Rslidy: Lightweight, Accessible, and Responsive HTML5 Slide Decks

Patrick Hipp

Rslidy: Lightweight, Accessible, and Responsive HTML5 Slide Decks

Patrick Hipp B.Sc.

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's Degree Programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Ao.Univ.-Prof. Dr. Keith Andrews
Institute of Interactive Systems and Data Science (ISDS)

Graz, 11 Sep 2021

Rslidy: Kompakte, Barrierefreie, und Responsive HTML5 Präsentationsfolien

Patrick Hipp B.Sc.

Masterarbeit

für den akademischen Grad

Diplom-Ingenieur

Masterstudium: Informatik

an der

Technischen Universität Graz

Begutachter

Ao.Univ.-Prof. Dr. Keith Andrews
Institute of Interactive Systems and Data Science (ISDS)

Graz, 11 Sep 2021

Diese Arbeit ist in englischer Sprache verfasst.

Abstract

Slide decks built for the web can be flexible and feature-rich for presenters, while at the same time being viewable in any web browser on any device worldwide. They are a viable alternative to proprietary presentation software such as PowerPoint and Keynote and offer unique features such as live code execution, responsive tables, and interactive charts.

This thesis first discusses the history and development of packages for web-based slide decks and gives an overview of currently available solutions. The four main approaches described are: text-based, JavaScript-based, hosted, and responsive slide decks.

Rslidy is a new lightweight open-source package supporting accessible and responsive HTML5 slide decks for the web. Rslidy's architecture, build system, and implementation are described and its pros and cons are assessed. Finally, potential future improvements and extensions are discussed.

Kurzfassung

Für das Web erstellte Slide Decks können flexibel und funktionsreich für Präsentatoren sein, während sie dabei in jedem Webbrowser und auf jedem Gerät weltweit aufgerufen werden können. Sie sind eine praktikable Alternative zu proprietärer Präsentationssoftware wie PowerPoint und Keynote und bieten einzigartige Funktionen wie Live-Code-Ausführung, responsive Tabellen und interaktive Diagramme.

Diese Arbeit diskutiert zunächst die Geschichte und Entwicklung von Paketen für webbasierte Slide Decks und gibt einen Überblick über die derzeit verfügbaren Softwarelösungen. Die vier beschriebenen Hauptansätze sind: textbasierte, JavaScript-basierte, gehostete und responsive Slide Decks.

Rslidy ist ein neues, leichtgewichtiges Open-Source Paket, das barrierefreie und responsive HTML5 Slide Decks für das Web unterstützt. Die Architektur von Rslidy, das Build-System und Details der Implementierung werden in dieser Arbeit beschrieben und Rslidy's Vor- und Nachteile werden abgewogen. Schlussendlich werden mögliche zukünftige Verbesserungen und Erweiterungen diskutiert.

Contents

List of Figures

List of Tables

List of Listings

Acknowledgements

In this chapter I want to express my dearest thanks to my colleagues and especially to my supervisor, Keith Andrews, for enduring all my questions, giving constant feedback and spending countless hours proofreading this thesis.

I would like to thank Michael Glatzhofer for introducing me to the development tools, Angelika Droisner and Ana Korotaj for performing a usability test on Rslidy, and lastly Christopher Kopel for his invaluable feedback with screen readers.

Patrick Hipp
Graz, Austria, 11 Sep 2021

Credits

I would like to thank the following individuals and organisations for permission to use their material:

- The thesis was written using Keith Andrews' skeleton thesis [Andrews 2019].
- The screenshot for Figure 3.18 was taken from Showoff [Puppetlabs 2019] (MIT License).
- The picture shown in Figure 7.1 was taken by Droisner and Korotaj [2019] (Creative Commons Attribution 4.0 License).
- The illustration shown in Figure 8.1 was taken from Shalamov et al. [2021] (Creative Commons Attribution 4.0 License).

Chapter 1

Introduction

Presentations can have a huge impact on the world. On the 5th of February 2013, U.S. Secretary of State Colin Powell went before the Security Council of the United Nations to give a presentation on the development of weapons of mass destruction in Iraq [Brock 2017]. His 45 PowerPoint slides were well-organised and convincing and are considered to be “one of the most famous PowerPoint presentations of all time.”

1.1 The Origins of Slide Shows

Back in the 1980s, presentation slides were typically made in one of two ways [Gaskins 2012, pages 15–19]:

- *Overhead transparencies*: Black and white slides were designed on a computer, printed onto paper on a laser printer, then copied onto overhead transparencies (clear plastic sheets) to be shown on an overhead projector.
- *35mm slides*: Colour slides were prepared on a workstation as say GIF images, then transferred (via expensive hardware) to individual 35 mm photographic slides. Once framed, these were sorted into a slide rack, then shown using a slide projector.

At the time, laser printers and copiers could only produce black and white images, and preparing 35mm colour slides was expensive and time-consuming.

PowerPoint, shown in Figures 1.1 and 1.2, changed all that and is now the most well-known presentation application. Microsoft estimates that 1.2 billion copies of PowerPoint now exist [Brock 2017, page 44]. Originally released in April 1987 by Forethought Inc. for the Macintosh [Gaskins 2012, page 16], PowerPoint 1.0 quickly made a name for itself, with sales of $ 1 million in its first month [Brock 2017, page 49]. At launch, PowerPoint was limited to black and white for overhead transparencies [Phi 2018], but support for colour 35mm slides was added in version 2.0 [Gaskins 2012, page 17]. Forethought was bought by Microsoft for $ 14 million just three months after the initial release of PowerPoint. PowerPoint made presentation software accessible to the public and defined the essential features for such applications. A non-profit and open source alternative for PowerPoint exists in LibreOffice Impress [The Document Foundation 2019].

Apple’s Keynote [Apple 2018] presentation software was originally custom-built for Steve Jobs himself to use during his Apple keynote talks [Roemmele 2013]. In January 2003, Keynote was released for MacOS [Apple 2003], as an alternative to PowerPoint. It is currently part of the iWork suite and is now free for both MacOS and iOS [Clover 2017]. Keynote comes with built-in support for Apple remotes and iCloud synchronisation, is cross-compatible with PowerPoint, and shares similar features.

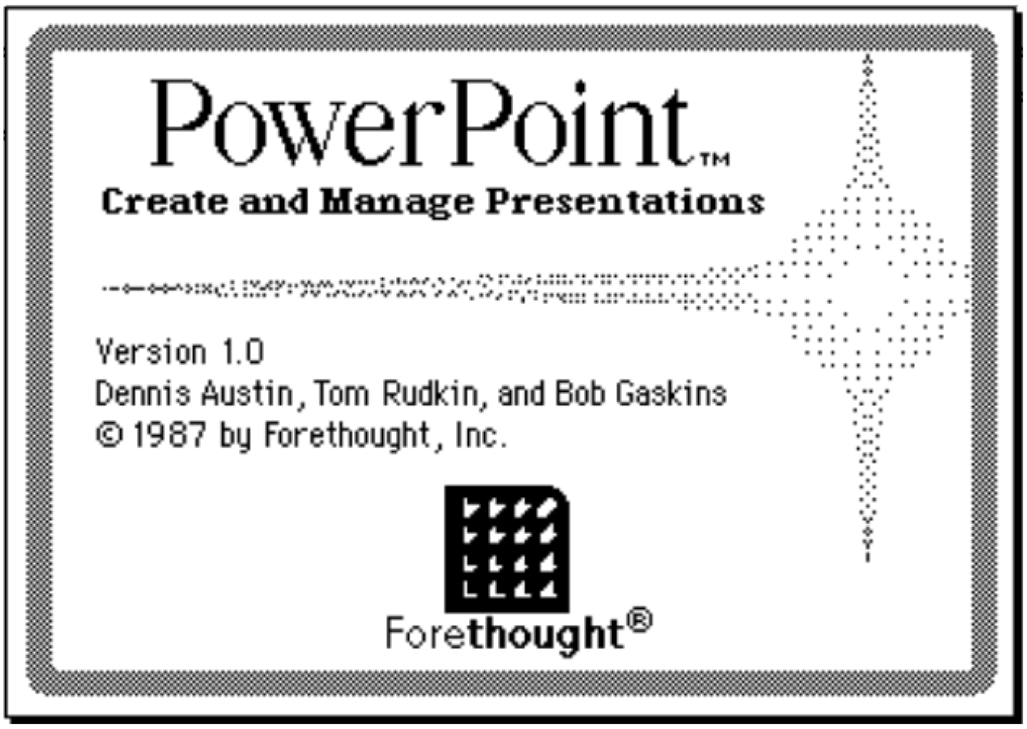
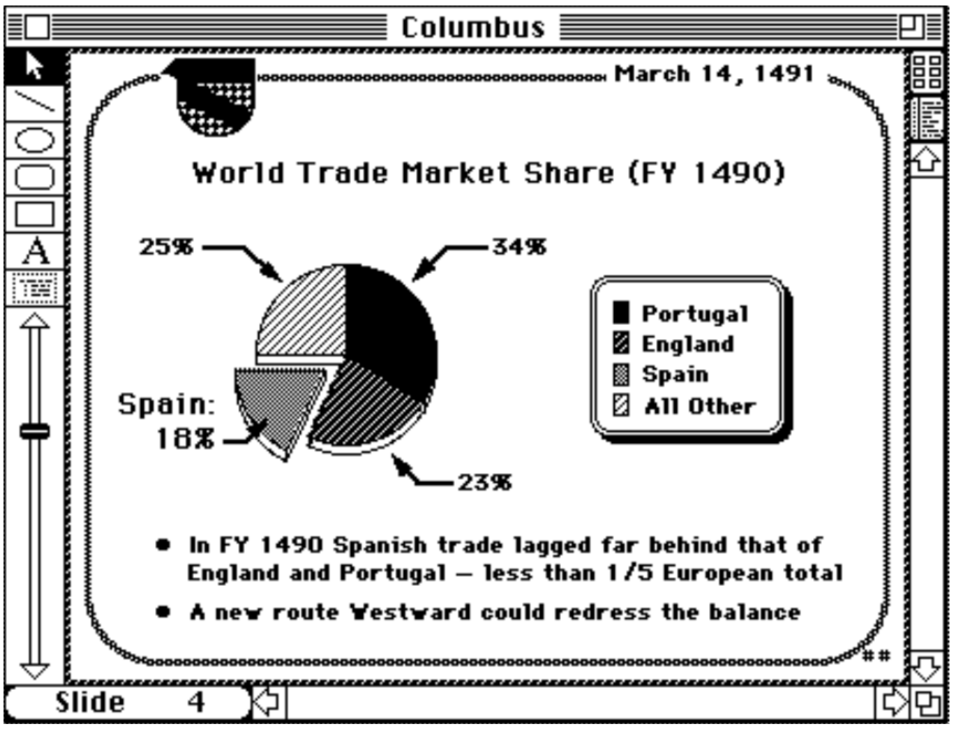

PowerPoint™
Create and Manage Presentations
Version 1.0
Dennis Austin, Tom Rudkin, and Bob Gaskins
© 1987 by Forethought, Inc.
Forethought®

Figure 1.1: The splash screen of PowerPoint 1.0 by Forethought, Inc. [Screenshot made by the author using emulation [Friend 2018].]



- In FY 1490 Spanish trade lagged far behind that of England and Portugal – less than 1/5 European total
- A new route Westward could redress the balance

Figure 1.2: One slide from an example presentation included in PowerPoint 1.0. [Screenshot made by the author using emulation [Friend 2018].]

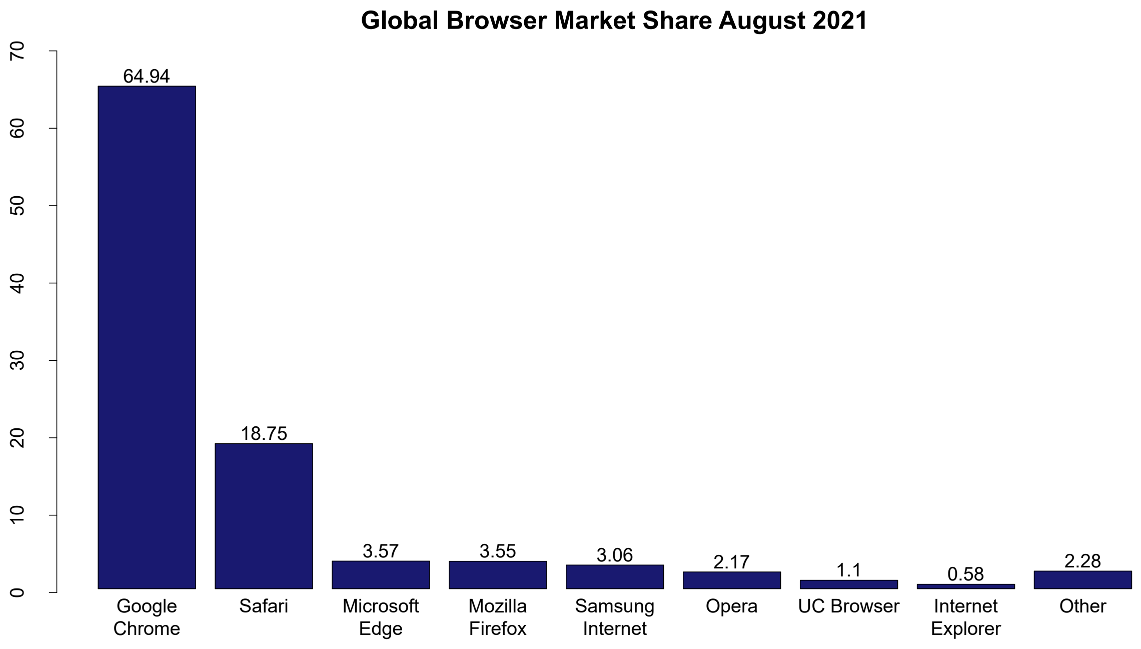


Figure 1.3: Browser market share over all platforms from August 2021. [Diagram made by the author based on data from StatCounter [2021].]

1.2 Slide Guidelines

Slides are the building blocks of every presentation. They can have a variety of different layouts, but most commonly they consist of a heading followed by text, bulleted lists, charts, images, or videos. The purpose of a slide is often to present information in small, easily digestible pieces, which then build up to form a larger picture. Gabrielle [2010] distinguishes between four kinds of slide decks: ballroom style presentations, briefing decks, discussion decks, and reading decks, depending on the size of the audience and the amount of interaction between speaker and members of the audience.

Doumont [2002, 2005, 2009] established four (originally three) rules for professional communication: 0) define your purpose, 1) adapt to your audience, 2) maximise the signal-to-noise ratio, and 3) use effective redundancy. Slides in a presentation should reinforce the message, focusing not on providing every detail, but rather on the implications that follow from them. Effective slides should not compete for attention with the speaker. This can be achieved by reducing the amount of text as far as possible and avoiding purely decorative graphics or backgrounds. For example, clever rephrasing should be used to reduce bulleted list items to no more than two lines of text. If more detailed tables, charts, or graphics would be helpful to promote the message, they can be distributed to audience members in the form of a handout.

Alley [2013] presents some guidelines for creating and giving scientific presentations. He points out that following PowerPoint's default slide layout leads to a topic–subtopic approach, which is often unsuitable for scientific purposes. In a topic–subtopic approach, the headline dictates the main topic and is followed by a bulleted list which further explores it [Alley 2013, page 110]. Research has shown that assertion–evidence slides lead to better comprehension of complex topics [Alley 2013, page 117]. In such slides, most of the words are found in the headline, presenting an assertion in the form of a sentence, while visual aids are used as evidence to support it [Alley 2013, page 121].

Top Ten Platforms		
1	Windows 10	18.23%
2	Android 10	14.76%
3	iOS 14	13.00%
4	Android 11	10.10%
5	Android 9	7.39%
6	Android 8	6.61%
7	Windows 7	6.11%
8	Mac OS X	4.52%
9	Android 5	3.35%
10	iOS 13	3.09%

Top Ten Screen Resolutions		
1	1366x768	6.96%
2	640x360	6.35%
3	1920x1080	6.06%
4	812x375	5.15%
5	780x360	4.89%
6	800x360	4.81%
7	896x414	4.76%
8	667x375	3.99%
9	1024x768	3.61%
10	760x360	3.21%

Table 1.1: The top ten platforms and screen resolutions globally in August 2021 [W3Counter 2021].

1.3 Modern Challenges

Over the last few years, a technological arms race took place between industry giants, causing mobile devices and web browsers to constantly evolve and change shape. This in turn directly affects the way these devices and browsers are used and how they are expected to behave in certain situations. As shown in Figure 1.3, Google Chrome, Safari, Microsoft Edge, and Mozilla Firefox are currently the most used web browsers, with Chrome leading by a wide margin. The dominance of Chrome in the web browser market share is likely due to the fact that Android is now the leading mobile operating system globally, as can be seen in Table 1.1, and it ships with Google Chrome preinstalled as the default web browser.

Developing an application for multiple devices, screen resolutions, and web browsers requires special effort, but is now considered to be best practice. Some of the challenges of modern web development include:

- *Support for all modern browsers*: Generally speaking, this includes Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, and Opera.
- *Support for all modern devices*: Devices can be broadly categorised into desktop, tablet, and smartphone, but hybrids also exists.
- *Adapt to changes in the device environment*: For example, detecting and utilising newly plugged-in peripheral devices, such as a mouse and keyboard, or redrawing the screen after device orientation has changed.
- *Sharing and communicating between different platforms*: Software that allows users to share their work, cooperate on projects, and communicate regardless of which platform they prefer to use.
- *Accessibility*: Intuitive controls, suitable for all kinds of users.
- *Inclusion of third-party software*: Ensure that third-party software is not blocked by parts of the application logic and provide an API where needed.
- *Resilient and responsive design*: Future-proof and device-independent design using flexible grids, fluid images, and media queries.

The above are important considerations for web-based presentation software, but some additional problems have to be dealt with too. These include issues such as zooming for images and charts (especially on smaller screens), having the ability to print a slide deck, slide navigation, speaker notes, and many more.

1.4 Thesis Structure

This thesis first covers the mechanics behind various web-based slide deck approaches and then presents Rslidy, a state-of-the-art responsive web-based slide deck solution. The thesis is structured as follows:

- Chapter 2 explains the basics of front-end web technologies and goes over some of the build tools used in modern web development.
- Chapter 3 presents various web-based slide deck solutions grouped into four categories: text-based, JavaScript-based, hosted, and responsive.
- Chapter 4 covers the history of Rslidy by revisiting early versions, discussing their shortcomings, and exploring reworked concepts.
- Chapter 5 introduces the current architecture of Rslidy and elaborates on some of its core features.
- Chapter 6 puts a focus on some of the more intricate parts of Rslidy and shines light on their implementation. In particular, the generation of slide thumbnails, the handling of user settings, and accessibility are discussed.
- Chapter 7 illustrates the process of user and device testing.
- Chapter 8 deals with possible shortcomings of Rslidy as well as going over future enhancements.
- Chapter 9 offers some concluding remarks.
- Finally, three appendices provide guides for the three kinds of Rslidy users. Appendix A contains a guide for slide viewers. Appendix B contains a guide for slide creators. Appendix C contains a guide for developers wishing to extend or modify Rslidy.

The Rslidy software is available on GitHub [Hipp and Andrews 2021].

Chapter 2

Front-End Web Technologies

Web development can be broadly categorised into two collaborating parts: client-side and server-side. Everything a user directly runs on their devices is referred to as a client-side application, or in more modern terms the front-end [Lindley 2019]. While front-end applications look like any other web page, there are many more tools and techniques in use behind the scenes. This chapter introduces various technologies used during front-end web development and discusses their advantages and disadvantages.

2.1 Web Fundamentals

The Hyper Text Markup Language (HTML) is the foundation of everything that is built on the web [Keith and Andrew 2016]. It is a markup language which defines the building blocks of a web page, such as headings, paragraphs, and images. Cascading Style Sheets (CSS) make it possible to style such elements dynamically and change where and how they are displayed on the screen [Cederholm 2015]. JavaScript is a programming language which can modify the behaviour of a web page [Marquis 2016]. In short, HTML defines the content of a website, CSS defines the layout of that content, and JavaScript adds interactivity. A collection of web pages under a URL (Uniform Resource Locator) is referred to as a website, while a web application consists of a user interface and application logic run by the client in a web browser. Together, these three technologies form the basis of every front-end web application.

2.2 The Document Object Model (DOM)

There are many ways for developers to interact with elements in a web application. Often, front-end developers opt to make use of browser-specific APIs or rely on external libraries, such as the legacy library jQuery [Resig 2006], to modify parts of a web page. The Document Object Model (DOM) is supported by all modern browsers and provides a universal and standardised way to access, modify, and create HTML elements [Lindley 2013]. This is done by traversing the HTML document through a tree-like structure, as shown in Figure 2.1, and providing JavaScript functions for accessing parent and child nodes, or for finding elements with specific attributes, such as IDs. If this tree structure grows too deep or too wide, performance issues can arise. It is therefore advised to keep the DOM tree as tidy and compact as possible.

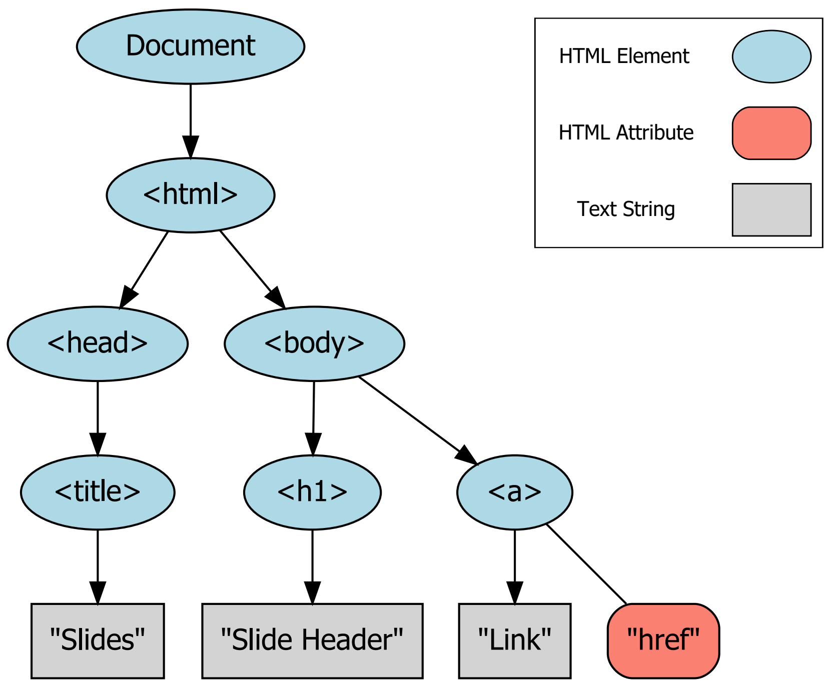


Figure 2.1: The Document Object Model (DOM) can be displayed as a tree structure, called the DOM tree. Standardised cross-browser JavaScript functions are provided to access and modify specific objects and their attributes. [Diagram created by the author in R.]

```
// style.scss
$color: blue;

@mixin solid-border($c) {
  border: 1em solid $c;
}

body {
  background: $color;
  @include solid-border(red);

  p {
    @include solid-border(green);
  }
}
```

```
// style.css
body {
  background: blue;
  border: 1em solid red;
}

body p {
  border: 1em solid green;
}
```

Listing 2.1: An example SCSS file using variables, mixins, and nesting and its resulting CSS file displayed side by side.

