refer to figure 3.3 to view how Internet Explorer 8 performed on the Acid 3 test, comparing it to the reference image.

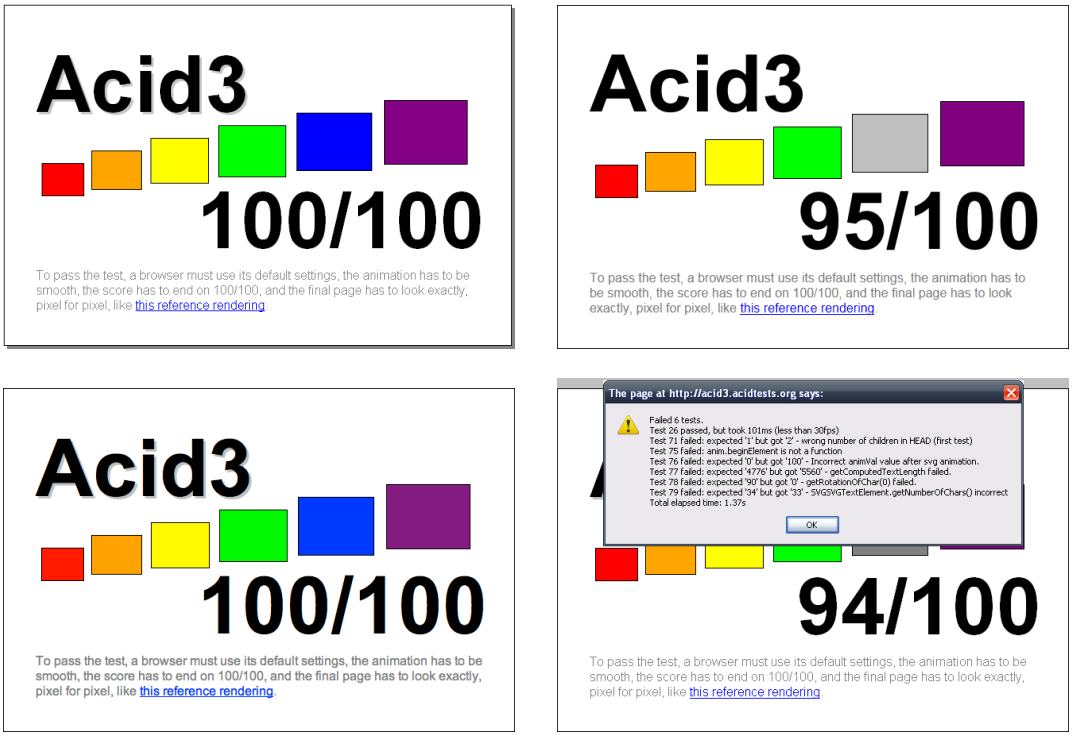



Figure 3.3: Acid 3 Reference Image (top left) comparison to and Internet Explorer 9 (top right), safari 5(bottom left) and Firefox 3.6 (bottom right).

According to the W3C specification for CSS 2.1 [LBÇH10] , many advantages can be perceived from the use CSS styling instead of individually style each HTML element based on attributes:

**Logical separation** by having the styling separated from the layout information, work can be easily divided between programmers and designers, having the first working on the HTML code and the second focusing on the CSS files. Also, the same HTML can be used with different style sheets to result in different themes for a page.

**Improved portability** since it makes it rather easy to reuse HTML code in different platforms or browsers. The requested server can verify the browser making the request, responding with the appropriate file for that browser. This can also guarantee portability for mobile devices, having the CSS customizing the same HTML in a way that will fit smaller screens.

**Bandwidth** can be saved, since CSS files can be stored in browser's cache and do not need to be download on every page view. HTML file size also is reduced since styling information is no longer contained in it.

**Consistency** can be achieve site-wide when the same CSS files are used, keeping the same style, without changing the HTML code. Also, it simplifies the programmer job: if there was the need to alter the font to an HTML element, such as *h1*, programmers needed to specify that information as an attribute to the element each time it occurred. With CSS, the rule only needs to be specified once, applying itself to all occurrences.

On the other hand, some minor disadvantages are also known, such as browser incompatibilities, which require an extra effort from the programmer to have customized CSS for each browser he wants to support. Figure 3.4 shows own different browsers displayed the same CSS styling. [Des10] . Still, this problem is disappearing as browser become conscious of the need to implement the standards for both creating a better web and please their users.


Figure 3.4: The Opera logo rendered with CSS in several browsers.

It is now uncommon to actually navigate in a web page that is not styled using CSS rules.

### 3.1.5 JavaScript

JavaScript is the scripting language used for client-side programming in web pages, providing dynamic interactivity to the static HTML specification. Despite what the name might suggest, other than marketing interests, JavaScript has no connection to the Java programming language [Ben99] . The language was initially developed by Brendan Eich

from Netscape under the name LiveScript. It was renamed to JavaScript after a marketing agreement between Netscape and Sun Microsystems[3] by the release time of Netscape Navigator 2.0 in September 1995 [New95] .

#### 3.1.5.1 Language Features

This section describes JavaScript internals and is based on the book *JavaScript: the definitive guide* [Fla02] .

**Imperative** JavaScript follows the structured programming syntax from C; it uses conditional expressions such as *if*, *while*, *for* or *switch*. Refer to source code 3.4 for an example. An example is available in source example 3.4.

**Source 3.4** A function to calculate the factorial of the given number and a call to it

```
function factorial(n){
    if( n==0 ) return 1;
    else return n * factorial( n-1 );
}

document.write(factorial(3));
```

**Dynamic** JavaScript is a dynamically typed language. This removes the need to specify the type for variables and allows to change a variable's type at any time. At the expense of strictness, this feature improves the language's simplicity.

The whole language itself is dynamic, being able to interpret and execute new code in run-time using the *eval* function.

**Functional** Functions in JavaScript are objects themselves. They have properties and methods and can be assigned to variables, passed to or returned in other functions and so on. The function itself is invoked using the *()* operator. Nested functions are also available, being created only when the parent functions is called. Source example 3.5 shows not only how nested functions work but also how a function could return another for later use.

**Prototype-based** Despite being object-oriented, JavaScript does not use classes as base for it's objects. Instead, functions with nested functions can be used for this purpose, like source example 3.6 shows.

The problem with that approach is that objects are always created as a copy in JavaScript. So, if we were to create one thousand Rectangles using source example 3.6, we would

[3] Sun Microsystems was behind the Java programming language

**Source 3.5** Example from *JavaScript: the definitive guide* of a function that creates and returns another function

```
// This function returns a function each time it is called
// The scope in which the function is defined differs for each call
function makefunc(x) {
    return function( ) { return x; }
}
// Call makefunc( ) several times, and save the results in an array:
var a = [makefunc(0), makefunc(1), makefunc(2)];

// Now call these functions and display their values.
// Although the body of each function is the same, the scope is
// different, and each call returns a different value:
alert(a[0]( )); // Displays 0
alert(a[1]( )); // Displays 1
alert(a[2]( )); // Displays 2
```

also have on thousand copies of the *area()* function, augmenting the memory required for execution.

To solve this issue, JavaScript uses the prototyping approach, were common properties (variables or functions) of an object are stored in a parent prototype object. Source example 3.7 shows how previous example 3.6 should be implemented.

#### 3.1.5.2 Beyond the Browser

Today, JavaScript does not belong only to the browser. Interpreters for non-web development exist, allowing the language to be used in different contexts. Just like the browser's interpreter provides access to the DOM environment (refer to chapter 3.1.3.3), each interpreter should provide access to it's host environment. Some examples are:

**Google Apps Script** allows automation of tasks across several Google products [Goo] .

**Unity** is a game engine with supporting scripting through JavaScript [Uni] .

**Adobe AIR** allows developers to create cross-platform applications using multiple technologies, such as Flash, Flex, HTML or JavaScript [CDGH08] .

**GNOME-Shell** is a component of GNOME 3.0, the next version of the GNOME Desktop Environment and it is largely written in JavaScript [Cut09] .

**webOS** is a mobile operating system initially developed by Palm and later purchased by Hewlett Packard. It uses the WebKit renderer as base for it's SDK, being all programing is done via JavaScript, using CSS and HTML for the layout, just like web applications.

**Source 3.6** How to use functions to create object like Java-style classes.

```
// This function will act as a constructor for
// our object
function Rectangle(w, h){
    // Initialize object properties.
    this.width = w;
    this.height = h;
    // Define methods for the object.
    this.area = function(){return this.width * this.height};
}
// Now, when we create a rectangle, we can invoke methods on it:
var r = new Rectangle(2,2);
var a = r.area();
```

**Node.js** is a framework for building network programs only by using JavaScript. It claims to have higher memory efficiency than common thread-based web servers since almost no functions perform I/O operations, removing deadlocks[4] that slow down applications [Nod] .

#### 3.1.6 Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It has several usages besides transferring HTML files, such as name servers and distributed object management systems. The protocol has been in use in the World-Wide Web since it's debut in 1990 [FGM+99] .

## 3.2 Novel Web Technologies

### 3.2.1 HTML 5

#### 3.2.1.1 Overview

HTML 5 is the latest draft for the HTML specification described in subsection 3.1.3. It's being developed by the Web Hypertext Application Technology Working Group (WHATWG), a growing community of people interested in evolving the Web. This community focuses primarily on the development of HTML and APIs needed for web applications [WHA11] .

Since March 2010 the specification is also a Working Draft at W3C, meaning that's recognized as the next HTML version and standard, ready for browser implementation

[4] A deadlock happens whenever two processes need to read or write the same information from the disk. One of them must stop and wait for the other to complete it's action before proceeding.

**Source 3.7** Using prototyping to improve JavaScript performance

```
// This function will act as a constructor for
// our object
function Rectangle(w, h){
    // Initialize object properties.
    this.width = w;
    this.height = h;
}

// Create the common function using prototyping
Rectangle.prototype.area = function(){return this.width * this.height;};

// Now, when we create a rectangle, we can invoke methods on it:
var r = new Rectangle(2,2);
var a = r.area();
```

but still evolving. The responsible editor for the specification [Hic10] is Ian Hickson from Google.

To ease the transition process for adopting this novel technologies, fallback methods are available to detect and work around non-supporting browsers. An use case scenario is when a video tag is not supported (see subsection 3.2.1.2), the flash plug-in can still be used to play the same video.

This section overviews HTML 5 and related specifications. The study done in this section will be lated used as a starting point for implementing our own application.

#### 3.2.1.2 New elements and features

The main innovation in the latest version of the specification consists on the new elements added. HTML 5 introduces a significant amount of elements that aim at providing interaction with the end user by using only the browser. In other words, HTML 5 aims to provide elements for web developers to create more interactive web applications.

By giving more power to HTML, it will also be possible to deprecate a number of technologies that were created to extend the browser, such as Flash or Silverlight[5].

Bellow is a description of some of the most relevant new elements in HTML 5 and some other specifications targeted for web development.

**Video** The <*video*> element is used for embedding video in a web page. Until now, this was generally achieved through the use of Adobe's Flash technology. With this new tag, the browser will now be able to natively decode video files streamed from the servers,

[5]Flash an Silverlight allow the development of Rich Internet Applications in supported platforms

resulting in a clear performance improvement and removing the need for the Flash plug-in.

The initial specification only predicted one video format to be streamed by <*video*>, the Open-Sourced Ogg Theora, due to it's openness. Nevertheless, many companies, like Apple [Pau09] , contested the decision in favor of other codecs, debating video quality, performance or legal rights; afterwards, new formats were introduced in the specification. Currently, the tag supports multiple video sources, leaving to the browser to choice of which to use, according to which it supports best. This forces the server to host multiple instances of the video to guarantee support across multiple browsers, as shown in source example 3.8. Browser's codec support is described in table 3.1.

Table 3.1: Supported codecs in browsers via the video tag.

| | **Native video format support** | | |
|---|---|---|---|
| **Browser** | **Ogg Theora** | **H.264** | **VP8 (WebM)** |
| Google Chrome | Yes | No | Yes |
| Internet Explorer | No | Yes | No |
| Mozilla Firefox | Yes | No | Yes |
| Opera | Yes | In Linux and FreeBSD with external libraries | Yes |

**Source 3.8** Video tag usage example. Note the two sources for video.

```
<video controls>
    <source src="foo.ogg" type="video/ogg">
    <source src="foo.mp4">
    Your browser does not support the <code>video</code> element.
</video>
```

**Audio** Similar to the video tag (subsection 3.2.1.2), the audio tag provides the audio playback. Until now, similarly to video, audio files had to be played using an external browser plug-in like Flash or Windows Media Player. The same API for controlling video applies in the interaction with the audio tag. Browser's supported codecs vary depending on browser and version. Using the audio element in exemplified in source example 3.9.

**Source 3.9** Audio tag usage. When there is no audio support, fallback text is show.

```
<audio src="horse.ogg" controls="controls">
    Your browser does not support the audio element.
</audio>
```

**Canvas** Canvas provides a two dimensional drawing interface inside the browser. Technically, it's a bitmap able to render graphs, games or other images. It has only two attributes, width and height. The API is available with a variety of graphical functions such as drawing lines, rectangles, transforming the canvas or even inserting text or images.

Source example 3.10 demonstrates how to instantiate a canvas element and how to draw on it. The result can be observed in figure 3.5.

**Source 3.10** Instantiating a canvas element and drawing in it.

```
var canvas = document.getElementById("example");
var context = canvas.getContext('2d');
canvas.setAttribute('width', '200'); // clears the canvas
canvas.setAttribute('height', '200'); // clears the canvas
context.fillRect(0,0,200,200);
context.fillStyle = 'red';
context.fillRect(0,0, 100, 100);
context.fillRect(100, 100, 200, 200);
```

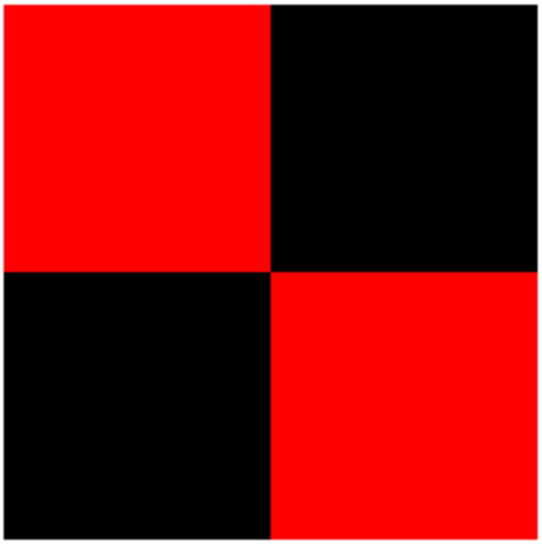


Figure 3.5: Result from code sample 3.10

The canvas element can also be used as placeholder for 3D graphics via WebGL (see subsection 3.2.2.5).

**Drag and Drop** Drag and drop is an event-based mechanism for interactively moving elements in the page. This mechanism allows for moving elements inside the browser and, in devices that support it, move files from the desktop into the browser. The specification provides a powerful touch or click detection API. In the markup, the moving elements must be marked as draggable and possible target element to drop it have to implement three handlers:

**dragenter** determines whether the element will accept the drop

**dragover** is used to give feedback to the user on the drag, like changing the target element color to know that the dragged object is over it.

**drop** is used to process the drop. For example, moving an image to a trash image could make that image disappear from the page, deleting the element.

JavaScript function must control the action. To drop elements, the dropping element must also have the drop attribute.

Drag and drop also allows dragging files from the computer to the browser and vice-versa. Gmail supports this for adding attachments to an email [dB10] .

**SVG** The SVG (Scalable Vector Graphics) is another new element in HTML 5 that allows to include this already adopted image format into the browser. SVG images have the advantage to be vectorial, meaning that they will not lose quality, even if stretched or zoomed in.

The SVG specification itself is also developed by W3C [RSN+08] .

**Forms** HTML forms have also been greatly improved from previous versions. A major improvement is the introduction of native support for client-sided data type validation.

New fields were also introduced. *Keygen* creates a new key pair for security proposes. When the form is submitted, the private key is stored locally in the local key store and the public key is sent to the server. *Object* can represent any external resource. It will be handle in different way depending on the type attribute, being threated either as an image or by an external plugin [Hic10] .

**Offline Web Applications** As previously noted, HTML 5 will introduce enormous capabilities to the development of Web Applications. Obvious issues with them is that they were only available while online. With Offline Web Applications, the browser can download all the needed resources to execute applications offline. This works pretty much as the common browser cache, but with all the needed resources, including HTML, CSS and JavaScript files.

In order to cache the needed files, the developer should create a *CACHE MANIFEST* specifying what files should be kept locally, adapting the HTML file . Every time the user attempts to browse the page, the manifest will be downloaded and the files updated if need. If offline, local files will be loaded.

The ApplicationCache, a JavaScript object, keeps track of the status of the cache, allowing the program to know it's state in the cache. Cache status vary according to data in table 3.2.

Table 3.2: Possible states for ApplicationCache

| Event name | Description |
|---|---|
| checking | The user agent is checking for an update, or attempting to download the manifest for the first time. |
| noupdate | The manifest hadn't changed. |
| downloading | The user agent has found an update and is fetching it, or is downloading the resources listed by the manifest for the first time. |
| progress | The user agent is downloading resources listed by the manifest. |
| cached | The resources listed in the manifest have been downloaded, and the application is now cached. |
| updateready | The resources listed in the manifest have been newly redownloaded, and the script can use swapCache() to switch to the new cache. |
| obsolete | The manifest was found to have become a 404 or 410 page, so the application cache is being deleted. |
| error | Some error occurred during the process of caching the application. |

### 3.2.2 Related Specifications

#### 3.2.2.1 Geolocation

The social component is becoming a requirement on new applications. People want to share information about themselves as much as they want to know about their friends. Sharing user's location is possible thanks to Geolocation.

Geolocation relies on information provided by the host device from WIFI networks, IP address information, GSM tower cell signals or attached GPS devices to acquire the user's location. To access the user's location, an application must be granted permissions in a per-domain level [Pop10] , as it is happening in the android browser in figure 3.6.

Geolocation's specification is prior to HTML 5 and is supported in all major browsers in their latest versions [Sun10] . An example of how to obtain the user's location inside the browser is shown in source example 3.11.

**Source 3.11** Script to obtain the position of the user.

```
if (navigator.geolocation) {
    // success and error are callback functions
    navigator.geolocation.getCurrentPosition(success, error);
} else {
    // function deals with non supporting browsers
    unavailable();
}
```

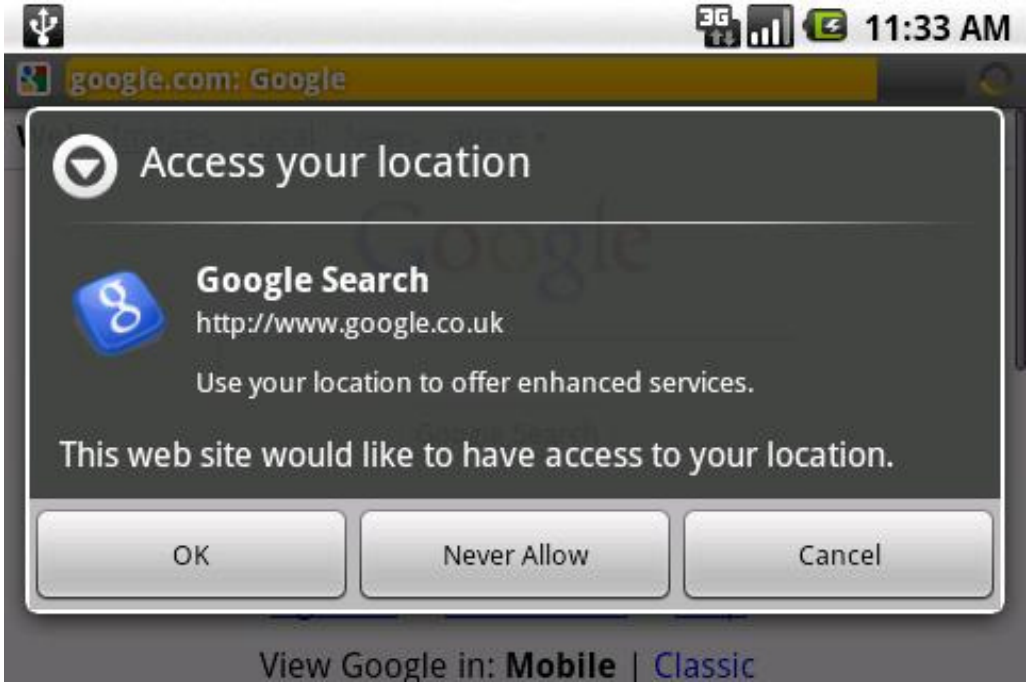



Figure 3.6: An Android phone requesting for user permission to share location information in the browser.

