rslidy: Responsive Presentation Slides in HTML5 and CSS3

Markus Schofnegger

Institute for Information Systems and Computer Media (IICM),
Graz University of Technology
A-8010 Graz, Austria

02 Nov 2015

Abstract

This thesis describes rslidy, a new version of an HTML-based web tool, which allows users to create responsive slide presentations. It is based on the idea of Slidy2 written by Dave Raggett, but extends it to be responsive, work on multiple browsers and platforms, and adds several new features. This is achieved by using responsive web design methods like CSS3 Media Queries.

In the first part of this thesis, front-end web technologies like HTML5, CSS3 and JavaScript used during the development of rslidy are described. Furthermore, other web-based slide tools are presented and compared according to their features and implementations. rslidy and its functionality and architecture are explained in the second half. This includes rslidy's code base and all of its major features. It is also shown how to use it for presentations and which settings can be changed.

Finally, this thesis explores features still missing in rslidy and shows how they could be implemented during future updates. For example, advanced web technologies like WebSockets for remote navigation are mentioned.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

rslidy is a new version of a web-based presentation tool. It allows users to create responsive presentations by adding slides to an HTML file. This chapter explains the benefits of web-based presentation tools, especially when compared to traditional software. It also describes the motivation behind developing rslidy and why the choice to extend the original Slidy2 has been made. Afterwards, existing solutions with different approaches like google-slides and older versions of rslidy are explained.

The third chapter gives a detailed overview of the new implementation. Special attention is paid to the key features like the new navigation panel and gestures for mobile devices. The compatibility of rslidy and ideas for future improvements are also mentioned.

Finally it is shown, how to use rslidy for a presentation. The necessary steps to combine the rslidy library with an existing HTML presentation file are exactly described and all supported customisation options are explained. This includes settings available in the JavaScript file and user-defined CSS rules.

1.1 Web-Based Presentation Slides

In contrast to conventional presentation software like Microsoft's PowerPoint, web-based presentation tools do not need to be installed on the user's computer. Instead, they can be accessed using a web browser, a tool often available by default on major platforms. This allows for a hassle-free deployment of presentations, without having to worry about compatibility issues and different file formats.

The long list of PowerPoint's features also includes animation effects like fade-ins, which can be used while navigating from one slide to another. These effects often help to distract the audience rather than improving the quality of the slides. [Mahar, Yaylacicegi and Janicki, 2008] showed the same presentation to two different groups of people, one with animations, the other without them. It was found that the people who saw the presentation without animations scored better in this specific study. This means that one of PowerPoint's popular features is not needed for a convincing presentation, making simple and lightweight web-based presentation tools an even more attractive choice.

One of the first HTML-based alternatives to PowerPoint was Slidy, which was developed by Dave Raggett [Raggett, 2006b] and is now hosted on the web servers of the W3C (World Wide Web Consortium) [Raggett, 2006a]. It implements basic features like slide navigation using the left and right arrow keys and also allows users to create their own slides within an HTML file.

rslidy is an attempt to improve Slidy's responsiveness and to add new features, making it an even more sophisticated web-based solution. While developing rslidy, special attention was paid to the compatibility with different devices. Furthermore, it also supports touch and motion events on most modern smart phones.

1.2 Motivation

Many web-based slide presentation solutions are currently available and some of them will be described later on. While most already work well, they often do not scale or are not usable on mobile devices. rslidy was developed with special attention to the following attributes:

- *Responsiveness*: rslidy is able to scale between multiple screen resolutions and to adapt the size and position of its control elements.
- *Compatibility*: Most browsers are supported by rslidy and it also works on mobile devices such as modern smart phones.
- *Minimalism*: rslidy is implemented using a single JavaScript file and the code architecture behind it is straight-forward and easy to understand.

1.3 rslidy's Framework

rslidy is built from scratch. It does not share any source code with the original Slidy. It has no dependencies on other third-party libraries. Due to rslidy's backward compatibility, old slides containing the `slide` or `incremental` classes can be used without problems. How to use rslidy is shown in Section A.2.

Chapter 2

Front-End Web Technologies

HTML5, CSS3 and JavaScript are today's most commonly used front-end web technologies. In this chapter, they are briefly described. Furthermore, Microsoft's TypeScript language is covered, since rslidy is written in TypeScript and then compiled into JavaScript.

2.1 HTML5

HTML stands for Hypertext Markup Language and was originally released in 1992 [W3C, 1992]. It is a markup language providing a specific syntax used to structure elements on a website. HTML is now in its fifth version called HTML5 and includes many new features like audio and video support and the `canvas` tag. Unlike XHTML2, a one-time rival to HTML5, it does not stop older web pages working thanks to its backward compatibility [MacDonald, 2014].

HTML is used to define the basic structure of a website. From within an HTML document, other files like CSS stylesheets or JavaScript libraries and functions can be included. HTML is executed client-side and is interpreted by the web browser.

2.2 CSS3

CSS (Cascading Style Sheets) files are used to define the appearance of HTML elements in the web browser. The first version of CSS was published by the World Wide Web Consortium (W3C) in 1995 [W3C, 1995]. It is now available in its third version called CSS3 [W3C, 2015a].

CSS uses *declarations* (what styles to apply) and *selectors* (where to apply them). *Selectors* can address specific sets of HTML elements. Apart from simply selecting an element by an ID, selectors also support more sophisticated methods like addressing an element only when it is hovered over or when it is nested within another user-defined element. The different parts of the CSS syntax is shown in Figure 2.1.

With CSS, many different rules can apply to the same element. For instance, multiple colours can be applied to the same element on a website. The *Cascading* in CSS means that more specific CSS rules override more general ones. *Cascading* can also be deactivated for a single property by adding the CSS keyword `!important` to its value, which makes this property ignore values inherited from parent elements. Some of the new CSS3 features used in rslidy include `transform`, the `calc()` function and CSS media queries.

The `transform` property can be used to make an HTML element and its content appear smaller [W3C, 2015b]. The `calc` function allows web developers to use multiple measurement units within one CSS rule. For instance, `percentage` and `px` can be used simultaneously. The `calc` function works in combination with CSS properties like `width` or `height`. Finally, media queries can be used to select the target screen resolution or orientation for CSS rules. This help to differentiate between various devices, but also between screen orientations like landscape and portrait mode.

```
h1 { color: blue; width: 50%; }
```

Figure 2.1: The CSS Syntax. The selector is red and defines the elements. Properties are blue and their corresponding values are orange in this example. Everything inside the curly brackets is referred to as the declaration.

2.3 JavaScript

JavaScript [Mozilla, 2015c] is the most commonly used programming language in web applications [Flanagan, 2011]. It was created by Brendan Eich at Netscape in 1995 and called LiveScript [W3C, 2012]. LiveScript and JavaScript were originally intended to be run inside the web browser, allowing web developers to interact with HTML elements.

Today, JavaScript is a modern programming language able to be executed both client-side and server-side. While the client-side version still runs inside web browsers, server-side versions of JavaScript include open-source solutions like Node.js [Node.js Foundation, 2015], which are executed directly on a web server. The basis of JavaScript is the ECMAScript standard [ECMA, 2015], currently in its sixth version. While ECMAScript 5 is widely supported, many of ECMAScript 6's features are still not available on some web browsers. However, development on ECMAScript 7 has already begun. This version will include several new features like easier usage of asynchronous functions.

All major web browsers support JavaScript and most JavaScript built-in functions are compatible with them. In interactive web applications like rslidy, JavaScript is often used to accept user inputs and to start procedures based on these inputs.

2.4 TypeScript

TypeScript [Microsoft, 2012] is a programming language and was developed by Microsoft in 2012. Unlike JavaScript, it supports type declarations. TypeScript is not compatible with today's web browsers, but it can be translated to a JavaScript file by using its compiler. In large projects, TypeScript is used to help web developers to quickly find errors regarding the declarations and usage of variables before actually executing the code.

2.5 Progressive Enhancement

Progressive Enhancement is a technique first mentioned by Steven Champeon and Nick Finck in 2003 [Champeon and Finck, 2003]. Its idea is to build web pages with functionality on all web-enabled devices in mind. This is possible by starting with a static HTML code base, which works without problems on every web browser. Only then, stylesheets (CSS) are added to the HTML code in order to change the appearance of HTML elements. Being the most critical part of a web site when it comes to compatibility, JavaScript is added at the end of this process. This method ensures a minimalistic yet still usable website even on devices, which support neither CSS nor JavaScript.

Progressive Enhancement is not the only way to handle compatibility issues on different devices. Another method called Graceful Degradation was already used before 2003 and still is. Its idea is to build a new website for larger displays and for the most advanced web browsers. Testing the website on older browsers or on mobile devices is not the main focus of the development and usually takes place at the end of it. This often results in older browsers and devices not displaying the page correctly. Due to these issue and since the rise of mobile web usage, Progressive Enhancement is considered the future of Web Design [Foster, 2012].

2.6 Responsive Web Design

Responsive Web Design is a technique to build and design web sites. It focuses on providing an optimal and mostly similar web experience across all devices. Ensuring this is especially difficult when devices with large differences in screen size should be able to use the web site without problems.

The method relies on changing the appearance of elements based on the screen resolution of the user's device. This is possible thanks to CSS3 Media Queries, for example. In general, appropriate usage of CSS is enough to ensure a good user experience on all devices [W3Schools, 2015].

Chapter 3

Presentation Slides in HTML

Many solutions are available for creating presentation slides in HTML. The focus in this chapter is laid on the features of each solution, their responsiveness, their compatibility with multiple devices and the technologies used.