2.3 Sass and Sassy CSS (SCSS)

Sass is a pre-processor syntax which is converted to CSS [Cederholm 2013]. It offers several advantages over regular CSS, such as shared variables, importing of files, inheritance, templates (referred to as mixins), and nesting in order to prevent code overhead. There are two syntaxes available for Sass, with Sassy CSS (SCSS) being the new and improved version. It is backwards compatible with CSS, meaning valid CSS code is also valid SCSS, but not the other way around. A simple example showing the conversion from SCSS to CSS can be seen in Listing 2.1.

2.4 TypeScript

TypeScript [Microsoft 2019; Syed 2019] is a programming language built on top of JavaScript. That means that valid JavaScript code is also valid TypeScript code. The main advantage of TypeScript is the inclusion of a type system, where variables and functions can be typed both implicitly and explicitly. TypeScript also includes advanced features planned for future JavaScript versions (ES6 and beyond), such as classes, lambda functions, and features previously only supported by external libraries, making them obsolete. These object-oriented programming principles make it easier for front-end developers to cooperate on larger projects. TypeScript can be transpiled into JavaScript, which then runs in the browser. Various options are handled via the configuration file: `tsconfig.json`.

2.5 Node.js

Node.js [Dahl 2019] is an event-driven cross-platform JavaScript runtime environment written in C/C++ that allows the execution of JavaScript without a web browser. It is most commonly used for server-side and networking tools, but also runs on a variety of devices, such as raspberry pies and household appliances. The event-driven structure of Node.js provides scalability to servers via callbacks, which are used to signal completion of a task, without having to rely on multi-threading. Developers can also write standalone tools and share them via the Node.js package manager.

2.6 Package Managers

A package manager installs and keeps track of all other tools and their dependencies. For web development, npm [NPM 2018] is commonly used. It is a manager for open source packages and comes bundled with Node.js. Developers joining a project can install all required tools with a simple

```
npm install
```

command, as long as they are included in the `package.json` configuration file. Npm also includes functionality for auditing packages and informing developers about known security risks and how to mitigate them.

2.7 Task Runners

A task runner is a customisable tool which aids the build process of a project. One of these tools, gulp [Schoffstall 2019], can be used to automate simple tasks and group them together. All tasks are defined in `gulp.js` and are written as JavaScript functions. Common tasks include transpiling TypeScript into JavaScript, merging and copying files, minifying files, and cleaning up the project directory. They can be invoked when needed individually or as a group from the command line by typing:

```
gulp <task-name>
```

A list of available tasks and their descriptions can be seen with:

```
gulp --tasks
```

2.8 Responsive Web Design

Ethan Marcotte [2014, 2015] describes the three main pillars of responsive web design as flexible grids, fluid images, and media queries. As the name implies, using this approach makes it possible to respond to changes in the browser environment and adjust how a web site or application is displayed and behaves. Flexible and device-independent design is the key to developing long term web applications, which can run on a multitude of display sizes and sniff for features rather than requiring them.

2.9 Web Accessibility

Scott Jehl [2014] argues that responsive web design has to rely on features certain users may not have access to. Additionally, some users may have to use additional software, such as screen readers, to interact with a web page. He coined the term responsible responsive design, which extends responsive web design with usability and accessibility design patterns, ensuring a comfortable web experience for all kinds of users. Usability patterns ensure a consistent and easy experience across many devices, for example by sticking to a list of web-safe gestures, such as tap and swipe. Accessibility patterns add meta-data for external software [Pickering 2016]. ARIA (Accessible Rich Internet Applications) [W3C 2017] is an accessibility specification for HTML5 and SVG. Developers can add ARIA roles and properties to DOM elements to better describe their semantics to assistive technology such as screen readers. ARIA properties can be updated dynamically to reflect user interaction, for example to indicate the current state of a checkbox.

Chapter 3

Web-Based Slide Decks

This chapter gives an insight into the history of slide decks for the web. A plethora of web-based slideshow solutions are grouped into four categories, based on their dominant design approach. The solutions in each category are discussed in chronological order. The four categories of slide decks are: text-based, JavaScript-based, hosted, and responsive. This chapter is based on a previous survey by the author of this thesis [Hipp 2019], which also contains a lengthy appendix with numerous examples.

3.1 Early Systems and Outliers

This chapter discusses the early days of presentation software for the web and describes the most important applications of that era. It also briefly covers some outliers, which use technologies like Flash or video encoding software to create online presentations.

3.1.1 Slidy and Slidy2

In 2005, Dave Raggett, a member of the W3C team, developed one of the first libraries for HTML-based slide presentations called Slidy [Raggett 2005]. Slidy turns an HTML file into a slideshow by including two files: a CSS style sheet to regulate the style of the presentation, and a JavaScript file to add presentation functionality. Additionally, an optional CSS file containing a print style sheet can be included. Slides are written in HTML and the contents of each slide are placed inside `<div class="slide">...</div>` elements.

Features include basic keyboard navigation, font scaling, handout notes, incremental bulleted lists, automatic numbering of slides, the ability to link to a specific slide, and a print style sheet. Disabling JavaScript or CSS displays all the slides in a single page, so in case of an error the main content can still be accessed.

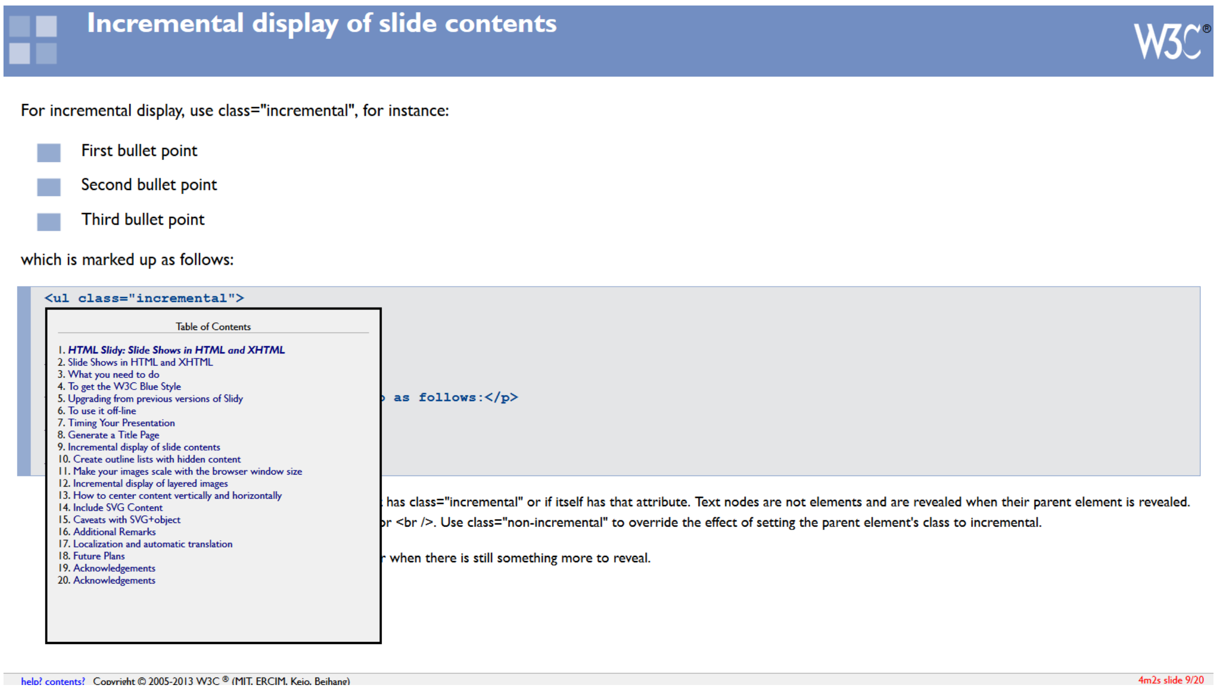
Raggett [2006a] later released an improved version of Slidy called Slidy2, which adds more functionality. The most important addition is an auto-generated table of contents, incorporating the main heading of each slide and supporting quick navigation. An example slide with a table of contents can be seen in Figure 3.1, generated from the code excerpt shown in Listing 3.1. Other new features are the addition of a timer and localisation support. Slidy2 is also fully backwards compatible with the original Slidy.

Slidy and Slidy2 were designed for use in a desktop environment, and do not scale well to narrower screens. Since keyboard shortcuts are typically unavailable on a mobile device, navigation quickly becomes a struggle. The footer is especially problematic on a touch screen, since it only takes up a tiny sliver of the screen, making the footer links extremely difficult to activate.

Further development plans for Slidy included a "what you see is what you get" editor, remote control of slides using HTTP requests, and the ability to import and export from or to other slide formats like PowerPoint and Open Office [Raggett 2006b], but these were never implemented.

```
<div class="slide">
  <h1>Incremental display of slide contents</h1>

  <p>For incremental display, use class="incremental", for instance:</p>

  <ul class="incremental">
    <li>First bullet point</li>
    <li>Second bullet point</li>
    <li>Third bullet point</li>
  </ul>
  ...
</div>
```

Listing 3.1: Slidy2. An example slide in Slidy2. The class `incremental` can be used for incremental display of bullet items.



Incremental display of slide contents

For incremental display, use class="incremental", for instance:

- First bullet point
- Second bullet point
- Third bullet point

which is marked up as follows:

```
<ul class="incremental">
```

```
 as follows:</p>
```

has class="incremental" or if itself has that attribute. Text nodes are not elements and are revealed when their parent element is revealed.

r <br />. Use class="non-incremental" to override the effect of setting the parent element's class to incremental.

r when there is still something more to reveal.

Table of Contents

help? contents? Copyright © 2005-2013 W3C ® (MIT, ERCIM, Keio, Beihang) 4m2s slide 9/20

Figure 3.1: Slidy2. Slidy2 features an auto-generated table of contents, which is displayed by pressing `T` on the keyboard or by clicking the `contents?` link in the footer. [Screenshot made by the author from Slidy2 [Raggett 2006a].]

```
<div class="slide">
  <h1>Navigation</h1>

  <ul>
    <li>Controls are...
    <ul>
      <li>Next slide: Space bar, return, right arrow...</li>
      <li>Previous slide: Up arrow, left arrow, page up...</li>
      <li>Toggle the slide styles: Click on the toggle button...</li>
    </ul>
    ...<a href="features.html">plus more</a>!
    </li>
    <li>To invoke the navigation menu: mouse into the lower right corner of the
        slide (below the navigation arrows)</li>
  </ul>
</div>
```

Listing 3.2: S5. An example slide in S5 and S6. S6 supports `<section>` in addition to `<div class="slide">`.

3.1.2 S5

S5 stands for Simple Standards-based Slide Show System and was developed by Eric Meyer [2005a] in 2005. Its features, code, and usage requirements are similar to Slidy and some additional slide themes are provided. A source code example from S5 is shown in Listing 3.2. Navigation is handled through keyboard shortcuts, mouse clicks, and a footer containing buttons and a drop-down menu, which only becomes visible when moving the mouse pointer near the bottom right half of the screen, as shown in Figure 3.2.

Navigation on mobile devices is extremely tedious. Fortunately, touch events in the browser automatically call equivalent mouse events, which makes it possible to advance slides in S5. However, in order to navigate to a previous slide, the footer still needs to be accessed, which is difficult without mouse hover. It is still possible to use the footer controls by first tapping the screen such that the emulated mouse pointer is close enough to cause a mouse hover event. However, these initial taps may also unintentionally advance the presentation.

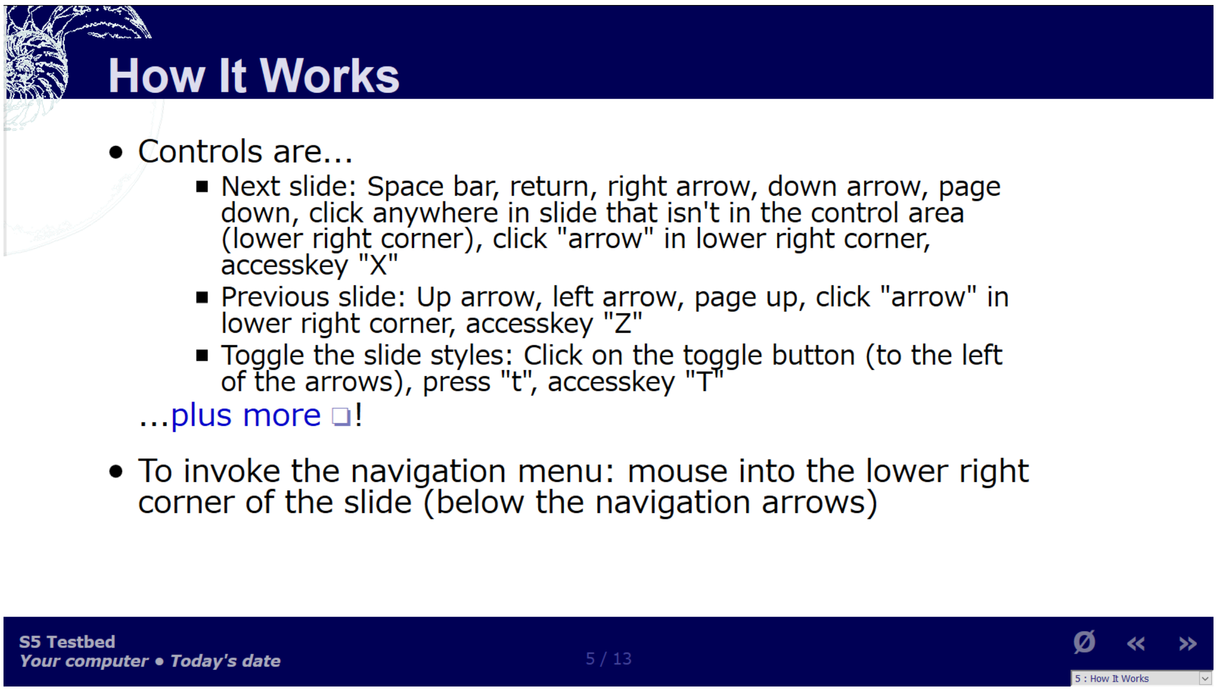
3.1.3 S6

Development of S6 was started by Bauer [2013] in the form of a rewrite of S5, focusing on easy extensibility through plugins. It makes use of the popular JavaScript library JQuery and supports both classic elements (`<div>`) and more modern HTML elements (`<section>`). S6 is one of the formats generated by the Ruby command line tool S9, a text-based slide deck solution discussed in Section 3.2.1. The code is in many cases identical to that of S5.

3.1.4 Diascope

In 2006, Martin Stoll created Diascope [Stoll 2016b], a slideshow generator for Linux which uses video encoding to render presentations. Diascope comes in the form of a binary executable and focuses primarily on smooth transitions, panning, zooming, and audio effects. The output of diascope is an MPEG2, MPEG4 or FLV video file ready for playing or embedding on a web page.

A short code example of a diascope presentation can be seen in Listing 3.3. It is a very powerful tool, but requires knowledge of video editing software. Even small changes to the code require a re-rendering of the output video file, making it unsuitable for inexperienced users.



How It Works

- Controls are…
  - Next slide: Space bar, return, right arrow, down arrow, page down, click anywhere in slide that isn't in the control area (lower right corner), click "arrow" in lower right corner, accesskey "X"
  - Previous slide: Up arrow, left arrow, page up, click "arrow" in lower right corner, accesskey "Z"
  - Toggle the slide styles: Click on the toggle button (to the left of the arrows), press "t", accesskey "T"

  …plus more ❑!
- To invoke the navigation menu: mouse into the lower right corner of the slide (below the navigation arrows)

S5 Testbed
Your computer • Today's date

Figure 3.2: S5. S5 slides can be navigated by keyboard and mouse controls. The on-screen buttons and drop-down menu only become visible when hovering over the bottom right half of the footer. [Screenshot made by the author from S5 [Meyer 2005b].]

3.1.5 Flash Presentations

Adobe Flash [Chun 2014] was a powerful tool for creating standalone interactive multimedia applications which could be embedded into a web site. Following its release in 1996, it became omnipresent on the web, with many of the more impressive looking websites making use of it to some degree. With support for mouse and keyboard interactions and built-in tools to easily create animations from keyframes, Adobe Flash could be used to create slideshows, which were then displayed on a web page with the Flash plug-in. Such slideshows could also be exported as Flash video for embedding in a web page.

In 2010 Steve Jobs [2010] described the shortcomings of Flash in a blog post entitled "Thoughts on Flash":

> “Flash was created during the PC era – for PCs and mice. Flash is a successful business for Adobe, and we can understand why they want to push it beyond PCs. But the mobile era is about low power devices, touch interfaces and open web standards – all areas where Flash falls short. [· · ·] New open standards created in the mobile era, such as HTML5, will win on mobile devices (and PCs too).”

Shortly afterwards, Apple announced that its products would no longer support Flash. Most developers today have switched away from Flash, and it was officially retired in 2020 [Adobe 2020; Barrett 2017].

```
format pal quality=1 interlaced mpeg2sound=mp2 mpeg2
base template_hq_i /tmp
set resize=resize
set font=PenguinAttack size=78 fill=black strokewidth=0

set dur=sec

# audio mysong.wav fade=0,2

# Title slide

create 2 white title=0,1,1,0 "Diascope\nSlideshow\nGenerator"
trns 0.3 luma dir=ro

set font=ArialB fill=gray stroke=white strokewidth=1

# Main show

create 0.8 blue title=0.8 "diascope.sf.net"
trns 0.4 luma dir=td sharp=1
create 0.8 red title=0.8 "diascope.sf.net"
trns 0.4 luma dir=td sharp=1
create 0.8 yellow title=0.8 "diascope.sf.net"
trns 0.4 luma dir=td sharp=1
create 0.8 blue title=0.8 "diascope.sf.net"

trns 0.3 luma dir=ri
audio silence

# End

create 1 black
```

Listing 3.3: Diascope. An excerpt from a simple diascope source file [Stoll 2016a]. Rendering parameters are followed by some basic slides with transitions.

3.2 Text-Based Slide Decks

Unlike other approaches, text-based slide decks do not require slide creators to possess any knowledge of HTML, CSS (styling), or JavaScript (scripting). Slides are created by editing plain text [Wikipedia 2019] files, such as ASCII [Pattis 2004] or Unicode [2018] or Markdown files [Gruber 2004], which are then turned into a web-based slideshow by an application. Markdown is a syntax for structuring plain text to be converted to HTML or other formats. The main advantage of this approach is the fast and easy way to write and edit slides, but the trade-off is less flexibility in terms of styling and interaction.

3.2.1 Slide Show (S9)

S9 is a command line tool written in Ruby by Gerald Bauer [2011]. It can turn a Markdown file into a variety of different slide decks, such as Slidy (Section 3.1.1), S5 (Section 3.1.2), and S6 (Section 3.1.3), among others. An example slide is shown in Listing 3.4. The user has to download and install template packs for each of these output formats, and can choose from a variety of theme packs. To install the S6 template pack, the command:

```
$ slideshow install s6blank
```

is run from the command line. A list of installed templates can be viewed with the command:

```
$ slideshow list
```

```
# Slide Heading

- First bullet point
- Second bullet point
- Third bullet point
```

Listing 3.4: S9. An example slide in S9 using Markdown. # is used for headings and - or * for list items.

```
!SLIDE code

# Syntax Highlighting

## is very simple

@@@ ruby
    # Variables and expressions.
    a = 10
    b = 3 * a + 2
    printf("%d %d\n", a, b);

    # Type is dynamic.
    b = "A string"
    c = 'Another String'
    print b + " and " + c + "\n"
@@@
```

Listing 3.5: Slidedown. In Slidedown, syntax highlighting is performed on code placed inside @@@ blocks. After the block definition, the programming language is specified.

To build a slide show from a file `slides.text`, the following command has to be executed:

```
$ slideshow build slides.text -t s6blank

=> Preparing slideshow 'slides.html'...
=> Done.
```

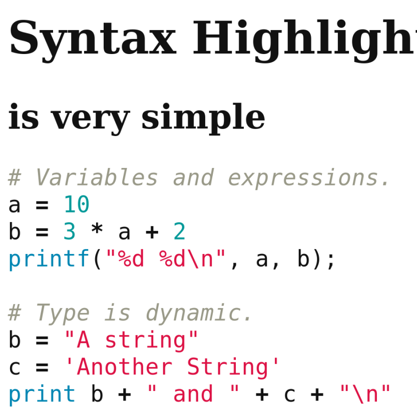
More experienced slide creators can include a variety of plugins and helpers in their presentations, or write their own scripts using Ruby. These can be used, for example, to provide syntax highlighting. It is also possible to create or edit templates and themes.

3.2.2 Slidedown

Slidedown [Nakajima and Croak 2012] is a very lightweight and minimalistic Ruby tool for creating basic slideshows from Markdown. A slideshow is generated by running the command:

```
$ slidedown slides.md > slides.html
```

The created presentations only support barebones keyboard controls: the left and right arrow keys are used for navigation and pressing escape exits the presentation. Custom style sheets are automatically included, if they are placed in the same directory as the presentation Markdown file, and support for syntax highlighting is provided. Some example code and the resulting slide are shown in Listing 3.5 and Figure 3.3 respectively.



Syntax Highlighting

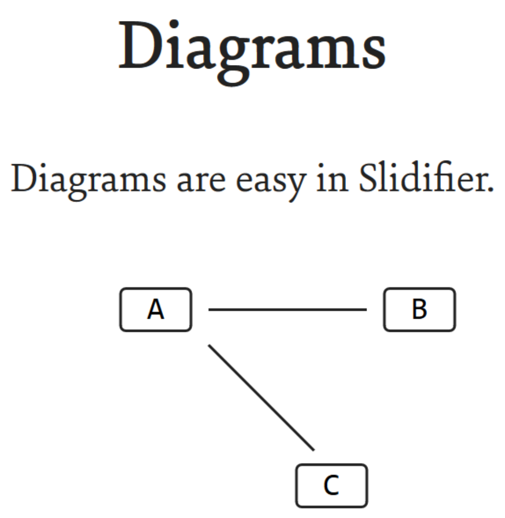
is very simple

```
# Variables and expressions.
a = 10
b = 3 * a + 2
printf("%d %d\n", a, b);

# Type is dynamic.
b = "A string"
c = 'Another String'
print b + " and " + c + "\n"
```

Figure 3.3: Slidedown. The code from Listing 3.5 turned into a slide with Slidedown. [Screenshot made by the author from Slidedown [Nakajima and Croak 2012].]

```
---

# Diagrams

Diagrams are easy in Slidifier.

\\diagram
  [ A ]*1       1*[ B ]
         *2



                *2
               [ C ]
\\diagram

---
```

Listing 3.6: Slidifier. Source code for a slide with a diagram in Slidifier. Connecting lines are assigned numbers and are connected using the * character; labels are enclosed in square brackets.



Diagrams

Diagrams are easy in Slidifier.

Figure 3.4: Slidifier. The output slide generated by Slidifier from the source code in Listing 3.6. [Screenshot made by the author from Slidifier [Ludvigsen 2016b].]