#### 3.2.2.2 Web Storage

Web storage is a local browser storage, similar to HTTP session cookies, that is, as key-values pairs. It has two different mechanisms, sessionStorage and localStorage, as described below [Hic11a] . It maintains data needed by an web application and it has no relation with offline storage, that locally stores files downloaded from web servers.

SessionStorage is designed for scenarios were the user is making transactions in a web site. Cookies can handle this situation, but are unique between the browser and the site (different tabs or windows also have the same cookies). With sessionStorage, multiple data can be kept from the same site, allowing multiple independent transactions to happen. This data is accessible from a single window or tab instead of browser-wide.

The second storage mechanism is localStorage. It work's just like sessionStorage but it stores data for offline usage. Data persistence is kept and must be authorized in a per-domain level. LocalStorage also improves cookies since data is kept locally and is not sent with every request. It is a core component for offline applications, where data objects would reside.

Source example 3.12 shows how to implement a simple localStorage variable to count the number of times the user access a web page.

**Source 3.12** A HTML block and script to show the number of times the user has accessed a page. Count is dynamically created in the local storage and increased each in each view.

```
<p>
You have viewed this page
<span id="count">an untold number of</span>
time(s).
</p>
<script>
    if (!localStorage.Count) {
        localStorage.Count = 0;
    }
    localStorage.Count = parseInt(localStorage.Count, 10) + 1;
    var count = document.getElementById('count');
    count.textContent = localStorage.Count;
</script>
```

#### 3.2.2.3 Indexed Database

The Indexed Database is a local set of key-values databases. Web Storage (subsection 3.2.2.2) is able to store simple key-values pairs but for structured and more complex data that is not enough. Indexed Database complements it, by introducing indexed search by key, iteration through data using cursors and a simple syntax requiring no SQL [PGSM10] .

Indexed Database could be used to store a small database to allow an application to work offline, synchronizing with the main server whenever the application demands or the user is online.

#### 3.2.2.4 Web Workers

Web workers allow multi-threading inside the browser. This specification will allow developers to run scripts in separated thread without slowing down the interactivity in the web page. Workers are not supposed to be used in small scripts, but when performance and memory requirements are demanding and might slow the web page [Hic11b] .

The code implementing a Web Worker should be inside it's own file. Source example 3.13 shows how to create a worker and creates a place to hold for the data it generates. Source example 3.14 calculates prime numbers and returns the value asynchronous message to the main thread that updates the result. More practical examples are web games that need to update a canvas and calculate the game's logic without slowing the rendered graphics.

**Source 3.13** HTML code for the Worker to place the result of the calculation.

```
<p>
The highest prime number discovered so far is:
<output id="result"></output>
</p>
<script>
    var worker = new Worker('worker.js');
    worker.onmessage = function (event) {
        document.getElementById('result').textContent = event.data;
    };
</script>
```

#### 3.2.2.5 WebGL

The WebGL specification aims at providing an OpenGL port to the browsers. This is a 3D accelerated port, using the graphics card [Mar11] .

WebGL opens a completely new world in the Web. It can be used to draw 3D models but it can also be used to play complex games in the browser. The first browser implementing WebGL in a stable release was Chrome [KB11] . A 3D fish tank[6] running in that browser can be observed in figure 3.8. Recent Firefox [Ngu09] versions also have WebGL implemented as shown in figure 3.7.

### 3.2.3 Integrating with Social Media Services

Most social media services have available APIs that allow developers to integrate them in their applications. We've learned how both Facebook and Twitter, two of the most used social networks, with an Alexa[7] rating of 2 and 9 by June 2011, respectively [Ale11] , implement their APIs.

We have observed that in both cases [Twi11, Fac11b] old implementations existed and are now being replaced by improved APIs, both with JavaScript SDK to simplify the access for web developers. Also, both offer plain HTTP requests that are replied with JSON encoded data. New implementations are based in OAuth authentication, which relies on having the user logging in the actual network and allowing external applications to access his data. The application then receives an authentication token with which it can make API calls for data from that user. Without the token, applications can still query both social networks for public data.

[6] From Google repository of WebGL examples available at http://code.google.com/p/webglsamples/.
[7] Alexa monitors activity in web pages, ranking them by amount of accesses with this data.

**Source 3.14** Worker's code. It updates the output element in the HTML using source example 3.13

```
var n = 1;
search: while (true) {
    n += 1;
    for (var i = 2; i <= Math.sqrt(n); i += 1)
        if (n % i == 0)
            continue search;
    // found a prime!
    postMessage(n);
}
```

### 3.2.4 JavaScript Frameworks

In order to ease the programing of web applications, there are many client-side frameworks available. This subsection describes some of those that could pose relevant to the project.

#### 3.2.4.1 jQuery

jQuery was probably one of the first JavaScript libraries to gain high reputation and is amongst the most used JavaScript libraries nowadays. From their homepage, *jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript* [jQu11] .

A major advantage of jQuery is the simplicity of the code created. Source example 3.15 shows two examples of how jQuery can simplify JavaScript programing.

**Source 3.15** How to select an element by it's ID and hide it. First with simple JavaScript and then with jQuery.

```
// Javascript
var e = var e = document.getElementById("example");
e.style.visibility = "hidden";

// jQuery
var e = $("#example");
e.hide();
```

Between web sites using jQuery are some big names such as: Google, Dell, Bank of America, Digg, NBC, CBS, Mozilla, and Wordpress.

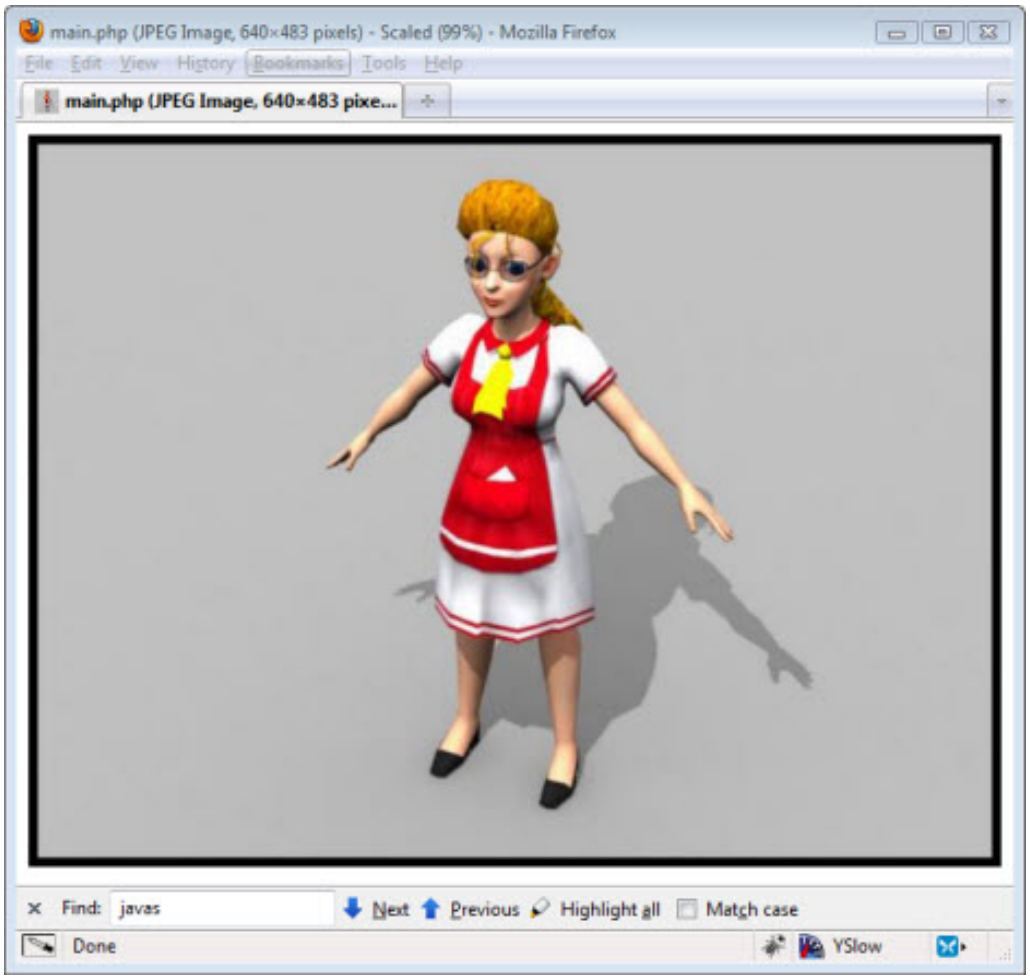



Figure 3.7: A 3D model rendered with WebGL in a development version of Firefox

#### 3.2.4.2 jQuery Mobile

With the appearance of HTML 5 and web applications, jQuery Mobile is a framework for unifying application's interfaces, built on top of jQuery [jM11] . It has a flexible theming engine and it focus it's development on proving cross-browser portability across mobile devices.

The programmer has the simplified job of writing HTML with some data attributes that configure the framework, being the layout automatically applied on page load just by including the framework's files. Highly branded applications, as seen in figure 3.9 can be achieved and deployed across a wide range of devices without the need to develop natively.

#### 3.2.4.3 Modernizr

Modernizr is a small and simple JavaScript library that helps you take advantage of emerging web technologies (CSS 3, HTML 5) while still maintaining a fine level of control over

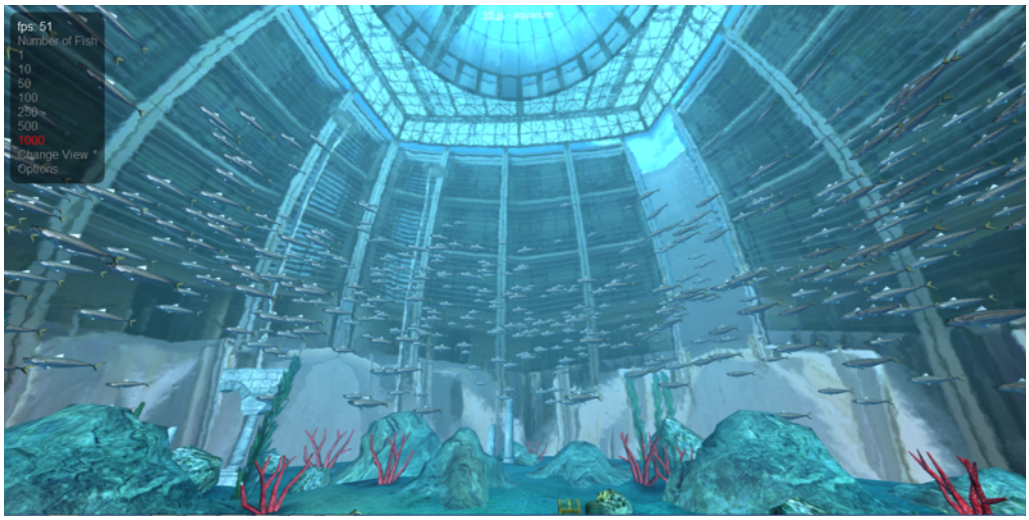



Figure 3.8: A 3D scene of an aquarium built with WebGL. Example from the WebGL samples from Google running with 1000 fishes inside Google Chrome

older browsers that may not yet support these new technologies.

It works by detecting the browser and it's supported features, creating a JavaScript object with boolean values corresponding to each feature it detects. Programmers can then query this object to know if they should use the latest technologies or fallback to older ones. This library will be useful while HTML 5 is not fully supported.

Currently, Modernizr is being used by many influent sites, like Twitter, Burger King, Posterous, or the NFL [Mod11] .

#### 3.2.4.4 Sencha Touch

Sencha Touch is a framework for HTML 5, CSS 3 and JavaScript development with high focus on power, flexibility and optimization. It has improved support for Android and iOS but more platforms are in development.

The framework includes a theme engine to simplify the design process, simple to use animations, automatic resolution adaption and a default icon package and multi-touch handling [Sen11] .

### 3.2.5 Case Studies and Applications

#### 3.2.5.1 Windows 8

Microsoft has recently announced [Ogg11] that the next version of their operative system, *Windows 8*, will have support for developing widgets and applications by using web standard technologies. They have also began to work on improving their Visual Studio to introduce Web Standards in it [gro11] .

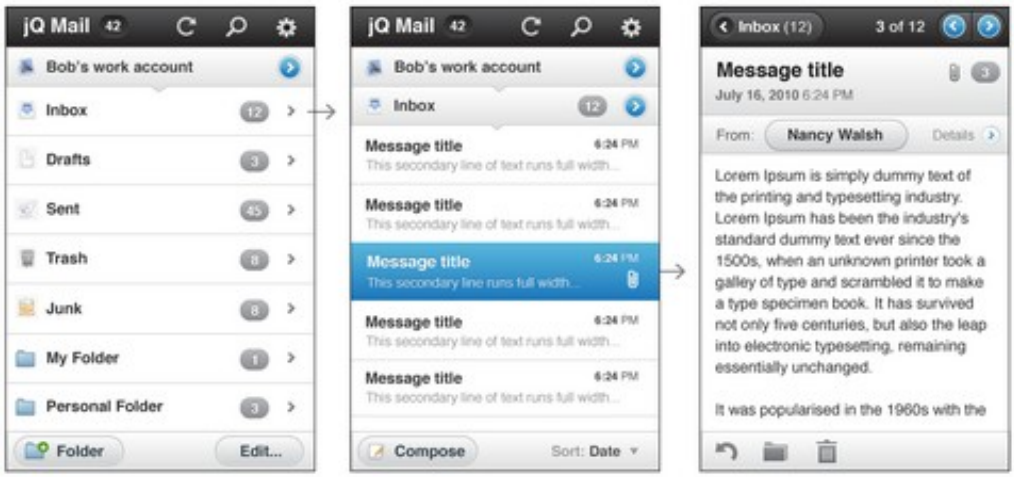



Figure 3.9: An email client built with jQuery Mobile.

#### 3.2.5.2 32 bit x86 Emulator

Fabrice Bellard, QEMU's author[8] has recently implemented a x86 emulator with JavaScript only. This unique achievement was exemplified in a page were only the JavaScript emulating code for the CPU was loaded, alongside with a Linux kernel image that is booted and executed inside the browser window [Bel11] .

#### 3.2.5.3 Chrome Web Store

The Chrome Web Store[9] is a centralized repository of free and paid web applications available to install in the Chrome Browser.

By installing an application from the store, it becomes available in the browser for future use. The applications themselves are built using web technologies (HTML, JavaScript and CSS). Many just load web applications from the developer's site, but it is possible to create any kind of application, offline included.

#### 3.2.5.4 WebOS

As it was previously referred, WebOS is a proprietary mobile operating system developed by Palm and latter purchased by HP.

The whole operating system was designed with the web in mind. Much so that the SDK itself is based in web technologies, using WebKit as rendering engine for the application, meaning that even native applications are developed using HTML, CSS and JavaScript [All09] .

[8] QEMU is an open-source machine emulator and virtualizer. More at http://wiki.qemu.org/Main_Page.
[9] Available from https://chrome.google.com/webstore

#### 3.2.5.5 MathBoard

MathBoard was initially and iPad application by PalaSoftware[10] The application has many subtle animations and a unique realistic look and feel, shown in figure 3.10. Early this year they have ported it to HTML 5 and it is now also available in the Chrome Web Store.

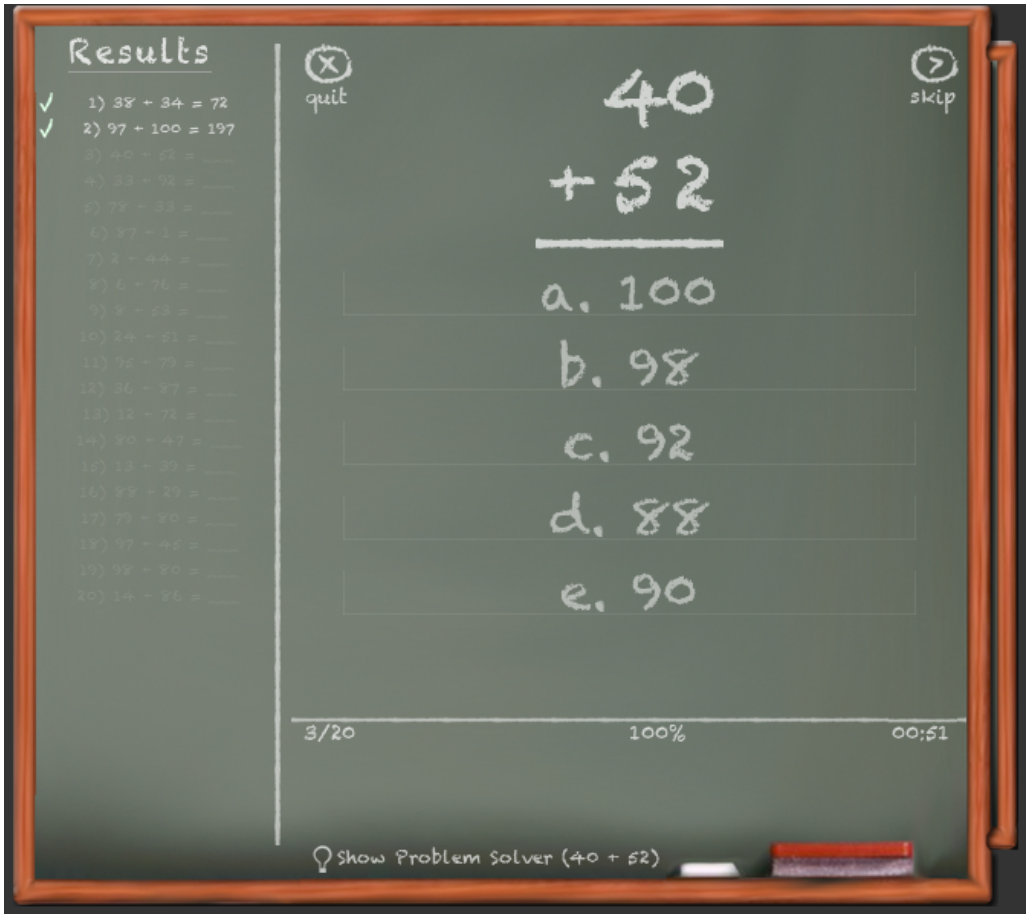



Figure 3.10: Mathboard on the Chrome browser.

According to the authors, the port was a success and HTML 5 is a great platform for creating web, desktop and mobile applications. CSS 3 enabled the look and feel to closely match the iPad interface and local storage allows the application to be used offline [Cho11] .

#### 3.2.5.6 Page Flip Effect

20 Things I Learned[11] was an educational web app created by F-i.com and Google Chrome team to introduce Web concepts such as Web applications or cloud computing.

[10] PalaSoftware is a software development company — http://www.palasoftware.com/.
[11] Available at http://www.20thingsilearned.com/.

They have decided to make the app look and behave like a book and to do so they had to create a page flip method. They have used the canvas API to create the book and render pages. To flip the page, they detect the user is using the mouse on the corner of the page and allow him to drag the corner to flip the page with a smooth transaction.