3.1 Slidy and Slidy2

One of the first web-based presentation tools was Slidy [Raggett, 2005] developed by Dave Raggett in 2006. It is now available in its second version, called Slidy2 [Raggett, 2006b].

3.1.1 Slidy2 Features

Slidy2 offers a very basic set of features. All slides are defined inside a single HTML file. Navigating within a presentation is done using the `Left` and `Right` arrow keys, or the `Pg Up` and `Pg Dn` keys. Jumping to the first and last slide is possible using the `Home` and `End` keys, respectively.

Slidy2 includes a status bar, located at the bottom of every slide. The status bar also contains only two clickable links: `help` and `contents`. The `help` link jumps to another HTML file, where the features and navigation of Slidy2 are explained. Slidy2 does not have a navigation panel, but offers a table of contents by clicking the `contents` link.

The status bar also shows the number of slides of the presentation and which slide is currently visible. It also includes a timer, helping the user to know how much time they have still left for the presentation. The total duration for the presentation in minutes can be changed using a meta tag in the slides' HTML file, for instance `<meta name="duration" content="3" />`, which would set the timer to three minutes. If the user does not want the status bar to appear in the presentation, it can also be hidden by using the `F` key.

Apart from contents inside `pre` tags, Slidy2 is very responsive and scales quite well to smaller screens, even when used in portrait mode. Navigation on devices with a touch screen works by swiping left or right. Additionally, it is possible to show the table of contents by swiping up and to hide it by swiping down. The largest disadvantage when using Slidy2 on a mobile device is the status bar being far too small, shown in Figure 3.1. The current slide number can hardly be seen and links cannot be clicked without zooming in. This means that the table of contents is very hard to use on smaller devices.

3.1.2 Slidy2 Implementation Details

Slidy2 is a very minimalistic presentation tool. Thus it can easily be used offline and is written entirely using HTML, CSS and pure JavaScript. Only three files are needed to use Slidy2:

- *slidy.js*
- *slidy.css*

Figure 3.1: Slidy2's status bar at the bottom of the screen is hardly visible on mobile devices.

- A user-defined HTML file containing the slides

Inside *slidy.js*, there is a single class called `w3c_slidy`. This class contains all the methods needed for the execution of Slidy2 and no other script files are included.

3.2 google-slides from Google I/O

A solution from Google was developed by Eric Bidelman in 2012 [Google, 2012] for Google I/O talks in the same year. It focuses on a modern and sleek design, many customisation options, and a huge set of features. Similar to dedicated presentation software, it also supports animations and transitioning effects.

3.2.1 google-slides Features

Google's google-slides is one of the most comprehensive solutions available. Its many customisation options can be changed within a single file named *slide_config.js*. Users are able to change the title, information about themselves, add different themes, and customise the general behaviour of the presentation. Options changed in this file affect the whole presentation, not just a selected set of slides.

Displaying the previous or the next slide is done by using the arrow keys or the `Pg Dn` and `Pg Up` keys, respectively. While there is also an overview mode available, which can be toggled by the user and shows the previous and the next slide of the currently visible slide, google-slides does not include an advanced navigation panel or any other navigation option. Thus, it is impossible to quickly jump to slides further away from the current one.

It is possible to change the appearance of the slides just by using different key buttons. A user can toggle a so-called *widescreen mode*, for instance, changing the aspect ratio of the slides from `16:9` to `4:3`. Other options include the possibility to highlight important sections on a slide. This can be used to show important parts of a displayed source code, for instance. The user is also able to define their own speaker notes, which can then be toggled on the corresponding slide by pressing the `P` key. These speaker nodes can be added within the HTML file itself and are overlaid on top of the displayed slide.

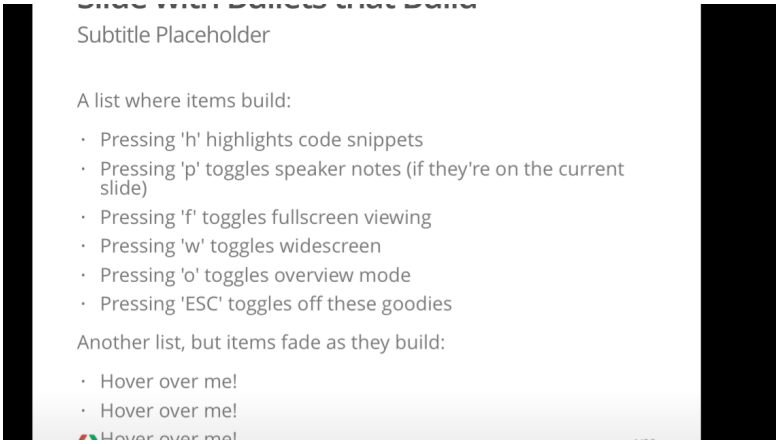
tp] 

Figure 3.2: Google's google-slides does not scale well in landscape mode on mobile devices. Both the title and the slide's content are not fully visible.

google-slides is currently supported on Chrome, Mozilla Firefox, Safari's desktop browser, Opera and on Microsoft's Internet Explorer. It also works on iOS 4.3 and later versions and on Chrome for Android. On mobile devices, it is only possible to switch between slides by using swipe gestures. Due to the lack of key buttons, other options like the afore-mentioned overview mode are not available on touchscreen devices. The biggest disadvantage, however, is that the whole presentation does not scale well when being used in landscape mode on mobile devices. This issue can be observed in Figure 3.2.

3.2.2 google-slides Implementation Details

google-slides works by using a basic HTML file and many different JavaScript files. It uses CSS for its styles. The following technologies are used:

- Compass, which is a CSS preprocessor used for browser compatibility (pure CSS can also be used, though)
- flexbox to define slide layouts
- RequireJS, a library used to load multiple JavaScript files or modules
- HTML5's window.postMessage to communicate with other browser windows (used for speaker mode)
- HammerJS for touch events

google-slides needs many different JavaScript files in order to work, making it a rather large package with many dependencies in comparison to Slidy2 or the new rslidy.

3.3 S5

S5 is short for Simple Standards-Based Slide Show System and, similar to Slidy and rslidy, it is a very light-weight presentation tool [Meyer, 2015]. It relies entirely on XHTML, CSS, and JavaScript and does not use any other web technologies. The initial version of S5 was written by Eric Meyer and released under a Creative Commons License in 2004 [Meyer, 2004].

3.3.1 S5 Features

Being one of the first alternatives to conventional presentation software, S5's set of features is very minimalistic when compared to google-slides and other more comprehensive solutions. One of the important advantages of

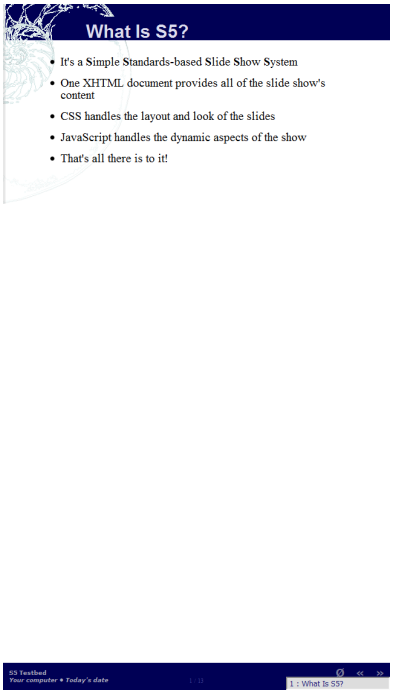


Figure 3.3: S5 has very small navigation controls on mobile devices.

S5 is the ability to auto-scale every text visible on the slides. Thus, the final screen resolution does not change the appearance of the presentation. On the other hand, zooming in and out on a web browser does not affect the size of the text and the auto-scale feature cannot be turned off while conducting a presentation. This makes it impossible to adjust the size of the content when needed.

Just like Slidy2 and rslidy, S5 also offers the possibility to create incremental content, for instance incremental lists. The user just adds the class `incremental` to lists and its items will be shown one at a time. This also works for other elements like images, allowing for very basic animation effects.

S5 implements an easy to use navigation system. The `Left` and `Right` arrow keys switch between slides. Alternatively, the `Up` and `Down` keys can be used. S5 does not include any possibility to navigate by using slide thumbnails, but it offers a table of contents, which can be found in the bottom right corner of every slide. These navigation controls can be shown by hovering over the bottom status bar with the mouse or by pressing the `C` key. If the user wants to see all the slides on one page without having to navigate, they can toggle the appearance of the presentation by hitting the `T` key.

S5 gives information about which slide is currently displayed in the bottom centre of the presentation. Unlike the navigation controls, the number of the current slide is always visible. It is possible to create bookmarks to specific slides. This can be done by adding a `#slide{number}` to the URL of the presentation, like in `http://www.example.com/demo.html#slide2`.

Being developed using very basic web tools, S5 is compatible with most major web browsers. An exception is Microsoft's Internet Explorer, where the table of contents located in the status bar is not usable.

Furthermore, S5 supports neither touch events nor tilt or shake gestures. The only possibility to navigate on a modern touchscreen device is by pressing the buttons provided in the bottom right corner or by using the table of contents. These elements are very small when compared to the rest of the content and quite hard to hit, which makes navigating on smaller touchscreens relatively difficult. This problem is shown in Figure 3.3.

3.3.2 S5 Implementation Details

S5 is a lightweight package using only XHTML, JavaScript and CSS. The most important files are:

- *slides.js*

- *slides.css*
- *s5-core.css*
- *framing.css*
- *pretty.css*
- *print.css*

S5's whole functionality is implemented in *slides.js*. This file does not contain any classes, instead everything is implemented using global functions and variables.