3.2.3 Slidifier

Rather than relying solely on Markdown, Slidifier [Ludvigsen 2016a] defines its own simple plain text syntax. A line containing `---` is used to separate slides and inline HTML can be used extend the plain text, for example to include an image. A unique feature of Slidifier is its inbuilt support for creating simple diagrams. The example in Listing 3.6 and Figure 3.4 illustrates this. A primitive form of keyword highlighting is supported as well.

3.2.4 Remark

Remark [Bang 2018b] uses a mixture of HTML and Markdown and provides many advanced features, such as a synchronised display, speaker notes, and a presenter mode, as shown in Listing 3.7 and Figure 3.5. Three dashes separate slides and three question marks introduce speaker notes. In presenter mode, a timer and speaker notes are displayed on the right and the current and next slide are displayed on the left. Remark supports touch gestures such as swipe to navigate and slide scaling for various screen sizes. There are some external tools for Remark, most noteworthy a PDF generator and an online "what you see is what you get" editor.

A Remark input file starts in HTML and allows for styles to be defined in the HTML header. The body then contains a `<textarea id="source">` tag, in which the Markdown text for slides is placed. Afterwards, Remark is called from within a script element as follows:

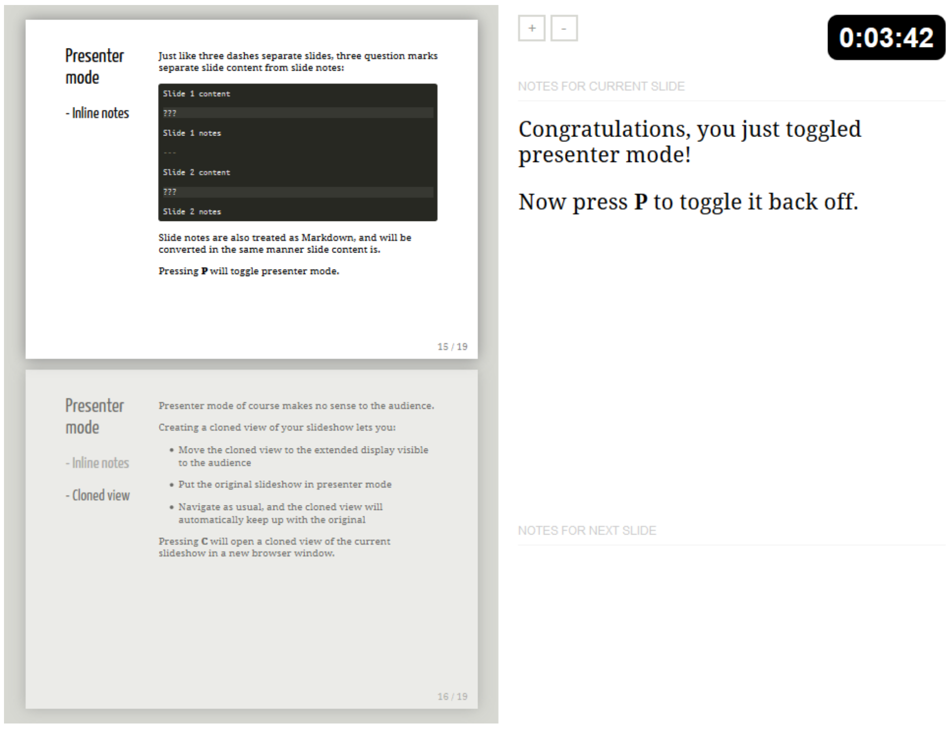
```
var slideshow = remark.create();
```

Remark further extends the Markdown syntax with slide properties. These can be placed at the beginning of a slide using a `key-value` syntax. The properties are:

- `name`: Assigns an identifier to a slide, in order to link to it directly using `file.html#identifier`.
- `class`: Adds a comma-separated list of CSS classes to the slide.
- `background-image`: Sets a background image for the specific slide.
- `count`: Slides where the count property is set to false will not influence the total number of slides. All incremental slides set this property to false by default.
- `template`: This property can be used to inherit the properties of another slide.
- `layout`: If set to true, the slide properties will be carried over to all subsequent slides, until layout is set to false again.
- `exclude`: Fully excludes a file from rendering.

```
<textarea id="source">
---
.left-column[
  ## Presenter mode
  ### - Inline notes
]
.right-column[
Just like three dashes separate slides,
three question marks separate slide content from slide notes:

Slide notes are also treated as Markdown, and will be converted in the
same manner slide content is.

Pressing __P__ will toggle presenter mode.
]
???
Congratulations, you just toggled presenter mode!

Now press __P__ to toggle it back off.
---
</textarea>
```

Listing 3.7: Remark. In Remark, classes are applied by `.class[]`, such as `.right-column[]`. Speaker notes are introduced by three question marks.



Presenter mode

- Inline notes

Just like three dashes separate slides, three question marks separate slide content from slide notes:

```
Slide 1 content
???
Slide 1 notes
---
Slide 2 content
???
Slide 2 notes
```

Slide notes are also treated as Markdown, and will be converted in the same manner slide content is.

Pressing **P** will toggle presenter mode.

15 / 19

Presenter mode

- Inline notes

- Cloned view

Presenter mode of course makes no sense to the audience.

Creating a cloned view of your slideshow lets you:

- Move the cloned view to the extended display visible to the audience
- Put the original slideshow in presenter mode
- Navigate as usual, and the cloned view will automatically keep up with the original

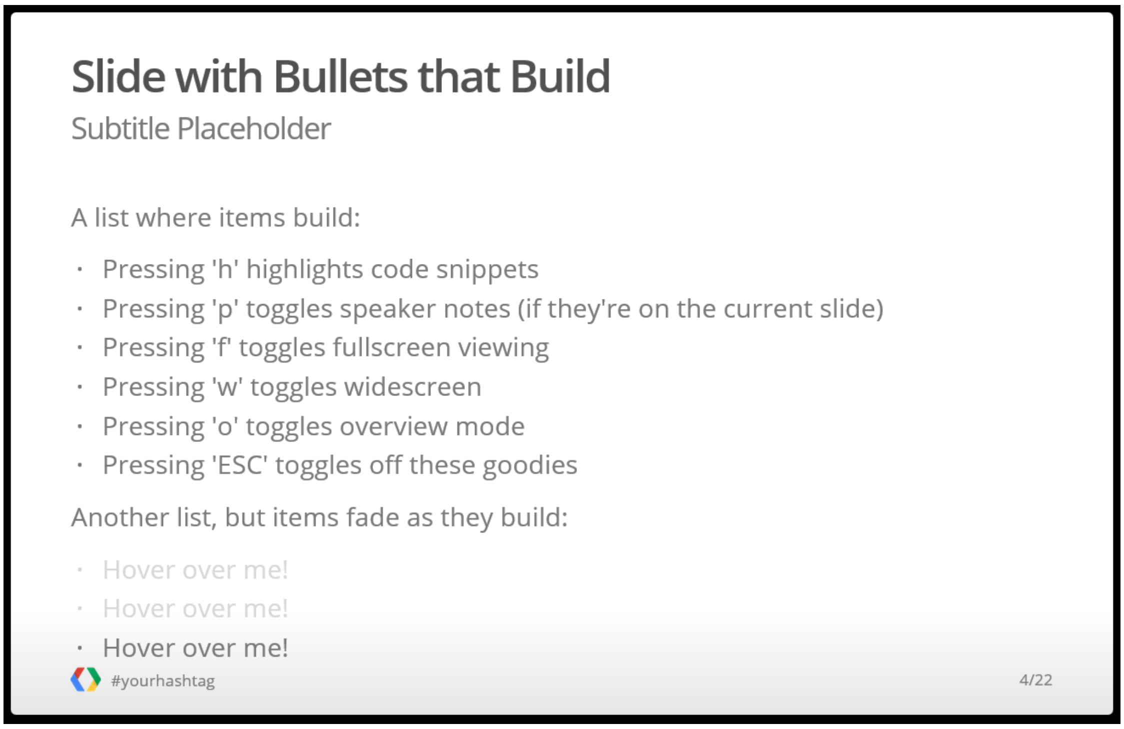
Pressing **C** will open a cloned view of the current slideshow in a new browser window.

16 / 19

Congratulations, you just toggled presenter mode!

Now press **P** to toggle it back off.

Figure 3.5: Remark. Pressing P in Remark toggles presenter mode. A separate synchronised window (called a cloned view) can be opened by pressing C, which can then be displayed on a projector. [Screenshot made by the author from Remark.js [Bang 2018a].]



Slide with Bullets that Build

Subtitle Placeholder

A list where items build:

- Pressing 'h' highlights code snippets
- Pressing 'p' toggles speaker notes (if they're on the current slide)
- Pressing 'f' toggles fullscreen viewing
- Pressing 'w' toggles widescreen
- Pressing 'o' toggles overview mode
- Pressing 'ESC' toggles off these goodies

Another list, but items fade as they build:

- Hover over me!
- Hover over me!
- Hover over me!

#yourhashtag 4/22

Figure 3.6: Google I/O Slides. As well as standard navigation, Google I/O Slides supports a number of keyboard shortcuts. [Screenshot made by the author from Google I/O Slides [Rota 2015].]

3.3 JavaScript-Based Slide Decks

Slide decks built with JavaScript offer a broad spectrum of functionality, rely heavily on scripting, and often make use of third-party packages. Many of these slide packages are listed in the NPM [2018] registry of open source packages for Node.js, and can be easily installed using a package manager. These packages are modern and feature-rich, but there are also some minimalistic slide decks which try to reduce the number of external dependencies. A possible downside to JavaScript-based slide decks is that the scripts are often computationally expensive.

3.3.1 Google I/O Slides

Google I/O Slides [Mahém et al. 2013] is an HTML template developed for Google I/O in 2011. Originally, it went under the name html5slides. It supports many of the standard features of slide decks such as keyboard navigation, syntax highlighting, and touch controls. A list of special keyboard shortcuts can be seen in Figure 3.6. A file called `slide_config.js` is used to configure the slide template, including title, fonts, themes, authors, and various options. The template makes use of external tools such as compass, a Ruby tool for converting SASS into CSS, and Python to convert Markdown into HTML. Slide content is placed inside `<slide>` elements.

Development was abandoned in 2015, likely in favour of Google Slides (see Section 3.4.1), a hosted slide deck approach. At the time of writing, all the repositories for Google I/O Slides have been either partially or fully removed, but a fork of the project is still available on GitHub [Rota 2015].

3.3.2 Fathom.js

Fathom.js [Dalgleish 2015] uses the jQuery Mobile library to optimise slide decks for mobile devices. It provides a list of event handlers for user actions like tap, swipe left, swipe right, scroll start, scroll stop, and orientation change. A special feature of Fathom.js is the ability to synchronise a slideshow to an

```
$('#presentation').fathom({
  timeline: [ 0, 5, 20, 30, 40, 50, '1:00', '1:15', 90, 120, 155 ],
  video: {
    source: 'vimeo',
    id: '16917950',
    parent: '#vimeo',
    autoplay: true
  }
});
```

Listing 3.8: Fathom.js. Fathom.js can synchronise a slide deck to an external video using a timeline of slide transitions. The timeline parameter supports both numbers and strings.

external video. This is achieved by providing a timeline of slide transitions and a video object, as can be seen in Listing 3.8. The project page is no longer maintained and recommends using Bespoke.js (see Section 3.3.5) instead.

3.3.3 Deck.js

Developed by Caleb Troughton [2016a], Deck.js is a modular and flexible JavaScript framework for generating customised slide decks. It is based on jQuery and Modernizr. The framework includes a file named `boilerplate.html`, which serves as a starting point for slide creators. The main functionality is provided by the `deck.core` module, which defines various states for the slide deck, which can be customised using CSS. Additional functionality is provided by extensions, which use core events and methods in various ways. An example of a navigation bar extension can be seen in Listing 3.9 and Figure 3.7. Using both the core module and extensions, slide creators can customise a presentation to include what they need and leave out what they do not. Deck.js has in-depth documentation [Troughton 2016b] and a wiki page [Troughton 2016d], where people can share their themes, extensions, and tools.

3.3.4 DZSlides

DZSlides [Rouget 2017a] is a single, ready-to-use HTML file template for creating slideshows. Both the CSS and JavaScript are already included in the file, making it easier to handle for slide creators. By using so-called *shells*, the presentation can be embedded or used in a presenter mode. An example of embedded mode can be seen in Figure 3.8. These shells are separate HTML files which link together with the original presentation. In order for these extensions to function, DZSlides implements a messaging system. The main template file can either receive or send a set of predefined messages. These include simple commands such as navigation, but also more advanced commands which can be used to register events or obtain the mouse cursor position.

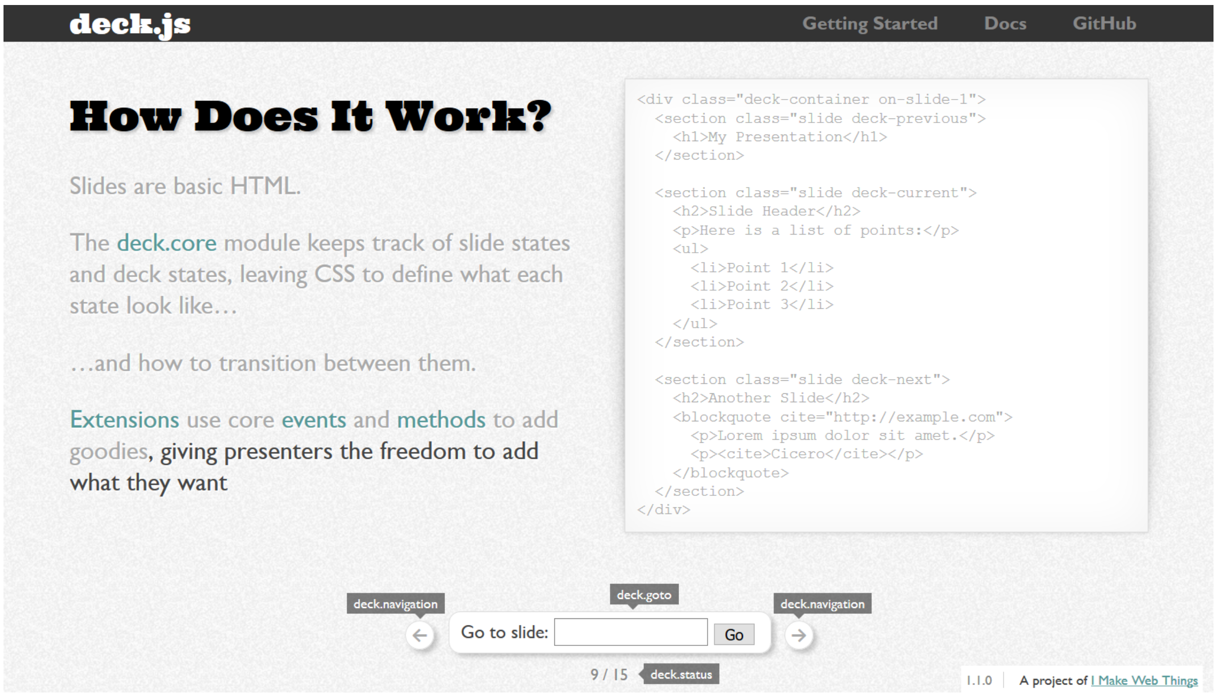
The slides automatically adjust their scaling along with the browser window, but they are internally fixed at a virtual resolution of `800x600`, making the presentation framework not fully responsive. Basic keyboard navigation is supported along with mobile-friendly touch controls.

3.3.5 Bespoke.js

Bespoke.js [Dalgleish 2018a] is a minimalist modular presentation library. The main package only provides very basic functionality and a simple control API, and everything else is managed via plugins. Due to the fragmented nature of the package, it is recommended to use a build tool such as gulp [Schoffstall 2019] to pull all the required plugins together and generate the presentation. Currently, 21 themes and 75 plugins are available. For example, `bespoke-touch` adds touch controls and `bespoke-math` adds support for LaTeX formulae. A custom theme can be seen in Figure 3.9

```
<section class="slide">
  <h1>Slide</h1>

  <ul>
    <li>First bullet point</li>
    <li>Second bullet point</li>
    <li>Third bullet point</li>
  </ul>
</section>

<!-- Core Module (Required) -->
<script src="core/deck.core.js"></script>

<!-- Extensions -->
<script src="extensions/menu/deck.menu.js"></script>
<script src="extensions/goto/deck.goto.js"></script>
<script src="extensions/status/deck.status.js"></script>
<script src="extensions/navigation/deck.navigation.js"></script>
<script src="extensions/scale/deck.scale.js"></script>

<!-- Finally, initialise the deck. -->
<script>
  $(function() { $.deck('.slide'); });
</script>
```

Listing 3.9: Deck.js. An example slide setup in Deck.js. After including the core module, additional functionality can be added through extensions.



How Does It Work?

Slides are basic HTML.

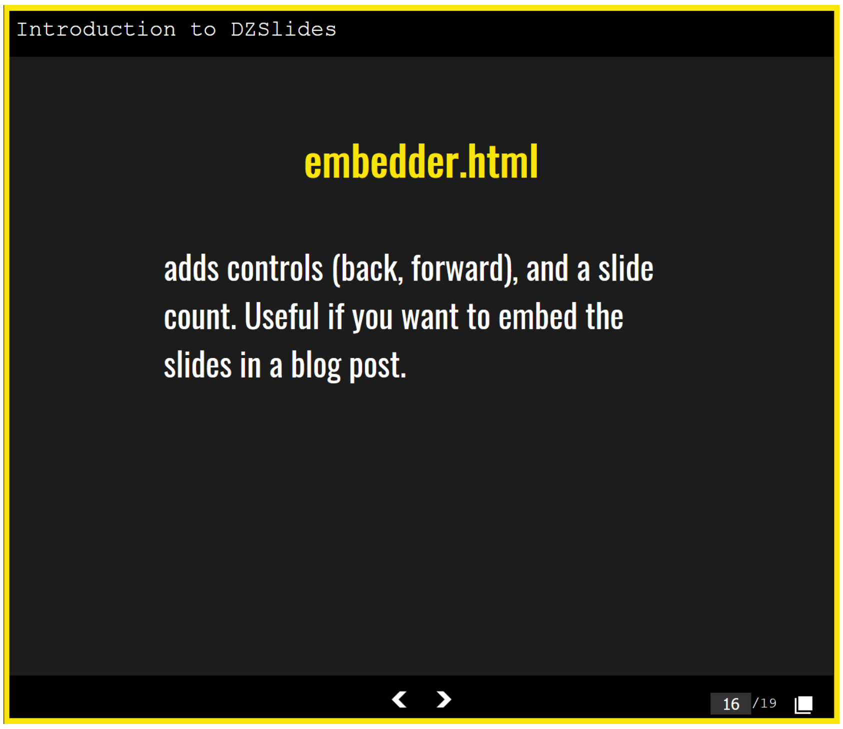
The deck.core module keeps track of slide states and deck states, leaving CSS to define what each state look like…

…and how to transition between them.

Extensions use core events and methods to add goodies, giving presenters the freedom to add what they want

```
<div class="deck-container on-slide-1">
  <section class="slide deck-previous">
    <h1>My Presentation</h1>
  </section>

  <section class="slide deck-current">
    <h2>Slide Header</h2>
    <p>Here is a list of points:</p>
    <ul>
      <li>Point 1</li>
      <li>Point 2</li>
      <li>Point 3</li>
    </ul>
  </section>

  <section class="slide deck-next">
    <h2>Another Slide</h2>
    <blockquote cite="http://example.com">
      <p>Lorem ipsum dolor sit amet.</p>
      <p><cite>Cicero</cite></p>
    </blockquote>
  </section>
</div>
```

Figure 3.7: Deck.js. One of the extensions for Deck.js provides a navigation bar, which interfaces to the core module's navigation, goto, and status functionality. [Screenshot made by the author from Deck.js [Troughton 2016c].]



Introduction to DZSlides

embedder.html

adds controls (back, forward), and a slide count. Useful if you want to embed the slides in a blog post.

Figure 3.8: DZSlides. In DZSlides, the `embedder.html` shell embeds the presentation into another web page and adds navigation controls in the form of a footer. The rightmost button on the footer opens a pop-up window to the embedded slideshow. [Screenshot made by the author from DZSlides [Rouget 2017b].]

To include a theme or a plugin, a simple `require('...')` command is needed to load the package, followed by a call to its constructor in the plugin array of Bespoke's main function. Plugins interact with the API of Bespoke, which contains a number of predefined variables and helper functions for navigation and events. Listing 3.10 shows some example code for generating a presentation with the `cube` theme and support for keyboard and touch controls.

3.3.6 Inspire.js

Inspire.js [Verou 2019a] aims to be a lean and minimalistic slideshow tool, focusing on reusability and extendibility using automatically loaded JavaScript extensions. It was formerly known as the CSS-based SlideShow System (CSSS). Unique features of Inspire.js include the ability to reuse a previously defined slide by its id, support for annotated videos, and support for live code examples. An example of video annotation can be seen in Listing 3.11 and Figure 3.10.

3.3.7 Reveal.js

With a broad range of functionality and elegance, Reveal.js [Hattab 2019a] is a highly advanced presentation framework. It supports vertically nested slides, Markdown content, keyboard and touch controls, PDF export, speaker notes, presenter mode, math functions, lazy loading, and more. An example presentation in presenter mode can be seen in Figure 3.11. Many of these features are provided by third party scripts, for example Markdown interpretation and syntax highlighting. Slideshows using Reveal.js can also be created and published through an online platform [Slides 2019], which provides a graphical slide editor.