Whenever the user turns a page, local storage records the page he is in. If the app is closed, when it is re-opened the user is prompter if he would like to continue reading from his previous page [Hat11] .

#### 3.2.5.7 Drag and Drop Download in Chrome

Box.net is a file hosting service. As soon as they've heard of Drag and Drop into and from the browser, they wanted to add that feature to their service. They are using Modernizr to verify browser compatibility first.

Implementation revealed some troubles. In a design first iteration, it was impossible to know if the file was being moved inside the browser (to another folder) or if it was being moved out of the browser to request it to be downloaded. Also, they wanted to avoid the user downloading files by accident. Problem was solved by altering the dragging script so that it would behave as a local move when dragging only with the mouse and as a request to download whenever the control key was pressed with the mouse.

Using this simples approach, appears just like any other folder from the computer, allowing to move the files not only inside it, but also to interact with local folder also with drags, while pressing the control key.

#### 3.2.5.8 Solitaire Offline

Developed as a demo for Sencha Touch (described in subsection 3.2.4.4), it is an implementation of the know Solitaire game, implemented with web technologies. The application needs to be loaded only once and it is then cached for offline use. It is freely available online[12].

Graphics are absolutely thrilling and using it feels just perfecti (fig. 3.11). Using it in the desktop feels like a native iPad app, with smooth CSS based animations and colors. If the browser is closed, the game state is saved and resumed once the user gets back on it [Spe10] .

## 3.3 Web vs Native

Questioning the adoption of web standards over native development is a common and valid question. There are many misconceptions related to the subject. The claims bellow

[12] http://touchsolitaire.mobi/app/

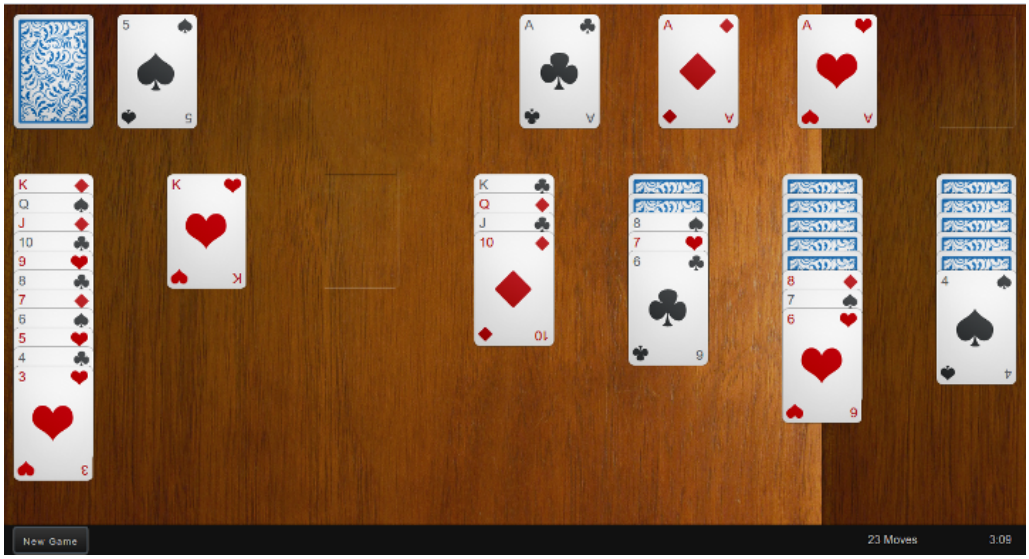



Figure 3.11: Offline Solitaire app built with Sencha Touch

are based on the article from Michael Mahemoff [Mah11] for HTML5ROCKS[13].

### 3.3.1 Common Technologies Comparison

**API Richness** While novel web technologies are quickly evolving, they will hardly ever be able to access most of every device's API, such as access all buttons (like Android's search or volume buttons) or request for enabling hardware. For situations where access to those API's really are needed, frameworks like PhoneGap[14] can encapsulate a web application in order to build a native application that executes inside a browser but can make operative system API calls.

**Performance** Until now it was impossible to take advantage of multi-core systems or GPUs inside the browser. With the introduction of CSS 3 GPU accelerated animations, web workers or WebGL, web applications can make the browser use the available hardware in a better way, allowing the creation of faster web applications. Furthermore, JavaScript engines have been greatly increasing in performance, making JavaScript more attractive even for non-web context applications.

**Monetization** Mobile marketplaces are filled with low cost applications that easily be bought by their users. Native apps also usually have in-app payment systems that can allow them to create a steady, longer-term income for developers. However, any payment

[13] Google sponsored web page promoting HTML 5 and related specifications at http://www.html5rocks.com/.

[14] PhoneGap allows to create hybrid applications for mobile devices such as Android, iPhone or Blackberries, by packaging a web application. This will allow to add native code to web applications or publish web applications in app markets. Read more ate http://www.phonegap.com/.

done from within a service provided by an application market has a fee for the company behind the market and that's clearly negative point for developers. Publicity is the one system that both native and web share and that might pose a good, but intrusive, way to monetize applications. On the Web, payments services exist that avoid transfers commissions. To reach the mobile markets, as previously suggested, web applications can still be wrapped into hybrid apps that can be submitted to mobile markets.

**Discoverability** Mobile devices have several ways to share applications but there's no better system to share information as it is the web. Online social networks such Twitter or Facebook can have huge influence in making a web application popular since it can be virally spread and accessed immediately from everyone with a browser, without the need for mobile markets or local installs.

### 3.3.2 Conclusion

It is hard to actually claim that web is better than native and vice-versa. Choosing web technologies is a decision that keeps summing advantages and by using it to create hybrid applications most issues might get solved. Both have their advantages and it is up to the developer to make his choice.

## 3.4 Summary

Despite the already long life of the Web, it's expansion has been constant since it's dawn. Also, more than ever, efforts are being put into standardizing it, by creating concise specifications, commonly grouped under the umbrella term of HTML 5 (that is actually the name for the latest version of the HTML standard) and having the support of browser developers to actually implement them into their engines.

Also, many developers are putting effort into improving existing tools and frameworks to allow others to create improved code for their own web applications. JavaScript plays a major role in this new Web, by bringing interactivity into pages, allowing them surpass the older static pages by actually becoming fast, reliable and complex web applications.

More than presenting a bit of the history of the web, this chapter overviews some web applications and frameworks that let us understand how the web is changing for a better, more capable place were we can find all kinds of applications.

# Chapter 4

# Designing a Storytelling Application for Older Adults

From chapter 2 we've learned that older adults are open-minded to new channels of communication with their friends and relatives, even by adopting digital devices. Still, it is also known that they are subject to age related changes that can limit adopting those devices, as explained in section 2.1.

This project aims at discovering if developing an application for the older adult to share their life-stories and casual thoughts with his friends and family is feasible. Also, it studies the state of the art of current web technologies, probing them on the readiness for developing fully featured applications. We want to create a starting point to verify if online communications can increase the well-being for older adults.

## 4.1 Specification and Methodology

As stated in section 2.3, this project is a joint collaboration between two master students and sponsored by Fraunhofer Portugal. The interaction was conceptualized by both students. However, the design and user research was conducted by Pedro Tenreiro and the implementation was kept by the author of this work

This section briefly overview the design process. Further details can be obtained from his thesis [Ten11] .

During the implementation phase, we embraced an agile process. We have both been actively working on the project, each responsible for two different areas, but still collaborating. While I was studying how to implement the application, Pedro was already doing user research with older adults. We regularly synchronized in order to maintain cohesion agreement on the decisions in both ends.

### 4.1.1 Participants of the HCI study

Due to the specificities of older adults, we have had the need to guide our research with the presence of older adults in order to set our base requirements. To do so, we have interacted with older adults from the day care center of Foco, at Boavista. The center currently hosts about 25 older adults, with an average age above 75. Health problems are common amongst them — we have registered 3 persons with Alzheimer, one with Parkinson, many with mobility impairments and most of them with earing or vision limitations.

#### 4.1.1.1 Interacting with the Older Adults

While working with the older adults, we have applied HCI methodologies to obtain detailed descriptions of what the system should do and how it should behave and look. Namely, User-Centered Design[1] was performed in order to better comprehend the user's characteristics and acceptance to new concepts. By doing so, we've intended to both understand what features should be considered for the application as well as how to design them in order to accommodate a smooth learning curve for the users.

#### 4.1.1.2 System Requirements

Requirements are the result of discussions held by the project members and field study. From the first, we though how online social networks are relevant to, and used, by many in their daily life. We then created a list of actions that should be possible to take in the application and generated ended up with figure 4.1.

Further than the requirements described in the diagram of figure 4.1, some more general requirements were listed to guide our work:

- Create a new communication channel for older adults to communicate with friends and family;
- Design an application that not only would be useful for them, but also interesting and motivating to use;
- Integrate the application with a widely adopted online social network.

Also, technical requirements were imposed before development started:

- The application was intended to be run from the Web;
- It should integrate with a major online social network where our users' friends probably already have an account;

[1] User-Centered Design is an HCI methodology that helps the designer to guide is work according to information gathered from a subset of his targeted audience [DFAB04] .

- Recent internet technologies commonly denominated by HTML 5 should be explored and applied whenever feasible;
- Should be compatible with recent smart phones browsers and tablets.

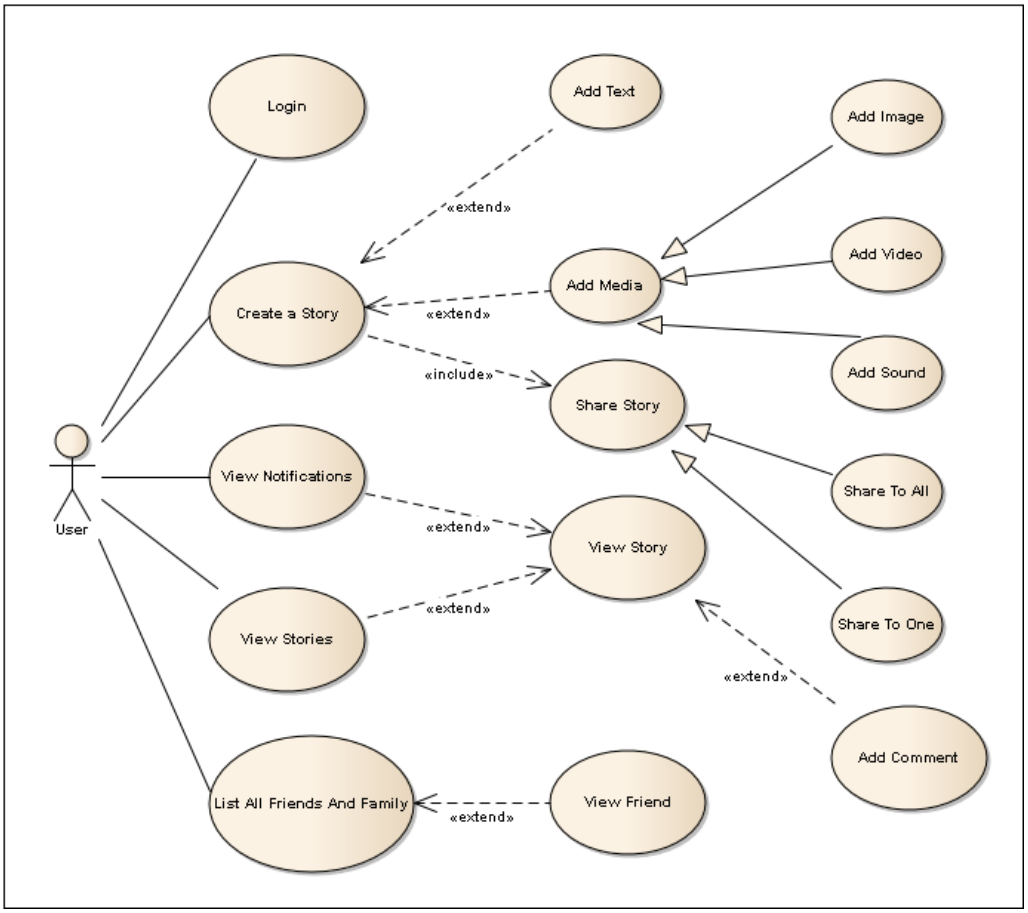



Figure 4.1: An use-case diagram from representing the possible user interactions with the system.

## 4.2 Summary

The process of designing interfaces can follow methodological approaches that guide the process into achieving better results. User-Centered Design was the main methodology we have adopted. While designing for older adults these procedures become mandatory due to the low technological capacities generally observed in this age group — every possible action to ease the application's adoption should be worked on. This chapter listed not only our requirements but also explained how we reached them and what methodologies were used while designing this project for older adults. During the design process we were also able to refine our requirements and plan our project's implementation.

# Chapter 5

# Implementation

In the previous chapter we describe our requirements and design methodology. In this chapter we describe our implementation process, including server and client architecture, used protocols, APIs, external tools an more.

## 5.1 High-level Architecture

The architecture of the application can be divided into two parts: the application itself, that runs on the user's browser and the server that generates and transfers the application to the client's browser. Figure 5.1 graphically shows how the two parts are composed, as well as how they interact with the outside data source, Facebook.

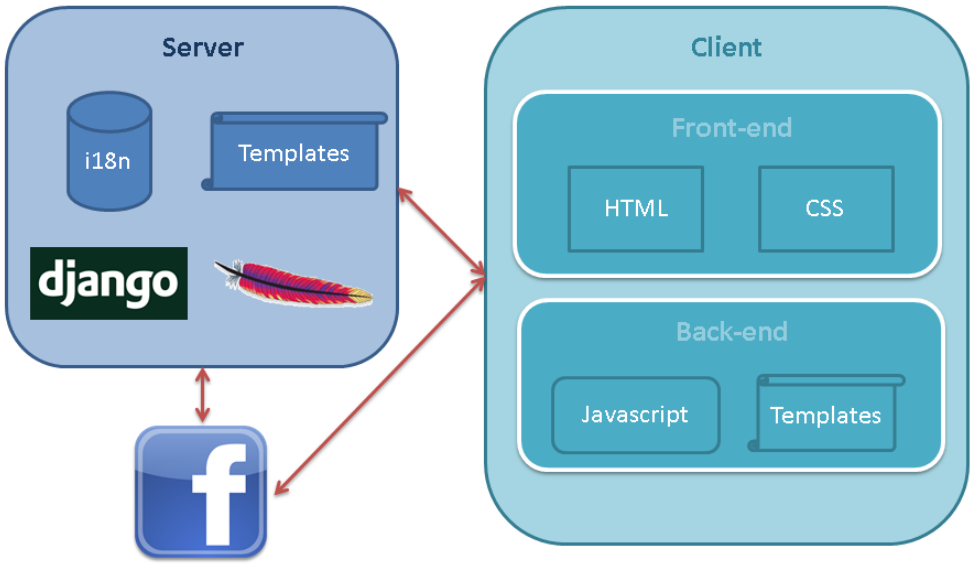



Figure 5.1: High level architecture. Arrows represent HTTP communications.

## 5.2 Client

The client side of this application is mostly autonomous. Once downloaded from the server, there's no further need to communicate with it unless there's the need for uploading videos or photos (details in section 5.4). Using JavaScript, new pages are locally rendered, as explained later in this section.

### 5.2.1 Components

The downloaded application is composed by a single HTML file and a number of JavaScript and CSS files. The application heavily relies on JavaScript to communicate with Facebook and update the front-end. Bellow is a description of the external components used.

#### 5.2.1.1 jQuery

jQuery needs no introductions to anyone who programed for the web in the latest years. It is a *fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development* [jQu11] . It is the heart of our applications and is used in an huge variety of ways, such as creating, editing or removing and adding new elements or contents in them, applying CSS and most of all selecting elements using the selectors API, implemented in jQuery according to the CSS selectors specification [jQu] .

#### 5.2.1.2 jQuery Mobile

The framework presented in section 3.2.4.2 is responsible for transforming our HTML markup into styled web pages by adding attributes to the markup used by the framework's CSS. It's use provided a lot of workarounds for improve compatibility with a wide range of browsers out of the box. Taking advantage of it's capabilities for embedding more than one page in a single HTML page by putting them in individual *divs*, we were able to include all the pages into a single HTML document, reducing the required number of connections to the server. Also, once loaded, except for uploading files, all connections are made from JavaScript directly to Facebook.

Refer to appendix A for an overview and description of each interface generated with the framework.

#### 5.2.1.3 jQuery Templates plug-in

jQuery Templates allow for client-side templating, allowing and easy to use dynamic way of changing the user interface. It is hosted by Microsoft and developed by Boris Moore

[Moo] , this plug-in allows client side templating using JavaScript. How we have used the client-side templating system is further detailed in section 5.4.1.

#### 5.2.1.4 jQuery Form plug-in

Working with forms through JavaScript and AJAX requires the serialization and validation of forms data. This plug-in simplifies both those processes. Since the application is only loaded once, and all pages are then locally rendered, using AJAX is mandatory to prevent from refreshing the browser.

#### 5.2.1.5 Pretty Dates

Trying to provide a friendlier way to display dates, we have altered the published code by James Padolsey [1] in order to be both translatable and compatible with Unix Time [2] and RFC 3339 [KN02] . The plug-in should be used and produces the output as seen in source block 5.1. The altered source code is available in appendix B.

**Source 5.1** Source code example of how to use the prettyDate function and respective output commented.

```
prettyDate("2011-05-10T22:24:17Z") // "5 weeks ago"
prettyDate("2011-06-09T15:40:17Z") // "Yesterday"
prettyDate("2011-06-10T10:40:17Z") // "5 hours ago"
prettyDate("2011-06-10T15:40:17Z") // "just now"
prettyDate("1307717015508") // "just now"
```

## 5.3 Server

This section describes the server, used by the client described in the previous section for retrieving the web applications and publish files in the online social network.

The server consists of an installation of the Django Framework[3]. Being a portable framework, it can be installed in many different environments. For development the embedded test server was used. Later on, for testing purposes we have deployed an instance of the Apache Web Server[4] and plugged the framework to it using modwsgi[5]. This second setup closely resembled how a production server could be setup.

Using the Django framework has proven very advantageous, namely in:

[1] Original code available from James Padolsey's blog, http://james.padolsey.com/JavaScript/recursive-pretty-date/.

[2] Unix Time is a time measuring system that counts the seconds past since midnight January 1st, 1970, in UTC time.

[3] Popular portable web framework based on the Python programming language. Read more in https://www.djangoproject.com/.

[4] Open-source web server.

[5] Python WSGI adapter module for Apache.

**Template engine** allowed to separately code each block of the application's HTML. The Django templating system allows the creation of page templates composed by multiple extensible blocks. Using that type of templating, creating a new page was a mere case of extending the template page and adding the desired data to it. This greatly reduced the amount of HTML code in favor of code reuse.

**Translations** The Django templating system, allied with a special utility for handling JavaScript files, allowed to translate the application using PO files. The chosen development language was English but it was further translated to Portuguese. The language the application is used in can be automatically chosen by identifying the user's language on Facebook after login.

**Easy deployment** due to the high portability in the Python language, the server can be installed in a huge variety of hosting environments.