The file *s5-core.css* contains S5's core rules and is not intended to be edited by the user. Instead, *framing.css* and *pretty.css* should be used. The former can be edited to adjust the basic layout of the slides, while the latter is responsible for colours, fonts, text alignments and similar properties. Finally, *print.css* contains important rules for a printer-friendly version of a presentation.

3.4 S9 (Slide Show)

Slide Show (S9) is a presentation tool which allows users to create and author slides in plain text [Bauer, 2015b]. It was developed by Gerald Bauer and the first version was released in February 2008 [Bauer, 2008]. S9 is based on S6, also developed by Gerald Bauer [Bauer, 2015a], and uses most of S6's architecture. However, it adds a new way to edit or create slides using a Markdown language [Gruber and Swartz, 2004]. Creating a presentation by using the HTML file only is also possible, though. Unlike the other solutions presented in this chapter, S9 is entirely written in Ruby.

3.4.1 S9 Features

S9 is a comparatively modern alternative to conventional presentation software. It inherits many features from S5 discussed in the previous section. The navigation controls are very similar and even the status bar located at the bottom of the presentation is the same. It is possible to navigate using the arrow keys or the `Space` key. Alternatively, the heading of the currently displayed slide can be clicked, which also navigates to the next slide. The current slide number is displayed at the bottom of the presentation. S9 also includes a navigation menu very similar to S5, which can be shown by moving the mouse to the bottom right corner or by pressing the `C` key.

Unlike S5, S9 does not automatically scale the contents visible during the presentation. This allows the user to manually zoom in or out, adjusting the font size to their needs.

S9's implementation of incremental slide content is different compared to other solutions. Instead of adding a class `incremental` to a list, the user adds a class `step` to a `div` element, which causes the software to display items with this class one at a time.

Slide authors do not edit HTML files directly. Instead, S9 uses the Markdown language developed by John Gruber and Aaron Swartz in 2004 [Gruber and Swartz, 2004]. A Markdown file is then converted into an HTML file using the provided command line tools.

S9 is compatible with most major browsers, but its table of contents navigation control is not usable in Microsoft's Internet Explorer. This issue could also be observed in S5. Moreover, touch events like swiping and motion gestures like tilting are not recognised, making navigation on touchscreen devices rather difficult. The table of contents works, but is hard to use due to its small size and location at the bottom.

S9 allows users to install custom templates, automatically replacing the CSS files responsible for the presentation's appearance. This can be done by using the command line tools provided. It is also possible to add new features like a syntax highlighter by installing new templates.

3.4.2 S9 Implementation Details

S9 is entirely written in Ruby and can be downloaded using `rubygems` [Quaranto, 2015] with the command `gem install slideshow`. Markdown is used to create or edit slide contents. The user creates a whole presentation in a single Markdown file and this file is then translated into a browser-compatible HTML file by S9.

To display the navigation controls and for animation effects when switching slides, S9 uses JavaScript, more specifically jQuery. Two JavaScript files are required, *jquery.js* and *jquery.slideshow.js*. *jquery.js* is the jQuery JavaScript library. The other file was originally used in S6 and provides the core functionality needed for the slide show.

3.5 remark.js

Remark.js is a web-based presentation tool developed by Ole Petter Bang which uses HTML and JavaScript [Bang, 2015a]. Users edit slides using the Markdown language [Gruber and Swartz, 2004] directly inside the HTML file. It aims for a modern and sleek design, but lacks many features provided by other solutions.

3.5.1 remark.js Features

Remark.js describes itself as a *"simple, in-browser, Markdown-driven slideshow tool targeted at people who know their way around HTML and CSS"* [Bang, 2015b]. Some of the key features mentioned by the author are:

- The usage of Markdown.
- Syntax highlighting compatible with many different programming languages.
- Auto-scaling of slide content.
- Support for touch gestures.

Just like in S9, in remark.js users can edit slides using the Markdown language [Gruber and Swartz, 2004], which is converted to browser-compatible HTML code by its JavaScript library. However, the Markdown language is used directly inside the HTML file and editing slides in plain HTML is not easily possible.

Remark.js supports the highlighting of programming languages using a third party plugin. Furthermore, it scales every slide's content automatically, which results in a very similar appearance among different devices. Just like S5, which also supports the auto-scaling of slide content, remark.js does not allow the user to disable this feature during a presentation. This means that font sizes and other content cannot be adjusted dynamically by the user.

On devices connected to a keyboard, the arrow buttons can be used to navigate. Alternatively, the mouse's scroll wheel can be used. Remark.js is able to detect touch events and swiping left or right displays the next or previous slide, respectively.

A more sophisticated navigation menu like a table of contents is not included in remark.js. However, it is possible to add custom slide notes and show them using the `P` key, which toggles a so-called *Presenter Mode*. While using this mode, a timer indicating the remaining presentation time is also visible. Moreover, the number of the currently displayed slide can be seen in the bottom right corner of the presentation. A full overview with all available commands listed can be shown by pressing the `H` key. Various options like the aspect ratio used for the slides or the navigation controls available can be customised by the user.

Since it uses HTML and JavaScript, Remark.js is compatible with most modern browsers. However, some styles and features like the auto-scaling feature do not work on Microsoft's Internet Explorer.

3.5.2 remark.js Implementation Details

Remark.js is entirely written in HTML, Markdown and JavaScript. It uses CSS to style its elements. The core functionality is not implemented within a single JavaScript file, instead remark.js divides the implementation

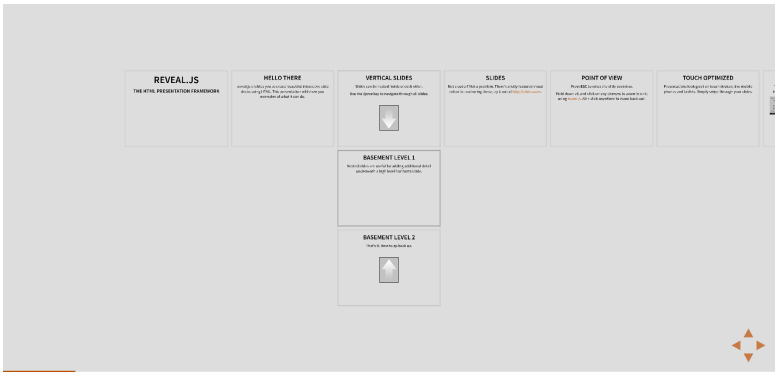


Figure 3.4: Reveal.js supports two-dimensional slide shows.

into many different modules. For instance, every component can be found in a different file. Syntax highlighting is added by using a third party library called highlight.js [Sagalaev, 2015]. Currently, more than 130 programming languages are supported.

Separate CSS files are not used, instead all CSS rules are applied within the HTML document itself and by JavaScript. This makes it rather difficult to customise the overall appearance when compared to other web tools using external CSS files.

3.6 reveal.js

Reveal.js is one of the largest packages presented in this chapter. It was developed by Hakim El Hattab and originally released as a tool called CSS 3D Slideshow [Hattab, 2015a]. This version was then updated in 2011, adding many new features.

3.6.1 reveal.js Features

Reveal.js is very different from basic HTML presentation tools due to its sophisticated navigation system. Instead of implementing a traditional one-dimensional slide show, reveal.js uses a two-dimensional arrangement for the slides, shown in Figure 3.4. This means that the user can also navigate vertically. Controls for the navigation are very similar to the other solutions presented in this chapter. Arrow keys switch between the slides in an horizontal way, but the `Up` and `Down` keys are used for vertical navigation.

To simply view all the slides one after another, without having to worry about horizontal and vertical navigation, the user can just use the `Space` key. An element in the bottom right corner shows in which directions the user can currently navigate to and a progress bar at the very bottom of the presentation indicates the number of slides remaining.

Reveal.js includes an overview mode, which can be toggled using the `Esc` key. The overview mode basically zooms away from the current slide, showing it together with some of its neighbouring slides. This is the only navigation menu available, a table of contents or similar features are not included.

Different themes are available and animated slide transitions are supported. These transitions can be customised by the user and it is possible to set a different transition for every different slide. Incremental lists are also supported, along with syntax highlighting and the embedding of HTML blockquotes. The presentation can be paused with the `B` key, which causes the current slide to fade out. By hitting the same key again, the presentation continues.

Reveal.js is compatible with all browsers tested, including Microsoft's Internet Explorer. Moreover, it is able to detect touch events and thus works flawlessly on mobile touchscreen devices. However, motion gestures like tilting and shaking are currently not supported.

3.6.2 reveal.js Implementation Details

Reveal.js is relatively modern and one of the most complete packages presented in this chapter. It has a large set of built-in features and is very customisable compared to other solutions. Furthermore, it is compatible with most devices. reveal.js is comparatively large and consists of multiple different files. The core functionality is implemented in a file called *reveal.js*, which uses only pure JavaScript. Reveal.js uses third party plugins to support some of its features, for instance *hightlight.js* [Sagalaev, 2015] and *zoom.js* [Hattab, 2015b]. The latter was also developed by Hakim El Hattab.

The user can edit and add new slides using the HTML file provided, where slides are separated with HTML5's `section` tag. Multiple CSS files are used to define the presentation's appearance.

3.7 diascope

Diascope is a slideshow program developed by Martin Stoll and last updated in February 2015 [Stoll, 2015]. In contrast to Slidy or rslidy, it is not a pure web-based presentation tool. Rather then relying on JavaScript and HTML, the contents of the slides are edited within a text file, which is then parsed by diascope. That is, diascope is a tool, which has to be installed on the user's Linux computer and can only be run from the command line.