Reveal.js can be configured both during initialisation and at runtime and comes with a long list of options for each of its features. Configuration is entirely optional and hides some interesting functionality. For cultures which write and read from right to left, the direction of the presentation can be configured. Even

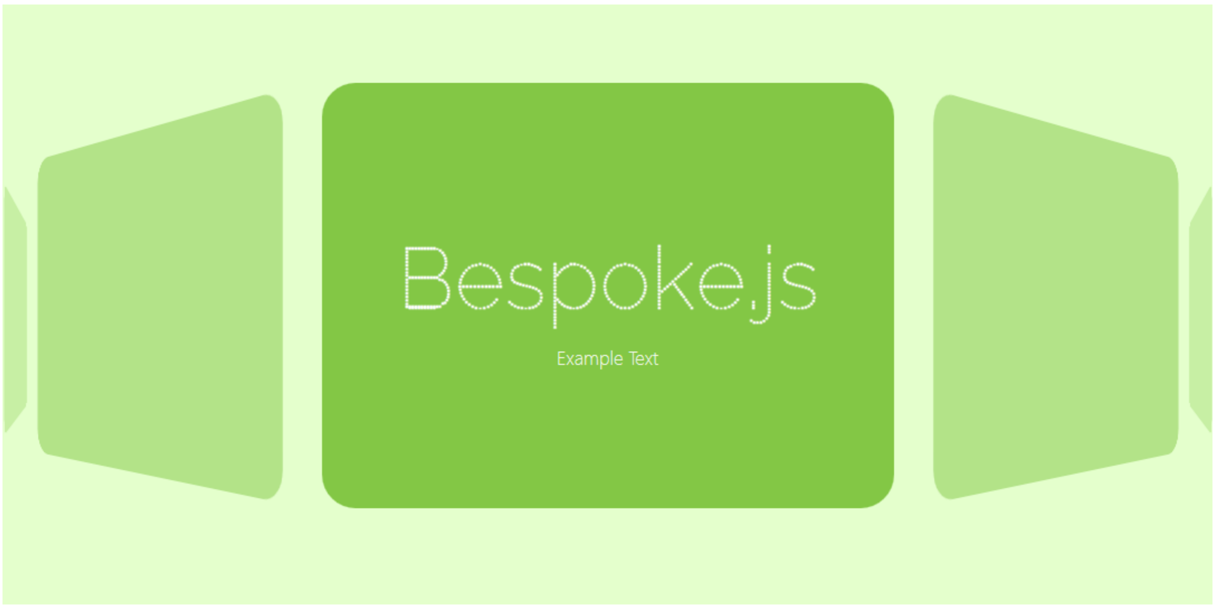


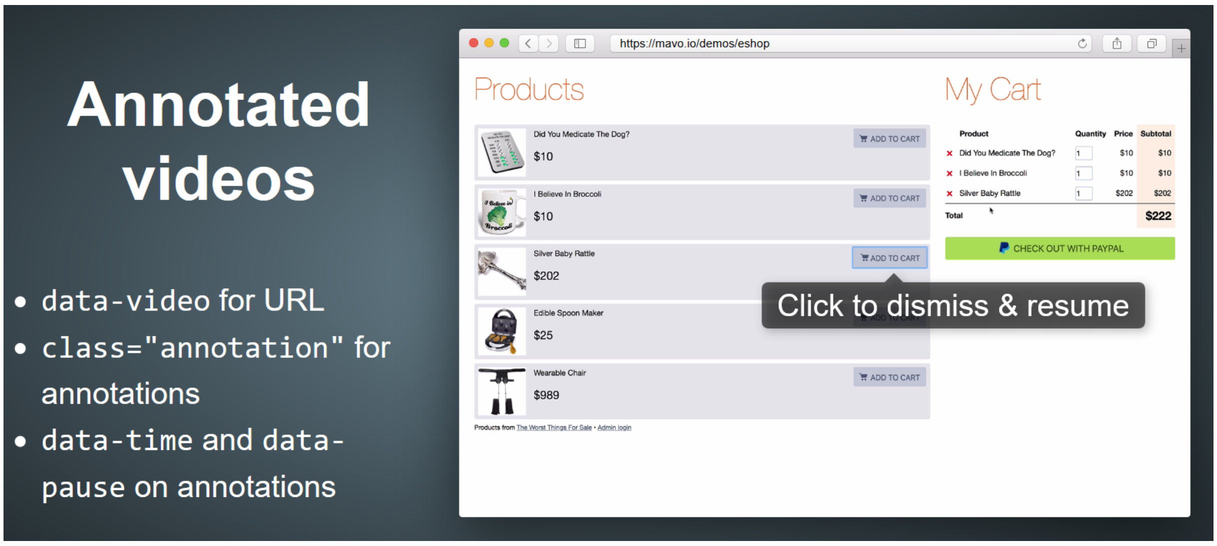
Figure 3.9: Bespoke.js. Bespoke.js currently features 21 custom themes. The theme here is `carousel`.
[Screenshot made by the author from Bespoke.js [Dalgleish 2018b].]

```
// HTML Body

<article id="presentation">
  <section>Slide 1</section>
  <section>Slide 2</section>
  <section>Slide 3</section>
</article>

// JavaScript

var bespoke = require('bespoke'),
  cube = require('bespoke-theme-cube'),
  keys = require('bespoke-keys'),
  touch = require('bespoke-touch');

var deck = bespoke.from('#presentation', [
  cube(),
  keys(),
  touch()
]);
```

Listing 3.10: Bespoke.js. Bespoke.js is a modular slide deck library, which uses a plugin ecosystem to provide additional functionality. In this example, the `cube` theme and two plugins are loaded.

```
<article class="slide" data-video="https://mavo.io/demos/eshop/video.mp4" style="--url: 'https://mavo.io/demos/eshop'" id="media-plugin">
  <div>
    <h1>Annotated videos</h1>
    <ul>
      <li><code>data-video</code> for URL</li>
      <li><code>class="annotation"</code> for annotations</li>
      <li><code>data-time</code> and <code>data-pause</code> on annotations</li>
    </ul>
  </div>
  <span class="annotation top-pointer" style="top: 44%; left: 59.5%; --pointer-left: 4em;" data-time="3000 to 3500" data-pause>Click to dismiss & resume</span>
  <span class="annotation top-pointer" style="top: 26%; left: 67.1%" data-time="5200 to 5700" data-pause="2000">Auto-resume after 2sec</span>
</article>
```

Listing 3.11: Inspire.js. Inspire.js uses the `annotation` class with the attributes `data-time` or `data-pause` to display video annotations.



Annotated videos

- `data-video` for URL
- `class="annotation"` for annotations
- `data-time` and `data-pause` on annotations

https://mavo.io/demos/eshop

Products

Did You Medicate The Dog?	$10
I Believe In Broccoli	$10
Silver Baby Rattle	$202
Edible Spoon Maker	$25
Wearable Chair	$989

Products from The Worst Things For Sale • Admin login

My Cart

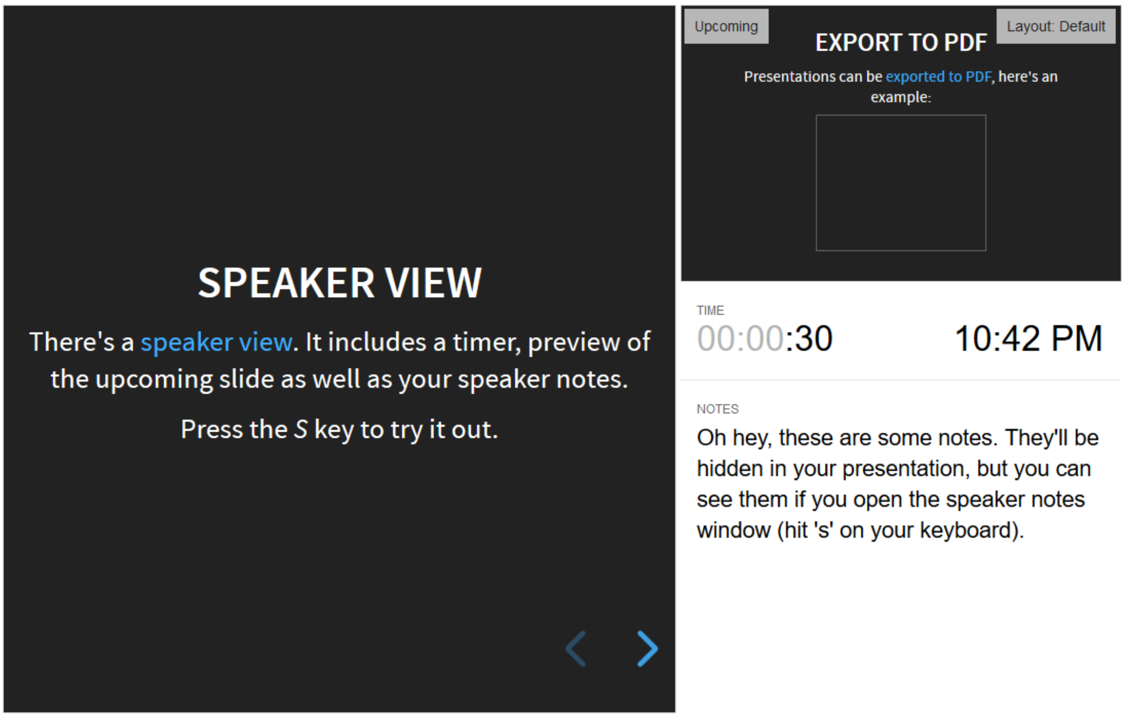
Product	Quantity	Price	Subtotal
Did You Medicate The Dog?	1	$10	$10
I Believe In Broccoli	1	$10	$10
Silver Baby Rattle	1	$202	$202
Total			$222

Click to dismiss & resume

Figure 3.10: Inspire.js. In Inspire.js, annotations temporarily pause a video to show text, which can either automatically disappear after a set time or be dismissed by mouse click. [Screenshot made by the author from Inspire.js [Verou 2019b].]

keyboard shortcuts can be remapped manually, as shown in Listing 3.12. A full list of these options are provided in the manual. Some of the more important options include:

- `controls`: If set to true, displays the arrow controls.
- `history`: If set to true, pushes slide changes into the browser history.
- `loop`: If set to true, loops the presentation.
- `autoSlide`: Automatically advances a slide after a set number of milliseconds.
- `rtl`: If set to true, changes the presentation direction to right-to-left.
- `previewLinks`: Adds an iframe preview window for links.
- `transition`: Changes the transition style between `none`, `fade`, `slide`, `convex`, `concave`, or `zoom`.



EXPORT TO PDF

Presentations can be exported to PDF, here's an example:

TIME

00:00:30 10:42 PM

NOTES

Oh hey, these are some notes. They'll be hidden in your presentation, but you can see them if you open the speaker notes window (hit 's' on your keyboard).

SPEAKER VIEW

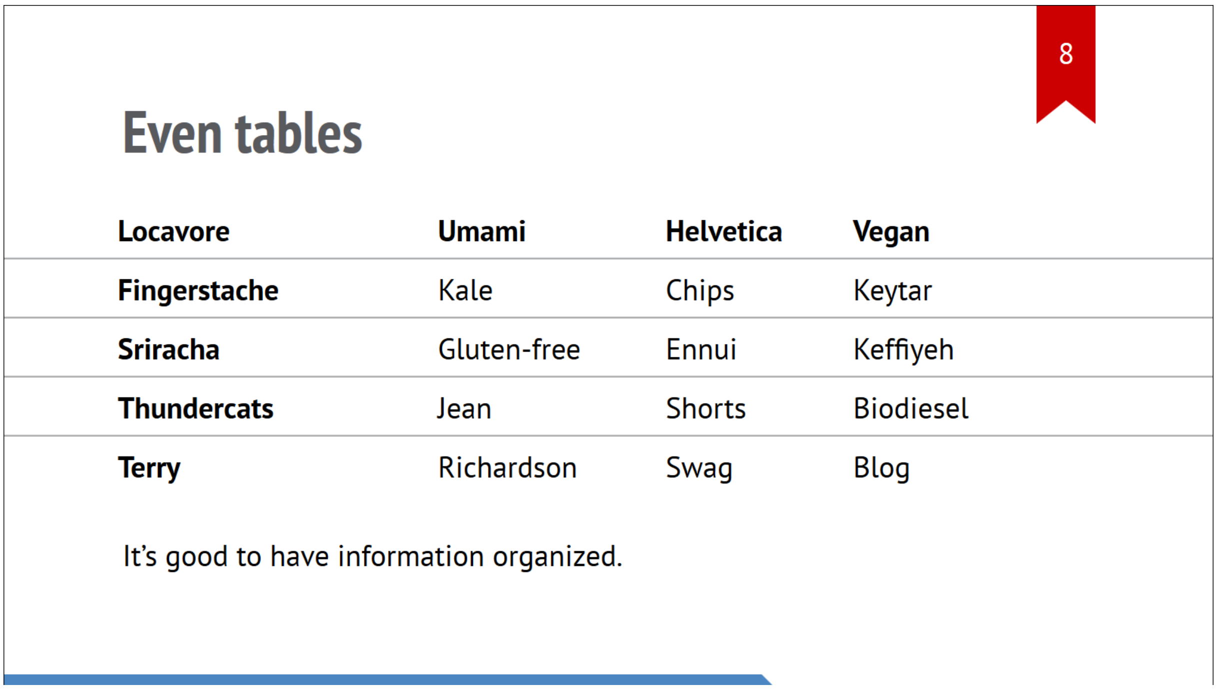
There's a speaker view. It includes a timer, preview of the upcoming slide as well as your speaker notes.

Press the *S* key to try it out.

Figure 3.11: Reveal.js. Reveal.js calls its presenter mode “Speaker View”. It shows the current and next slide, speaker notes, and a timer. [Screenshot made by the author from Reveal.js [Hattab 2019b].]

```
<section>
  <h1>Slide</h1>

  <ul>
    <li>First bullet point</li>
    <li>Second bullet point</li>
    <li>Third bullet point</li>
  </ul>
</section>

<script>
  Reveal.initialize({
    keyboard: true,
    rtl: true,
    transition: 'slide',
    transitionSpeed: 'fast'
  });

  Reveal.configure({
    keyboard: {
      13: 'next',
      27: function() {},
      32: null
    }
  });
</script>
```

Listing 3.12: Reveal.js. The example code above initialises Reveal.js in right-to-left mode and remaps some keyboard shortcuts. To unmap its default setting, a shortcut is set to `null`.



8

Even tables

Locavore	Umami	Helvetica	Vegan
Fingerstache	Kale	Chips	Keytar
Sriracha	Gluten-free	Ennui	Keffiyeh
Thundercats	Jean	Shorts	Biodiesel
Terry	Richardson	Swag	Blog

It's good to have information organized.

Figure 3.12: Shower. An example slide with stylised tables in Shower. The progress bar is visible at the bottom and the current slide number can be seen in the upper right corner. [Screenshot made by the author from Shower [Makeev 2018].]

3.3.8 Shower

Resembling the look of PowerPoint, Shower [Makeev 2019] tries to recreate classic printable slideshows. An example slide can be seen in Figure 3.12. Unlike most other slide decks, Shower also provides a means for slide creators to deploy and host their presentations using Netlify [2019], in addition to being able to self-host. A rebuild is automatically triggered when files are updated and no extra tools need to be installed for hosting the presentation.

The tool comes bundled with a "template archive" containing the two main themes, Ribbon and Material, and the core files. Shower can also be downloaded using npm. This includes a gulp configuration file with predefined tasks: `prepare`, which builds the slide deck, and `publish`, which handles the upload process to Netlify. The source code for an example slide in Shower is shown in Listing 3.13.

3.3.9 Impress.js

Impress.js [Szopka and Ingo 2018a] is a presentation framework unlike traditional slide decks. It is heavily inspired by Prezi (see Section 3.4.2) and focuses on stunning transitions and animations, which are made possible by CSS3 transformations. Impress.js supports translation, scaling, and rotation, in both 2D and 3D. The entire presentation is placed on an infinite canvas, which is then traversed by a camera. An example of its bird's-eye view can be seen in Figure 3.13.

In order to achieve this, every slide has to provide additional parameters for position, rotation, and scaling as HTML data attributes, which can be tedious to maintain. An example can be seen in Listing 3.14. Impress.js, as the name implies, is designed to impress with stunning visuals, and is therefore computationally expensive. It is intended to be presented on a powerful computer with a modern browser and does not guarantee mobile device support. Impress.js does not rely on third party packages.

```
<section class="slide">
  <h1>Key features</h1>
  <ul>
    <li>Built on HTML, CSS and JavaScript</li>
    <li>Works in all modern browsers</li>
  </ul>
</section>
```

Listing 3.13: Shower. An example slide in Shower.

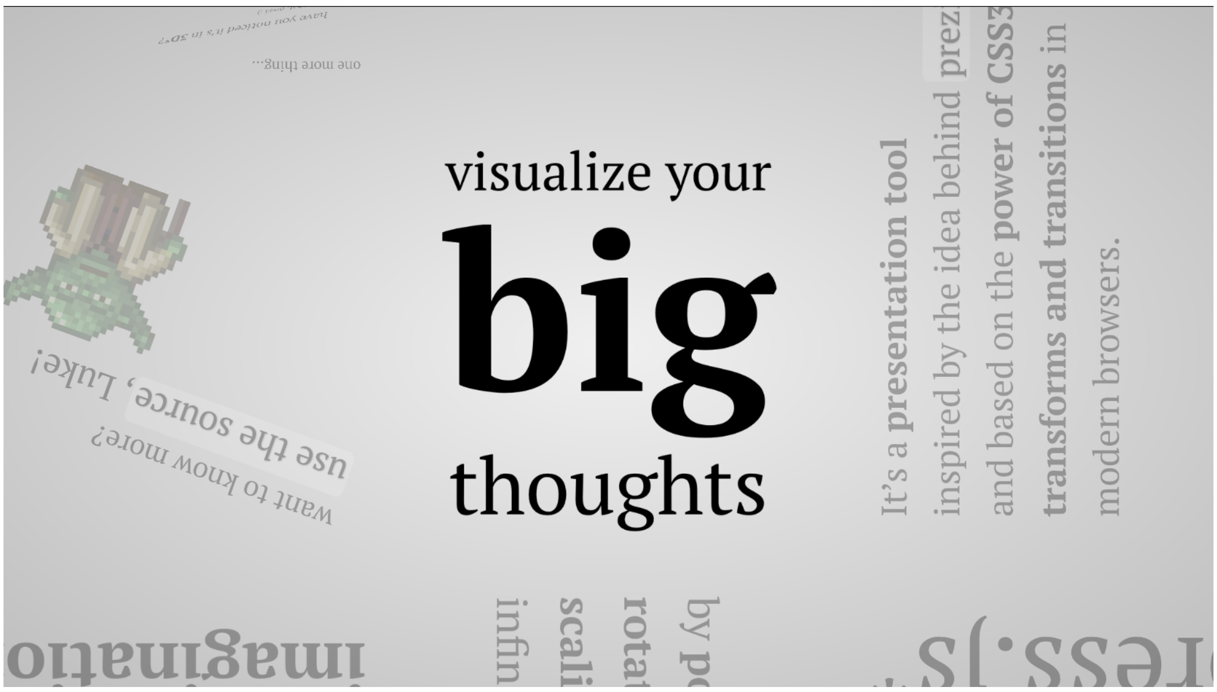
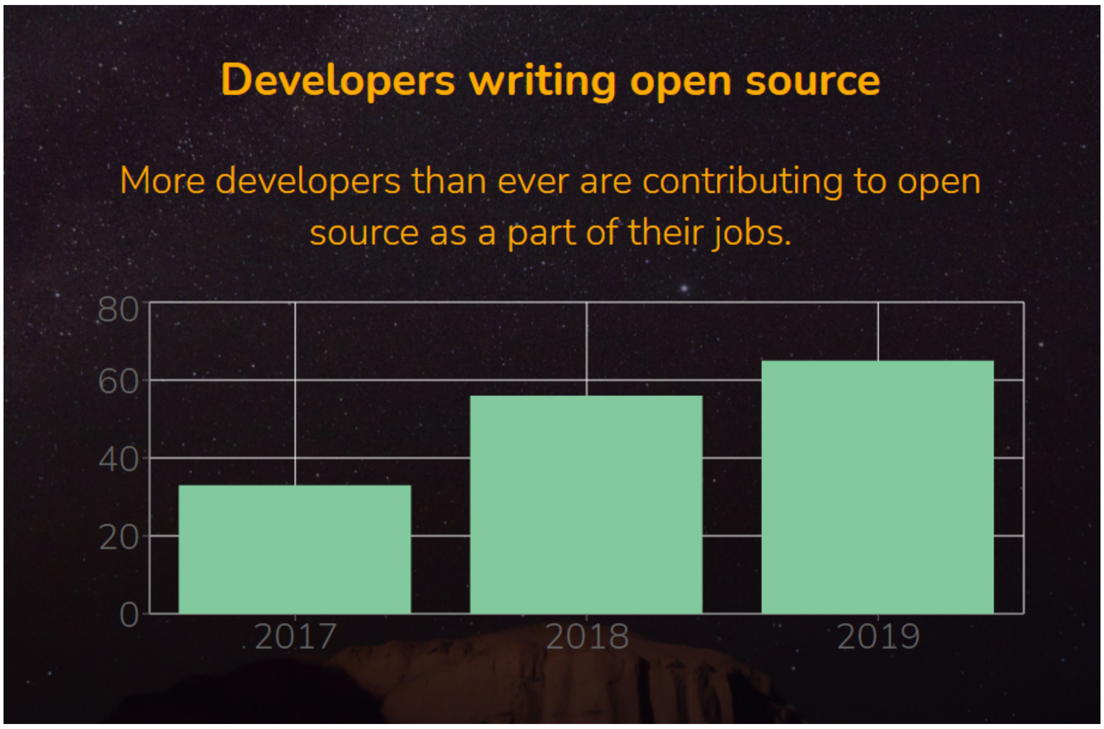


Figure 3.13: Impress.js. Impress.js places all the slides on an infinite canvas and then traverses it with a camera. [Screenshot made by the author from Impress.js [Szopka and Ingo 2018b].]

```
<div class="step" data-x="6200" data-y="4300" data-z="-100" data-rotate-x="-40" data
    -rotate-y="10" data-scale="2">
  <ul>
    <li>First bullet point</li>
    <li>Second bullet point</li>
    <li>Third bullet point</li>
  </ul>
</div>
```

Listing 3.14: Impress.js. In Impress.js, animations are handled with HTML data attributes. The position of the centre of a slide is defined by `data-x`, `data-y`, and `data-z`. Rotation can be adjusted with `data-rotate-x`, `data-rotate-y`, and `data-rotate-z`. The scale is defined by `data-scale` and defaults to 1.


Developers writing open source

More developers than ever are contributing to open source as a part of their jobs.

Figure 3.14: MDX Deck. React components within MDX Deck allow for easy creation of charts and other dynamic content. See the corresponding code in Listing 3.15. [Screenshot made by the author from MDX Deck [Dzielak 2019].]

3.3.10 MDX Deck

MDX Deck [Jackson 2020] is a hybrid between text-based and JavaScript-based slide decks. Unlike regular Markdown, MDX allows the direct embedding of React JSX components [React 2020]. With this combined approach, slide content can be written in an easy to digest human-readable form, while not having to sacrifice any scriptability or extendibility.

MDX Deck ships with a variety of built-in components, like support for headers and footers, speaker notes, and steps, as well as a variety of themes and layouts. It also supports external components for syntax highlighting and live code execution. Themes are handled via react imports and apply to the entire slide deck, but each individual slide can also have a custom CSS layout. An example slide can be seen in Figure 3.14 with the corresponding Markdown in Listing 3.15 .

Keyboard shortcuts are simplistic and handle general navigation, as well as an overview mode, grid mode, and presenter mode. In presenter mode, speaker notes as well as a timer are shown and a miniature depiction of the next slide is displayed.

At present, there seems to be no universal print support for MDX slide decks. Only the currently visible slides are printable. A workaround for printing all slides is possible by opening the browser's print dialog from the grid overview mode.

```
# Developers writing open source

More developers than ever are contributing to
open source as a part of their jobs.

‘‘‘jsx
import ColumnChart from "./components/ColumnChart"

<ColumnChart xAxis={[’2017’, ’2018’, ’2019’]}
             yAxis={[’33%’, ’56%’, ’65%’]} />
‘‘‘
```

Listing 3.15: MDX Deck. In addition to using Markdown for the generation of slide content, JSX can be used to import and display React components. These components can be reused within the slide deck. The resulting slide can be seen in Figure 3.14.

3.4 Hosted Slide Decks

Neatly packed into a web application, hosted slide decks take a drastically different approach from those previously covered in this survey. They not only host the finished presentations for viewing, but also the tools for editing them. While offering some unique advantages, this approach also has its downsides. Some of the tools listed in this section require a monthly subscription fee. Often, the slide decks can only be viewed with a stable internet connection, offline access and the ability to self-host is not provided.