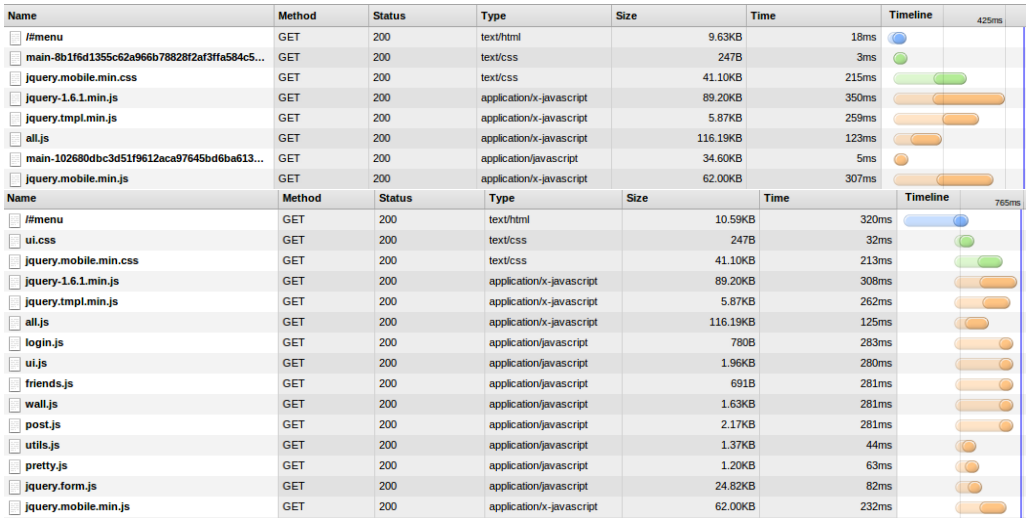
To aid the development process, some python extensions were used with Django:

**Django-mediagenerator** improves HTTP transactions and is a simple way to improve loading time, as well as reducing required network traffic. In fact, while the HTML code consist in the most relevant piece of any web application, the 80/20 Performance Rule [The06] applies to most cases, being most of the loading time spent on images and separated files instead of the HTML itself. To provide a faster experience, using Django-mediagenerator, we were able to automatically compress all CSS and JavaScript files into two compressed files, reducing the number of requests to the server from 9 to 2. Furthermore, the extension is also able to minify[6] the two generated files by using an external minification tool. To accomplish it, we've configured the plug-in to use Yahoo's YUI Compressor[7]. This has proven successful using a server in the local network has seen in figure 5.2. On a remote server it's expected to further reduce loading time since there are fewer requests to the server and establishing the connection takes most of the time when requesting small files.

**Django-extensions and Werkzeug** were used together to enable an improved development server, with better debug capabilities, allowing an in browser interactive debug console whenever a server-side error occurred. The debug console could be used to inspect variables at the time of the error, as well as to interactively execute commands in the environment, making it easier to perceive the error.

[6] Minification is a technique applied to JavaScript and CSS documents for compacting the files by removing unused spaces or characters, without altering their behaviour.

[7] YUI Compressor is a free compress tool for minification of JavaScript and CSS code.



| Name | Method | Status | Type | Size | Time | Timeline 425ms |
|---|---|---|---|---|---|---|
| /#menu | GET | 200 | text/html | 9.63KB | 18ms | |
| main-8b1f6d1355c62a966b78828f2af3ffa584c5... | GET | 200 | text/css | 247B | 3ms | |
| jquery.mobile.min.css | GET | 200 | text/css | 41.10KB | 215ms | |
| jquery-1.6.1.min.js | GET | 200 | application/x-javascript | 89.20KB | 350ms | |
| jquery.tmpl.min.js | GET | 200 | application/x-javascript | 5.87KB | 259ms | |
| all.js | GET | 200 | application/x-javascript | 116.19KB | 123ms | |
| main-102680dbc3d51f9612aca97645bd6ba613... | GET | 200 | application/javascript | 34.60KB | 5ms | |
| jquery.mobile.min.js | GET | 200 | application/x-javascript | 62.00KB | 307ms | |

| Name | Method | Status | Type | Size | Time | Timeline 765ms |
|---|---|---|---|---|---|---|
| /#menu | GET | 200 | text/html | 10.59KB | 320ms | |
| ui.css | GET | 200 | text/css | 247B | 32ms | |
| jquery.mobile.min.css | GET | 200 | text/css | 41.10KB | 213ms | |
| jquery-1.6.1.min.js | GET | 200 | application/x-javascript | 89.20KB | 308ms | |
| jquery.tmpl.min.js | GET | 200 | application/x-javascript | 5.87KB | 262ms | |
| all.js | GET | 200 | application/x-javascript | 116.19KB | 125ms | |
| login.js | GET | 200 | application/javascript | 780B | 283ms | |
| ui.js | GET | 200 | application/javascript | 1.96KB | 280ms | |
| friends.js | GET | 200 | application/javascript | 691B | 281ms | |
| wall.js | GET | 200 | application/javascript | 1.63KB | 281ms | |
| post.js | GET | 200 | application/javascript | 2.17KB | 281ms | |
| utils.js | GET | 200 | application/javascript | 1.37KB | 44ms | |
| pretty.js | GET | 200 | application/javascript | 1.20KB | 63ms | |
| jquery.form.js | GET | 200 | application/javascript | 24.82KB | 82ms | |
| jquery.mobile.min.js | GET | 200 | application/x-javascript | 62.00KB | 232ms | |

Figure 5.2: Comparison between load times of core components with and without JavaScript and CSS compression.

**Facebook's python SDK** was originally created by Facebook's team, now forked and maintained by the community, due to upstream abandonment, using Github[8]. During the development of this project, we have made some contributions to the SDK code, being the most relevant the ability to upload media (photos or videos) to the social network, amongst some other minor fixes[9].

## 5.4 Implementation Details

### 5.4.1 Client-Side Templates

In order to keep the application with an app-like flow, we had to reduce waiting times. To do so, as previously stated, every needed files are loaded at the initialization and there's no further need to request pages to the server. To do so, we use jQuery Templates.

jQuery Templates allow us to specify templates to be applied to any element in the HTML document. Each template is transfered inside the HTML page as a script block, with an associated ID for future identification, as exemplified in source code 5.2. To use it, the library is called with a JavaScript object that it uses to populate the template with values. The result can then be applied to an existing HTML element.

Our client templates are generated themselves by Django templates. This has proven to be an issue since both identify variables for substitution by being between double curly brackets. To work around this issue, we have replaced the double curly brackets by square

[8]Github is an online service providing git repositories for collaborative project development.

[9]Fixes were uploaded to subssn21 fork of the upstream facebook sdk version in https://github.com/subssn21/python-sdk.

**Source 5.2** Example of a jQuery Template block used to generate a list entry with friend details. Passed variables are *index*, a number and a JavaScript object, *item*.

```
{% load i18n %}
<script id="list_friends_tmpl" type="text/x-jquery-tmpl">
    <li>
    <a href="#" onclick="showFriend(${index});">
        <img src="http://graph.facebook.com/
            ${$item.id}/picture?type=normal"/>
        <h3>${$item.name} </h3>
    </a>
    </li>
</script>
```

brackets on the server side and substituted them once the page is ready by curly brackets again, using a regular expression, making them usable as expected by the library.

### 5.4.2 Client-side translations

Using Django's *gettext* JavaScript implementation, we were able to mark our strings inside each JavaScript file for translation. Django automatically scans all JavaScript files and mark all strings inside a *gettext* function for translation. Translation is then done the same way it is done for Django's templates. On the client side an extra JavaScript file implementing the *gettext* function is loaded; it returns a translated string when called with the original string as parameter. In the same JavaScript file is a dictionary mapping all the original strings to translated strings.

### 5.4.3 Talking with Facebook

**An Online Social Network** Facebook is the most adopted online social network worldwide. It's current number of active users is above 500 million. It is translated in more than 70 languages and about 70% of the users are from outside the United States. It reaches users both by the full and mobile web interface and mobile applications available for most mobile platforms [Fac11c] . We have chosen to integrate with Facebook due to the huge number of active users in it, increasing the chances that our users will already have friends in it.

**Facebook's API** relies on the HTTP protocol. HTTP requests are created by choosing the appropriated HTTP method from POST, GET and DELETE and crafting the appropriated URL that accesses the desired data [Fac11b] . The API itself follows the expected HTTP usage. The POST method is used for sending data to the server. GET requests data from the server. The DELETE method request deletion of an object to the server. Since these

Table 5.1: Implemented templates using jQuery Templates. The input is a dictionary object from JavaScript whilst the output is the generated HTML.

| Name | Input | Output |
|---|---|---|
| List Empty | — | Message reporting empty list. |
| List Friends | friend | Friend's photo and name. |
| List Notifications | notification | Notification type (comment or post) with author's photo and name. |
| List Wall | post | Post message with author's photo and name. |
| Show Friend | friend | Friend's photo and name with available personal detail. |
| Show Post | post | Author's photo and name, post's message, including video or image, if any. Also comments on the post and form to comment on it. |

are native HTTP methods, their well documented [FGM+99] and plenty of examples of how to use them are available online. Initially a REST server handled all Facebook API's requests but a more recent implementation, called Graph API, has deprecated the previous implementation in favor of JSON oriented responses. At this point, the Graph API is not yet as feature rich as the previous REST API was.

#### 5.4.3.1 Authentication

Authentication is done using OAuth 2.0[10]: after the user authenticates with the service, a token is acquired from the server and is then used on following requests for identification while valid. OAuth tokens have a time to live and must be renewed after expiring [Fac11a] [E. 11] .

After authorization is granted, the application can access all the information available on the user's profile, his posts, photos, events or friends. Also, it is possible to publish new content. Source example 5.3 shows how to publish a new message using curl[11].

**Source 5.3** Example of how to post a message, after acquiring the authentication token as the user *username*.

```
curl -F 'access_token=...' \
    -F 'message=Testing facebook API.' \
    https://graph.facebook.com/username/feed
```

#### 5.4.3.2 Javacript SDK

An SDK is also available for asynchronous interaction with the API. This SDK can be included in any HTML document and it handles communication between web pages and Facebook by recurring to JavaScript objects and functions. Source block 5.4 exemplifies

[10] OAuth is an open standard for authorization

[11] A terminal application able to craft HTTP requests.

how to retrieve a user's public personal information using the SDK, showing the JSON string that is returned.

**Source 5.4** Using Facebook's JavaScript SDK to retrieve a user's information based on the user name. Commented is the JSON string returned by Facebook.

```
FB.api('/john', function(data){
    var text = JSON.stringify(data, null, '\t')
    console.log( text );
});

/*********

{
    "id": "779695190",
    "name": "John Chan",
    "first_name": "John",
    "last_name": "Chan",
    "link": "https://www.facebook.com/John",
    "username": "John",
    "gender": "male",
    "locale": "en_US",
    "updated_time": "2011-05-23T02:02:28+0000"
}

**********/
```

The JavaScript SDK communicates with the Graph API and, as such, it is limited to it's implementation. This results in some limitations for the user. One example is the inability to currently POST files to the Facebook server.

To load the SDK into our application we needed to include the JavaScript file available from the Facebook servers. This file is the only external JavaScript file in our application and can not be kept local due to cross-site scripting prevention techniques included in browsers — JavaScript can only receive data from the same domain from where it's code was download [Moz10] .

**Querying for Data** Facebook's Graph API can be queried in several ways using the SDK. The most straightforward one is by using the *api* function. It expects a string identifying the query and a function to be asynchronously executed once the server responds.

An alternative and more advanced way to query Facebook is by using FQL (Facebook Query Language). Similar to SQL, it allows to specify optimized queries that can be executed faster and that only return the exact needed data, opposed to the above approach that commonly returns all the available data related to the queried object.

#### 5.4.3.3 Posting Data

The *api* function from the SDK also allows for posting data asynchronously by specifying the HTTP method as POST and providing a dictionary with the data to be posted. We use it to post comments and wall posts. An example demonstrating how to post a comment to a chosen wall post is exemplified in source example 5.5

**Source 5.5** Posting a comment to a post with ID passed as *post_id*. The SDK expects the two extra arguments identifying the call as a HTTP POST connection and settings the POST arguments in the *msg* JavaScript object.

```
function post_comment(post_id, comment){
    var msg = {'message':comment};
    FB.api('/' + post_id + '/comments',
            'post',
            msg,
            function(response){
                if(!response || response.error){
                    // Something went wrong, handle it
                } else {
                    // Success, do something
                }
            }
            );
}
```

While most of the requests are done with serialized data as a JavaScript dictionary object, serializing files, in this case photos or videos, is still impossible to do. This special case had to be handled differently and is specified in the next section.

### 5.4.4 Uploading Files

Being Online Social Network's main purpose to enabling their users with the ability to share ideas and experiences, sharing photos and videos is an essential part of it. It is currently impossible to load local files and serialize them into a JavaScript object so that it can be posted to Facebook using the previously mentioned SDK. There were two approaches to solve this problem.

**Post the form with an iframe** consists in creating an iframe inside the DOM and set an HTML form to have it as *target*, while having the Graph API as *action*. By doing so, an HTTP POST would be sent to the Facebook servers and the request would be returned and made available inside the created iframe. We cannot simply post to Facebook since that would make us leave the current page.

There are some constraints to it though, since the iframe's content would be filled with data returned from the Facebook domain. The same-origin policy would prevent us from reading the response. We would end up with a working posting method but be unable to guarantee that the data was currently sent.

**Have the server handle the upload** increases the complexity but is a more reliable way to post files. It requires a server-side handler to receive the post data, including the file, and then have it uploading the file to Facebook and return the operation's result to the application. To prevent changing the page, this is done similarly to the previous approach, inside an iframe, but we're able to read the response since we're posting to the same domain from where our JavaScript are loaded from.

Implementing this method requires the client to provide the OAuth's authentication token obtained from Facebook to the server so that the server has permissions to upload the image to Facebook with the user's credentials. The token is saved in the browser at login time as a cookie, being sent with the request to the server.

Implementing the upload of media files to Facebook was possible by using Facebook's python-sdk, to which we had to contribute since there was no support for uploading these media files as previously stated. An overview of the contributions is available in appendix B.

We have opted for having the server handling the upload so that we had control over the success of the upload.

### 5.4.5 Bootstrapping

After all files are downloaded and the user is presented with the login window, bootstrapping procedures are executed. These procedures are executed after page load, detected and asynchronously initiated by jQuery's *.ready()* function. These are required to instantiate the described items in this section and transform the static HTML document into a dynamic web application.

**Templates** require the validation of all jQuery Template blocks in order to substitute the double square brackets by double curly brackets, a workaround done in the server side, needed to prevent Django and jQuery templating systems incompatibility.

**Facebook SDK** is the next step and starts the authentication process with Facebook. This action is done asynchronously to wait for the response from their server. If the user still has a valid session, the application automatically jumps to the main menu. Otherwise, it remain on the login screen, allowing the user to login with his Facebook account.

**jQuery Mobile** is automatically setup just by including it in our JavaScript files. Still, we configured it with some options, namely *a*) disable form submission handling, so that we can do this ourselves; *b*) disable URL hash listening, so that the page is not changed if the user presses the back button by accident; and *c*) set the initial hash to /.

**Updating data** is done and scheduled at boot time so that the screens are filled with the most recent data available from the user. The schedule is done using *setTimeout* and the update happens every 60 seconds — this is an initial value that's subject to change after evaluating user's behaviour. If something is posted, then the updating procedure is executed, reseting the timer.

## 5.5 Known Issues

Currently there are a number of known issues, mostly due to the new Facebook's API (Graph API) limitations.

**Commenting issues** on the API have been recently reported on the Facebook forums [rt] . These appear to occur from time to time and we were unable to find any feedback by the Facebook's team regarding them. The message *(#100) Error finding the requested story* is returned whenever the attempt to post a comment fails. We've handled this issue by alerting the user that it is impossible to post the comment at that time, suggesting him to retry later.

**Large lists** might be slow to handle in less capable devices. The worst case happens when the user has many friends and lists them. When pagination is not used, a very large number of friends might appear to freeze the browser while the list loads. To work around this issue, pagination was included in the application. Still, testing with the older adults revealed that it would increase the system complexity, being currently disabled. Still, if further testings reveal it as usable, it can be applied to any of the existing lists.

## 5.6 User Interface Design

Following the decisions from chapter 4 we have forked [12] jQuery Mobile and made the necessary changes to it. Several CSS changes were need so that the overall looks of the applications used most of the available space to show bigger font sizes. Figure 5.3 shows how the application lists the friends from the current user. Remaining interfaces are available in appendix A.

[12] Forking is the process of cloning the source code from a project and adapting it to different needs.

## 5.7 Summary

The combination of chosen technologies allowed us to successfully create a client-server architecture that takes advantage of latest browser technologies and is able to generate it's own front-end code from data acquired from Facebook. The implementation is also characterized with an adaptive interface providing the required portability. Getting back to the community, several bug reports and some patches were submitted to some of the adopted projects, like the support for video and image uploading to Facebook's python SDK.

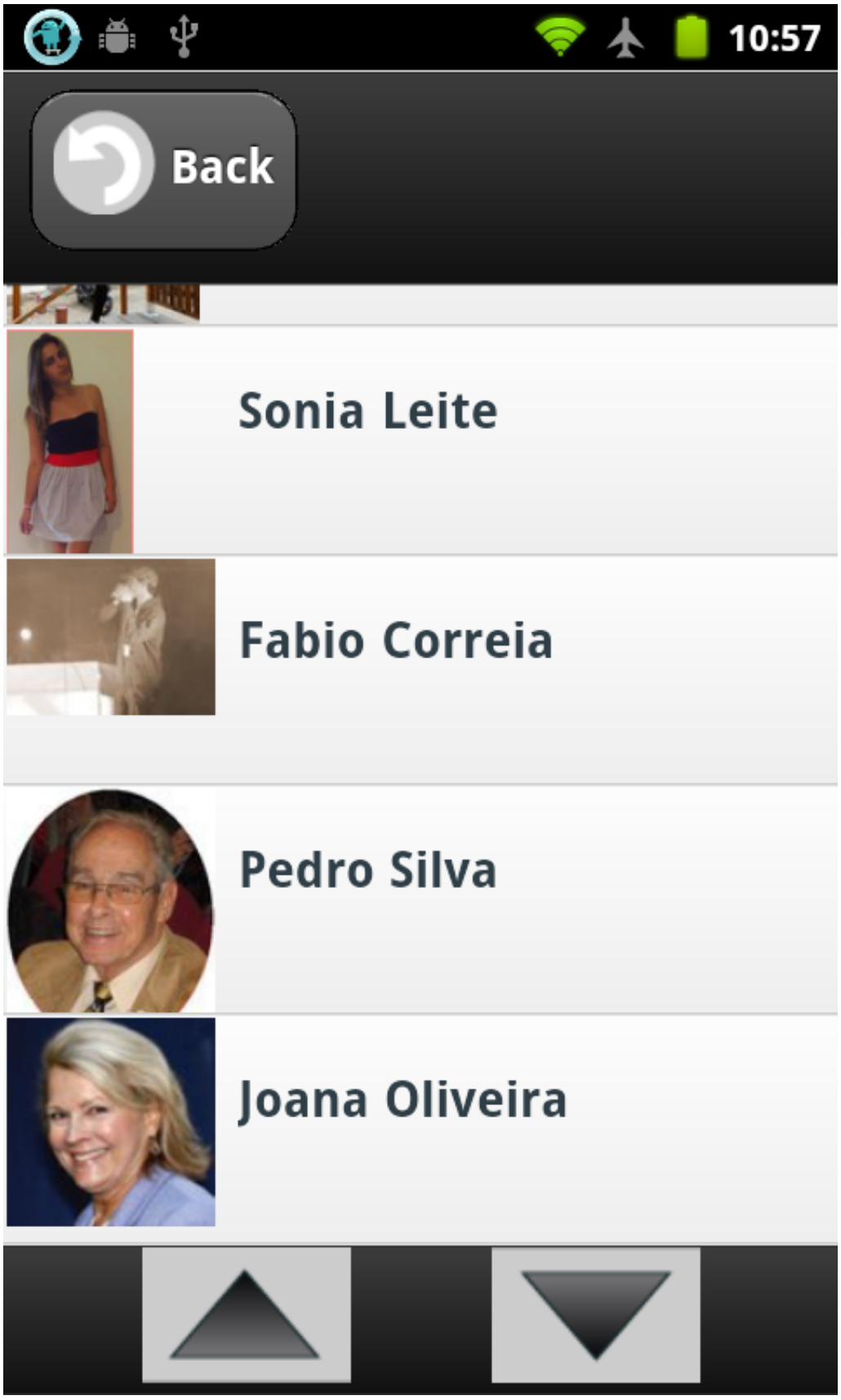



Figure 5.3: List the user's friends in a HTC Desire.