3.7.1 diascope Features

The user defines slides inside a text file. This file is then used by diascope to create the presentation. The output is a video file, which can be played offline and also made available online due to comprehensive encoding options. The possible output video formats are `dv`, `mpeg2`, `mpeg4` and `flv`. The codec `mpeg4` is supported in every major browser [W3C, 2015c], including most mobile devices. Thus, online presentations with HTML5's `video` tag are easily possible.

Due to an actual video file being played during a presentation, there are numerous options for animation effects and the embedding of other media files, which are not supported when using pure HTML. This makes diascope very similar to conventional presentation software like Microsoft's PowerPoint, as far as animation effects or the embedding of other media files are concerned.

On the other hand, diascope only supports images for slides. It does not allow the user to write the slide contents directly and instead forces them to create pictures. Alternatively, the implemented `create` command can be used to add text to slides.

Furthermore, navigating inside the resulting video file is not possible, which means that switching slides manually cannot be done. The only way to simulate navigation is to seek inside the video file itself, which is inconvenient when compared with the other tools presented in this chapter.

When it comes to mobile devices, creating a presentation supported on all major web browsers is possible with the encoding options provided. Its responsiveness, however, cannot be compared to HTML-based presentations, because font sizes and similar attributes do not scale when using a pure video file.

diascope saves rendering time by recycling as many parts as possible from a previous run. It also supports multi-core rendering and the number of processing units can even be adjusted be the user.

To try out a new presentation before publishing it, the quality of the resulting video file can also be changed. This makes the encoding process a lot faster and allows for quick changes.

3.7.2 diascope Implementation Details

The programming language used for diascope is C. It can be started by compiling it on a Linux machine and running the resulting executable file. It is currently not possible to use diascope on other operating systems. A simple compile command for a provided text file defining the presentation would look like this:

```
diascope presentation.txt
```

3.8 The First rslidy and its Successor rslidy2

Previous versions of rslidy were developed with special attention to compatibility and responsiveness. The first rslidy was released in 2014 and its successor rslidy2 was released in 2015. The former rslidy from 2014 is referred to as *rslidy1* in this section.

Features

Both versions are similar when it comes to their features. Navigation works using the arrow keys. In rslidy1, the left mouse button can be used to switch to the next slide, in rslidy2 the `Home` and `End` keys lead the user to the first and last slide, respectively. rslidy1 includes a more sophisticated navigation menu than Slidy, by displaying the slide contents as thumbnail images. rslidy2 copies this feature, but also implements the table of contents found in Slidy. Both versions include a status bar showing the current slide and the total number of slides. In rslidy2, the buttons `Help` and `OptionsMenu` are also located in the status bar. The former redirects to a separate HTML file explaining how to use the software, while the latter allows the user to change some settings.

rslidy1 and rslidy2 are compatible with the browsers tested, except for rslidy2 not working in Microsoft's Internet Explorer. While both versions are able to detect swiping gestures used for navigation, rslidy2 also detects tilting and shaking gestures on supported devices. However, rslidy2's navigation menu suffers from a display bug on iOS and thus can hardly be used.

Implementation Details

Both rslidy1 and rslidy2 were developed by using only HTML, CSS and pure JavaScript. rslidy1 uses HammerJS [Tangelder, 2015] to recognise touch events, while rslidy2 uses Full-Tilt [Tibbett, 2015] to handle device motion events like tilting and shaking.

Chapter 4

The New rslidy

The new version of rslidy was written from scratch, but it still shares some features with the original Slidy2 and with previous versions of rslidy. While developing it, special attention was paid to a new minimalistic codebase and to its responsiveness across different devices. In this chapter, the new version of rslidy, its architecture and all of its improvements are described in detail.

4.1 Architecture of rslidy

The rslidy source code is completely written in TypeScript, which is then translated to a browser-compatible JavaScript file with TypeScript's compiler.

4.1.1 Files

By default, following files are used in rslidy:

- *rslidy.css*: This is the main stylesheet and is not intended to be edited by the user. Its main purpose is to ensure responsiveness among different platforms.
- *slides-default.css*: This file contains basic rules for the appearance of the slides. It can be modified according to the user's needs (see Appendix A).
- *rslidy.js* or alternatively *rslidy.min.js*: All of rslidy's functionality is implemented in *rslidy.js*, which was automatically created by TypeScript's compiler. The file *rslidy.min.js* is just a minified version of the larger file and was created using Grunt [Grunt, 2015].
- *rslidy.ts*: This is the TypeScript file which was used for rslidy's development.

The original TypeScript file, called *rslidy.ts*, contains two different classes: `Rslidy` and `Utils`. The former is separated into four different sections:

- Initialisation
- Styling and DOM Manipulation
- Key/Mouse Events and Listeners
- Navigation

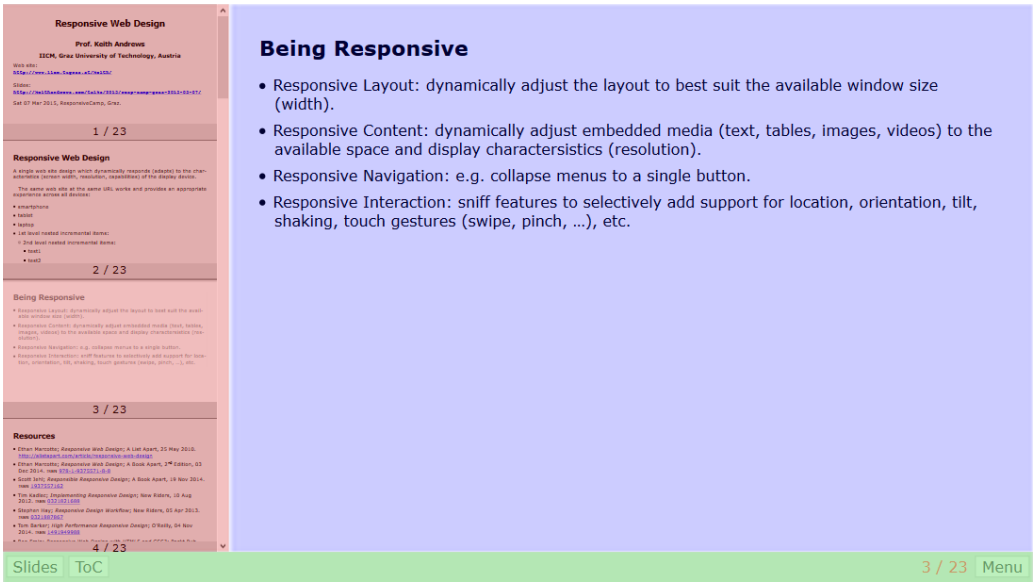


Figure 4.1: The user interface of rslidy. After the initialisation, the website is divided into three different parts. The red area is the navigation area containing both the Slides Overview and the Table of Contents, the blue area is the content area (slides) and the green area at the bottom is rslidy's status bar.

4.1.2 Initialisation

This section contains the constructor and all other methods needed by rslidy to start. It is also responsible for all member variables of the class. This includes the JavaScript settings, which can be edited by the user, as further explained in Appendix A. The main method in this section is the `init()` method. It creates a new instance of the `Utils` class, is responsible for the DOM (Document Object Model) manipulation, and changes settings according to the platform or operating system being used. The `init()` method also adds all event listeners. They are used to recognise button actions and touch gestures on supported devices, and also ensure the correct positioning of elements after the window is zoomed or resized.

4.1.3 Styling and DOM Manipulation

rslidy reads the content of the `body` tag found in the HTML file and modifies it to create a presentation. In particular, three main elements are added by rslidy:

- Navigation Area (containing the Slides Overview and the Table of Contents)
- Status Bar
- Menu

The slide navigation control is added by splitting the content of the original body into two different parts, one of them being the new navigation area. Both the Slides Overview and the Table of Contents are generated and added to this navigation area. The status bar and the menu are simply added to the body as new HTML elements, whose positions are defined by rslidy's stylesheet file. A detailed overview of rslidy's GUI (Graphic User Interface) along with the new elements added is shown in Figure 4.1.

All methods in this section are called once during the `init()` method, except for `adjustOverviewPanel()`. This method is responsible for resizing the thumbnail images inside the Slides Overview panel and is called whenever the browser window is resized or zoomed.

4.1.4 Key and Mouse Events and Listeners

Key and mouse events and listeners for them are used to recognise button actions and touch events. All listeners are added to their related HTML elements by calling the `addListeners()` method. Listeners added to the HTML page include:

- Button listeners (overview buttons, menu button, and all the buttons in the menu).
- Touch event listeners for swiping gestures.
- Window listeners for resizing.
- Window listeners for tilting and shaking.

Adding listeners is part of the initialisation and thus this method is called in the body of the `init()` method.

4.1.5 Navigation Methods

All other methods found inside the `Rslidy` class are responsible for the navigation itself, such as the transition between two slides. Special features, for instance incremental lists, are also handled within these methods.

4.1.6 The `Utils` class

Finally, *rslidy.ts* contains a `Utils` class. This class provides functions often used by rslidy, but not directly related to it. For example, it implements methods for casting between different data types or obtaining the current width of the browser window.

4.2 Cross-Platform Compatibility

The *'r'* in rslidy stands for *responsive*. rslidy was developed for use on different platforms and screen resolutions. rslidy is able to react to different screen sizes and device capabilities.