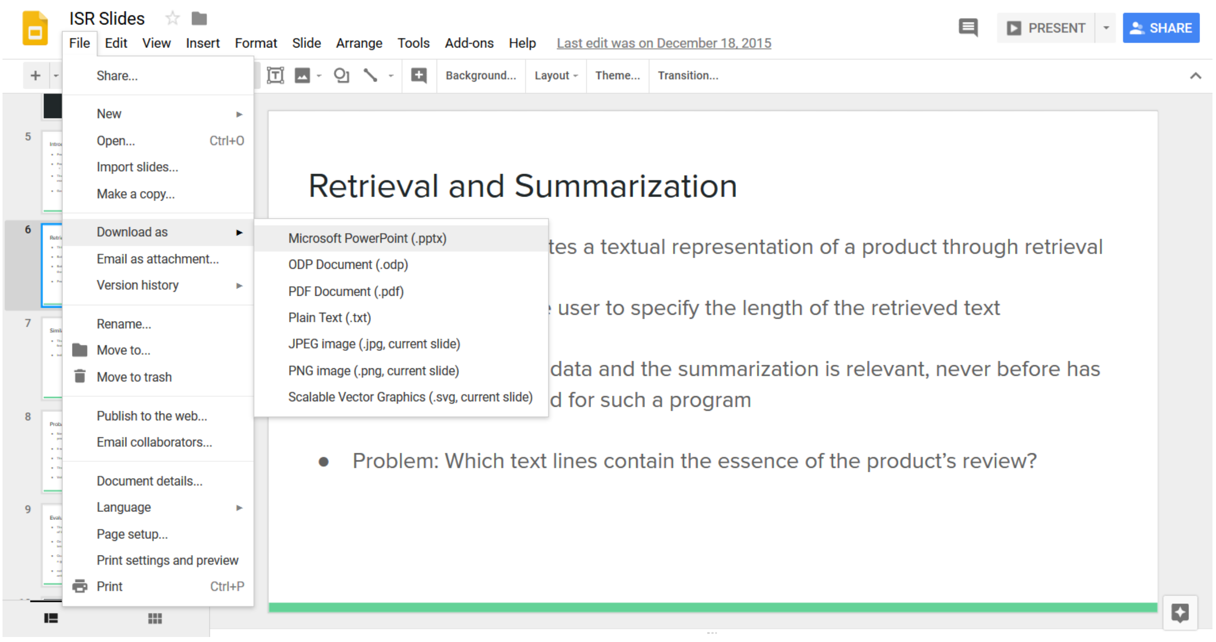
3.4.1 Google Slides

Google hosts a free online office suite as part of Google Drive. This includes Google Slides [Google 2019], an online presentation tool compatible with PowerPoint, which means users can import from and export to the PowerPoint file format freely. An example of the export menu is shown in Figure 3.15. The features of Google Slides mostly mirror those of PowerPoint.

Google Slides provides a set of templates and layouts and focuses on simple and fast creation of slideshows with basic functionality. The most prominent advantage of working in Google Slides is that multiple slide creators can edit a presentation simultaneously. Every change is committed to Google Drive’s cloud servers immediately and slides can even be edited as they are presented. An offline version does not exist.

3.4.2 Prezi

Prezi [Halácsy et al. 2019a] takes a camera with a bird’s-eye view over an infinitely expandable canvas and moves it between fixed points. It was developed as a standalone app for iOS in 2011. An Android and a web browser version using Flash were released later on, and a HTML5-based hosted version was released in 2017 called “Prezi Next”. Prezi provides public and reusable template slide decks on their website, an example of which is shown in Figure 3.16. With the acquisition of Infogram, a data visualisation platform, Prezi expanded its PowerPoint-like online editor with several tools for charts and figures. Beyond that, it provides analysis tools for slide creators, showing how much time viewers spend on each slide, how many people viewed the presentation, and how many of them shared it. Unfortunately, the analysis tools are only available for Premium users and offline access is only provided to Plus users and higher. A list of the available account plans can be seen in Figure 3.17.



Retrieval and Summarization

tes a textual representation of a product through retrieval

user to specify the length of the retrieved text

data and the summarization is relevant, never before has d for such a program

- Problem: Which text lines contain the essence of the product's review?

Figure 3.15: Google Slides. Google Slides supports exporting to various file formats, including PowerPoint and PDF. [Screenshot made by the author from Google Slides.]

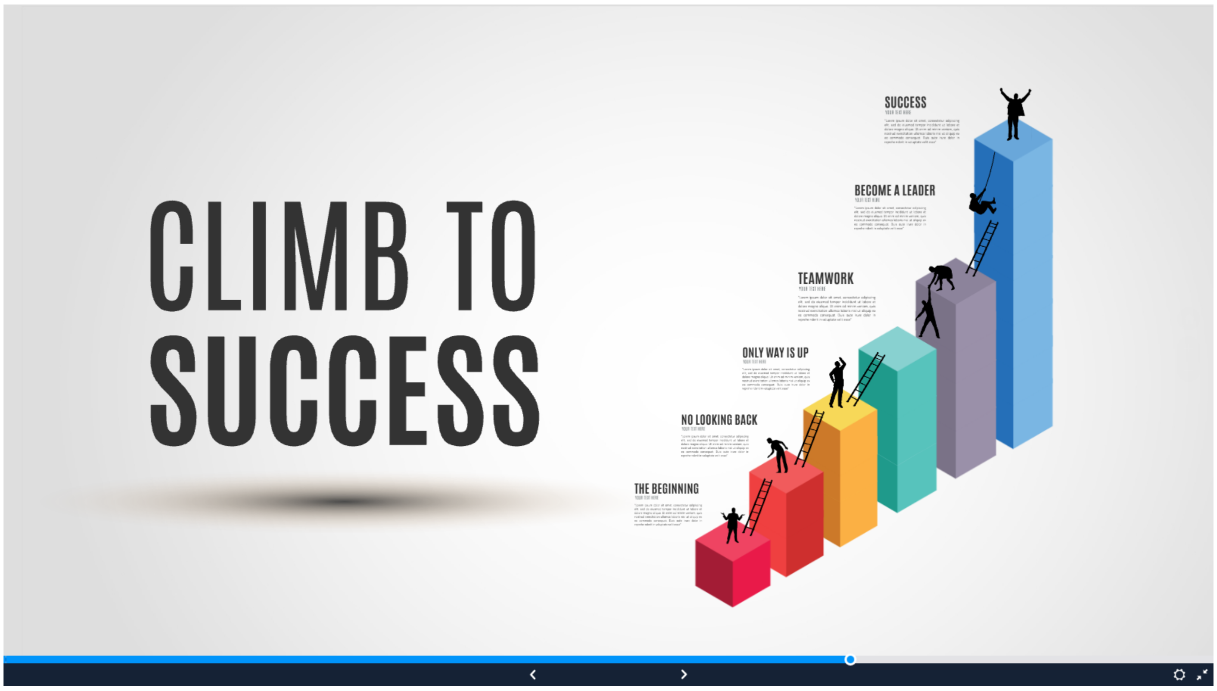
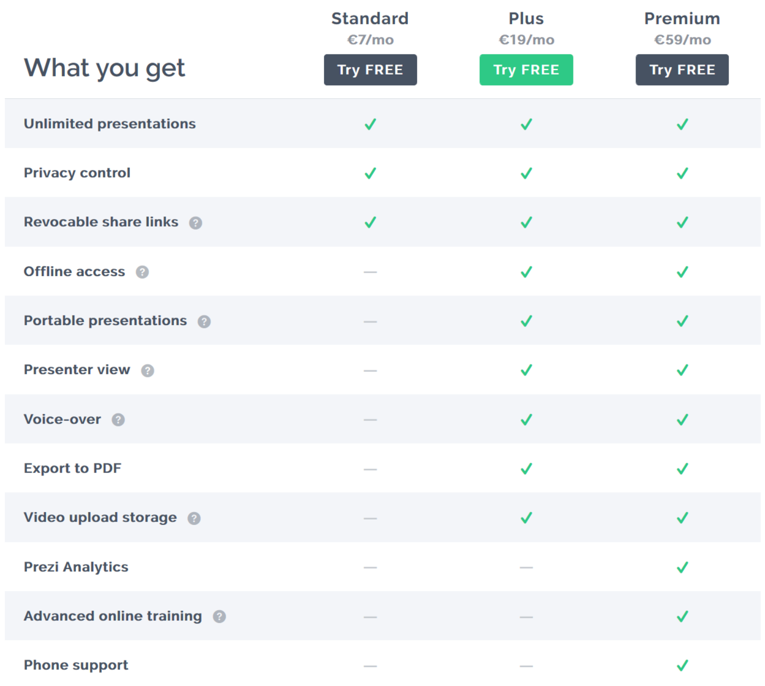


Figure 3.16: Prezi. A slide showing the entire canvas of Prezi's "Climb to Success" template. [Screenshot made by the author from Prezi.com [Prezibase 2015].]



What you get	Standard €7/mo Try FREE	Plus €19/mo Try FREE	Premium €59/mo Try FREE
Unlimited presentations	✓	✓	✓
Privacy control	✓	✓	✓
Revocable share links	✓	✓	✓
Offline access	—	✓	✓
Portable presentations	—	✓	✓
Presenter view	—	✓	✓
Voice-over	—	✓	✓
Export to PDF	—	✓	✓
Video upload storage	—	✓	✓
Prezi Analytics	—	—	✓
Advanced online training	—	—	✓
Phone support	—	—	✓

Figure 3.17: Prezi. A list of Prezi features included with each subscription plan. Features like offline access, portable presentations, presenter view, and exporting to PDF are not included in the Standard plan, and analysis tools are only included in the Premium plan. [Screenshot made by the author from Prezi.com [Halácsy et al. 2019b].]

3.4.3 Showoff

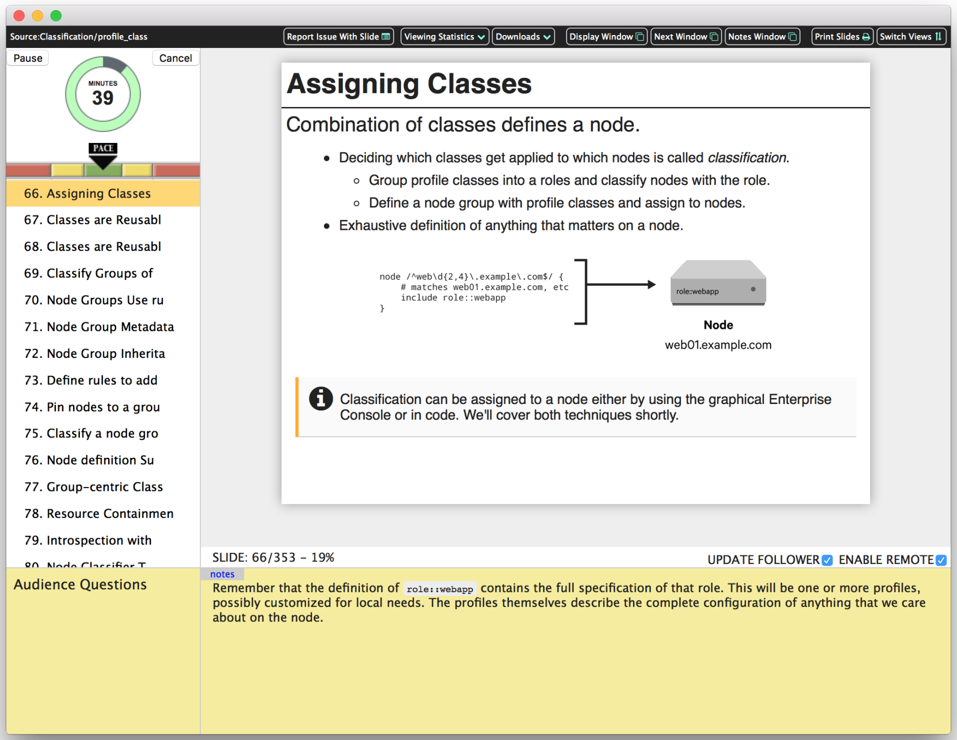
"Don't just present; interact with your audience!" is the core idea behind Showoff [Puppetlabs 2019]. It supports Markdown, live code execution, speaker notes, and a presenter mode. Unlike traditional slide decks, Showoff also includes live quizzes, polls, and feedback questions in order to boost audience interaction. An example can be seen in Figure 3.18. Showoff is written in Ruby and can be easily installed like any other Ruby gem.

Showoff is intended for use in a face-to-face classroom setting. To use its interactive features, the speaker executes the command `showoff serve` and distributes the local IP address URL to members of the audience. For these live viewers, the slideshow can either be synchronised to the main presentation or navigated independently. It is also possible to create a slide deck for deployment to a web server, but without the interactive audience features.

3.4.4 Slidebean

Slidebean [2014a] is a hosted slide deck creation suite that offers automated tools for optimising the style and layout of slides. Slide creators only need to provide the content of the slides and Slidebean takes care of everything else. It also provides a variety of slide templates for pitches, sales, or academics as well as a searchable list of many stock images to select from, which will then automatically be arranged onto the slides. Of course, the slide creator is also free to alter the design and positioning of elements on the slides to their liking. An example slide is shown in Figure 3.19.

As with other hosted slide deck approaches, the presentation is then accessible via a link. If accessed this way, Slidebean provides a variety of analytical features which track how much time users spent looking at different slides and how they interacted with them. However, in order to make use of these features, a paid plan [Slidebean 2014b] must be used, the cheapest of which currently lies at $8 per month. The free trial version does not allow exporting of slide decks to PDF or HTML for offline usage.



66. Assigning Classes
67. Classes are Reusabl
68. Classes are Reusabl
69. Classify Groups of
70. Node Groups Use ru
71. Node Group Metadata
72. Node Group Inherita
73. Define rules to add
74. Pin nodes to a grou
75. Classify a node gro
76. Node definition Su
77. Group-centric Class
78. Resource Containmen
79. Introspection with

Assigning Classes

Combination of classes defines a node.

- Deciding which classes get applied to which nodes is called *classification*.
  - Group profile classes into a roles and classify nodes with the role.
  - Define a node group with profile classes and assign to nodes.
- Exhaustive definition of anything that matters on a node.

```
node /^web\d{2,4}\.example\.com$/ {
    # matches web01.example.com, etc
    include role::webapp
}
```

Classification can be assigned to a node either by using the graphical Enterprise Console or in code. We'll cover both techniques shortly.

SLIDE: 66/353 – 19%

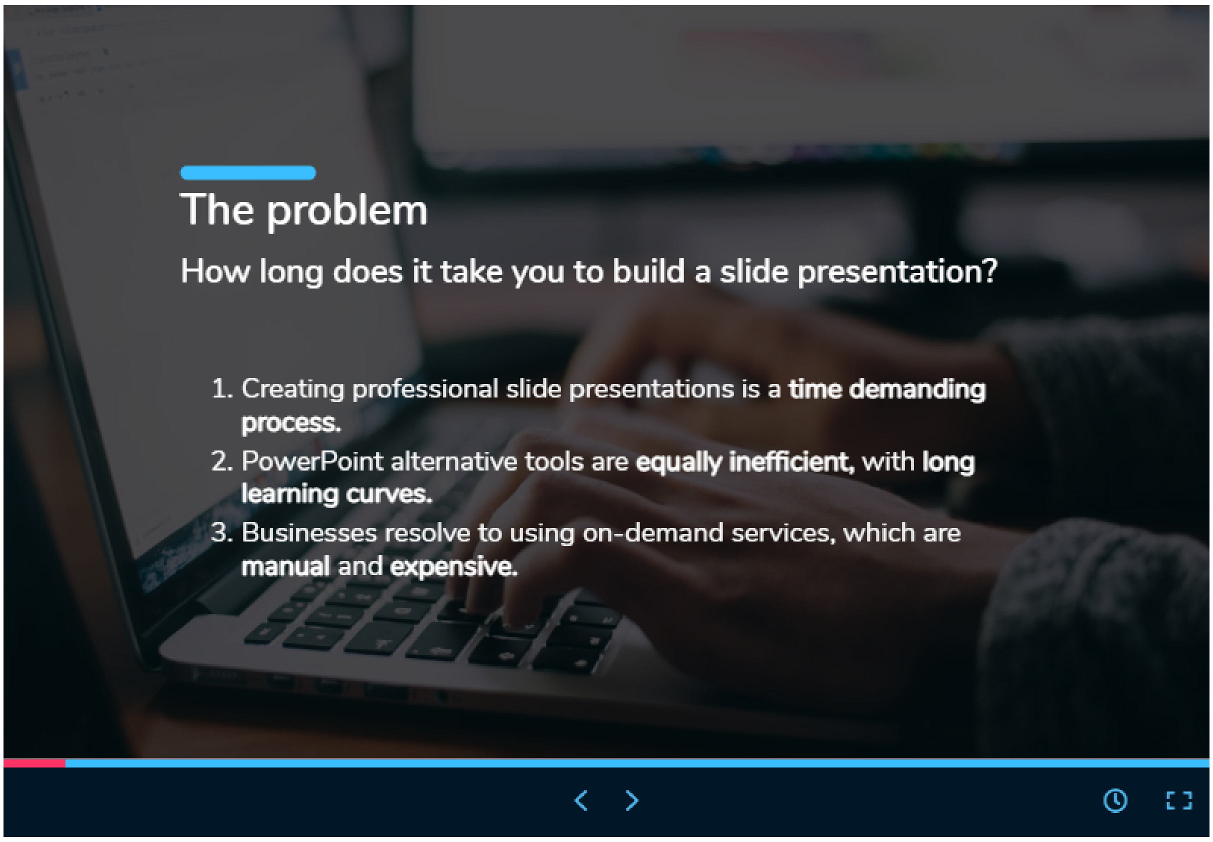
Remember that the definition of `role::webapp` contains the full specification of that role. This will be one or more profiles, possibly customized for local needs. The profiles themselves describe the complete configuration of anything that we care about on the node.

Figure 3.18: Showoff. Showoff encourages interaction with the audience. The person hosting a Showoff presentation can see a list of audience questions at the bottom of the screen. [Screenshot taken from Showoff [Puppetlabs 2019] (MIT License).]

3.4.5 Beautiful.ai

Beautiful.ai [2018b] offers AI assistance during the slide creation process. A set of so-called smart slide templates is provided, which adapt to changes in layout or content using AI technology. These templates include common slide layouts as well as bar charts, flow charts, Venn diagrams, and many other types of chart. Beautiful.ai offers a free image and icon library. An example slide is shown in Figure 3.20.

Slide decks can be exported to PDF or PowerPoint, but Beautiful.ai also offers an offline desktop player for download. Paid plans [Beautiful.ai 2018a] include analytic tools and give access to custom fonts as well as a team management feature which allows multiple creators to collaborate on slide decks as well as share slide libraries with each other. The team plan is currently priced at $38 per user per month.



The problem

How long does it take you to build a slide presentation?

1. Creating professional slide presentations is a **time demanding process.**
2. PowerPoint alternative tools are **equally inefficient,** with **long learning curves.**
3. Businesses resolve to using on-demand services, which are **manual** and **expensive.**

Figure 3.19: Slidebean. Slidebean wants to optimise the slide creation process via automated styling and placement of content with AI. [Screenshot made by the author from Slidebean [Slidebean 2014c].]

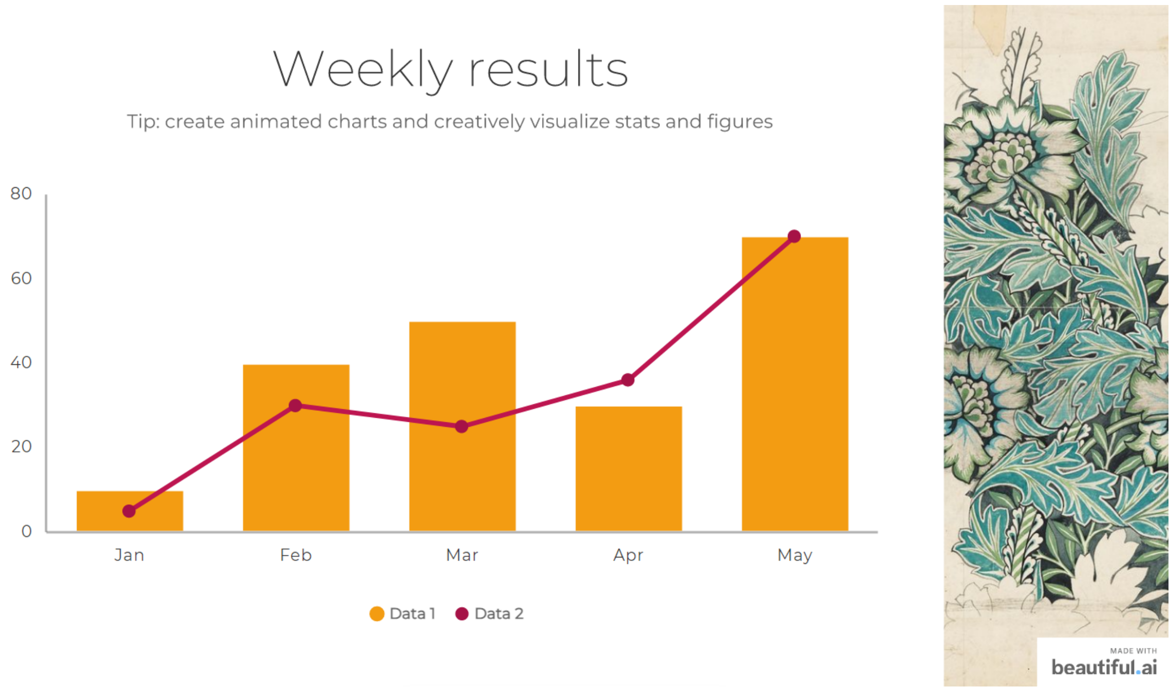


Figure 3.20: Beautiful.ai. Smart slide templates offer AI powered tools for creating charts, graphs, and many other data visualisations. [Screenshot made by the author from Beautiful.ai [Beautiful.ai 2018b].]