# Chapter 6

# Results and Discussion

We believe that an application is only as successful when it is well received and widely adopted by it's target audience. From that, we infer that there's the need to provide prototypes that can be tested by target users in order to receive feedback and ideas for improvement, as well as detecting possible issues before release. This chapter describes our final usability test with the older adults. We also discuss some conclusions generated from our approach to this project.

## 6.1 Usability Tests

We needed to validate our design and gather feedback from the target audience to guide our work. To do so, together with the participants from the Foco day care center, we have continuously tested our prototypes, even when these were only in a paper prototype form in order to inform our design. We started by using hand sketches and later computer designed images to simulate the real application with the older adults, to whom we asked to tell us how they would perform some tasks. This was an User-Centered Design approach that allowed us to get feedback early on.

By the time we had a working prototype we performed a last, more thorough, usability test. There was a total of seven individual tasks in this test. Each task was started from the main menu in the storytelling application, whenever not otherwise specified, so after each task the user was requested to get back to the main menu.

For each task, we explained the user what we wanted him to do and let him try to do it by himself, without guidance. Whenever he asked for help, we clarified the task's objective and let him know that we actually want to know how easy it is for them to do it without help so that we can verify the design and find common difficulties.

**A - Find and view a person's page with a given name** : the user had to enter the list of friends, search for a person with a given name view that person's details.

**B - Return to the initial menu** : following the previous test, we have evaluated how complex it was for the test subjects to return to the main menu.

**C - Count the number of recent comments** : we asked the user to find how many new comments were available in both his and received stories. He had to be able to distinguish new stories from new comments in the notifications list.

**D - Find a specific story that was created at a given time by a certain person** : we told the test subject that a story had been posted at a given time and from a given friend and asked him to find that story and to open it. This test finished in the story page since it is the starting point of the following one.

**E - Comment on a story** : this was a continuation of the previous test, where we asked the user to write a comment on the story he had found.

**F - Create a story** : by selecting the share option on the menu and then create a story by writing a text.

**G - Add an image and share the story** : we asked the user to attach an image to the story he just created and then share it, getting back to the main menu afterwards.

We have created several Facebook accounts for the tests since many users did not have accounts and we needed them to already have content on the accounts to perform the test. This was not subject to testing since we expect that those who adopt the application have someone to set the account for them.

The test was held in the common area of the Foco daycare center. We have applied the same test with two types of devices, an iPad tablet and an HTC Desire smart phone (refer to appendix A for the devices' specifications). Seven people were willing to perform the usability test. They were slitted into two groups, four did the tests in the Desire, while the others used the iPad. Background knowledge of every test subject relative to the application was similar, having been shown previously some paper prototypes and explained what we were trying to achieve. No subject had experience using computers or other similar devices.

We interacted with each older adult individually, preventing the others from seeing the application until it was their turn to perform the test. While performing the test, we ranked each task execution according to how complex it was for them to execute it, from easy, average or hard. The chart from figures 6.2 and 6.1 represent how the users performed on each task.

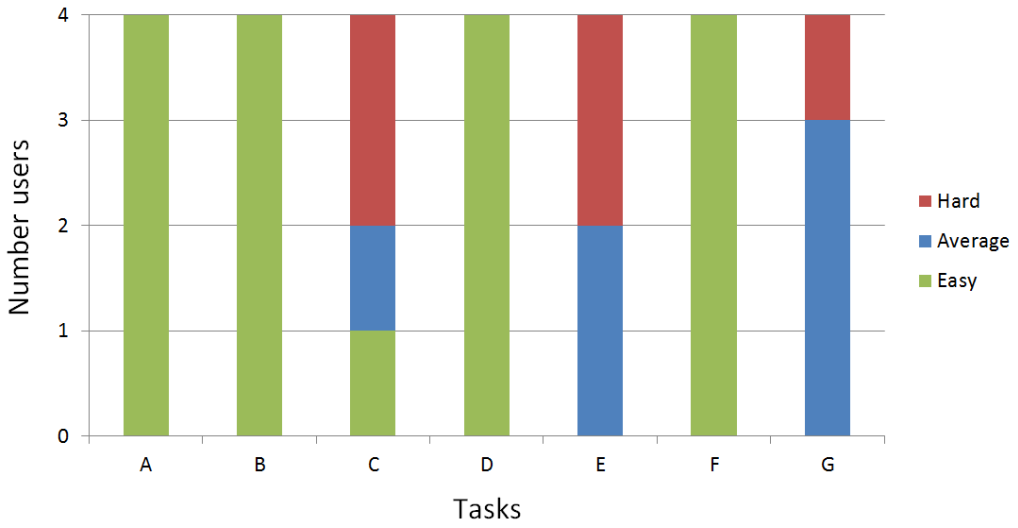



Figure 6.1: Global user performance per task on the HTC Desire.

## 6.2 Analysing Gathered Feedback

By analysing both charts and from our own observations we ended up with some good conclusions about our design and implementation.

**Navigation Design** From tasks **A**, **B** and **D** we can conclude that navigating in the application is easy for them. Only one user took longer than expected to find the back button in task **B** since he was searching for a physical button in the device for doing this. When we asked them to count how many recent comments were available on task **C** there was an increased difficulty on the HTC Desire due to the smaller screen, making the users have to scroll the list to find all comments.

**Sharing data** It was with test **E** that we have identified the most relevant difficulty in older adults to adopt new technologies. When we asked them to write a comment, they promptly found where text area to write it in, but had major problems while writing in both devices keyboards. Either by finding the keyboard too small or by finding it hard to hold the device while typing with the other hand, writing was generally complex to all but one person who put the iPad on a table and actually wrote with both hands, saying *"If I take too long is because I'm used to AZERTY keyboards"*, showing that she had previous experience with a typing machine.

Finding how to write and share a story was also easy to do for tests **F** and **G**. The iPad browser is unable to upload files by design and as such the file input was disabled in test **G**. We considered the test successful for those who correctly reached the menu and told them skip the task where they should select a file. This task worked perfectly with the Desire with older adults being able to choose a photo that existed in the device.

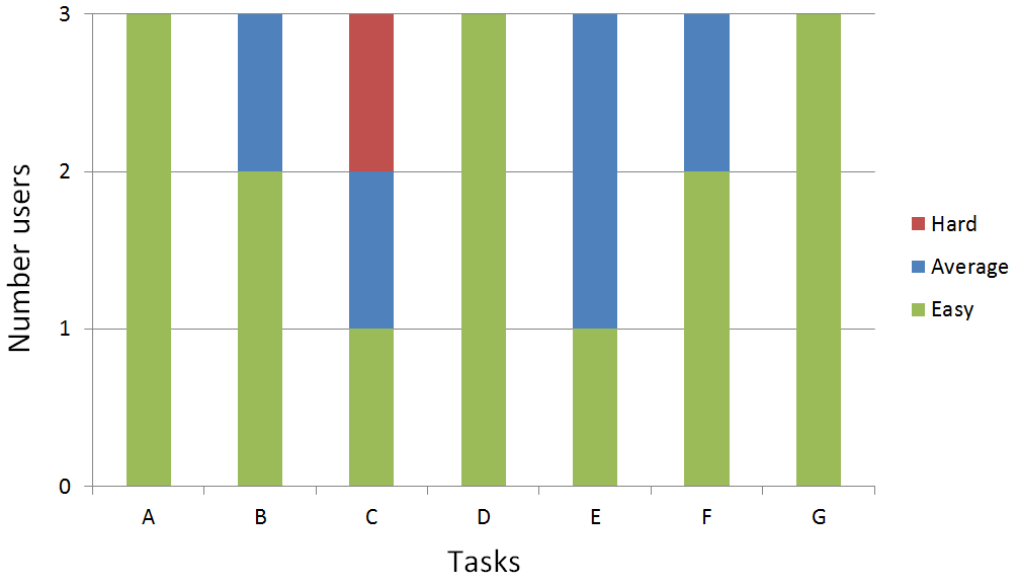



Figure 6.2: Global user performance per task on the Apple iPad.

**Other Issues** During these tests we were also able to identify some room for improvement in both implementation and design details.

**Feedback on Loading** A commonly observed issue was the lack of expected feedback once a user presses a button. Since it is common to have to query Facebook's servers to generate a new page, whenever they tried to do so and the result was not immediate, they pressed the button again and again until the action was performed. We believe we are able to prevent this behaviour by adding a visual *loading* information, as exemplified in figures 6.3, while the new page is not ready.

**Swaps for touches** Another observed issue we observed was that users many times not only pressed a button, but actually swiped their fingers on top of it, making the page scroll instead of selecting. This is implemented at the OS level and might be hard to workaround. Still, we believe we can try to catch the swipe event and use it to programmatically generate a tap in the same element that generated the scroll.

## 6.3 Results Evaluation

### 6.3.1 Discussion

From the results obtained we can infer that the design was in general well received amongst test users, but still has room for improvement. From the users' reactions we observed that our navigation system is easy for them to interact with. Still, we should provide them with visual feedback for every action they perform, preventing situations were the user is unsure if the application is slow to respond or idle. This is most needed

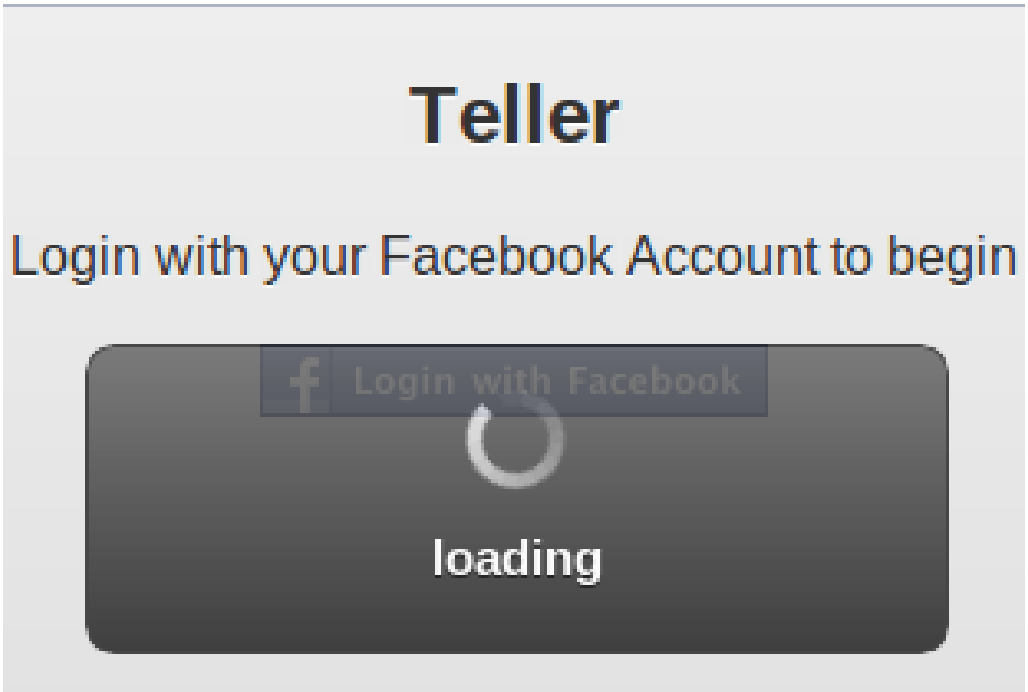



Figure 6.3: Possible visible information that a new page is loading.

when we are loading data from the social network, as well as to posting data to it, action subject to network delays.

From a technical point of view, there is also room for improvements. We believe that if we can transform user swipe actions into tap actions using the point where the user first touched the device, we might be able to improve the responsiveness of the applications, since they often swipe by mistake. We also believe that there is room for performance improvements in this prototype. Despite that, our approach was successful in several ways: *a*) Our server's framework has excellent templating and translation support, allowing to easily change the application's design; *b*) yet on the server, since generating pages with data pulled from Facebook is done on the client, load is expected to be very low and most processing can be cached; *c*) using jQuery Mobile on the client has proven a major success due to it's excellent adaptive methods

Globally, our tests indicate that we made the right choices while developing this prototype and are now getting near a marketable working product.

### 6.3.2 Technological Guidelines

**Reuse Proven Working Code** — JavaScript frameworks can save developers a lot of time by helping them to do their jobs and to actually do many of it for them. Also, they are generally free and community-maintained, guaranteeing that they are peer-reviewed and highly stable.

**Use Client-Side Templating** — JavaScript engines are becoming really fast in modern browsers. Adopting one of the many libraries that allow client-side templating pose as a good way in reducing the need for network communications, hence, speeding up web applications. With that in mind, programmers can begin to pass the server load of always generating full pages to only answering clients with the required data to fill a template.

**JavaScript Translations** — There's no need to have multiple JavaScript files when the JavaScript contains translatable strings. These can be imported from an external JavaScript file that is sent with the HTML to the client depending on the locale used or by using a *gettext* like function like in our application.

**Web Development eases Rapid Prototyping** — Using web development can reveal to be an agile development strategy as writing interface code for it with CSS and HTML is a fast operation and non-functional prototypes can be used early in the development for testing purposes while functionality is added in parallel. Interface can be adapted based on test results with minimal effort by updating the markup and style sheets.

## 6.4 Summary

Following the need to validate our design, we have performed a number of tests with the available population of older adults. From their feedback we evolved into a working prototype that was generally well received by all. Still, we perceived how hard it is for them to type in touch-based screens, as well as some improvements that we may include in our application.

Based on our tests and gathered experience during this project, we were able to write some guidelines for developers who are about to begin a similar process of developing online applications.

# Chapter 7

# Conclusions

## 7.1 Thesis Summary

From chapter 2 we have learned that older adults are receptive to new technologies and can be easily motivated to share their stories as long as the adoption of the technology is not a complex task for them.

Chapter 3 revisited the history of the web and introduced some new features from modern browsers that can in fact open them to a wide new range of possibilities, creating content for the web that needed to be developed natively, until now, making the web a resourceful environment for cross-platform development.

The specification on chapter 4 defines our application's requirements and development methodology. It also briefly introduces some HCI techniques that were applied during this phase.

In chapter 5 we described our architecture by presenting our process and tools, detailing on the implementation's internals.

Finally, in chapter 6 we presented our field testing methods and results and discuss those results.

## 7.2 Main Results

From our initial goals (section 1.3), this thesis concludes the following:

**Digital Storytelling is motivating for older adults.**

From our literature review in chapter 2 we identified several projects that are working with older adults to provide storytelling applications for them to interact mostly with their families. By creating a similar approach that enables content dissemination for a wider range of receptors, by using Facebook, the user can receive feedback from an higher number of people, resulting in a motivational experience for him. With this point we answer our two first goals, understanding digital storytelling and

existing implementations, as well as evaluating the presence of older adults in the web.

**HTML 5 and related technologies are viable for many future applications.**
New specifications and effort from the teams behind browsers to comply with them will guarantee a better result for developers that embrace the web as their target platform. Also, speedups in JavaScript engines provide browsers with the ability to perform better, hence, to be able to perform more complex tasks.

**Prototyped an online storytelling application for older adults.**
From the knowledge gathered from our literature review and studying HCI methods, we were able to design and implement a storytelling application, with the features previously specified: with a simplified design (using appropriated amount of information, font sizes, minimum clicking size and so on) and built with novel web technologies, guaranteeing portability.

From our tests results we were able to verify that older adults are able to use these technologies. Also, from observing their reactions, we noticed curiosity, which could be transformed into motivation to actively use the application, as long as feedback was returned from friends and family. A more extensive user study should be conducted for us to understand their willingness to adopt the application

Concluding, we are able to claim that HTML 5 is ready for mainstream adoption, as long as we consider users with up to date browsers. Users with older browsers that do not know how or can't update their browsers will certainly be unable to access the new features. For those, fall back methods should be implemented.

## 7.3 Dissemination of Research Results

To share our scientific results and help others working in similar areas to improve our guide their work, we submitted our results as papers for two conferences. Also, we are studying possible approaches to the market.

### 7.3.1 Scientific Dissemination

A demonstration paper has been accepted for publication at Interact 2011[1].

A second article was also accepted for publication in the International Conference on Entertainment Computing[2] (ICEC) describing how our work could increase older adults

[1] An international conference on Human-Computer Interaction to be held in Lisbon. Detail in http://interact2011.org/.

[2] Mote details at http://www.icec2011.org/.

well being by providing them with a new communication channel to interact with friends and family.

Both articles are attached to this thesis in appendix C.

### 7.3.2 Industrial Dissemination

Fraunhofer Portugal is the owner of the source code for this application and, as such, commercialization will depend on them. We believe that such type of application could be monetized by being sold both at day care centers or directly to older adults. Also, it is possible to freely provide the application for end users by monetizing it with unobtrusive ads.

## 7.4 Future Work

**Writing a Full Paper** with design and technical specifications of the project to be submitted to the ACM Conference on Human Factors in Computing Systems[3] (CHI) conference. We want to leave the iPad or Desire with older adults for a prolonged time for them to test the application in their daily lives. For that, we would have to contact their relatives to guarantee that they have Facebook accounts and motivate them to interact with the older adult in the social network. Such a study would evaluate the amount of communication between them during that period of time, as well as to get detailed feedback from both parties, allowing us to understand how the older adult uses the application and use that data to improve it.

**Improve Keyboard Interaction** since it was the major drawback we felt during our tests. As soon as it becomes ready, we could study the adoption of the Sound Capture API and introduce it into the application for sharing ideas and stories using the user's voice. At the same time, studying keyboards for older adults in touch-based devices could be subject for a new study.

[3] More details at http://www.chi2011.org/.

# Appendices

# Appendix A

# Interfaces

## A.1 Introduction

This appendix describes and exemplifies each of the composing screens of our application. In it, we show how the same screen looks using a first generation iPad and an HTC Desire, described in table A.1.

Table A.1: Comparison of test devices specifications

| | HTC Desire | Apple Ipad |
|---|---|---|
| Display | 3.7-inch, 480x800px | 9.7 inch, 1024x768px |
| Input type | Capacitive | Capacitive |
| Operative System and Version | Android 2.3.3 | iOS 4.3.3 |
| Browser Layout Engine and Version | WebKit 533.1 | WebKit 533.17.9 |

For each figure in this chapter, the one on the left is from the HTC Device, being the one of the right for the one from Apple.

## A.2 Interfaces

All interfaces share a common general look and feel, result from the use of jQuery Mobile and customized with the design specifications that resulted from Tenreiro's HCI studies with the older adults.

### A.2.1 Login

The login screen (fig. A.1) is the entry point of our application. Since we don't keep any data, the first step is to present the user with the option to login with his Facebook account, using OAuth. If the user has already logged in, this step is automatically skipped.