4.2.1 The Stylesheet

rslidy's appearance is mainly defined by the *rslidy.css* file. To ensure compatibility among different screen resolutions, relative CSS units, especially `percentage` and `em`, are preferred to physical units like `px`. Platform-specific CSS prefixes were used when needed as a fallback, including:

- `-ms-*`
- `-moz-*`
- `-webkit-*`

For example, the CSS rule `transform: scale3d(x, y, z)` used to create the slides' thumbnail images does not work on iOS devices without the `-webkit-*` prefix.

CSS3 media queries are used to define different CSS rules for different screen resolutions and aspect ratios. The content area, for instance, is only wrapped in landscape mode.

4.2.2 Web Browsers and Platforms

rslidy is compatible with most major web browsers, both on desktop computers and on mobile devices. It was tested on current versions of Mozilla Firefox, Microsoft Edge, Microsoft Internet Explorer, Google Chrome, Apple Safari and Opera. Apart from the Low Light Mode, all of rslidy's features work well on every web browser mentioned. The reason for the Low Light Mode not working properly is that it uses the CSS3 `filter: invert(x)` property. CSS3 filters are not enabled by default on Microsoft Edge. On Microsoft's Internet Explorer, the `invert` value is not supported at all, making the Low Light Mode completely unusable.

On mobile devices, Apple Safari for iOS and Google Chrome for iOS were tested. rslidy works without any problems and all touch and motion events are fully compatible with these mobile browsers.

4.2.3 Changing the Size of the Presentation

Zooming and resizing the window is fully supported in the new rslidy. The user can adjust the zoom level by using the browser's built-in features and rslidy will automatically recalculate the size and position of all affected elements. For easier usage on mobile phones, it is possible to adjust the zoom level by using the respective buttons, shown in Figure 4.3.

4.2.4 Feature Sniffing

rslidy adapts automatically to the current environment. For example, motion events like tilting and shaking are disabled if they are not supported by the user's web browser and options to turn them on are greyed out.

4.3 Slides and Navigation

The different navigation options of rslidy are described in this section. Moreover, it is explained how to use incremental lists and how to edit or add new slides. Furthermore, the status bar is mentioned, which helps users to change basic settings during a presentation.

4.3.1 Slides Navigation

One of the most important goals when developing a web-based presentation tool is to provide an easy way to navigate through the presentation's slides. Just like other presentation tools explained in Chapter 3, the user is able to press the `Left` and `Right` arrow keys to navigate to the previous and next slide, respectively. Alternatively, the `Pg Down` and `Pg Up` keys can be used. Users can quickly jump to the next slide by pressing the `Space` key or the left mouse button. Moreover, the status bar contains two links, which allow to navigate without using a keyboard. These links are not visible on very small screens. When using the navigation keys while the Slides Overview is open, the current slide is highlighted and automatically scrolled to in the overview panel.

rslidy also allows the user to change slides by modifying the URL. Adding `#3` to an opened HTML slide file URL (like `demo.html#3`) navigates to the third slide, for instance.

4.3.2 Slide Overviews

rslidy implements a Table of Contents and a thumbnail-based Slides Overview. These can be seen in Figure 4.2.

The Table of Contents contains a list of all slide titles (headings). These are automatically fetched from the main HTML file. By selecting a title, the system navigates to the corresponding slide.

The new Slides Overview was created using the CSS3 properties `transform: scale3d(x, y, z)` and `transform-origin: 0px 0px 0px`. The aspect ratio of every thumbnail image is calculated dynamically using the aspect ratio of the browser window. It is recalculated when the size of the window changes during the presentation.

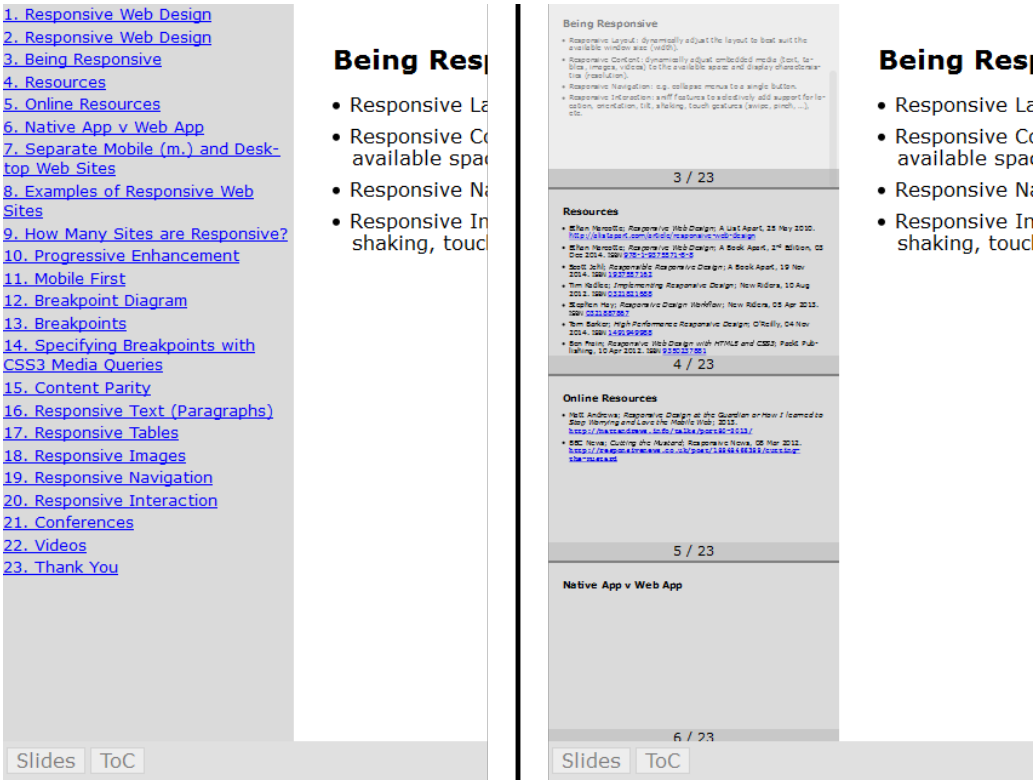


Figure 4.2: The navigation controls implemented in rslidy. On the left, the original Table of Contents can be seen. On the right, the new thumbnail-based Slides Overview is shown. The current slide is clearly highlighted in the overview panel.

Both overviews hide themselves after a slide has been selected by the user. This behaviour can be disabled by changing rslidy's settings found in the JavaScript file (see Section A.6).

Unlike former versions of rslidy, which used a hamburger icon, the new navigation controls are activated using the buttons on the left side of rslidy's status bar. Alternatively, the table of contents can be shown by clicking on the number of the current slide, located on the right side of the status bar.

4.3.3 Incremental Lists

rslidy supports incremental lists. They are defined by adding the `incremental` class to the HTML `ol` or `ul` elements. Incremental lists allow the user to reveal items one by one. To reveal the whole list at once, `Shift - Right Arrow` can be pressed. A fully visible list can be hidden by pressing `Shift - Left Arrow` key.

4.3.4 Status Bar

The status bar is always located at the bottom of the presentation. This is done by using CSS's `position: absolute` property. The `position: fixed` property could also be used, but causes flickering issues on iOS devices.

Besides the navigation buttons and the current slide's number already mentioned, a menu button can be found on the right side of the status bar. When activated, it shows a menu, which allows the user to toggle the detection of motion gestures and the Low Light Mode. It also contains a button for a help section. By clicking it, a dialog box is shown, which shortly explains the key bindings and how navigation in rslidy works. Moreover, the status bar contains two links, which allow users to navigate without using a keyboard. These links are hidden on devices with very small screens.

The right side of the status bar together with the opened menu is shown in Figure 4.3.

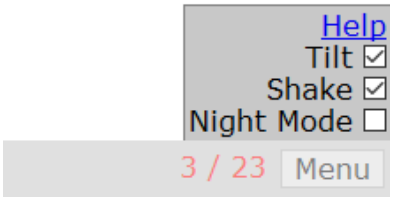


Figure 4.3: rslidy's menu contains checkboxes for toggling motion gestures, the Low Light Mode and the help section.

4.3.5 Editing and Adding New Slides

Slides for the presentation can be edited using the main HTML file. All slides are stored in this file and use the class `slide` to be correctly detected by rslidy. Slides can be created by adding new `div` elements for them and are numbered automatically by rslidy. In terms of HTML classes used, the new version is fully compatible with older versions of rslidy and with the original Slidy2, making it possible to use older slide decks with the new rslidy.

4.4 Touch Events and Motion Gestures on Mobile Devices

Motion gestures on mobile devices have become an important interaction method in the last few years. Studies have shown that over 80 percent of smart phone users would occasionally use them [Ruiz, Li and Lank, 2011] when trying to complete a task.

rslidy detects both touch events and motion gestures by using various JavaScript events. Their listeners are added to the corresponding elements in the `addListeners()` method called during rslidy's initialisation.

The detection of all gestures described in this section can be customised by the user, for example to increase or decrease the sensibility of the detection.

4.4.1 Swiping

Swiping means to hold down the finger on the screen while moving it and releasing it after movement. This type of interaction allows users to navigate through the available slides on mobile devices, which do not have a keyboard attached to them. Swiping to the left leads the user to the next slide, while swiping to the right leads them to the previous one.

There are various third party libraries available, which add touch gesture support to a JavaScript project, for example HammerJS [Tangelder, 2015]. They were not used with rslidy, instead a custom solution was developed using the three relevant JavaScript events:

- `touchstart`, fired when a user starts touching the screen
- `touchmove`, repeatedly called while moving the finger while touching the screen
- `touchend`, fired when a users stops touching the screen

Monitoring the time between these events' activations allows rslidy to detect swipe gestures made by the user. By calculating the direction of the movement, rslidy distinguishes between right-to-left and left-to-right swipes.