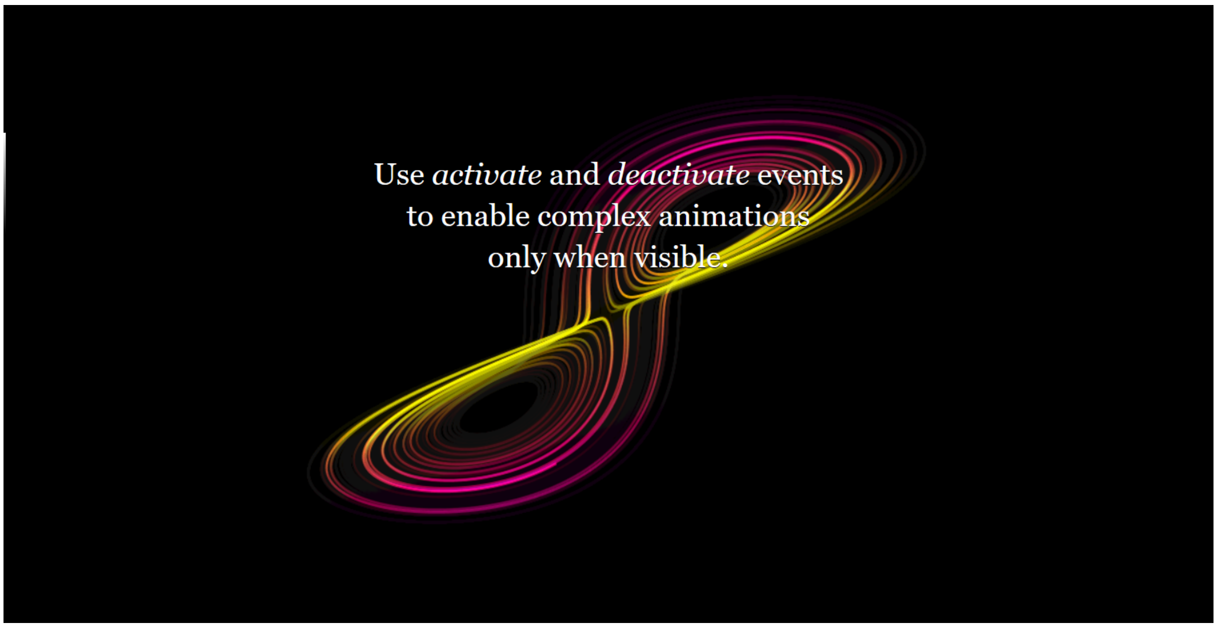

Use *activate* and *deactivate* events
to enable complex animations
only when visible.

Figure 3.21: Stack. In order to conserve power, Stack provides the `activate` and `deactivate` events. With these events, expensive calculations can be disabled unless a specific slide is displayed. [Screenshot made by the author from Stack [Bostock 2015b].]

3.5 Responsive Slide Decks

The key idea behind responsive web design (Section 2.8) is to respond to changes in the browser environment and offer whatever features are possible through flexible, robust, and device-independent design. Responsive slide decks can thus tackle the challenges of modern web development described in Section 1.3.

3.5.1 Stack

Stack [Bostock 2015a] takes a unique approach to slide decks by stacking slides from top to bottom, rather than relying on the usual left to right transitions. Slide navigation is handled primarily by scrolling, which is a basic feature supported by every browser on every device. Relying on scrolling allows Stack to support touch controls without explicitly coding for them. A small bar is displayed on the left side of the screen indicating the distance which needs to be scrolled for the next slide to appear. In addition to mouse wheel, touch pad, or touch screen, basic keyboard controls are also supported.

Stack resizes the content of slides dynamically with CSS media queries and JavaScript, maintaining a constant aspect ratio. The framework also provides two events `activate` and `deactivate` to execute certain content only when a specific slide is displayed. This can be used to prevent expensive scripts or animations from executing while they are not visible. An example can be seen in Figure 3.21 and Listing 3.16.

3.5.2 Scrolldeck.js

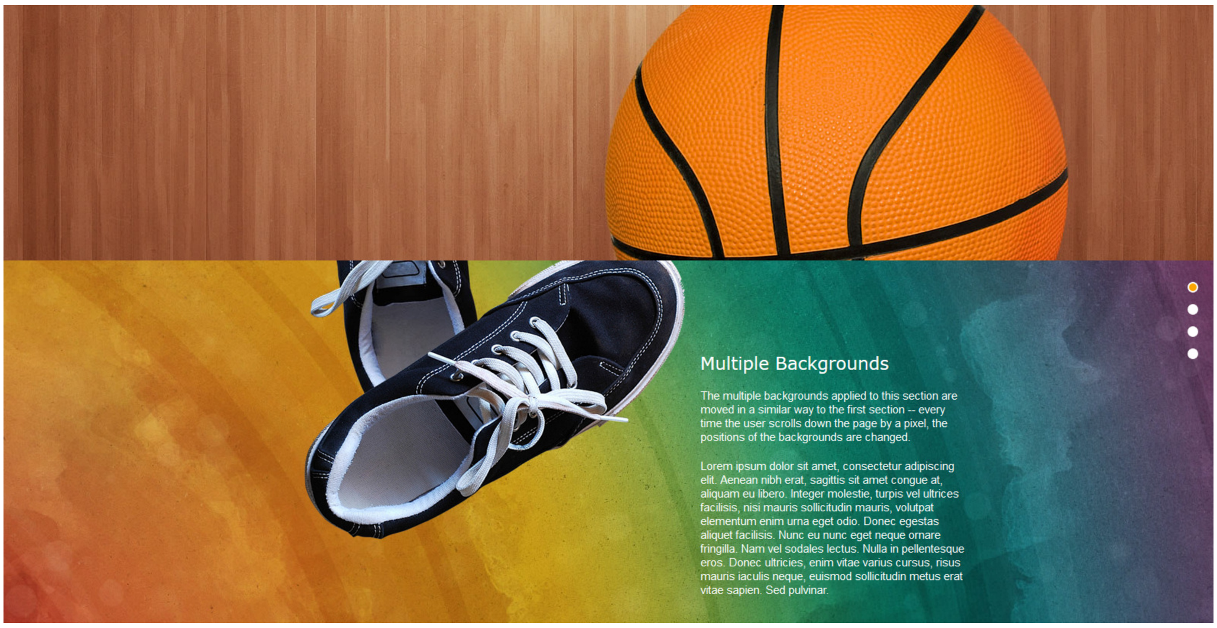
Scrolldeck.js [Polacek 2015b] builds on the idea behind Stack (see Section 3.5.1) and enhances it using jQuery and Scrollorama. This includes support for animations, incremental lists, parallax scrolling, and improvements in the responsive scaling of content. The slides are no longer fixed to a constant aspect ratio and scale more freely. An optional sticky header is also supported, which displays the title and some user-defined navigation shortcuts. An example of how to set up Scrolldeck.js can be seen in Listing 3.17. Parallax scrolling is shown in Figure 3.22.

```
<section>
  <p>
    Use <i>activate</i> and <i>deactivate</i>
        events
    <br>to enable complex animations
    <br>only when visible.
  </p>
</section>

var mystack = stack()
    .on("activate", activate)
    .on("deactivate", deactivate);

function activate(d, i) {
  if (i === 3) start();
}

function deactivate(d, i) {
  if (i === 3) stop();
}
```

Listing 3.16: Stack. An example slide in Stack. On a slide transition, the events `activate` and `deactivate` are called. In this example, special functions are called for the slide with index 3.

```
<!-- Load the required dependencies and scrolldeck.js -->
<script src="js/jquery-1.8.2.min.js"></script>
<script src="js/jquery.scrollTo-1.4.3.1.min.js"></script>
<script src="js/jquery.scrollorama.js"></script>
<script src="js/jquery.easing.1.3.js"></script>
<script src="js/jquery.scrolldeck.js"></script>

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

  <ul>
    <li>First bullet point</li>
    <li>Second bullet point</li>
    <li>Third bullet point</li>
  </ul>
</div>

<!-- Initialize scrolldeck.js -->
<script>
  $(document).ready(function() {
    var deck = new $.scrolldeck({
      buttons: '.nav-button',
      slides: '.slide',
      duration: 600,
      easing: 'easeInOutExpo',
    offset: 0
    });
  });
</script>
```

Listing 3.17: Scrolldeck.js. Scrolldeck.js requires the scripts `jquery`, `scrollTo`, `scrollorama` and `easing` to be linked before its own script file. Initialisation and configuration is then handled in the `$(document).ready` function.



Multiple Backgrounds

The multiple backgrounds applied to this section are moved in a similar way to the first section -- every time the user scrolls down the page by a pixel, the positions of the backgrounds are changed.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean nibh erat, sagittis sit amet congue at, aliquam eu libero. Integer molestie, turpis vel ultrices facilisis, nisi mauris sollicitudin mauris, volutpat elementum enim urna eget odio. Donec egestas aliquet facilisis. Nunc eu nunc eget neque ornare fringilla. Nam vel sodales lectus. Nulla in pellentesque eros. Donec ultricies, enim vitae varius cursus, risus mauris iaculis neque, euismod sollicitudin metus erat vitae sapien. Sed pulvinar.

Figure 3.22: Scrolldeck.js. Scrolldeck.js can blend together two slides using parallax scrolling. [Screenshot made by the author from Scrolldeck.js [Polacek 2015a].]

Chapter 4

History of Rslidy

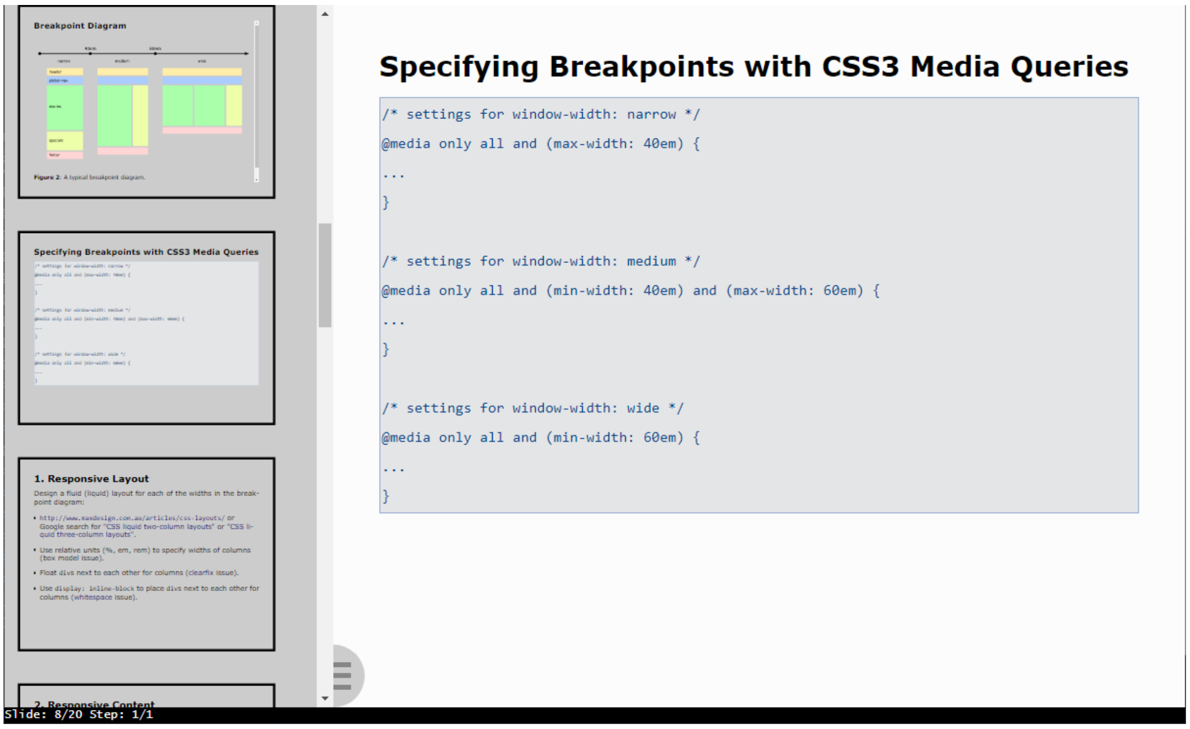
Rslidy is a lightweight and responsive slide deck currently in development at Graz University of Technology. Rslidy, in its current state [Hipp and Andrews 2021], is feature-rich, responsive, resilient, and supports a variety of input methods. It is backwards compatible with Slidy and Slidy2 (see Section 3.1.1). However, it took many iterations, a variety of different tools, and the contributions of many talented students to reach this state. This chapter chronicles the keystones and turning points during its development.

4.1 Rslidy: First Iteration

Rslidy was originally developed by a group of students [Kasper et al. 2014] for the course Information Architecture and Web Usability [Andrews 2020]. It was developed as an enhancement to Dave Raggett's Slidy [Raggett 2005], with an added layer of responsiveness, support for touch inputs, and a table of contents with slide previews. This table of contents can be accessed through an on-screen button, as can be seen on the example slide in Figure 4.1.

Similar to the original Slidy [Raggett 2005] and Slidy2 [Raggett 2006a], slides were specified inside `<div class="slide">` elements and incremental display of list items was implemented by providing the CSS class `incremental` to be added to the `<ul>` or `<ol>` elements.

Some features of this first version of Rslidy included: swipe and tap inputs via the third-party library Hammer.js [Hammer.js 2016], support for key inputs and overrides using `<meta>` tags, experimental remote control support via WebSockets, and basic animations. It served as a general proof of concept for responsive slide decks and laid the foundation for the Rslidy codebase.



Breakpoint Diagram

Figure 2: A typical breakpoint diagram.

Specifying Breakpoints with CSS3 Media Queries

1. Responsive Layout

Design a fluid (liquid) layout for each of the widths in the break-point diagram:

- http://www.maxdesign.com.au/articles/css-layouts/ or Google search for "CSS liquid two-column layouts" or "CSS liquid three-column layouts".
- Use relative units (%, em, rem) to specify widths of columns (box model issue).
- Float divs next to each other for columns (clearfix issue).
- Use display: inline-block to place divs next to each other for columns (whitespace issue).

2. Responsive Content

Slide: 8/20 Step: 1/1

Specifying Breakpoints with CSS3 Media Queries

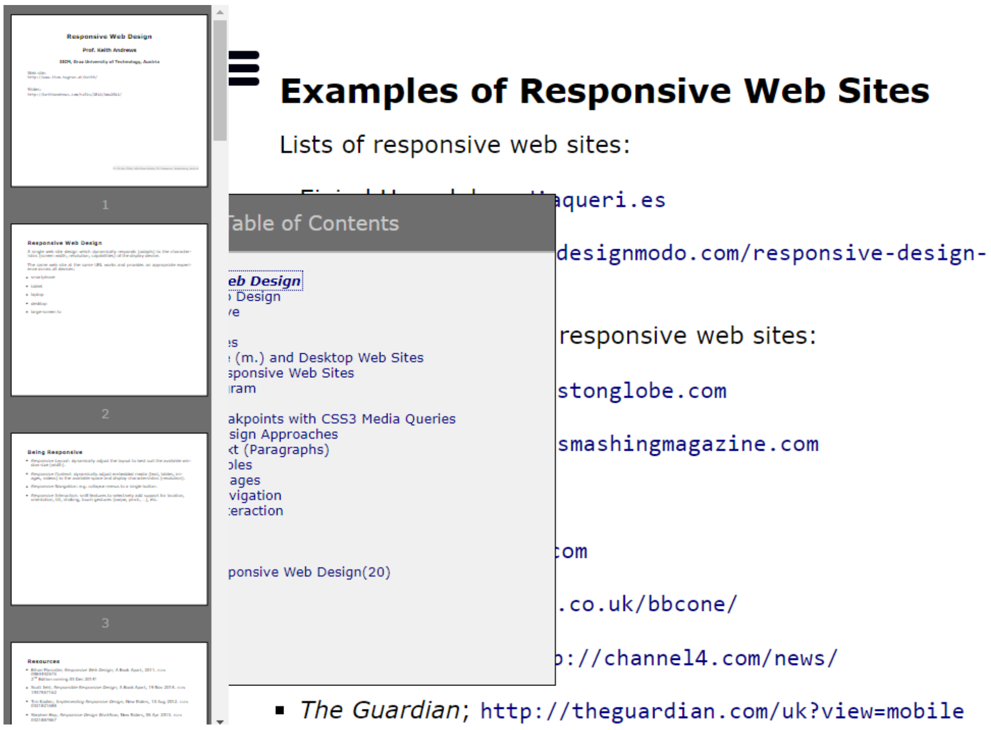
```
/* settings for window-width: narrow */
@media only all and (max-width: 40em) {
...
}

/* settings for window-width: medium */
@media only all and (min-width: 40em) and (max-width: 60em) {
...
}

/* settings for window-width: wide */
@media only all and (min-width: 60em) {
...
}
```

Figure 4.1: The first iteration of Rslidy delivered a rudimentary interface for web-based slide decks. [Screenshot made by the author from Rslidy [Kasper et al. 2014].]

4.2 Rslidy: Second Iteration

The second iteration of Rslidy was also handled by a group of students [Garolla et al. 2015] for the course Information Architecture and Web Usability [Andrews 2020]. The primary focus of this iteration lay in enhancing the support for mobile devices and providing an improved universal style sheet for slides. Tilt and shake controls were added using the third-party library Full-Tilt [Tibbett 2015], while touch gesture support was still provided by Hammer.js [Hammer.js 2016]. Additionally, a help page was added and several minor improvements were made.

An example slide can be seen in Figure 4.2, which shows some of the UI issues with this iteration. The table of contents was partially obscured when a sidebar was open. The same was true for the settings menu, which allowed toggling of tilt and shake controls.


Responsive Web Design

Prof. Keith Andrews

IICM, Graz University of Technology, Austria

1

Responsive Web Design

2

Being Responsive

3

Resources

Examples of Responsive Web Sites

Lists of responsive web sites:

queri.es

designmodo.com/responsive-design-

responsive web sites:

stonglobe.com

smashingmagazine.com

om

.co.uk/bbcone/

://channel4.com/news/

- *The Guardian*; http://theguardian.com/uk?view=mobile

Table of Contents

eb Design
Design
e

s
(m.) and Desktop Web Sites
sponsive Web Sites
ram

akpoints with CSS3 Media Queries
sign Approaches
t (Paragraphs)
les
ages
vigation
eraction

ponsive Web Design(20)

Figure 4.2: The second iteration of Rslidy had some UI issues, for example the table of contents could become partially obscured. [Screenshot made by the author from Rslidy [Garolla et al. 2015].]

4.3 Rslidy: Third Iteration

Rslidy was overhauled from the ground up by Schofnegger [2015] in the scope of a Bachelor's Thesis. The source code now comprised a single TypeScript [Microsoft 2019] file and several pure CSS [Cederholm 2015] files. Grunt [Grunt 2020] was used both as a simple build system and to bundle the files. After running grunt, one minified JavaScript and one minified CSS file were produced, which could then be embedded into an HTML slide deck, as shown in Listing 4.1.

This iteration of Rslidy provided some basic keyboard shortcuts for navigation, swipe and motion gestures without third-party dependencies, as well as a minimalistic on-screen interface. The interface, along with an example slide, is shown in Figure 4.3. The interface consisted of a toolbar (status bar) beneath the slide, containing five buttons and a slide counter. From left to right, the buttons are:

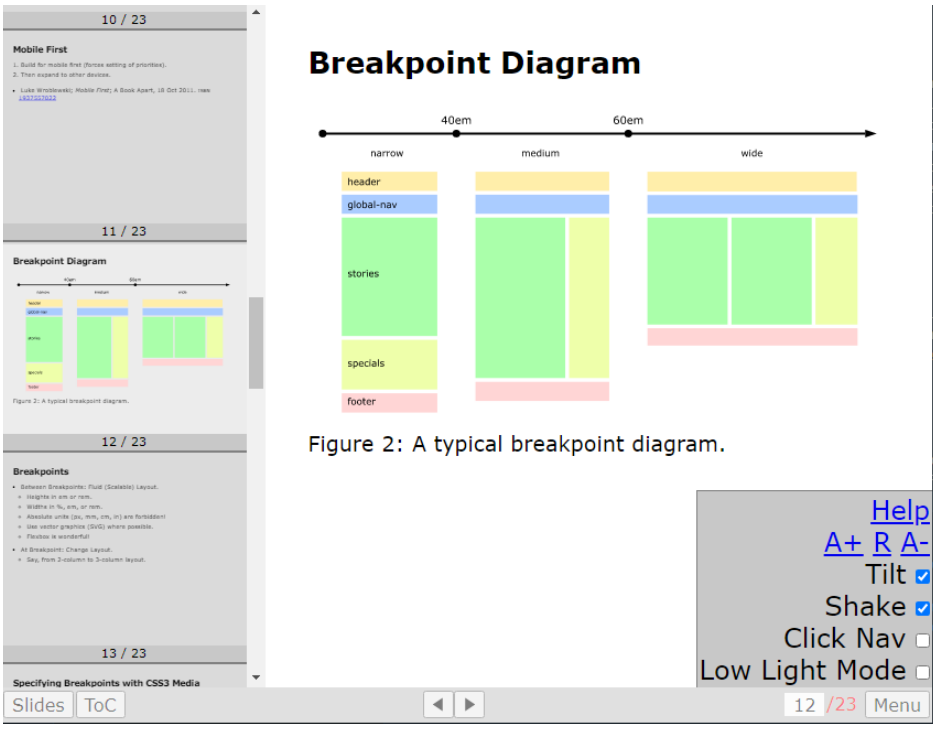
- Slides: Opens a scrollable panel of slide thumbnails on the left side of the main slide window.
- ToC: Opens a Table of Contents (list of slide headers) on the left side the main slide window.
- ◀ and ▶ : Move to the previous and next slide, respectively.
- Menu: Opens a menu to adjust Rslidy settings, such as slide font size, motion gestures, click to navigate, and a low light mode. Additionally, a short help text in a popup window can be toggled on or off.

While providing a variety of basic functionality, this iteration also had a variety of shortcomings. The UI did not dynamically adjust to the screen size and could not be adjusted from within the menu. This made it very hard to access the on-screen buttons on narrower devices. Additionally, no keyboard shortcuts were provided beyond those for navigation.

Another major shortcoming of this iteration lay in its maintainability. Since all the code was contained in a single TypeScript file and no modular approach was used, the scope of certain functions was not clearly indicated. One change could have several repercussions in various parts of the application. In

```html
<head>
  <link rel="stylesheet" href="../css/rslidy-combined.min.css"/>
  <script src="../js/rslidy.js"></script>
</head>

<body>
  <div class="slide">
    <h1>Breakpoint Diagram</h1>

    <figure>
      <div class="figure" style="width:100%;">
        <img src="images/breakpoint.svg" alt="Breakpoint Diagram" />
      </div>
      <figcaption>
        <p class="caption">
          <span class="fig-label">Figure 2</span>: A typical breakpoint diagram.
        </p>
      </figcaption>
    </figure>
  </div>
</body>
```

Listing 4.1: The third iteration of Rslidy required one CSS and one JavaScript file to be included in the head of the HTML file. Slides were specified inside `<div class="slide">` elements. The resulting slide can be seen in Figure 4.3.



10 / 23

Mobile First

1. Build for mobile first (forces setting of priorities).
2. Then expand to other devices.

- Luke Wroblewski; *Mobile First*; A Book Apart, 18 Oct 2011. ISBN 1937557022

11 / 23

Breakpoint Diagram

Figure 2: A typical breakpoint diagram.

12 / 23

Breakpoints

- Between Breakpoints: Fluid (Scalable) Layout.
  - Heights in em or rem.
  - Widths in %, em, or rem.
  - Absolute units (px, mm, cm, in) are forbidden!
  - Use vector graphics (SVG) where possible.
  - Flexbox is wonderful!
- At Breakpoint: Change Layout.
  - Say, from 2-column to 3-column layout.