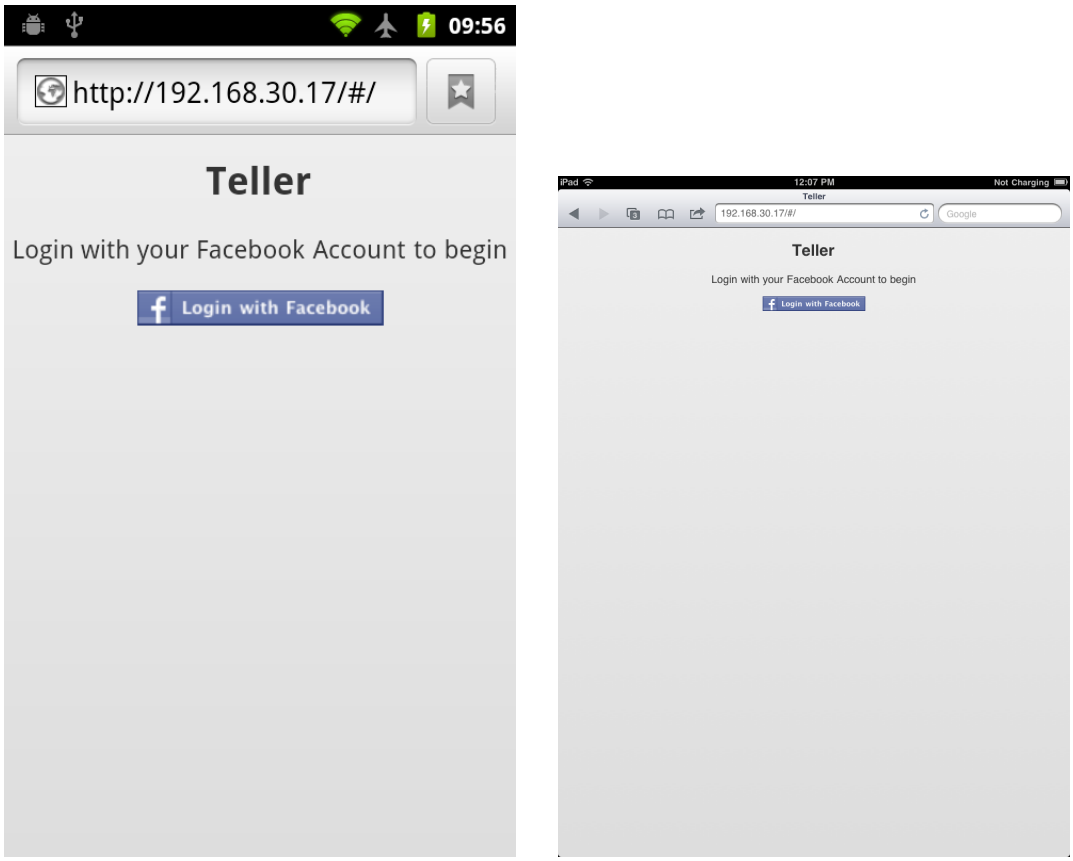



Figure A.1: Login Screen

### A.2.2 Menu

The main menu (fig. A.2) presents the user with the four main available options, *a*) View Notifications; *b*) View the most recent posts on his wall; *c*) share a story and *d*) view his list of friends.

### A.2.3 Lists

All lists share a common structure. They are built on the client site using a template based on data previously queried from Facebook.

#### A.2.3.1 Empty List

Whenever a list is requested but the request for data to Facebook returns empty, this page is displayed (fig. A.3). It can happen if the user opens the notification's menu and there's no recent activity to show.

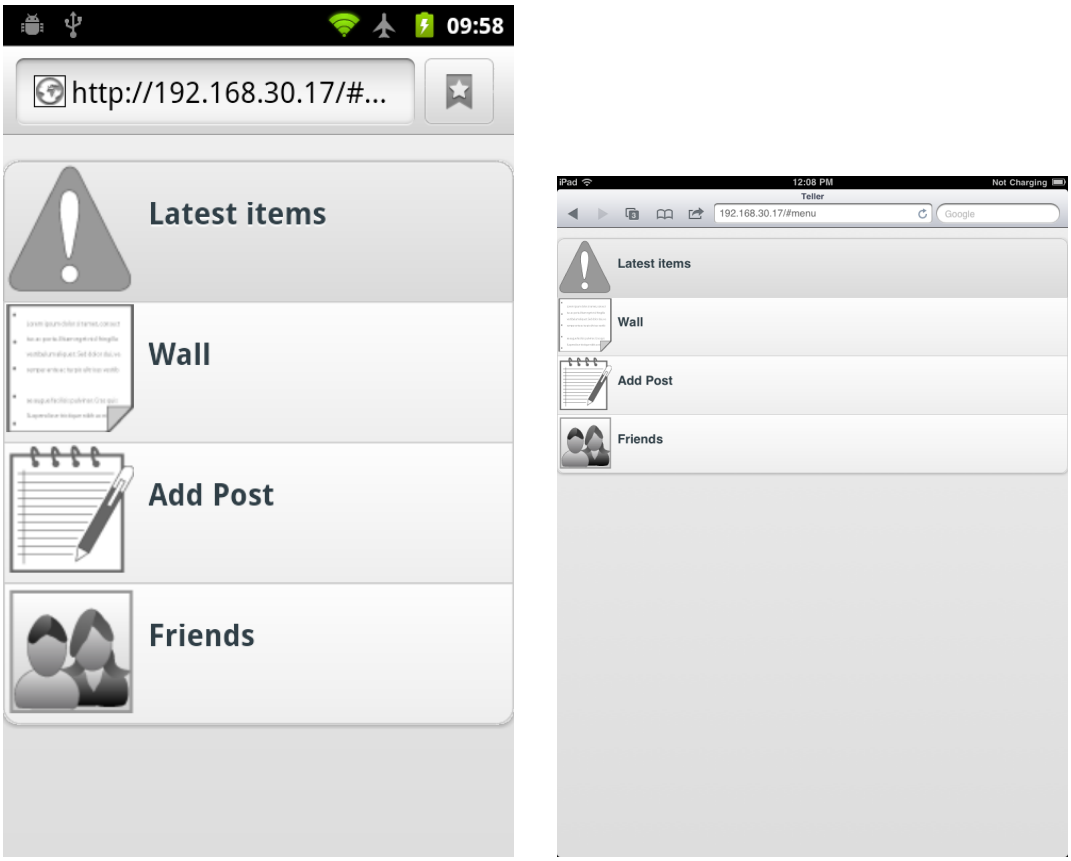



Figure A.2: Main Menu

#### A.2.3.2 List Wall

Shows the user's Wall (fig. A.4)with some information on each post: the author's photo and name, the time past since the post was created, the message from the post, if any (might be empty if the post only has an image or video). This list is similar to the one displayed in the notification's menu, but that one has an extra self-explanatory information on each line: *New Post* or New Comment

#### A.2.3.3 List Friends

Shows the complete list of friends (fig. A.5) from the user with a picture and his name. Clicking on each entry brings up the individual user page.

### A.2.4 Post

An individual post page (fig. A.6). It shows on top a picture of the post's author and his name. Bellow it indicates how old the post is. Next, the post message that is a composed by any text associated to the post, as well as showing video or image files when these are present.

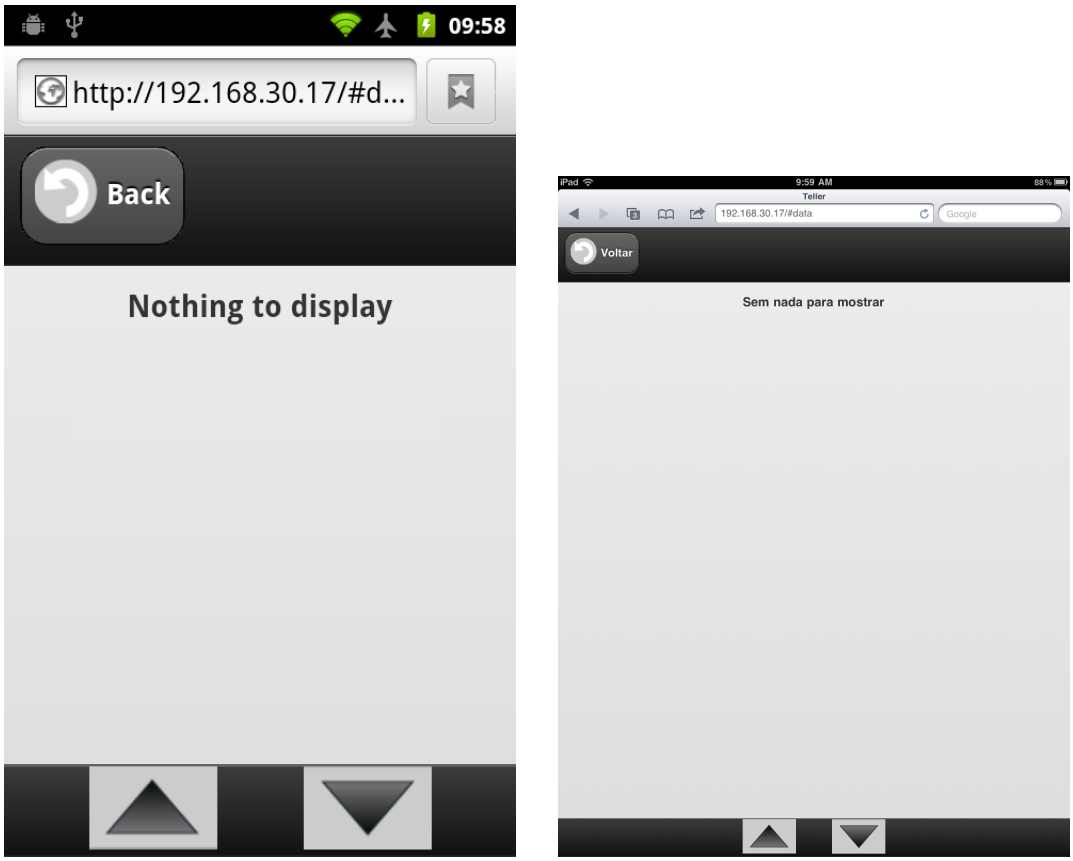



Figure A.3: List without data to display

While showing images or video, these are centered on the screen. An example of a post with video and another with an image are shown in image A.7, where screen on the left shows a video from Youtube ready to be played while the one on the right shows an uploaded image.

### A.2.5 Friend

A detailed page for showing a friend's details (fig. A.8) is also included. Information was kept minimal for design purposes, with a photo from the friend and information from where he lives.

### A.2.6 Share

The sharing page (fig. A.9) allows the user to share his ideas or stories. The user is presented with a menu for choosing what he wants to share. By choosing to share a story, he has access to a text area where he can write his story. He can then cancel or save the created story before getting back to the share menu. The same happens with the option to add photo or video, being presented with a file choosing option. At the menu, he can

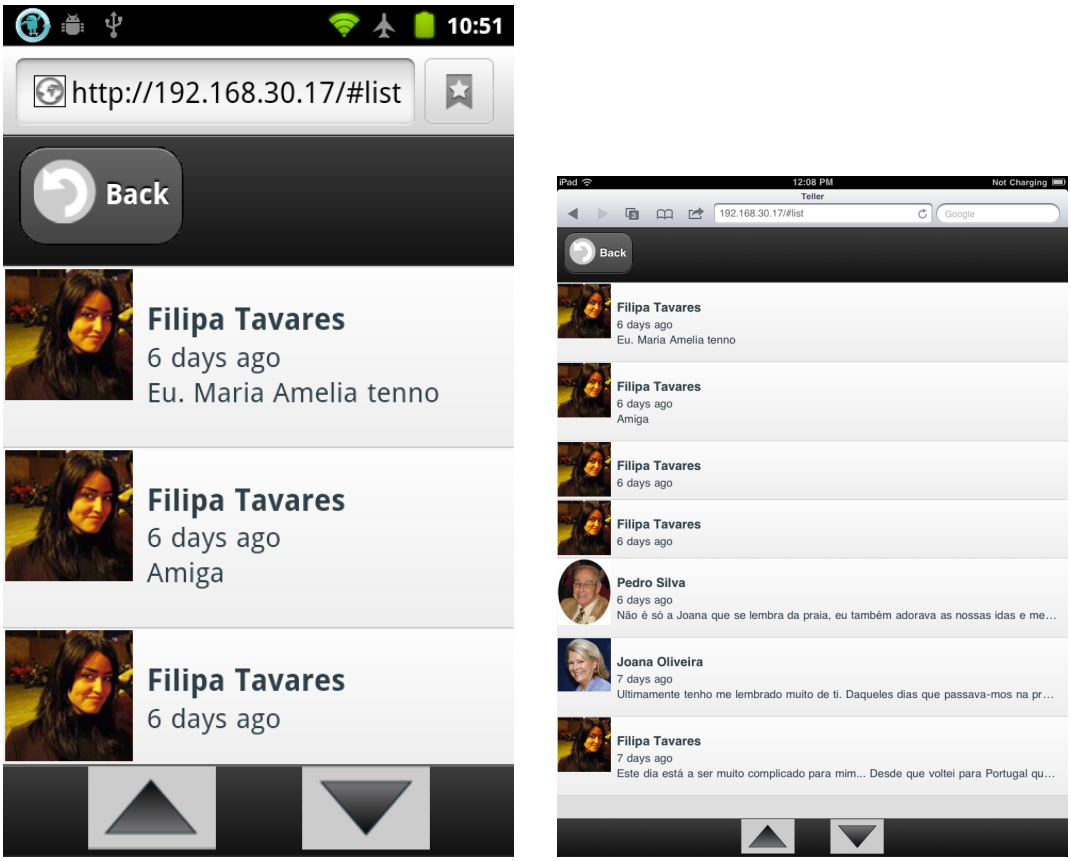



Figure A.4: List from the user's posts

then finally choose to submit the story, being redirected to the main menu. Figure A.10 partially shows this process. If he presses back at anytime, the current story is discarded.

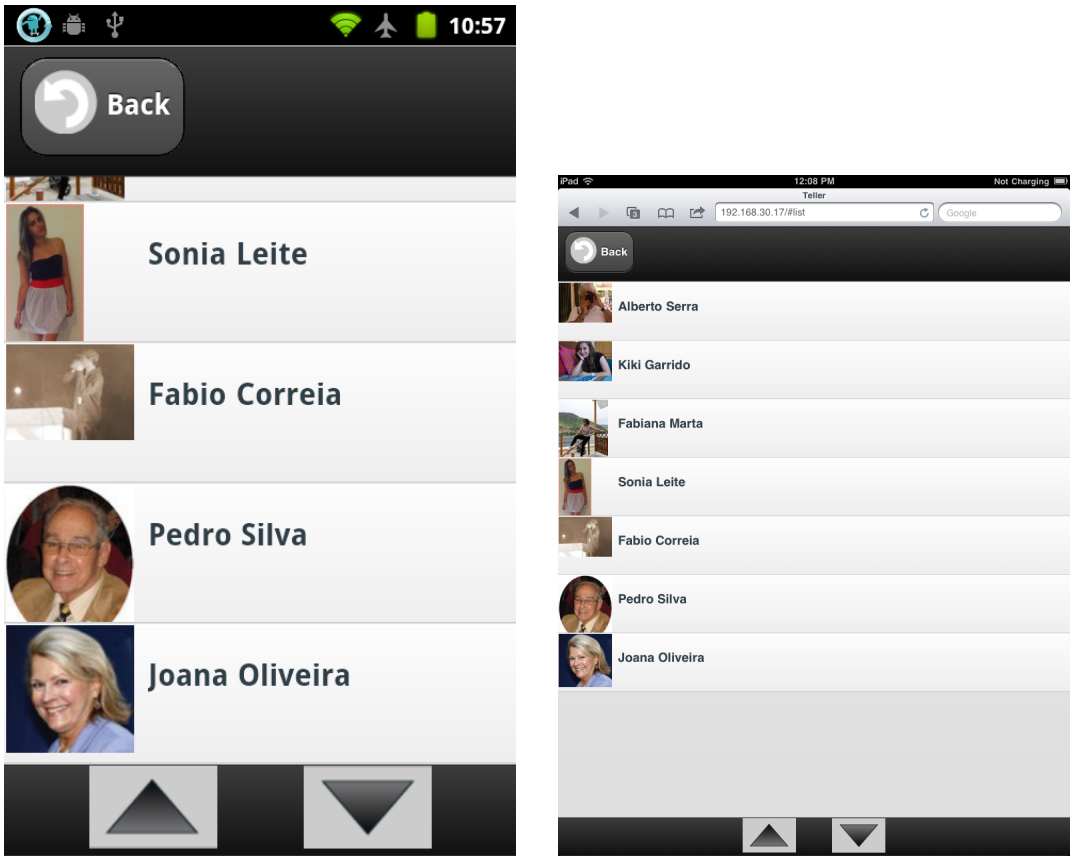



Figure A.5: List the user's friends

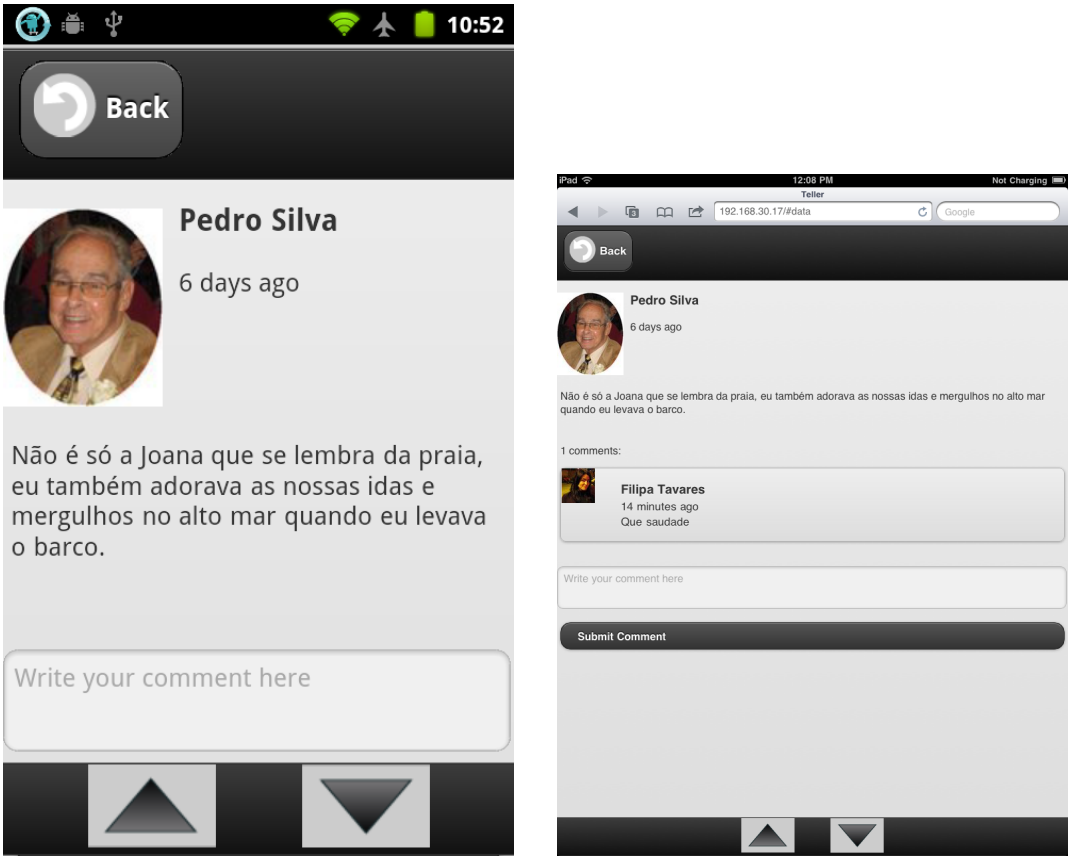



Figure A.6: Show a post from or to the user

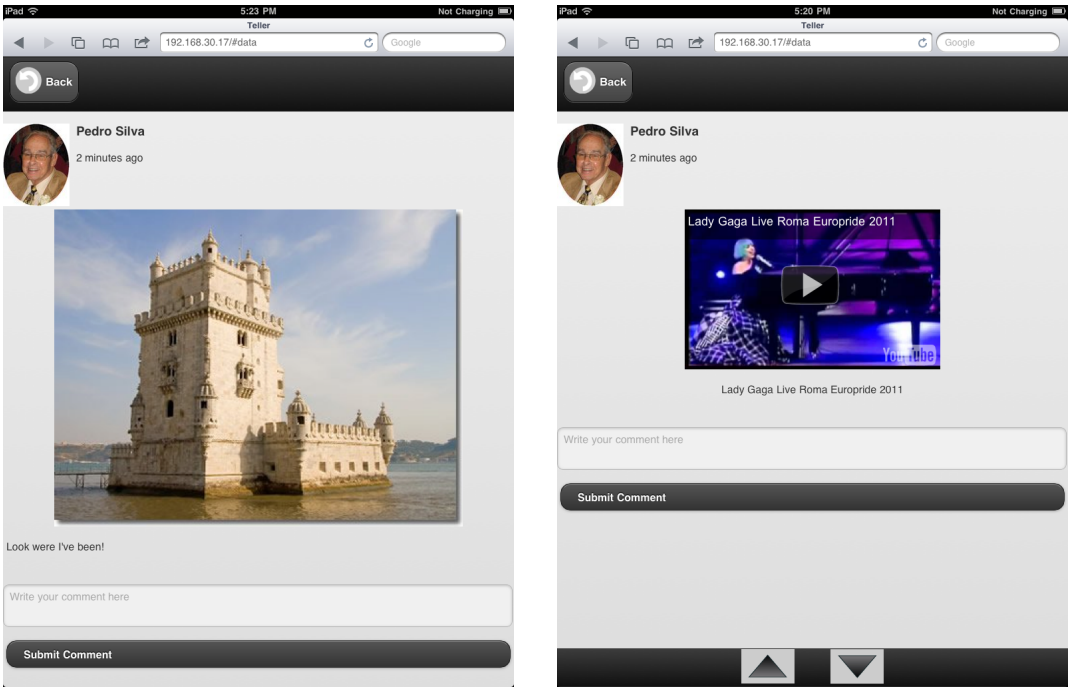



Figure A.7: Show a post with media content.