Swipe gestures rely on the events mentioned above. These are supported on all major mobile browsers and also on some desktop browsers [Mozilla, 2015d].

4.4.2 Tilting

In addition to swiping gestures, users can tilt the device to the left and to the right in order to navigate to the previous and next slide, respectively. rslidy detects this kind of gesture using JavaScript's `deviceorientation` event. It is supported on most mobile browsers, except for Internet Explorer Mobile and Opera Mobile [Mozilla, 2015b]. If tilt gestures are not desired by the user, they can be disabled using rslidy's settings menu shown in Figure 4.3.

In rslidy, tilting is implemented by calculating the time between orientation updates. The lower the intervals, the higher the change rate of the orientation is. This rate can be described as the *speed* of tilting the device. Using this method, slow or slight tilting is not recognised as an action done on purpose.

rslidy adjusts its calculation to the current orientation of the device itself. If the device is switched to landscape mode, the detection is basically rotated by 90 degrees.

4.4.3 Shaking

While tilting means to rotate the device around one or more of its axis, shaking refers to the rapid movement of a smart phone without necessarily rotating it. rslidy detects a user shaking the device using JavaScript's `devicemotion` event. Just like `deviceorientation`, this event is supported on most mobile browsers, with the exception of Microsoft's Internet Explorer Mobile and Opera Mobile [Mozilla, 2015a].

In rslidy, shaking the device causes the presentation to reset to the first slide. Similar to tilt gestures, there is a certain threshold required for shake gestures to cause rslidy to react. This threshold can be customised by the user. Shake gestures can also be turned off using rslidy's settings menu shown in Figure 4.3.

4.5 Low Light Mode

The Low Light Mode is a new feature, which was not included in previous versions of rslidy. Its main advantage is that darker colours can help to reduce eye strain when an interface is used in a dark environment [Few, 2012].

In rslidy, CSS filters are used, especially the `filter: invert(x)` property, in order to change the colours of the whole presentation. Images and other graphical representations are not affected by the Low Light Mode and keep their original appearance. The colours used for the dark representation of rslidy can be customised by the user.

The CSS `filter: invert(x)` property is not supported by all browsers. By default, it is disabled on Microsoft's Edge browser, but can be activated by the user. On Microsoft's Internet Explorer, it does not work at all. Apart from these two web browsers, the Low Light Mode works on all other major desktop and mobile browsers without any restrictions.

In Figure 4.4, rslidy's appearance in its Low Light Mode is shown. For comparison, rslidy's normal appearance is shown in Figure 4.5. The Low Light Mode can be toggled using rslidy's settings menu shown in Figure 4.3.

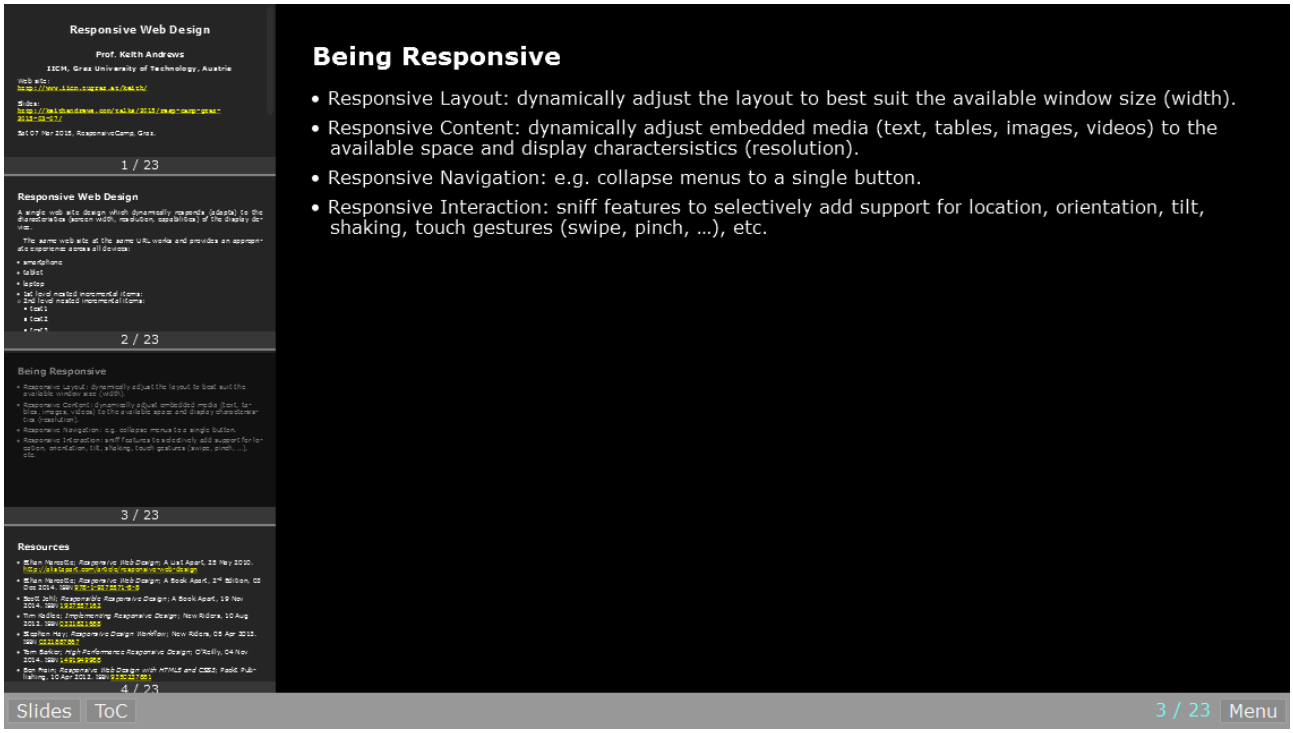


Figure 4.4: In Low Light Mode, darker colours are used to reduce a user's eye strain in darker environments.

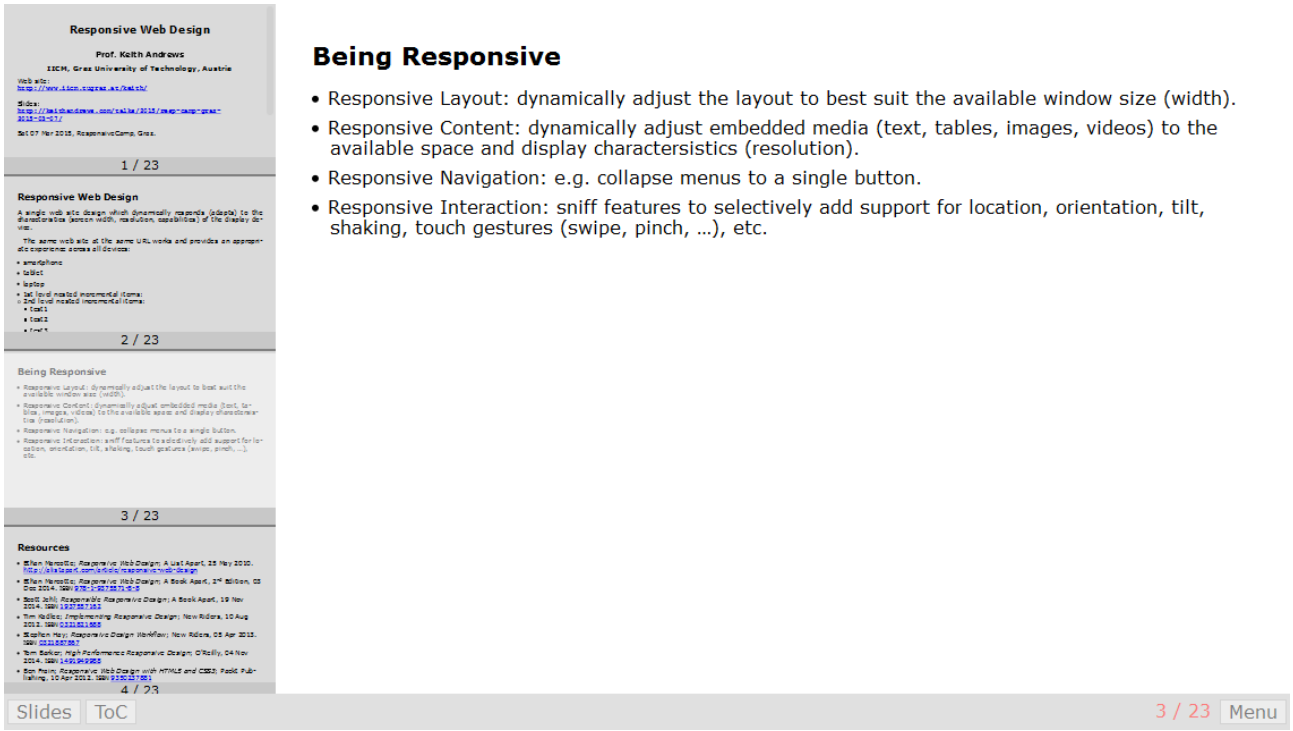


Figure 4.5: rslidy's normal appearance with the Low Light Mode disabled.

Chapter 5

Selected Details of the Implementation

5.1 Low Light Mode

Inverting the colours of all documents except for images does not always work when using SVG graphics. In this case, the whole figure and also HTML's `figcaption` elements have to be separately inverted as shown in Listing 5.1. The `switchElementsClass(elements, class)` method is used to toggle the existence of a class for a given list of HTML elements.