13 / 23

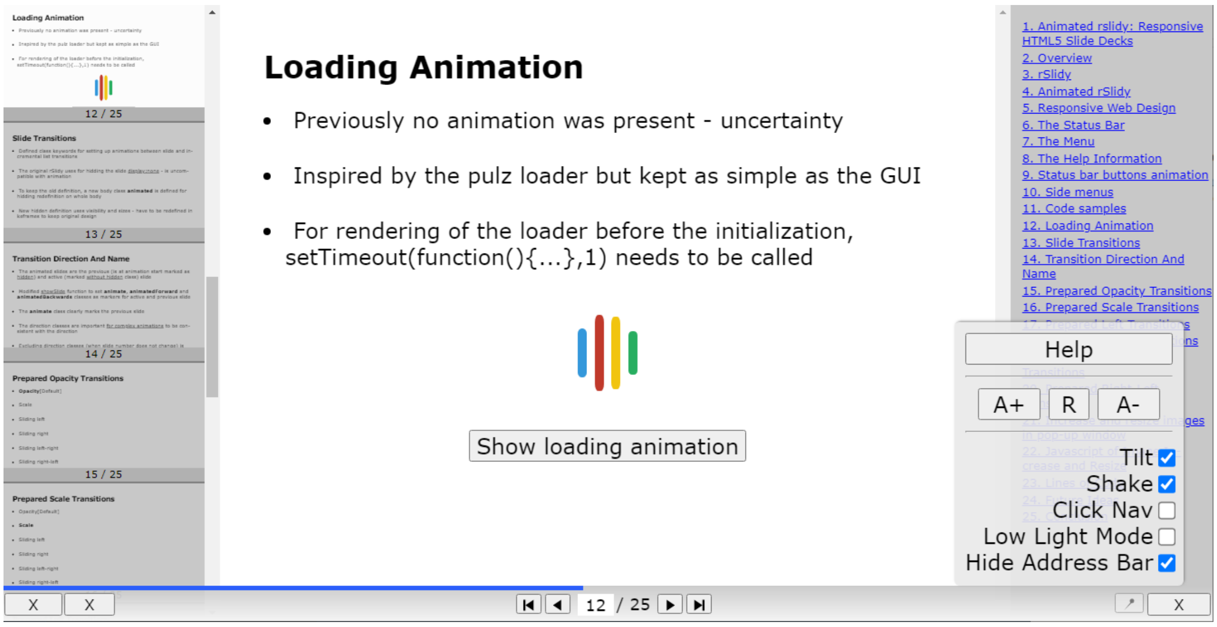
Specifying Breakpoints with CSS3 Media

Breakpoint Diagram

Figure 2: A typical breakpoint diagram.

Tilt ✓
Shake ✓
Click Nav ☐
Low Light Mode ☐

Figure 4.3: The third iteration of Rslidy provided a simple but efficient interface for web-based slide decks. [Screenshot made by the author from Rslidy [Schofnegger 2015].]



Loading Animation

- Previously no animation was present - uncertainty
- Inspired by the pulz loader but kept as simple as the GUI
- For rendering of the loader before the initialization, setTimeout(function(){...},1) needs to be called

Show loading animation

12 / 25

Slide Transitions

- Defined class keywords for setting up animations between slide and incremental list transitions
- The original rSlidy uses for hidding the slide display:none - is uncompatible with animation
- To keep the old definition, a new body class **animated** is defined for hidding redefinition on whole body
- New hidden definition uses visibility and sizes - have to be redefined in keframes to keep original design

13 / 25

Transition Direction And Name

- The animated slides are the previous (is at animation start marked as hidden) and active (marked without hidden class) slide
- Modified showSlide function to set **animate, animatedForward** and **animatedBackwards** classes as markers for active and previous slide
- The **animate** class clearly marks the previous slide
- The direction classes are important for complex animations to be consistent with the direction

14 / 25

Prepared Opacity Transitions

- **Opacity**[Default]
- Scale
- Sliding left
- Sliding right
- Sliding left-right
- Sliding right-left

15 / 25

Prepared Scale Transitions

- Opacity[Default]
- **Scale**
- Sliding left
- Sliding right
- Sliding left-right
- Sliding right-left

1. Animated rslidy: Responsive HTML5 Slide Decks
2. Overview
3. rSlidy
4. Animated rSlidy
5. Responsive Web Design
6. The Status Bar
7. The Menu
8. The Help Information
9. Status bar buttons animation
10. Side menus
11. Code samples
12. Loading Animation
13. Slide Transitions
14. Transition Direction And Name
15. Prepared Opacity Transitions
16. Prepared Scale Transitions

Help

A+ R A-

Tilt ✓
Shake ✓
Click Nav ☐
Low Light Mode ☐
Hide Address Bar ✓

X X 12 / 25 X

Figure 4.4: The fourth iteration of Rslidy provided several UI enhancements, as well as support for animations and third party tools. The on-screen buttons flip and display an X while they are active. [Screenshot made by the author from Rslidy [Kogovšek et al. 2017].]

addition, pure CSS was used, which at the time did not support variables, making it difficult to maintain a uniform design approach.

4.4 Rslidy: Fourth Iteration

The main goals of the fourth iteration of Rslidy were to enhance the Rslidy UI, introduce animated slide transitions, implement a zoomable image viewer, and provide support for third-party tools, such as syntax highlighting for code snippets. It was developed by a group of students [Kogovšek et al. 2017] for the course Information Architecture and Web Usability [Andrews 2020].

The improved UI is shown in Figure 4.4. A progress bar was added to the toolbar and the table of contents was moved to the right of the slide. The toolbar was enhanced with two buttons for quick navigation to the first and last slides. A pin button was added, which can be used to pin or unpin the toolbar. When loading a slide deck, a splash screen was displayed until Rslidy finished initialisation.

Animated slide transitions were implemented by providing CSS classes to be added to the <body> element:

- `animated opacity`: Slowly fades out to a white screen, before fading in the next slide. The default slide transition.
- `slidein`: Slides the current slide off screen in one direction, followed by sliding in from another direction. Can be set to left, right, left-right, or right-left.
- `scaleup`: Gradually scales down the size of the current slide until it is no longer visible. The next slide is shown by inverting this animation.

In addition, a simple, pop-up based image viewer was added as well, as shown in Figure 4.5. A row of buttons was provided at the top of the pop-up to adjust the magnification level of the image.

Unfortunately, all of these changes were made directly to the JavaScript file of the previous iteration of Rslidy, so they had to be either slowly integrated back into future iterations or abandoned entirely. The

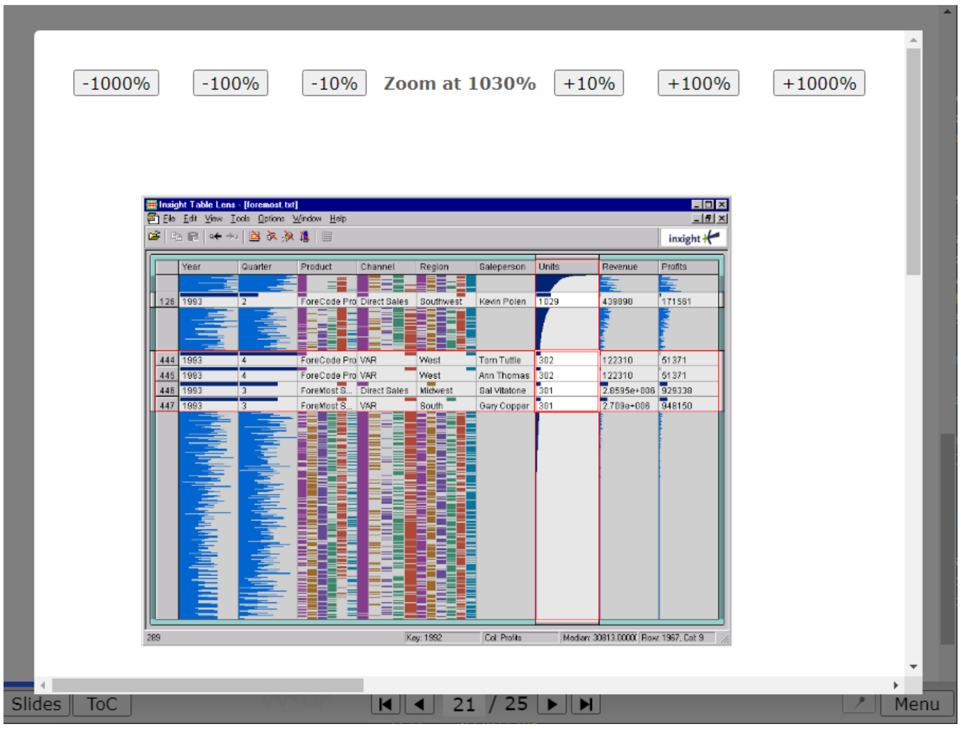


Figure 4.5: The fourth iteration of Rslidy provided a simple image viewer, allowing for magnification of images. [Screenshot made by the author from Rslidy [Kogovšek et al. 2017].]

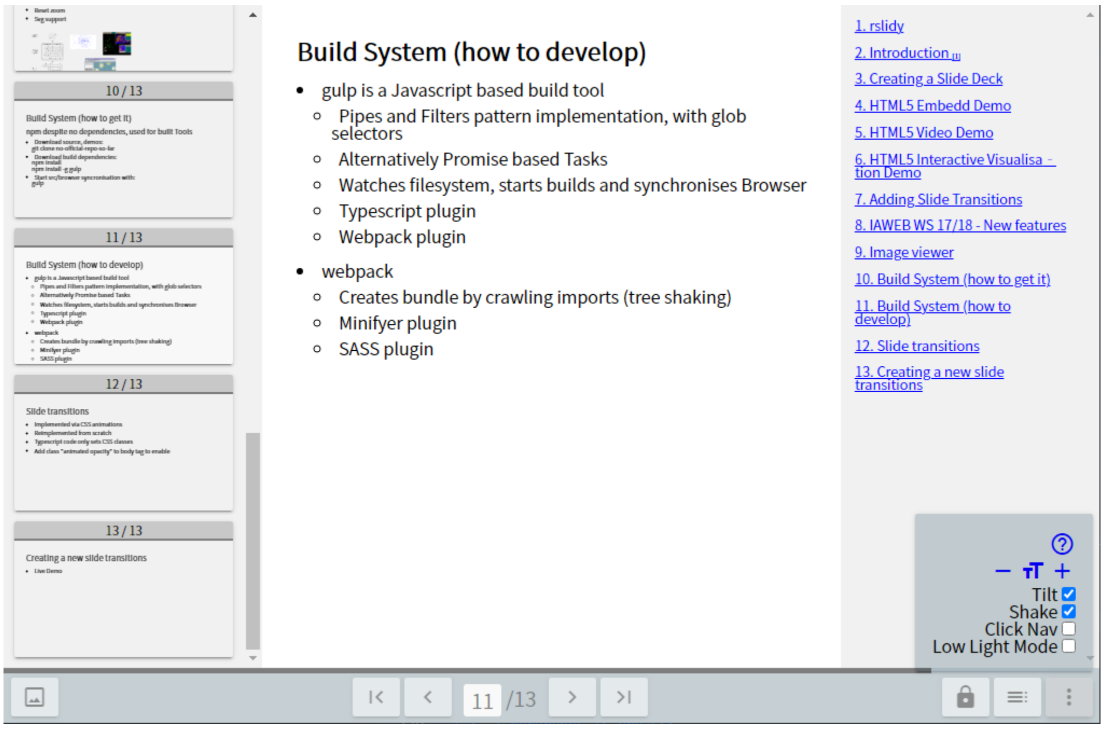
toolbar was still static and did not adjust itself relative to the screen size, and no additional keyboard shortcuts were provided.

4.5 Rslidy: Fifth Iteration

This iteration tackled the maintainability concerns of previous iterations by moving to a modular design approach, while also porting and polishing desirable aspects of the fourth iteration to TypeScript and SCSS. It was also developed by a group of students [Eibl et al. 2018] for the course Information Architecture and Web Usability [Andrews 2020].

The build system was upgraded from Grunt to Gulp [Schoffstall 2019] to better accommodate the modular structure and allow easier access to front-end web development tools. One of the tools used was webpack [Webpack 2021], which can bundle and optimise TypeScript and SCSS files among other formats. Other tools, such as BrowserStream [Tarr 2012], were used to simplify the development workflow. BrowserStream watches for changes in the source files, toggles the build script, and updates the browser window with the newly packed files.

Several enhancements were made to Rslidy's UI. The Material Components library [Google 2020] was used for fonts and icons, as shown in Figure 4.6. However, this was deemed to be overly bloated and was later removed. The problems regarding responsiveness and usability mentioned in previous iterations were not addressed.


- Reset zoom
- Svg support

10 / 13

Build System (how to get it)

npm despite no dependencies, used for built Tools

- Download source, demos:
  git clone no-official-repo-so-far
- Download build dependencies:
  npm install
  npm install -g gulp
- Start src/browser syncronisation with:
  gulp

11 / 13

Build System (how to develop)

- gulp is a Javascript based build tool
  - Pipes and Filters pattern implementation, with glob selectors
  - Alternatively Promise based Tasks
  - Watches filesystem, starts builds and synchronises Browser
  - Typescript plugin
  - Webpack plugin
- webpack
  - Creates bundle by crawling imports (tree shaking)
  - Minifyer plugin
  - SASS plugin

12 / 13

Slide transitions

- Implemented via CSS animations
- Reimplemented from scratch
- Typescript code only sets CSS classes
- Add class "animated opacity" to body tag to enable

13 / 13

Creating a new slide transitions

- Live Demo

Build System (how to develop)

- gulp is a Javascript based build tool
  - Pipes and Filters pattern implementation, with glob selectors
  - Alternatively Promise based Tasks
  - Watches filesystem, starts builds and synchronises Browser
  - Typescript plugin
  - Webpack plugin
- webpack
  - Creates bundle by crawling imports (tree shaking)
  - Minifyer plugin
  - SASS plugin

1. rslidy
2. Introduction [1]
3. Creating a Slide Deck
4. HTML5 Embedd Demo
5. HTML5 Video Demo
6. HTML5 Interactive Visualisa - tion Demo
7. Adding Slide Transitions
8. IAWEB WS 17/18 - New features
9. Image viewer
10. Build System (how to get it)
11. Build System (how to develop)
12. Slide transitions
13. Creating a new slide transitions

Tilt ✓
Shake ✓
Click Nav ☐
Low Light Mode ☐

Figure 4.6: The fifth iteration of Rslidy relied on a third party library for fonts and icons. [Screenshot made by the author from Rslidy [Eibl et al. 2018].]

Chapter 5

The Architecture of Rslidy

Nowadays, Rslidy is feature-rich, responsive, resilient, and supports a variety of input methods [Hipp and Andrews 2021]. Some of its features include: slide transitions, a navigable progress bar, an image viewer with pan and zoom controls, a low light mode, print style sheet, navigation with touch, tilt, shake and swipe controls, a table of contents, and an overview with slide thumbnails. The software architecture behind Rslidy tries to stay minimalistic and devoid of third-party dependencies, while upholding accessibility and usability principles [Nielsen 1994] wherever possible. These include:

- *Communication of System Status*: The user should always be aware of the current system status and as little as possible should be obscured by tasks running in the background.
- *Consistency*: Users have expectations of how certain design elements should work. These should be consistent across platforms.
- *Flexibility*: The user should be free to use their preferred method of interacting with the software. This directly ties into accessibility.
- *Help and Documentation*: Ensure access to short and precise documentation for both the code and the application.

5.1 Build System

Rslidy is written in TypeScript, SCSS, and HTML5. The build system consists of gulp, which is a task runner written in JavaScript, and webpack, which can transpile and bundle all TypeScript files into a singular minified JavaScript file, as well as bundle all SCSS files into a singular minified CSS file. The resulting JavaScript and CSS files are then copied to the build folder, as well as to every example slide deck. Additionally, a template for usage with S9 (see Section 3.2.1) is generated. This allows swift generation of Rslidy slide decks from Markdown.

5.2 Modules

Rslidy uses a modular code base to reduce the scope of individual modules and promote better maintainability. A module consists of a TypeScript file, as well as an optional SCSS file when needed. Currently, Rslidy consists of the following TypeScript modules:

- *Content*: Displays and manages the content of slides and keeps track of the current slide index.
- *HTML Definitions*: Contains all the HTML definitions necessary for drawing Rslidy's UI.
- *Icon Definitions*: Contains definitions for all icons used within Rslidy's UI.

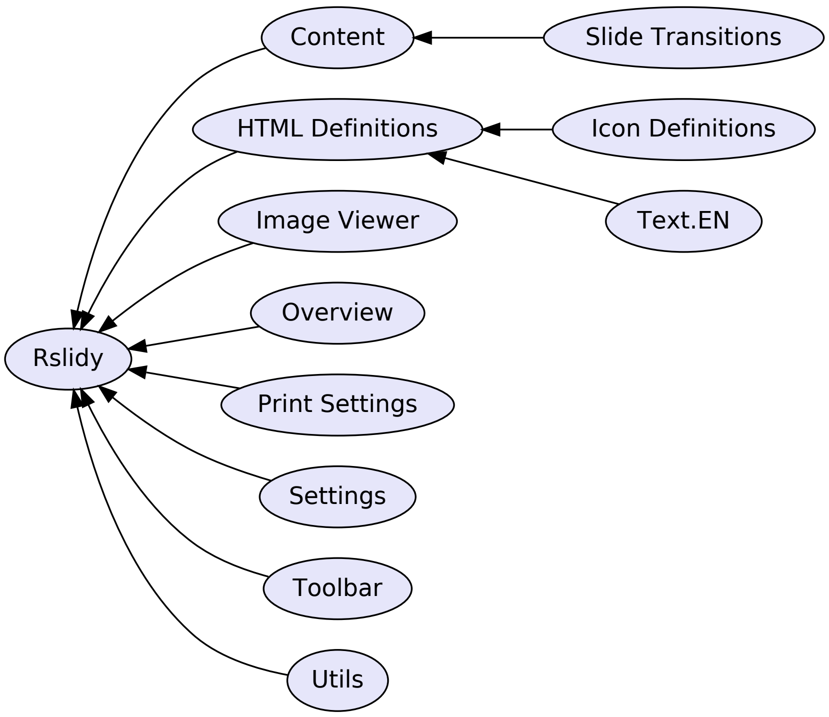


Figure 5.1: Rslidy is comprised of several modules, whose dependencies are illustrated here. [Diagram created by the author in R.]

- *Image Viewer*: Handles the Image Viewer component and its initialisation.
- *Overview*: Handles the rendering of the Slide Overview Panel, which shows a list of slide thumbnails, as well as the Table of Contents Panel.
- *Print Settings*: Handles the Print Settings Panel and ensures print settings are persisted between sessions.
- *Rslidy*: Handles the main initialisation, as well as that of other modules and event listeners.
- *Settings*: Handles the Settings Panel and ensures settings are persisted between sessions.
- *Slide Transitions*: Handles the rendering of slide transitions.
- *Text.EN*: Contains all of the text found within Rslidy's UI. In order to localise Rslidy for a different language, only this file needs to be modified.
- *Toolbar*: Handles the state of the Toolbar as well as all the buttons that can be found on it.
- *Utils*: Provides useful utilities which are shared between modules. These utilities deal with creation of HTML elements, CSS classes, and parsing of content.

The dependencies between these modules are illustrated in Figure 5.1.

In addition, there are three standalone SCSS models:

- *Variables*: For ensuring consistent fonts, colours, and breakpoints across all style sheets.
- *Normalise*: For ensuring consistent cross-browser behaviour and styling.
- *Reset*: For ensuring consistent cross-browser base layout.

5.3 Icons

Icons are stored as standalone SVG files in the folder `src/icons`. This allows quick modification of individual icons using external programs. A gulp task is then run to bundle them into their respective module. Illustrators do not need to worry about file size optimisation, since the gulp task takes care of it using SVGO [Belevich 2019].

5.4 Examples

The folder `examples` is copied from the `src` folder to the `build` folder during Rslidy's build process. It contains a number of fully-fledged slide decks, which illustrate best practice responsive content, such as responsive tables and responsive images. Some of these example slide decks are discussed in Section B.6.

5.5 Tests

The folder `tests` is also copied from the `src` folder to the `build` folder during the build process. It contains a number of specific slide examples used to test particularly tricky situations. They are further discussed in Section 7.1.

Chapter 6

Selected Details of the Implementation

While the previous chapter offered a broad overview of the architecture of Rslidy, this chapter describes some of the more intricate parts of Rslidy. The emphasis is placed on clarifying design decisions and shining light on the concrete implementation in the code. In particular, the generation of slide thumbnails, the handling of user settings, and accessibility provisions are discussed.

6.1 Slide Thumbnails

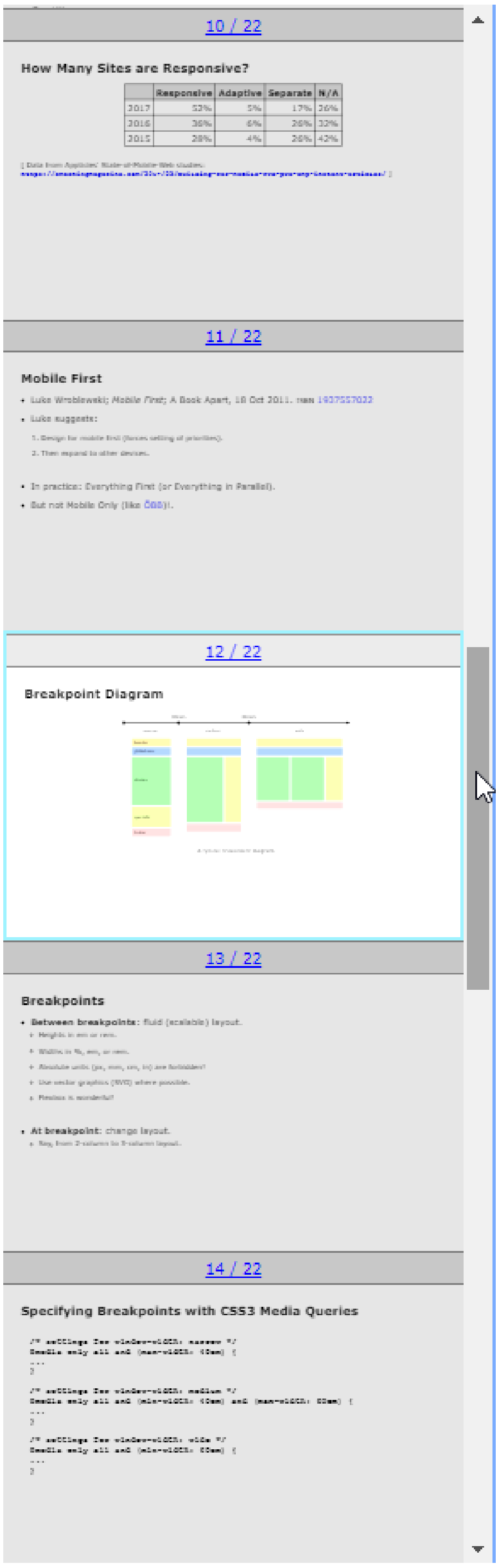
Slide thumbnails are visible whenever the user opens the Slide Overview Panel or hovers over a section of the Progress Bar. They depict a miniature version of a slide which serves as a preview in order to quickly navigate to a desired slide. In order to keep the HTML DOM [Lindley 2013] as concise as possible, slide thumbnails are generated on the fly and deleted once they are no longer visible. However, this approach is only feasible for individual thumbnails, like those shown when hovering over the Progress Bar. Since the Slide Overview Panel contains as many thumbnails as there are slides, generating them all over again and again can become quite computationally expensive for larger slide decks. Thus, a hybrid solution was used which generates slide thumbnails once for the Slide Overview Panel and individually on the fly for the Progress Bar.