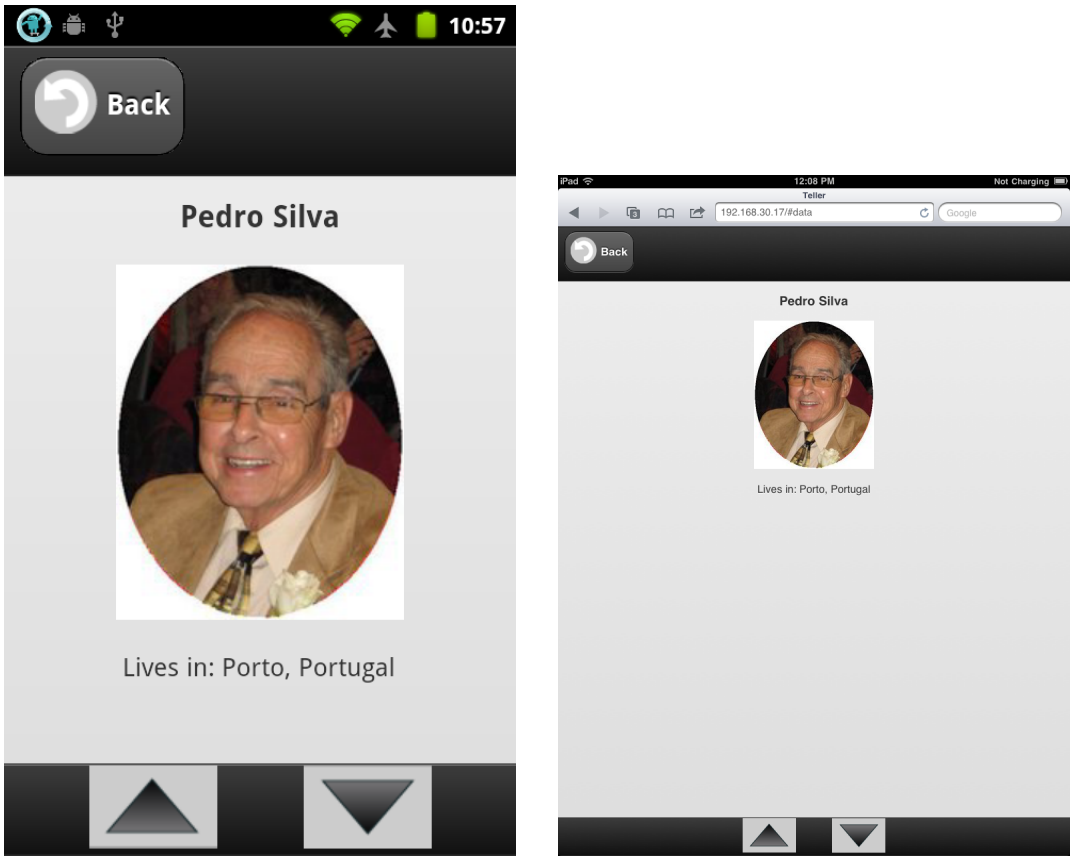



Figure A.8: Showing a friend's profile

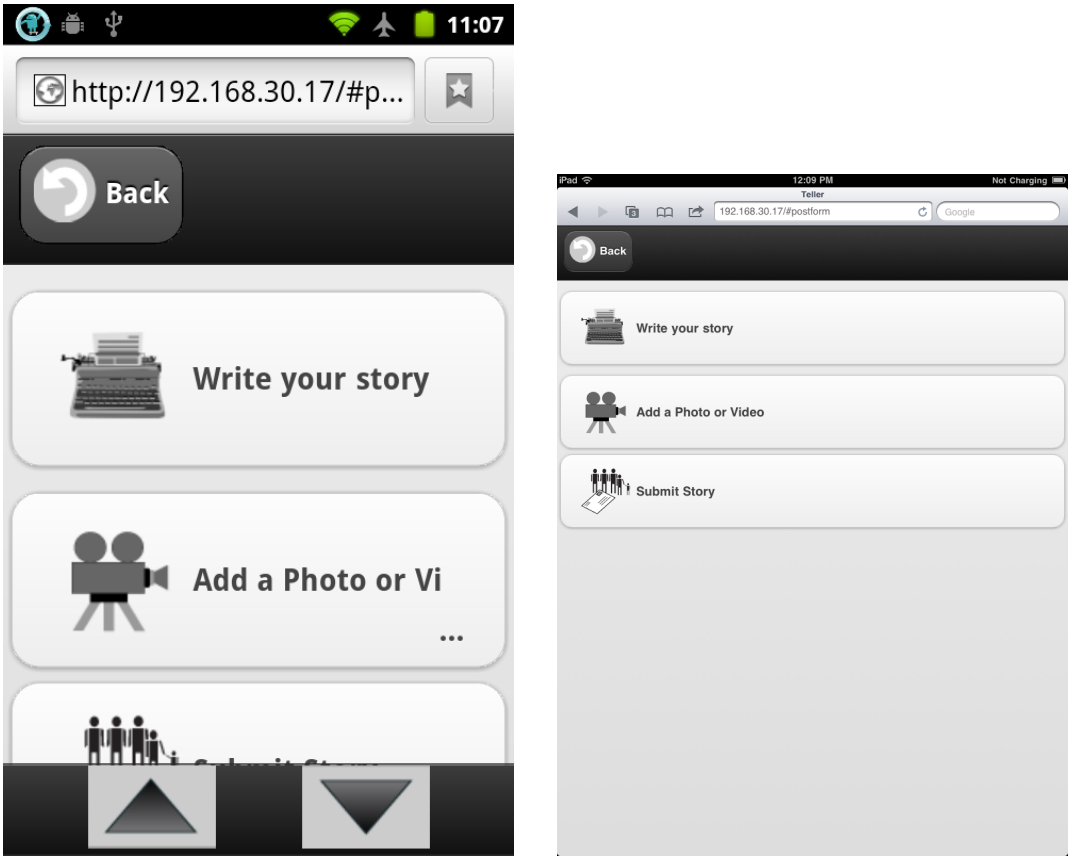



Figure A.9: Share a story menu

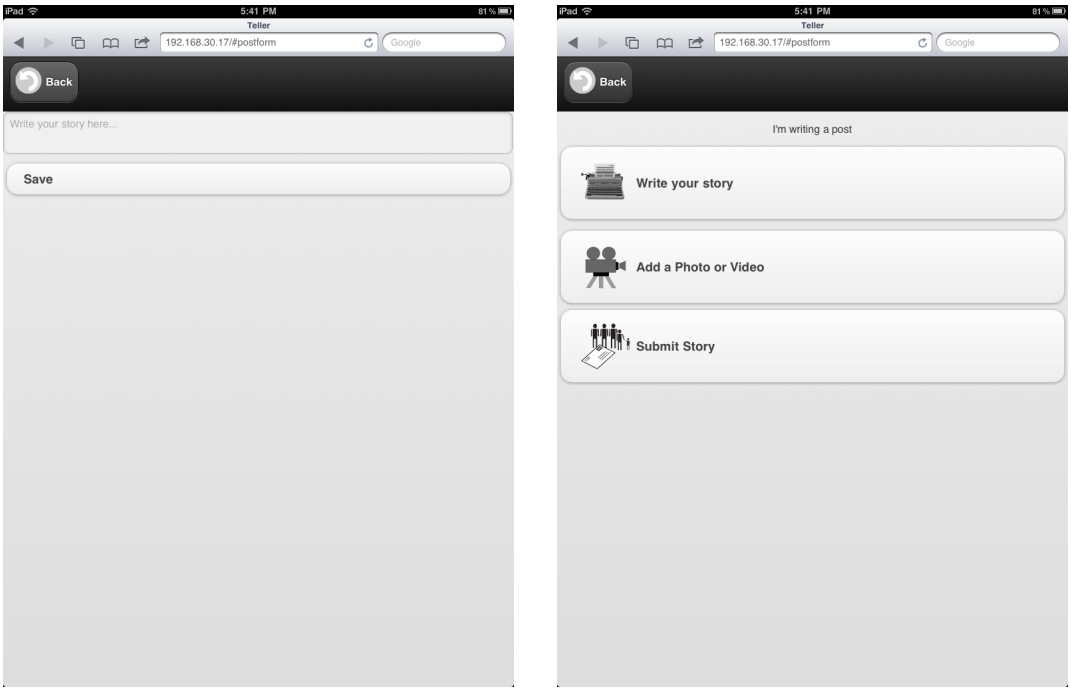



Figure A.10: Writing a story and then getting back to the share menu by hitting save.

# Appendix B

# Contribution to Open-Source Projects

This thesis contributed to two open-source projects with source code and to another one with feedback and bug reports.

## B.1 Facebook's Python SDK

When we initially started to use the Python SDK for Facebook, we had to deal with file uploads and noticed that it's implementation was still rather limited. So, we've downloaded the source from a community maintained fork from the original code and implemented a more generic upload method that allows for also uploading videos. Source block B.1 represents most of our commit to the repository, merged into the community fork [1].

## B.2 Pretty Dates

To show better dates, we have used the original code from James Padolsey. Still, his code was not complaint with Unix time, so we altered it has shown in source block B.2 to also handle Unix time.

## B.3 jQuery Mobile

As adopters of the framework, we have found some bugs and details to be improved in it. We have contacted the developers and reported some issues that are to be fixed in next releases, such as:

- Always show the tool bars, to be fixed in future versions;
- Improve the speed of updating a list, already fixed in latest version;

[1] More detail in the merge log in https://github.com/subssn21/python-sdk/commit/8fc070949adc69b55eb3912596e2e8d0ecde340a.

- Center images when used as thumbnails in lists, marked as medium priority;
- Fix horizontal scrolling bug in iPad, marked as high priority;
- Improve the handling for fixed footer, marked as medium priority.

With these bug reports we have reported and described how to reproduce the bug so that the team might work in fixes for it.

**Source B.1** Contributed code to a community maintained fork of the Facebook's Python SDK.

```
def put_photo(self, image, message=None, album_id=None, **kwargs):
    """Shortcut for put_media to upload a photo"""
    self.put_media(image, message, album_id, fxtype='photos',
            kwargs=kwargs)

def put_video(self, image, message=None, album_id=None, **kwargs):
    """Shortcut for put_media to upload a video"""
    self.put_media(image, message, album_id, fxtype='videos',
            kwargs=kwargs)

def put_media(self, fx, message=None, album_id=None,
        fxtype=None, **kwargs):
    """Uploads a file using multipart/form-data
    fx: File like object for the image
    message: Caption for your image
    album_id: On photos, None posts to /me/photos
         which uses or creates and uses
    an album for your application.
    fxtype: one of 'photos' or 'videos' depending on media type
    """
    object_id = album_id or "me"
    #it would have been nice to reuse self.request;
    # but multipart is messy in urllib
    post_args = {
          'access_token': self.access_token,
          'source': fx,
          'message': message
    }
    post_args.update(kwargs)
    content_type, body =
         self._encode_multipart_form(post_args, fxtype)
    req = urllib2.Request("https://graph.facebook.com/%s/%s" %
            (object_id, fxtype), data=body)
    req.add_header('Content-Type', content_type)
    data = urllib2.urlopen(req).read()
    try:
        response = _parse_json(data)
        if response and response.get("error"):
            raise GraphAPIError(response["error"].get("code", 1),
                                response["error"]["message"])
    except ValueError:
        response = data

    return response
```

**Source B.2** Altered code from the original pretty date code from James Padolsey

```
/*
* Altered code from the Recursive pretty date from James Padolsey
* http://james.padolsey.com/javascript/recursive-pretty-date/
*/
var prettyDate = (function() {

    return function(time) {

        var time2 = +time;
        if( !isInt(time2) ){
            time2 = +new Date(time);
            if( !isInt(time2) ){
                time2 = +Date.parse(time)
            }
            if ( !isInt(time2) ) {
                alert('oi');
                alert(time);
                time2 = +time;
            }
        }
```

# Appendix C

# Submitted Papers

## C.1 Interact 2011

Interact 2011[1] is the 13$^{th}$ edition of the Conference on Human-Computer Interaction by the International Federation for Information Processing[2] The following paper is a demonstration paper were we describe our work from both a design and technical point of view. It was submitted and accepted with a rating of 5 out of 5 by both reviewers.

[1] More detail at http://interact2011.org/.
[2] More details at http://www.ifip.org/.

# Storytelling meets the Social Web: an HTML5 cross-platform application for older adults

Tiago Boldt Sousa[1], Pedro Tenreiro[1], Paula Alexandra Silva[1], Eduarda Mendes Rodrigues[2]

[1] Fraunhofer Portugal, Rua Alfredo Allen 455/461 4200-135 Porto PORTUGAL
[2] Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias, s/n 4200-465 Porto PORTUGAL

**Abstract.** This demonstration presents a storytelling application specifically designed for older adults to share stories and thoughts. Studies claim that older adults commonly have difficulties in engaging with online social networks [1], but increased social inclusion and sense of well-being has been observed in those who engage [2]. While following a user-centered design approach, we have developed an HTML5 device-independent and intuitive social web application which addresses older adults' specific needs and age-related impairments, allowing them to connect to their friends and family through storytelling.

**Keywords:** Storytelling, Online Social Networks, User-Centered Design, Older Adults, HTML5

## 1 Introduction

Unlike a few decades ago, today's households no longer have more than two generations [4], resulting in older adults living by themselves. Distance from family can result in feelings of loneliness and can limit their ability for sharing their legacy, something which is paramount for them. Furthermore, transportation and communication problems are known constraints due to the decline of their physical and mental conditions, resulting in a general decline of their well-being and happiness [1]. While they can be positively influenced by the use of online social networks [2], younger people are the mainstream audience of these services [3]. Feature-rich services and web pages with large amounts of information are common and difficult to use by those lacking computer skills; this results in a general inhibition towards the exploration of technology from the elderly. To address these problems and to fight the social exclusion of older adults, we developed a prototype of a storytelling application that leverages an existing online social network (Facebook) for connecting older adults with their relatives.

## 2 Cross-platform Storytelling on the Social Web

### 2.1 Application Design

The prototype application was implemented using a set of novel web technologies commonly denominated as HTML5, taking into account special design con-

straints identified through user-centered design, which highlighted common impairments and difficulties of older adults' interaction. Several observations and usability tests were pursued with a sample of 15 older adults to inform the design. We have used the test results to accommodate a small learning curve, as well as a pleasant experience for any older adult user. Those studies provided us with our main goals which were to enable text and image sharing, as well as to allow follow-up comments on them. The inclusion of meta-data in each story (for example the user's mood or location) appeared as interesting, but secondary.

### 2.2 Prototype Implementation

The implementation of this storytelling application followed a user-centered design approach, in order to create an easy to adopt application by senior users. This demonstration aims at showing how the application enables older adults to share stories through online social networks, specifically Facebook. Our target users consist of older adults without any prior experience with information and communication technologies. As such, the client application only provides access to a small subset of all Facebook's features, those identified as the most relevant for the older adults, while also keeping the application easy to use. Despite primarily aiming at older adults, any individual with a Facebook account will be able to use the storytelling application as well as to interact with storytellers through the Facebook service itself. The prototype uses elements from the recent HTML5 specification, such as localStorage for keeping local data. This guarantees seamless portability between compatible browsers, enabling the user to choose her preferred device to use the application.

## 3 Conclusions and Future Work

Our application provides an alternative communication channel between older adults and their friends and family, hoping to increase their well-being through the elimination of feelings of loneliness.

Due to the fast-paced evolution of the HTML5 standard, extra features might be included in the project before the date of the demonstration. While sharing text and images are the initial features, video and audio capture should also be integrated into the application as soon as they are available in web browsers (already described in the current HTML5 specification).

## References


1. Chen, Y.: Usability Analysis on Online Social Networks for the Elderly (2009)
2. Gibson, L., Moncur, W., Forbes, P., Arnott, J., Martin, C., Bhachu, A.S.: Designing Social Networking Sites for Older Adults (2010)
3. Pingdom: Study: Ages of Social Network Users (2010)
4. Williams, B.K., Sawyer, S.C., Wahlstrom, C.M.: Marriages, Families & Intimate Relationships. Boston, MA: Pearson (2005)

## C.2 ICEC 2011

We submitted a paper for the International Conference on Entertainment Computing[3] (ICEC) describing how our work could increase older-adults well being by providing them with a new communication channel to interact with friends and family. We are still waiting or the author notification date to acknowledge if it was accepted.

[3]Mote details at http://www.icec2011.org/.

# Cross-Platform Social Web Application for Older Adults with HTML 5

**Abstract.** Web social networks can be very important for older adults that often live far from their family and friends. Despite its benefits, older adults are still far from older adults because the user interfaces are not adapted to them or because their friends are not on the web social networks they can use. This paper introduces a mobile web application designed with older adults characteristics in mind and that connects to a popular social network.

**Keywords:** storytelling, web social networks, user-centered interface design, older adults, html5

## 1 Introduction

Older adults may face a reduction in personal contact mainly due to the loss of loved ones, mobility difficulties or geographical separation [11].

To keep contact, families have used mail, telephones and web social networks. Although the first two are well mastered by older adults, the latter is still far from them due to user interfaces that very complex and which design does not consider older adults characteristics [11]. Nevertheless, web social networks can be good for older adults key in fighting loneliness and isolation, allowing them to stay connected with the loved ones who have moved away [5].

This paper describes the work in progress of the development of a storytelling application for older adults, introducing the problems that older adult face in their daily life and their difficulties to use social networks sites. The paper also describes the methods that are and have been used to design the application, their results and some details of the implementation.

## 2 Use of Social Networks by Older Adults

Older adults, can find themselves in situations of social exclusion [11], due to retirement, moving difficulties or simply because they live far from friends and family [11].

A study with US elderly citizens reveals that most retirees are staying at home and one-third of those aged 75 and older live alone [10]. Loneliness may cause severe conditions, such as depression and insomnia [11]. This is due to the fact that, as human-beings we need to communicate with each other in order to share their knowledge and experiences [2][1].

Thus, social exclusion is often aggravated by the onset of health conditions. The use of ICTs (Information and Communication Technologies) is known to

play a fundamental role in the psychological wellbeing of older adults, by enabling them to socialize, reducing feelings of loneliness and alienation [5]. Online social networks, in particular, offer excellent opportunities for older adults to fight isolation and loneliness, by enabling them to keep in touch with their relatives and friends, bypassing the physical distance issues.

According to Chen [11], *"online social networks typically exclude the elderly"*. One reason for this is that, unlike younger people, the elderly have difficulties in adopting new technologies and software. Partially, this is due to the fact that the current generation of older adults grew up without computers and, therefore, have some difficulty in perceiving the value and usefulness ICTs has in their daily activities [8].