Another problem is to allow the user to customise the colours used during Low Light Mode. As the CSS's `filter: invert(x)` also inverts colours explicitly applied by stylesheets, the colours entered by the user have to be inverted in order to keep them when inverted again by rslidy's Low Light Mode implementation. This means that the hexadecimal value of the inverted colour has to be manually calculated using the given colour of every affected element. The JavaScript code responsible for this procedure is shown in Listing 5.2.

5.2 SVG Graphics on iOS

One of iOS's compatibility problems is that the `image` tag cannot be used within SVG files together with a Base64-encoded image. This would cause this very part of the SVG graphic to not appear on iOS devices, regardless which browser is used. Therefore, a workaround had to be found. The solution is to embed existing SVG elements using HTML's `object` tag. The SVG image is now not rendered as a simple image any more, but instead as a multimedia element, which works on iOS devices. The code used to change the tags is shown in Listing 5.3.

This modification leads to a major change within the HTML code written by the user. For this reason, it is turned off by default. It can be toggled by using the settings found in the JavaScript file described in Section A.6. The corresponding method is only called on iOS devices. This means that tags are not changed on already supported platforms.

```
// Invert images
var imgs = document.getElementsByTagName("img");
this.utils.switchElementsClass(imgs, class_color_invert);

// Invert figures
var figures = document.getElementsByTagName("figure");
this.utils.switchElementsClass(figures, class_color_invert);

// Invert figure captions
var figcaptions = document.getElementsByTagName("figcaption");
this.utils.switchElementsClass(figcaptions, class_color_invert);
```

Listing 5.1: Code used to invert all images and figures again.

```
var color_rgb = getComputedStyle(elements[i]).getPropertyValue("color");
var color_hex = "#";
var rgx = /\d+/g;
var match;
while ((match = rgx.exec(color_rgb)) != null) {
    var inverted = 255 - match[0];
    if (inverted < 16)
        color_hex += "0";
    color_hex += inverted.toString(16);
}
elements[i].style.color = color_hex;
```

Listing 5.2: In Low Light Mode, colours chosen by the user have to be manually inverted in order to stay the same after applying the CSS filters.

```
// Elements to consider
var items = document.querySelectorAll('.svg');

// Change the tags
for (var i = 0; i < items.length; i++) {
    var item_new = '<object style="display:table-cell;pointer-events:none;z-index:-10;"
        class="svg" data="' + items[i].getAttribute("src") + '" type="image/svg+xml
       "></object>';
    items[i].outerHTML = item_new;
}
```

Listing 5.3: This listing shows how `object` tags can be added for SVG images on iOS devices.

Chapter 6

Future Work

In this chapter, possible future features of rslidy are discussed. A few of them are already implemented in previous versions of rslidy or in other web-based presentation tools.

6.1 More Touch Events and Key Bindings

Currently, rslidy support only left and right swipes to navigate. In the future, additional touch events like double taps could be used to show the navigation panel, for example. Moreover, Swiping up or down could be used to skip incremental lists or multiple slides.

Compared to other web-based presentation tools mentioned in Chapter 3, rslidy supports only a small amount of key-based actions. Further key bindings could be used to quickly toggle the navigation panel or various settings like the Low Light Mode.

6.2 Remote Navigation using Mobile Devices

The first version of rslidy published in 2014 includes a WebSocket Remote Control possibility. With a server set up and running, users can remotely control a presentation taking place on a certain device by using their smart phone, for example.

This feature was implemented using the master-slave-principle. A master and multiple slaves are able to connect to the server and whenever the master sends a command, the server forwards it to all slaves. This means that even multiple presentation devices can be controlled with one smart phone acting as the master. A similar feature is currently not implemented in the new version of rslidy, but could be developed in the future.

6.3 Slide Editor

To make rslidy even more user-friendly, an editor for slide contents could be included. This feature would allow easier modifications of the slide deck when compared to editing the plain HTML file. Additionally, slide decks could be quickly modified during the presentation without having to open another file.

As an alternative to a fully functional editor, a markup language like Markdown could be used [Gruber and Swartz, 2004]. This solution is already implemented in other web-based presentation tools like S9 and remark.js.

6.4 More Information during the Presentation

An indicator showing how much time the user has still left for their presentation was already implemented in Slidy and Slidy2. Other solutions like remark.js also include this feature.

A similar option is not available in rslidy, but could be a useful feature for upcoming versions. The basic timer in Slidy2 could even be improved in order to show the time for each slide instead of the total amount of time for the whole presentation.

Another feature would be the possibility to add speaker notes to the presentation. They would be part of specific slides and could be shown by the user whenever needed. This feature is included in google-slides, for example.

Chapter 7

Concluding Remarks

Web-based presentation tools exist for many years now, but they still have trouble to completely replace conventional software like Microsoft's PowerPoint or Apple's KeyNote. While being feature-rich, they often lack important attributes, for example the compatibility between multiple platforms and the responsiveness on mobile devices.

In this work, we learned how elevated a presentation tool based on HTML, CSS and pure JavaScript can be. These technologies alone can be used to achieve a convincing final product. This shows that conducting a quality presentation is not only possible with conventional computer software any more and by using web technologies, every presentation tool can instantly be made cross-platform.

Apart from describing already existing solutions, rslidy, a new approach, was presented and explained in detail. Based on the idea of the original Slidy2, it is a lightweight web tool and allows users to create their own presentations. While developing rslidy, special attention was paid to its responsiveness across different devices and screen resolutions. It also includes more features and is overall more user-friendly and adaptive than its previous versions. Furthermore, some ideas for future improvements were shown and also briefly explained. By implementing the features mentioned, rslidy could become even more elevated and competitive when compared to other solutions.

In conclusion, we can say that web-based presentation tools are indeed a strong alternative to conventional presentation software and more often than not the fastest solution when slide decks need to be made available on many different devices.

Appendix A

User Guide

In this chapter, detailed instructions are given on how to use and customise the new version of rslidy. This includes the adding of rslidy to an own HTML presentation and a description of the HTML file structure and important class names. Moreover, all of rslidy's settings are described. By changing them, the appearance as well as the behaviour of the presentation can be altered to satisfy a user's needs.

A.1 Extracting the rslidy Archive File

Following file structure can be found after having extracted the *rslidy.zip* archive:

```
rslidy.zip
├── css
│   ├── normalise.css
│   ├── reset.css
│   ├── rslidy.css
│   ├── rslidy-combined.min.css
│   └── slides-default.css
├── js
│   ├── rslidy.js
│   ├── rslidy.ts
│   └── rslidy.min.js
└── demo.html
```

The archive not only contains files needed by rslidy, but also additional files. They help the user to customise their own copy of rslidy.

normalise.css and *reset.css* are not necessarily needed by rslidy, but contain normalisation rules like `margin: 0` for some HTML elements. The *rslidy.css* file is needed by rslidy in order to work. This file is not intended to be modified by the user. Instead, the *slides-default.css* file should be used to add custom styles to the presentation. However, if custom styles are not necessary, *rslidy-combined.min.css* can be used, which is a minified version containing all other CSS files.

rslidy.js is the basic JavaScript file directly created by the TypeScript compiler. It is the large version, thus not minified, and contains all comments from the original *rslidy.ts*. If code customisations are not needed by the user, the *rslidy.min.js* file should be used, as it is the smallest and most portable version of rslidy.

The two HTML files contain a simple demo presentation (*demo.html*) and a help file also used by rslidy. The demo presentation file can be removed without compromising the functionality of rslidy. An absolute minimum version of rslidy without the user's presentation would contain the files shown in Figure A.1.

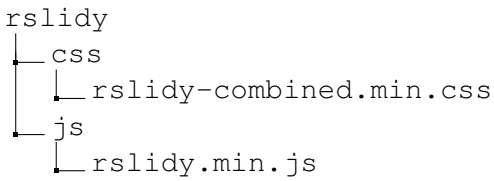


Figure A.1: This figure shows the files needed for the absolute minimum version of rslidy.

```
1 <link rel="stylesheet" href="css/reset.css"/>
2 <link rel="stylesheet" href="css/normalise.css"/>
3 <link rel="stylesheet" href="css/rslidy.css" />
4 <link rel="stylesheet" href="css/slides-default.css"/>
5 <script src="js/rslidy.js" charset="utf-8" type="text/javascript"></script>
```

Listing A.1: These commands are necessary to include rslidy.

A.2 Adding rslidy to a Project

After having prepared the files needed for rslidy and the presentation file itself, rslidy has to be added to the HTML file. This can be done by using HTML's `link` and `script` tags as shown in Listing A.1. Depending on the user's file structure, the location of the files might certainly be different.

A.3 The `slide` and `titleslide` Classes

It is possible to edit existing slides or to add new ones by editing a single HTML file. Slides are represented as `div` elements with certain classes assigned to them. The HTML code used to create a slide together with its `div` element and the `slide` class is shown in Listing A.2. Additionally, the class `titleslide` can be used together with `slide` to mark the first slide of a presentation.

A.4 Using Incremental Lists

Similar to the original Slidy and Slidy2, the new version of rslidy also supports incremental list, allowing the user to show list items one at a time. Incremental lists can be used adding the class `incremental` to an HTML list element such as `ol` or `ul` as demonstrated in Listing A.3.

It is possible to use nested lists with multiple levels by adding list elements to `li` elements of other lists. In this case, every list element needs to have its own `incremental` class added.

A.5 Customising the Appearance of the Presentation

If user-specific CSS rules are needed, they should be added to the *slides-default.css* file, which already contains a very basic set of rules. One of the sections in this file defines the colours used for rslidy's Low Light Mode feature discussed in 4.5. These rules override the values obtained by inverting the original colours and thus represent the final colours displayed in the presentation.

A.6 JavaScript Settings for the Presentation

Using the JavaScript file, many different settings of rslidy can be changed. This allows users to adjust the presentation to their specific needs.

```
<div class="slide">
<h1>Sample Slide</h1>

<p>
Some content for the sample slide.
</p>

</div>
```

Listing A.2: rslidy uses the class `slide` to mark slide content.

```
<div class="slide">
<h1>Sample Slide with an Incremental List</h1>

<ul class="incremental">
<li>List Item 1</li>
<li>List Item 2</li>>
<li>List Item 3</li>
</ul>

</div>
```

Listing A.3: In rslidy, incremental lists can be created using the `incremental` class with `ul` elements.

A full list of available settings is shown in Table **??**. All these settings can by found inside the *rslidy.js* file and are located in the `Rslidy()` function used to initialize the `Rslidy` object. In the original *rslidy.ts* file, these settings can be found inside the constructor method of the `Rslidy` class.

Variable Name	Description
boolean `close_menu_on_selection`	If true, the settings menu closes after having selected an option.
boolean `close_navigation_on_selection`	If true, the navigation panel is closed after having selected a slide.
boolean `start_in_low_light_mode`	If true, rslidy starts in Low Light Mode.
boolean `block_slide_text_selection`	If true, text on slides cannot be selected.
boolean `svg_fix_on_ios`	If true, `object` tags are added for SVG images when using iOS devices.
float `custom_aspect_ratio`	The aspect ratio for the thumbnail images in the navigation panel. Values like 4:3 and 16:9 can be used. If 0, the aspect ratio is dynamically calculated and updated by rslidy.
int `custom_width`	The width to calculate the custom aspect ratio. Only used when `custom_aspect_ratio` is greater than 0.
float `overview_slide_zoom`	Sets the zoom for the thumbnail images in the navigation panel. Default is 1.0.
int `doubletap_delay`	Maximum delay in ms between two taps to be recognised as a double tap. Currently not used.
float `min_slide_zoom`	The minimum zoom level (font size) of the content area in em units. Default is 0.2.
float `max_slide_zoom`	The maximum zoom level (font size) of the content area in em units. Default is 3.0.
float `zoom_step`	Defines for how many em units the zoom level of the content area changes in each step. Default is 0.2.
float `tilt_sensitivity`	The sensitivity used for tilt gestures. Higher sensitivity means less tilting required. Default is 3.0.
float `shake_sensitivity`	The sensitivity used for shake gestures. Higher sensitivity means less shaking required. Default is 1.0.

Table A.1: The settings for rslidy inside the JavaScript file.

Bibliography

Bang, Ole Petter [2015a]. *Remark*. 2015. http://remarkjs.com/ (cited on page 12).

Bang, Ole Petter [2015b]. *Remark - Description*. 2015. http://gnab.github.io/remark/#3 (cited on page 12).

Bauer, Gerald [2008]. *Slide Show (S9) - Versions*. 2008. https://rubygems.org/gems/slideshow/versions (cited on page 11).

Bauer, Gerald [2015a]. *S6*. 2015. https://github.com/slidekit/s6 (cited on page 11).

Bauer, Gerald [2015b]. *Slide Show (S9)*. 2015. http://slideshow-s9.github.io/ (cited on page 11).

Champeon, Steven and Nick Finck [2003]. *Inclusive Web Design For the Future with Progressive Enhancement*. 2003. http://hesketh.com/publications/inclusive_web_design_for_the_future/ (cited on page 4).

ECMA [2015]. *ECMAScript 2015 Language Specification*. Ecma International. 2015. http://ecma-international.org/ecma-262/6.0/ (cited on page 4).

Few, Stephen [2012]. *Black or White: What Color Works Best for the Background of a Screen?* 2012. https://perceptualedge.com/blog/?p=1445 (cited on page 23).

Flanagan, David [2011]. *JavaScript: The Definitive Guide: Activate Your Web Pages*. 6th edition. O'Reilly Media, 2011. ISBN 0596805527 (cited on page 4).

Foster, Aidan [2012]. *Progressive Enhancement - A Technique For Building Future Friendly Websites*. 2012. http://responsivedesign.ca/blog/progressive-enhancement-a-technique-for-building-future-friendly-websites (cited on page 4).

Google [2012]. *google-slides*. 2012. https://code.google.com/p/io-2012-slides/ (cited on page 8).

Gruber, John and Aaron Swartz [2004]. *Markdown*. 2004. http://daringfireball.net/projects/markdown/ (cited on pages 11, 12, 27).

Grunt [2015]. *Grunt - The JavaScript Task Runner*. 2015. http://gruntjs.com/ (cited on page 17).

Hattab, Hakim El [2015a]. *Reveal JS*. 2015. http://hakim.se/projects/reveal-js (cited on page 13).

Hattab, Hakim El [2015b]. *Zoom JS*. 2015. https://github.com/hakimel/zoom.js (cited on page 14).

MacDonald, Matthew [2014]. *HTML5: The Missing Manual*. 2nd edition. O'Reilly Media, 2014. ISBN 1449363261 (cited on page 3).

Mahar, Steve, Ulku Yaylacicegi and Thomas N. Janicki [2008]. "Less is More When Developing PowerPoint Animations". In: *The Proceedings of the Information Systems Education Conference 2008*. 2008. http://isedj.org/7/82/ (cited on page 1).

Meyer, Eric [2004]. *s5 - Release*. 2004. http://meyerweb.com/eric/thoughts/2004/10/18/ (cited on page 9).

Meyer, Eric [2015]. *s5*. 2015. http://meyerweb.com/eric/tools/s5/ (cited on page 9).

Microsoft [2012]. *TypeScript*. 2012. http://typescriptlang.org/ (cited on page 4).

Mozilla [2015a]. *DeviceMotionEvent*. Mozilla Foundation. 2015. https://developer.mozilla.org/de/docs/Web/API/DeviceMotionEvent#Browser_compatibility (cited on page 23).

Mozilla [2015b]. *DeviceOrientationEvent*. Mozilla Foundation. 2015. https://developer.mozilla.org/de/docs/Web/API/DeviceOrientationEvent#Browser_compatibility (cited on page 23).

Mozilla [2015c]. *JavaScript*. Mozilla Foundation. 2015. https://developer.mozilla.org/en-US/docs/Web/JavaScript (cited on page 4).

Mozilla [2015d]. *Touch Events*. Mozilla Foundation. 2015. https://developer.mozilla.org/en-US/docs/Web/API/Touch_events#Browser_compatibility (cited on page 22).

Node.js Foundation [2015]. *NodeJS*. Node.js Foundation. 2015. https://nodejs.org/en/ (cited on page 4).

Quaranto, Nick [2015]. *RubyGems*. 2015. https://rubygems.org/ (cited on page 12).

Raggett, Dave [2005]. *HTML Slidy: Slide Shows in XHTML*. 2005. http://w3.org/2005/03/slideshow.html (cited on page 7).

Raggett, Dave [2006a]. *HTML Slidy: Slide Shows in HTML and XHTML*. 2006. http://w3.org/Talks/Tools/Slidy2/ (cited on page 1).

Raggett, Dave [2006b]. "Slidy - a web based alternative to Microsoft PowerPoint". In: *Proceedings of XTech 2006*. 2006. http://w3.org/2006/05/Slidy-XTech/slidy-xtech06-dsr.pdf (cited on pages 1, 7).

Ruiz, Jaime, Yang Li and Edward Lank [2011]. "User-Defined Motion Gestures for Mobile Interaction". In: *CHI 2011*. 2011. http://yangl.org/pdf/motiongestures-chi2011.pdf (cited on page 22).

Sagalaev, Ivan [2015]. *highlight.js*. 2015. https://highlightjs.org/ (cited on pages 13, 14).

Stoll, Martin [2015]. *diascope*. 2015. http://diascope.sourceforge.net/index.php (cited on page 14).

Tangelder, Jorik [2015]. *hammer.js*. 2015. http://hammerjs.github.io/ (cited on pages 15, 22).

Tibbett, Rich [2015]. *Full-Tilt*. 2015. https://github.com/Full-Tilt/Full-Tilt (cited on page 15).

W3C [1992]. *Basic HTTP as defined in 1992*. World Wide Web Consortium. 1992. http://w3.org/Protocols/HTTP/HTTP2.html (cited on page 3).

W3C [1995]. *Cascading Style Sheets, level 1*. World Wide Web Consortium. 1995. http://w3.org/TR/WD-css1-951117.html (cited on page 3).

W3C [2012]. *A Short History of JavaScript*. World Wide Web Consortium. 2012. http://w3.org/community/webed/wiki/A_Short_History_of_JavaScript (cited on page 4).

W3C [2015a]. *CSS Snapshot 2015*. World Wide Web Consortium. 2015. http://w3.org/TR/css3-roadmap/ (cited on page 3).

W3C [2015b]. *CSS3 transform Property*. World Wide Web Consortium. 2015. http://w3schools.com/cssref/css3_pr_transform.asp (cited on page 3).

W3C [2015c]. *HTML5 Video*. World Wide Web Consortium. 2015. http://w3schools.com/html/html5_video.asp (cited on page 14).

W3Schools [2015]. *CSS Responsive Web Design*. 2015. http://w3schools.com/css/css_responsive_intro.asp (cited on page 5).