Nowadays, older adults are giving more importance to online social networks, acknowledging their benefits. In fact, according to Madden [6] the engagement of older adults in social media services has doubled between April 2009 and May 2010 to 26%. This study points out three main reasons for this increased adoption of social networks by older adults: i) reconnecting with people from their past by finding them through common friends or groups; ii) seeking online support for those suffering from chronic diseases and finding someone to share their experience with; iii) creating a bridge in generational gaps, facilitating communication of the older adult with other age groups.

Thus, older adults seem eager to start using social Web applications to communicate with friends and family but might be reluctant to do so due to the complex design of existing social media services. This calls upon careful design of future social Web applications, specifically intended to facilitate the social inclusion of older adults.

## 3 Designing Social Web Application for Older Adults

The authors are developing a social Web application, designed for the older adult in which they should be able to communicate with friends and family to share their life-stories or general thoughts and feelings from their daily life.

We believe that an user-centered design approach is fundamental to gather the needed information to start prototyping the design for such application. For this reason, we conducted informal interviews, observation, and usability tests in a day care centre. These techniques were crucial to understand the needs, challenges and obtain early feedback from the targeted audience.

### 3.1 Sharing Life-Stories and Toughs

A social networking application should enable the user to share memories from their past(e.g.: childhood, jobs, or other subjects), and their daily life.

Storytelling presents advantages, not only for the storytellers themselves, but also for the story listeners. These stories carry life experiences to the next generations, the story listeners, establishing a bound between the two, providing them with wisdom that might influence their future [2][1].

The authors believe a storytelling application for older adults should integrate with Facebook, because this is the most adopted social network world wide with more than 500 million users [4]. In order to do so, we've been using the Facebook API to post the stories and thoughts of the user as a *Wall Post*.

This will enable the use of *Comments*, which will potentiate the interaction between the user and his connections.

### 3.2 User-Centered Approach Results

Usability tests showed that older adults struggle to perceive small sized items. Thus, items as text and images should be adapted to the user's ocular characteristics, adopting a size suitable for the user to read or watch the images with minimal effort. With the same usability tests we were able to define that the minimum usable font size is of 2.8mm (25px) so the older adults can read comfortably on mobile devices.

By applying informal interviews to older adults we managed to confirm the literature that they get confused when confronted with large amounts of information such as more then a message in the same display or even multiple notifications [11] [3]. When presented with many different items to process, older adults distract themselves easily, difficulties the usage. This might result in a loss of interest to use the application. With these interviews we concluded that it's important to reduce the application's complexity, by introducing only the essential features needed by older adults into the application.

In the day care centre almost all older adults need to listen or read questions or instructions more than once in order to understand them. This allow us to infer that whenever showing information to the users, we should always give them the necessary time to understand each piece of information, removing the possibility to use information that automatically disappear from the screen [3].

Nowadays older adults have higher rates of illiteracy than any other age bracket [9]. This way proper images and sounds must exist to allow the replacement of the written text, so that illiterate older adults are able to use the system. It's relevant for older adults that our application allows them to record their voice or even a video to simplify the input process, removing the need for user to type text.

With this information in mind, we know that we'll need to not only stimulate the interest from the older adult, but also to provide him with a pleasant and easy experience while using a social application. Therefore our application ensure that the font size is appropriated for older adults, that there are no large amounts of information in each screen, e. g.: only a story each time, and also ensure that only the more important functionalities are presented to older adults so they can perform their actions without get confused with functionalities that they will not use such as events, groups or even external applications.

### 3.3 Implementation

The fast-paced growth of technology has led to the emergence of a number of different electronic devices. As the variety of devices grew, the technologies used to create software for them also expanded. HTML 5 aims to accelerate development by creating a web standard that allows developers to create rich web applications that can be written once and that run in multiple devices, allowing older adults to choose the device that bests suits them, considering portability, screen size, input method or even device costs. Because of the older adults' difficulties on adopting new technologies, this is an important feature for them to feel comfortable while trying the application.

## 4 Conclusions

We have recognized that older adults are subject to feelings of loneliness resulting from isolation. Studies also verify that older adults are receptive to new technologies and social networks as a mean to improve their well being and connect them with their acquaintances. With this in mind, our work intends to provide a way for the older adults to communicate with their friends and family, helping them to strength relationships and reduce feelings of loneliness. The application will be designed as intuitive and pleasant as possible, adapting to common impairments and limitations of the older adult.

HTML 5 would allow to create the cross-platform application application we need, providing the user the freedom to choose the device that best fits him.

## References


1. Life storytelling, occupation, social participation and aging. Occupational Therapy Now pp. 23–24 (Sep 2007)
2. Beth Sanders: Reminiscence: Impacting brain fitness, lifelong learning, healthy aging & wellness (Apr 2010)
3. Fisk, A.D., Rogers, W.A., Charness, N., Czaja, S.J., Sharit, J.: Designing for Older Adults: Principles and Creative Human Factors Approaches. Taylor & Francis Group, LLC, second edn. (2009)
4. Jenna Wortham: Facebook Tops 500 Million Users (Jul 2010)
5. Maria Karavidas and Nicholas K. Lim and Steve L. Katsikas: The effects of computers on older adult users. Science Direct (May 2004)
6. Mary Madden: Older Adults and Social Media Social networking use among those ages 50 and older nearly doubled over the past year. Tech. rep., Pew Research Center (Aug 2010)
7. Pingdom: Study: Ages of social network users (Feb 2010)
8. Santana, P.C., Rodríguez, M.D., González, V.M., Castro, L.A., Andrade, A.G.: Supporting emotional ties among mexican elders and their families living abroad pp. 2099–2103 (2005)
9. Sarah Poff Roman: Illiteracy and Older Adults: Individual and Societal Implications. Customer Services for Taylor & Francis Group Journals., 25 Chestnut Street, Suite 800, Philadelphia, PA 19106 (2004)

10. Teresa Steinfatt: Social Networks Reduce Loneliness and Isolation for the Elderly (Jul 2010)
11. Yu Chen: Usability Analysis on online Social Networks for the elderly (2009)

# References


[Ale11] Alexa. Top Sites — The top 500 sites on the web., February 2011. Accessed February 2011 from http://www.alexa.com/topsites.

[All09] Mitch Allen. Palm webOS: Developing Applications in JavaScript using the Palm Mojo Framework, August 2009.

[Bel11] Fabrice Bellard. Javascript pc emulator - technical notes, may 2011. Accessed June 2011 from http://bellard.org/jslinux/tech.html.

[Ben99] Brent W. Benson, Jr. Javascript. *SIGPLAN Not.*, 34:25–27, April 1999.

[BKA+09] Rafael Ballagas, Joseph 'Jofish' Kaye, Morgan Ames, Janet Go, and Hayes Raffle. Family communication: phone conversations with children. In *Proceedings of the 8th International Conference on Interaction Design and Children*, IDC '09, pages 321–324, New York, NY, USA, 2009. ACM.

[BL] Tim Berners-Lee. The WorldWideWeb browser.

[BL89] Tim Berners-Lee. Information Management: A Proposal, March 1989. Accessed February 2011 from http://www.w3.org/History/1989/proposal.html.

[BL92] Tim Berners-Lee. HTML Tags, November 1992. Accessed February 2011 from http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html.

[BLC95] Tim Berners-Lee and Dan Connolly. Hypertext Markup Language - 2.0. Technical report, W3C, November 1995. Accessed February 2011 from http://www.w3.org/MarkUp/html-spec/html-spec_toc.html.

[CDGH08] Mike Chambers, Daniel Dura, Dragos Georgita, and Kevin Hoy. *Adobe® AIR™ for JavaScript Developers*. Adobe, April 2008. Accessed February 2011 from http://onair.adobe.com/files/AIRforJSDevPocketGuide.pdf.

[ÇEG+09] Tantek Çelik, Elika J. Etemad, Daniel Glazman, Ian Hickson, and Peter Linss andJohn Williams. Selectors Level 3. W3C Proposed Recommendation, W3C, December 2009. Accessed February 2011 from http://www.w3.org/TR/css3-selectors/.

[CH06] L.L. Carstensen and C.R. Hartel. *When I'm 64*. National Academies Press, 2006.

[Che09] Yu Chen. Usability Analysis on Online Social Networks for the Elderly. Technical report, Helsinki University of Technology, 2009.

[Cho11] Jeremy Chone. Case Study: HTML5 MathBoard, January 2011. Accessed February 2011 from http://www.html5rocks.com/tutorials/casestudies/mathboard.html.

[Cli09] Stephanie Clifford. Online, 'a reason to keep on going'. *New York Times*, June 2009. Accessed February 2011 from http://www.nytimes.com/2009/06/02/health/02face.html?_r=2.

[Cut09] Paul Cutler. Behind the Scenes with Owen Taylor. *The GNOME Journal*, July 2009. Accessed February 2011 from http://gnomejournal.org/article/74/behind-the-scenes-with-owen-taylor.

[dB10] Adam de Boor. Drag and drop attachments to save them to your desktop, August 2010. Accessed February 2011 from http://gmailblog.blogspot.com/2010/08/drag-and-drop-attachments-to-save-them.html.

[DBQ09] Allison Druin, Benjamin B. Bederson, and Alex Quinn. Designing intergenerational mobile storytelling. In *Proceedings of the 8th International Conference on Interaction Design and Children*, IDC '09, pages 325–328, New York, NY, USA, 2009. ACM.

[Des10] David Dessandro. Opera Logo with CSS, March 2010. Accessed February 2011 from http://desandro.com/articles/opera-logo-css/.

[DFAB04] Alan Dix, Janet E. Finlay, Gregory D. Abowd, and Russell Beale. *Human-Computer Interaction*. Prentice Hall, 3rd edition, 2004.

[E. 11] E. Hammer-Lahav. The OAuth 2.0 Authorization Protocol. Technical report, IETF, 2011.

[Fac11a] Facebook. *Authentication*, February 2011. Accessed February 2011 from http://developers.facebook.com/docs/authentication/#user-login.

[Fac11b] Facebook. *Graph API*, February 2011. Accessed February 2011 from http://developers.facebook.com/docs/reference/api.

[Fac11c] Facebook. Statistics, June 2011.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFCs 2817, 5785.

[Fla02] David Flanagan. *Javascript: the definitive guide*. O'Reilly & Associates, Inc., 2002.

[GMF+10] Lorna Gibson, Wendy Moncur, Paula Forbes, John Arnott, Christopher Martin, and Amritpal S Bhachu. Designing Social Networking Sites for Older Adults. Technical report, University of Dundee, 2010.

[Goo] Google. Google Apps Script. Accessed February 2011 from http://code.google.com/googleapps/appsscript/index.html.

[Gre09] Samuel Greengard. Facing an age-old problem. *Commun. ACM*, 52:20–22, September 2009.

[Gro00] W3C HTML Working Group. XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition). W3C Recommendation, W3C, January 2000. Accessed February 2011 from http://www.w3.org/TR/xhtml1.

[gro11] groovecoder. Web standards in visual studio, June 2011. Accessed at http://groovecoder.com/2011/06/19/web-standards-in-visual-studio/.

[Hat11] Hakim El Hattab. Case Study: Page Flip Effect from 20thingsilearned.com, January 2011. Accessed February 2011 from http://www.html5rocks.com/tutorials/casestudies/20things_pageflip.html.

[HCLS09] John Helmes, Xiang Cao, Siân E. Lindley, and Abigail Sellen. Developing the story: Designing an interactive storytelling application. Technical report, Microsoft Research Cambridge, March 2009.

[Hic10] Ian Hickson. Html 5. W3C working draft, W3C, October 2010. Accessed February 2011 from http://www.w3.org/TR/2010/WD-html5-20101019/.

[Hic11a] Ian Hickson. Web storage. W3C working draft, W3C, February 2011. Accessed February 2011 from http://www.w3.org/TR/2011/WD-webstorage-20110208/.

[Hic11b] Ian Hickson. Web Workers. W3C Working Draft, W3C, February 2011. Accessed February 2011 from http://www.w3.org/TR/2011/WD-workers-20110208/.

[ICD] ICDL. ICDL Mission. Accessed February 2011 from http://en.childrenslibrary.org/about/mission.shtml.

[jM11] jQuery Mobile. jquery mobile: Touch-optimized web framework for smartphones & tablets, May 2011. Accessed May 2011 from http://jquerymobile.com/.

[jQu] jQuery. jQuery API — Selectors. Accessed March 2011 from http://api.jquery.com/category/selectors/.

[jQu11] jQuery. jquery — write less, do more., February 2011. Accessed February 2011 from http://jquery.com/.

[KB11] Erik Kay and Aaron Boodman. A dash of speed, 3D and apps, February 2011. Accessed February 2011 from http://chrome.blogspot.com/2011/02/dash-of-speed-3d-and-apps.html.

[KN02] G. Klyne and C. Newman. Date and time on the internet: Timestamps, 2002.

[KSZP05] Kurniawan, Sri, Zaphiris, and Panayiotis. Research-derived web design guidelines for older people. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, Assets '05, pages 129–135, New York, NY, USA, 2005. ACM.

[LB08] Håkon Wium Lie and Bert Bos. Cascading style sheets (CSS1) level 1 specification. W3C recommendation, W3C, April 2008. Accessed February 2011 from http://www.w3.org/TR/2008/REC-CSS1-20080411/.

[LBÇH10] Håkon Wium Lie, Bert Bos, Tantek Çelik, and Ian Hickson. Cascading style sheets level 2 revision 1 (css 2.1) specification. W3C recommendation, W3C, December 2010. Accessed February 2011 from http://www.w3.org/TR/2010/WD-CSS2-20101207/.

[Mah11] Michael Mahemoff. Html5 vs native: The mobile app debate, June 2011. Available from http://www.html5rocks.com/en/mobile/nativedebate.html.

[Mar11] Chris Marrin. WebGL Specification. Final Draft(1.0), Khronos, February 2011. Accessed February 2011 from http://www.khronos.org/registry/webgl/specs/latest/.

[Mea03] Daniel Meadows. Digital Storytelling: Research-Based Practice in New Media. *Visual Communication*, February 2003. Accessed February 2011 from http://vcj.sagepub.com/content/2/2/189.

[Mod11] Modernizr. Modernizr, February 2011. Accessed February 2011 from http://www.modernizr.com/.

[Moo] Boris Moore. Jquery api — templates. Last accessed on June 2010 from http://api.jquery.com/category/plugins/templates/.

[Moz10] Mozilla. Same origin policy for javascript. *MDN Docs*, 2010.

[New95] PR Newswire. Netscape and sun announce javascript(tm), December 1995. Accessed February 2011 from http://web.archive.org/web/20070916144913/http://wp.netscape.com/newsref/pr/newsrelease67.html.

[Ngu09] Hubert Nguyen. WebGL aims at providing in-browser hardware-accelerated 3D without plug-ins, April 2009. Accessed February 2011 from http://www.ubergizmo.com/2009/08/webgl-aims-at-providing-in-browser-hardware-accelerated-3d-without-p

[Nod] Node.js. Noje.js. Accessed February 2011 from http://nodejs.org/.

[Ogg11] Erica Ogg. Sinofsky shows off windows 8 at d9. *CNET*, June, 2011.

[Pau09] Ryan Paul. Decoding the HTML 5 video codec debate. *ars technica*, July 2009.

[Per11] João Pedro Pereira. O facebook não é só para jovens. *Público*, 2011. Accessed June 2011 from http://www.publico.pt/Tecnologia/o-facebook-nao-e-so-para-jovens_1498043?all=1.

[PGSM10] Andrei Popescu, Eliot Graff, Jonas Sicking, and Nikunj Mehta. Indexed database api. W3C working draft, W3C, August 2010. Accessed February 2011 from http://www.w3.org/TR/2010/WD-IndexedDB-20100819/.

[Pin10] Pingdom. Study: Ages of Social Network Users, 2010.

[Pop10] Andrei Popescu. Geolocation api specification. W3C candidate recommendation, W3C, September 2010. Accessed February 2011 from http://www.w3.org/TR/2010/CR-geolocation-API-20100907/.

[PPM+01] Denise C. Park, Thad A. Polk, Joseph A. Mikels, Stephan F. Taylor, and ChristyMarshuetz. Cerebral aging: integration of brain and behavioral models of cognitive function — stateof the art. Technical report, University of Michigan, 2001.

[Rag05] Dave Raggett. Getting started with HTML, May 2005. Accessed February 2011 from http://www.w3.org/MarkUp/Guide/.

[RHJ99] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. HTML 4.01 Specification. W3C recommendation, W3C, December 1999. Accessed February 2011 from http://www.w3.org/TR/html401/.

[RLAK88] Dave Raggett, Jenny Lam, Ian Alexander, and Michael Kmiec. *Raggett on HTML 4*. Addison Wesley Longman, 1988.

[RSN+08] Nandini Ramani, Andrew Shellshear, Craig Northway, Vincent Hardy andErik Dahlström, Ola Andersson, Jon Ferraiolo, Doug Schepers, Dean Jacksonand Chris Lilley, Andrew Emmons, Anthony Grasso, Cameron McCormack, AndreasNeumann, Scott Hayman, Antoine Quint, and Robin Berjon. Scalable vector graphics (SVG) tiny 1.2 specification. W3C recommendation, W3C, December 2008. Accessed February 2011 from http://www.w3.org/TR/2008/REC-SVGTiny12-20081222/.

[rt] raghurajah and themesmok. Commenting api issues reports. From the Facebook Developers forums, accessed Jun 2010 from http://forum.developers.facebook.net/viewtopic.php?pid=247021 and http://forum.developers.facebook.net/viewtopic.php?pid=349538.

[Sen11] Sencha. Sencha touch, February 2011. Accessed February 2011 from http://www.sencha.com/products/touch/.

[Spe10] Ed Spencer. Offline Apps with HTML5: A case study in Solitaire, June 2010. Accessed February 2011 from http://edspencer.net/2010/06/offline-apps-with-html5-a-case-study-in-solitaire.html.

[Sun10] Veera Sundar. Geolocation in HTML5 – browser and device support, February 2010. Accessed February 2011 from http://veerasundar.com/blog/2010/02/geolocation-in-html5-browser-and-device-support/.

[Ten11] Pedro Tenreiro. User-centered design of a mobile storytelling application for older adults. Master's thesis, Faculdade de Engenharia de Universidade do Porto, 2011.

[The06] Tenni Theurer. Performance research, part 1: What the 80/20 rule tells us about reducing http requests. *Yahoo! User Interface Blog*, 2006. Accessed May 2011 from http://yuiblog.com/blog/2006/11/28/performance-research-part-1/].

[TLT04] Chrispin Thurlow, Laura Lengel, and Alice Tomic. *Computer Mediated Communication — Social Interacation and the Internet*. Sage Publications Ltd, February 2004.

[Twi11] Twitter. *API Documentation*, February 2011. Accessed February 2011 from http://dev.twitter.com/doc.

[Uni] Unity. Unity scripting. Accessed February 2011 from http://unity3d.com/unity/features/scripting.

[VKPV10] René Vutborg, Jesper Kjeldskov, Sonja Pedell, and Frank Vetere. Family storytelling for grandparents and grandchildren living apart. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, NordiCHI '10, pages 531–540, New York, NY, USA, 2010. ACM.

[W3Ca] W3C. About W3C. Accessed February 2011 from http://www.w3.org/Consortium/.

[W3Cb] W3C. W3C Facts. Accessed February 2011 from http://www.w3.org/Consortium/facts.

[W3Cc] W3C. W3C Members List. Accessed February 2011 from http://www.w3.org/Consortium/Member/List.

[WHA11] WHATWG. Frequently asked questions, January 2011. Accessed February 2011 from http://wiki.whatwg.org/wiki/FAQ#What_is_the_WHATWG.3F.

[Wri00] Kevin Wright. Computer-mediated social support, older adults, and coping. *Journal of Communication*, 2000.

[WSW05] B. K. Williams, S. C. Sawyer, and C. M. Wahlstrom. *Marriages, Families & Intimate Relationships*. Boston, MA: Pearson, 2005.