Since thumbnail creation is needed for both of these use cases, wrapper functions are used to avoid duplication of code in the code base. A suffix is passed to the function to discern between Slide Overview Panel and Progress Bar and aids in finding the corresponding HTML Elements. The code for thumbnail generation is shown in Listing 6.1. The code grabs the necessary HTML elements in the first step. In the case of the Slide Overview Panel, all slides are collected, in the case of the Progress Bar, only one specific slide is collected. Next, the aspect ratio of the current browser window is calculated and some custom settings are read. In the case of the Slide Overview Panel, the slide thumbnails adjust to the width of the panel, however the preview thumbnails shown via the Progress Bar have a set width. Finally, the collected slides are scaled down according to the parameters above and the position of the slide thumbnail is adjusted. The resulting thumbnails are shown in Figure 6.1.

A possible shortcoming of this approach can be found in the limitations of the HTML DOM. If referencing of elements was allowed within the DOM, duplication of content could be entirely avoided. Cross-checking with similar slide decks, such as Google Slides [Google 2019], showed a similar approach was used. Another valid approach would have been the creation of actual image files for each slide thumbnail and storing them until needed. However, this would not be as responsive, since the image files cannot adjust to changes in CSS, like low light mode for example, and would need to be regenerated entirely every time the browser window is resized.

```
public adjustPanel(suffix: string): void {
  var thumbnail_wrappers: any = document.getElementsByClassName(
    "rslidy-slide-thumbnail" + suffix
  );
  var overview_items: any = document.getElementsByClassName(
    "rslidy-overview-item" + suffix
  );

  var aspect_ratio: number =
    window.rslidy.custom_aspect_ratio != 0
      ? window.rslidy.custom_aspect_ratio
      : window.rslidy.utils.getCurrentAspectRatio();
  var overview_slide_zoom: number = window.rslidy.overview_slide_zoom;
  var custom_width: number = window.rslidy.custom_width;
  var final_width: number = custom_width / overview_slide_zoom;
  var overview_width = this.slide_view.clientWidth;
  if(suffix == "-pv") overview_width = this.preview_width;

  var relative_width: number = window.rslidy.utils.getRelativeWidth(
    overview_width,
    aspect_ratio,
    final_width
  );

  for (var i: number = 0; i < overview_items.length; i++) {
    overview_items[i].style.width =
      window.rslidy.utils.getSlideWidth(aspect_ratio, final_width) + "px";
    overview_items[i].style.height =
      window.rslidy.utils.getSlideHeight(aspect_ratio, final_width) + "px";

    var scale_value: string = "scale3d(" +
      relative_width + ", " + relative_width + ", " + relative_width + ")";
    var origin_value: string = "0 0 0";
    overview_items[i].style.transform = scale_value; // safety first
    overview_items[i].style.transformOrigin = origin_value; // safety first
    overview_items[i].style.MozTransform = scale_value;
    overview_items[i].style.MozTransformOrigin = origin_value;
    overview_items[i].style.webkitTransform = scale_value;
    overview_items[i].style.webkitTransformOrigin = origin_value;
    overview_items[i].style.msTransform = scale_value;
    overview_items[i].style.msTransformOrigin = origin_value;

    const w = overview_width;
    const h = w /
      (window.rslidy.utils.getSlideWidth(aspect_ratio, final_width) /
        window.rslidy.utils.getSlideHeight(aspect_ratio, final_width));

    if(suffix == "-pv")
      thumbnail_wrappers[i].style.width = (w - 20) + "px";
    thumbnail_wrappers[i].style.height = (h - 20) + "px";
  }
}
```

Listing 6.1: The function `adjustPanel()` is called by the wrapper functions `adjustOverviewPanel()` and `adjustPreviewPanel()`, which pass along a corresponding suffix to generate slide thumbnails for the Slide Overview Panel or the Progress Bar respectively. First, the HTML items are gathered and aspect ratios are calculated. Then, the necessary size adjustments are made in a for loop over all items.



10 / 22

How Many Sites are Responsive?

	Responsive	Adaptive	Separate	N/A
2017	53%	5%	17%	26%
2016	36%	6%	26%	32%
2015	28%	4%	26%	42%

[Data from Appticles' State-of-Mobile-Web studies: [illegible]]

11 / 22

Mobile First

- Luke Wroblewski; *Mobile First*; A Book Apart, 18 Oct 2011. ISBN 1937557022
- Luke suggests:
  1. Design for mobile first (forces setting of priorities).
  2. Then expand to other devices.
- In practice: Everything First (or Everything in Parallel).
- But not Mobile Only (like [illegible]).

12 / 22

Breakpoint Diagram

13 / 22

Breakpoints

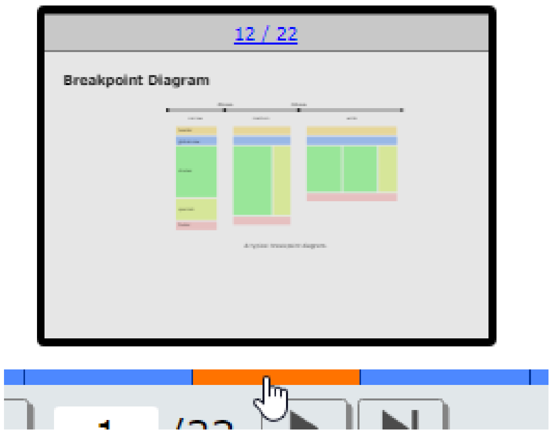
- **Between breakpoints:** fluid (scalable) layout.
  - Heights in em or rem.
  - Widths in %, em, or rem.
  - Absolute units (px, mm, cm, in) are forbidden!
  - Use vector graphics (SVG) where possible.
  - Flexbox is wonderful!
- **At breakpoint:** change layout.
  - Say, from 2-column to 3-column layout.

14 / 22

Specifying Breakpoints with CSS3 Media Queries

```
/* settings for window-width: narrow */
@media only all and (max-width: 40em) {
...
}

/* settings for window-width: medium */
@media only all and (min-width: 40em) and (max-width: 60em) {
...
}

/* settings for window-width: wide */
@media only all and (min-width: 60em) {
...
}
```

(a) Slide Overview Panel



12 / 22

Breakpoint Diagram

(b) Progress Bar

Figure 6.1: Rslidy's slide thumbnails offer a quick preview of the contents of each respective slide. They can be viewed in the Slide Overview Panel or by hovering over a section of the Progress Bar. [Screenshots made by the author from Rslidy.]

6.2 Settings

The Settings and Print Settings of Rslidy can be altered in order to adjust the behaviour of the application. The functionality of every individual setting is described in Section A.6.4). Here, the internal handling of settings is described, including how settings are persisted between sessions using JSON [JSON 1999] and the browser's local storage.

JSON is shorthand for JavaScript Object Notation and is a subset of JavaScript, which allows for a simple textual representation of data. In Rslidy, it is used to store and retrieve a collection of name-value pairs corresponding to the user settings. Listing 6.2 shows the code required for this process. First, a `Data` interface is established with the necessary data structures and their corresponding names. In this case, two numbers, for the current slide font and UI font adjustments, and five booleans, for the checkbox type settings, are needed.

The function `loadSettings()` is called whenever a slide deck running Rslidy is opened. It looks into the local storage of the browser and searches for an item called `rslidy`. If no item is found, or access to the local storage was restricted, the function stops and logs errors to the console. If an item is present, the `JSON.parse(item)` function is used to retrieve the data. The rest of the code deals with restoring the settings and enabling them when needed.

Similarly, the function `saveSettings()` is called whenever a slide deck running Rslidy is closed. This function first checks to see if settings are already present in local storage. If this is the case it removes them, before storing the current settings again. Rslidy now creates the JSON structure containing the current settings and uses `JSON.stringify(data)` to turn it into plain text, which is then stored in local storage as an item called `rslidy`. Any errors that could occur here are again logged.

The advantage of this approach is that both JSON parse and stringify, as well as the operations required for accessing the local storage are native JavaScript functions, requiring no third party tools and fully supported by all modern browsers. However, browser permissions might have to be granted in order to access local storage. Downsides to this approach are that settings are only stored locally and are not synced between devices.

```
interface Data {
  slidefont: number;
  uifont: number;
  tilt:  boolean;
  shake: boolean;
  space: boolean;
  margintap: boolean;
  lowlightmode: boolean;
}

loadSettings(): void {
  try {
    var item = localStorage.getItem("rslidy");
  } catch(e) {
    console.log(e);
  }
  if (item === null || item === undefined) return;

  var data: Data = JSON.parse(item);
  if(data.slidefont != undefined)
    while(data.slidefont != this.slidefont)
      this.changeSlideFont(null, data.slidefont > 0 ? 1 : -1);
  if(data.uifont != undefined)
    while(data.uifont != this.uifont)
      this.changeUIFont(null, data.uifont > 0 ? 1 : -1);
  this.view.querySelector("#rslidy-checkbox-tilt").checked = data.tilt;
  this.view.querySelector("#rslidy-checkbox-shake").checked = data.shake;
  this.view.querySelector("#rslidy-checkbox-space").checked = data.space;
  this.view.querySelector("#rslidy-checkbox-margintap").checked = data.margintap;
  this.view.querySelector("#rslidy-checkbox-lowlightmode").checked = data.
      lowlightmode;
  if (data.lowlightmode) window.rslidy.toggleLowLightMode();
}

public saveSettings(): void {
  try {
    localStorage.removeItem("rslidy");
    localStorage.setItem("rslidy", this.generateJSON());
  } catch(e) {
    console.log(e);
  }
}

generateJSON(): string {
  const data: Data = {
    slidefont: this.slidefont,
    uifont: this.uifont,
    tilt: this.view.querySelector("#rslidy-checkbox-tilt").checked,
    shake: this.view.querySelector("#rslidy-checkbox-shake").checked,
    space: this.view.querySelector("#rslidy-checkbox-space").checked,
    margintap: this.view.querySelector("#rslidy-checkbox-margintap").checked,
    lowlightmode: this.view.querySelector("#rslidy-checkbox-lowlightmode").checked
  }
  return JSON.stringify(data);
}
```

Listing 6.2: Settings in Rslidy are persisted between sessions. A data interface is used in combination with JSON to store and retrieve the respective settings in the browser's local storage.

6.3 Accessibility

Most slide content is static and can thus easily be made accessible by providing the corresponding meta information, such as alternative text for for images. However, Rslidy's buttons and checkboxes can change state as the interface is used. For better accessibility, these state changes are reflected by dynamically adjusting corresponding ARIA properties.

For buttons, this can easily be accomplished by adding a few lines of code to the function associated with each button. For example, in the case of the Table of Contents Panel, the following lines are added:

```
let visible = document.getElementById("rslidy-overview-toc")
  .classList.contains("rslidy-overview-visible");
document.getElementById("rslidy-button-toc")
  .setAttribute("aria-expanded", String(visible));
```

First, a boolean called `visible` is set to true when the Table of Contents Panel is visible on screen, and is set to false when it is not visible. This can easily be checked by probing the CSS class list. Then, the attribute `aria-expanded` is set to the corresponding value. Similar code is used for other button states within Rslidy.

For checkboxes, the process is somewhat more complicated, since they do not directly call a function when pressed. Rather, their values are checked in different parts of the code as needed. To solve this problem the code shown in Listing 6.3 is used. A query selector with the wildcard expression `label[id *="checkbox"]` is used to gather all of Rslidy's checkboxes at once. Then, in a for loop iterating over all gathered elements, the initial ARIA attributes of the checkboxes are set and event listeners for `change` and `keyup` events are defined. The latter makes it possible to toggle checkboxes with the Space or Return ⏎ keys. The `change` event updates the `aria-checked` attribute accordingly.

```
setTabindexAndCallbacks() {
  var labels = document.querySelectorAll(`label[id*="checkbox"]`);

  for(let i = 0; i<labels.length; i++) {
    let l = labels[i];
    l.setAttribute("tabindex","0");
    var input = (<HTMLInputElement>l.querySelector("input"));
    l.setAttribute("role","checkbox");
    l.setAttribute("aria-checked", String(input.checked));
    input.addEventListener("change", function() {
      l.setAttribute("aria-checked", String(this.checked));
    });
    l.addEventListener("keyup", e=>this.checkboxCallback(e,l));
  }
}

checkboxCallback(e: any, l: Element) {
  if(l != document.activeElement)
    return;
  var key: number = e.keyCode ? e.keyCode : e.which;

  if(key == 13 || key == 32) { //return and space
    var i = (<HTMLInputElement>l.querySelector("input"));
    if(!i.disabled) {
      i.checked = !i.checked;
      l.setAttribute("aria-checked", String(i.checked));
      i.dispatchEvent(new Event("click"));
    }
  }
}
```

Listing 6.3: To make checkboxes fully accessible, keyup and change event listeners are added. Checkboxes in Rslidy can be toggled with the Space or Return ⟵ keys.

Chapter 7

Testing

Rslidy was put through three different kinds of testing: feature testing, device testing, and user testing. After collecting the test feedback, the software was adapted accordingly.

7.1 Feature Testing

Feature testing was conducted manually to ensure Rslidy's features work properly with different use cases. For this purpose, several example slide decks were created. They can be found in the `src/tests` folder and deal with the following features:

- *Animations*: Four slide decks ensure the different slide transitions work properly. They are `animation-fade.html`, `animation-slidein.html`, `animation-zoom.html`, and `unanimated.html`.
- *Event Listeners*: The slide deck `events.html` ensures the slide creator can set event listeners from slides to execute live code examples.
- *Stress Tests*: These slide decks deal with outliers, such as unusually long, nested, or empty slide decks. They are `long-slides.html`, `many-slides.html`, `sections.html`, `nested-sections.html`, and `no-slides.html`.
- *Device Orientation*: The slide deck `orientation.html` offers a live visualisation of the current device orientation and was used to test tilt and tip gestures.
- *Experimental Features*: Some experimental features, such as speaker notes, can be tested using `notes.html`.
- *Overrides*: The slide deck `overrides.html` ensures that custom settings are handled properly. For more information see Section B.8.

These feature tests, along with the example slide decks, should be reviewed whenever changes to the code are made, to check that nothing was broken in the code update.

7.2 Device Testing

Rslidy was tested on a wide variety of different devices and operating systems. In addition to laptops and personal computers, the following mobile devices were used:

- *OnePlus 5T*: An Android phone with a 6″ display and 1080×2160 pixel (18:9 ratio) resolution. The Android version used was 8.1.0. [GSMA 2021b]

- *Samsung Galaxy S III I9300*: An older Android phone with a 4.8″ display and 720×1280 pixel (16:9 ratio) resolution. The Android version used was 4.1.0. [GSMA 2021c]
- *Apple iPad Mini 2*: An iOS tablet with a 7.9″ display and 1536×2048 pixel (4:3 ratio) resolution. The latest iPadOS version was used (12.4.6). [GSMA 2021a]

Particular attention was paid to touch interaction and motion events on these devices.

7.3 User Testing

A thinking aloud test with six participants was performed by a group of students [Droisner and Korotaj 2019] as part of a project for the course Information Architecture and Web Usability [Andrews 2020]. The test environment is shown in Figure 7.1. Three of the test users used Firefox on a Windows 10 laptop, the other three used Safari on an iPad mini running iOS 12. The participants were asked to perform six tasks of increasing difficulty within a total time frame of twenty minutes:

1. Please open the slide deck and spend a few minutes looking around the presentation.
2. Please change the size of the text.
3. Please read the presentation from start to end.
4. Take a closer look at the diagram on Slide 12.
5. Explore the interface's options and possibilities.
6. Print the slides to a PDF and save it locally.

Afterwards, a short interview was conducted and a feedback questionnaire was filled out by each participant.

Most users praised the simple, easy-to-use design of Rslidy. However, a number of improvements were also suggested. Firstly, the Help Button was hidden inside the Settings Panel and participants said it was hard to access. This was fixed by making it an individual button on the toolbar. Secondly, some panels remained open and would not close automatically when the participants expected them to close, for example by opening another panel or advancing the slides. This was also fixed. Some crashes, however, could not be fully reproduced. It is suspected the web connection was lost during one of the tests.

In addition to the thinking aloud tests, Christopher Kopel, a visually impaired student, gave valuable feedback in terms of the accessibility of Rslidy in combination with the screen readers JAWS [FS 2021] and NVDA [NVA 2019]. After his feedback, several ARIA roles and properties and keyboard interactions were incorporated into Rslidy. For additional information see Section A.8.

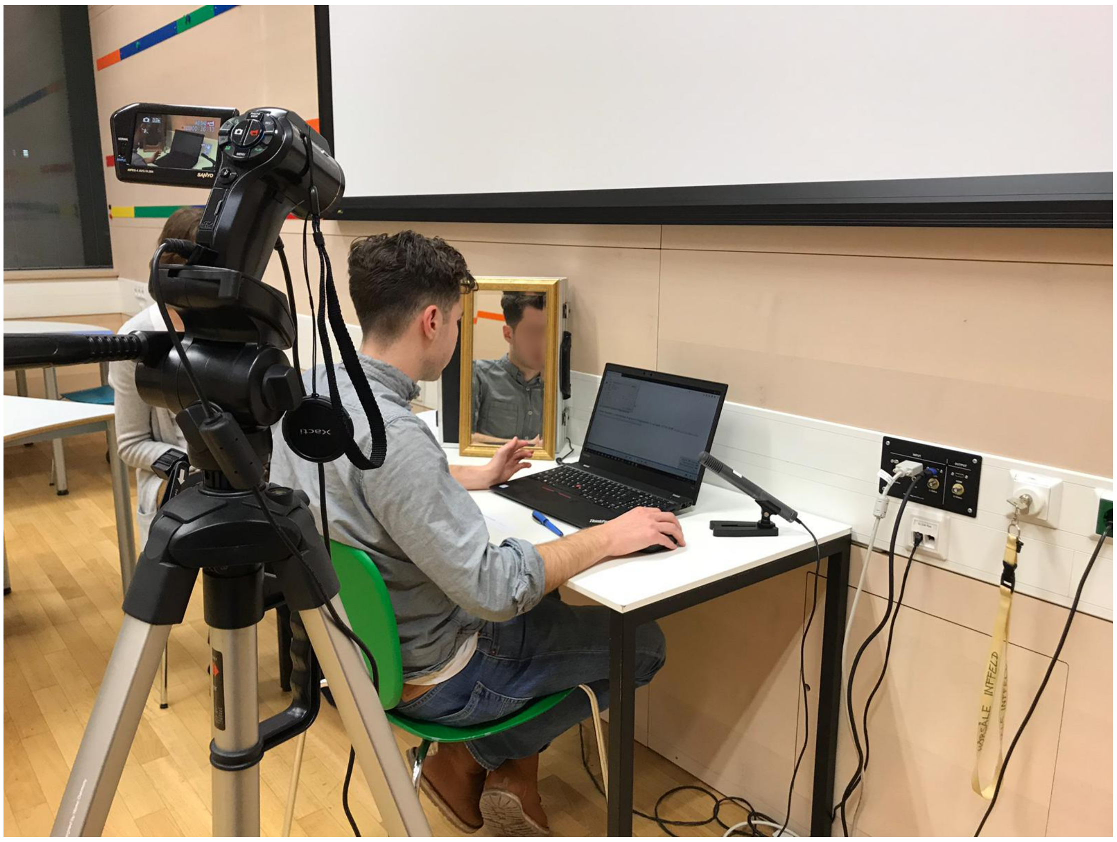


Figure 7.1: The thinking aloud test was captured both by external camera and screen recording. A mirror was used to capture the facial expressions of the participant. [Photograph etxracted from Droisner and Korotaj [2019] and used under the terms of a Creative Commons Attribution 4.0 License). The face of the test user has been deliberately blurred.]

Chapter 8

Future Work

This chapter explains some of the current shortcomings of Rslidy and considers some possible future enhancements. Some potential features were not able to be implemented yet due to lack of universal browser support, or because they did not fit the current scope of the project.

8.1 Remote Control Support

Support for navigation by remote control might be possible within Rslidy's current implementation, as long as the remote allows for mapping of specific buttons to keyboard keys. However, support for proprietary remote control software has not been considered so far.

8.2 Screen Wake Lock

Currently only supported in Chrome, the Screen Wake Lock API [LePage and Steiner 2021] provides a way for browsers to urge the operating system to avoid locking or dimming the screen. Using this, presenters using Rslidy slide decks on laptops or mobile devices could keep the screen awake without having to alter their device's power saving settings.

8.3 Speaker View

A speaker view with notes and a timer was considered during development. However, since modern browsers do not encourage popups, the functionality was left incomplete. Possible solutions could include running a local hosted instance and communication via API requests, however this would go against the minimalist philosophy of Rslidy and would require installation of third party tools for the end user. Speaker notes can be still accessed in one of the example test slide decks called `notes.html` by pressing the N key. Code for a functional timer is also present within Rslidy's code. A dedicated view for the presenter, with a preview of the next slide, was not yet implemented.

8.4 Container Queries

Container Queries are a new proposed CSS standard by Ethan Marcotte [Marcotte 2017]. They function similar to media queries, which allow the creation of CSS rules depending on the viewport or browser dimensions. However, with container queries every individual HTML container could be queried in the same way. An example for how to use them can be seen in Listing 8.1. Using container queries would allow for more flexible, responsive, and readable style sheets in Rslidy. There is currently some support for container queries in Chrome Canary if enabled via the `#enable-container-queries` flag.

```
.container:(max-width: 50em) {
  .links {
    display: none;
  }

  p {
    font-size: 120%;
  }
}
```

Listing 8.1: For containers less than 50 em wide, hide the links and increase the font size.

8.5 Web Speech API

The Web Speech API [Mozilla 2021] could prove useful for two accessibility use cases: issuing voice commands to interact with Rslidy, and providing live subtitles while giving a presentation. However, this feature has not yet been fully supported by every browser and requires sending voice data to a speech recognition engine.

8.6 Accessibility Object Model

The Accessibility Object Model (AOM) [WICG 2020] is a proposed standard for better handling accessibility in browsers. It mainly deals with five use cases:

1. Default accessibility components, which help keep the DOM tidy and do not need to be filled with ARIA roles and properties.
2. Relationships between ARIA elements without having to specify IDs, as this can also cause the DOM to grow exponentially.
3. Allowing events from external software or from assistive technology, such as screen readers.
4. Support for virtual nodes within the accessibility tree. These can be used to add accessibility options on top of otherwise inaccessible UI elements built with `<canvas>` elements.
5. Native developer support for probing and inspecting the Accessibility Object Model directly within the browser.

Once the AOM becomes an established standard, Rslidy should be adapted to support it.

8.7 Generic Sensor API

The Generic Sensor API [W3C 2021] could provide a better way of handling sensor data within Rslidy to address some of the bugs the current implementation faces. Currently, the gyroscope is not fully accessible via event listeners, and only device orientation events can be used. However, these orientation sensors work by referencing a local point in space below the device and calculating the angular difference between that point and top and bottom end of the device. This causes a problem when the device is held in an upright position, since all three points are aligned on a straight line, a slight movement of the device causes large and inaccurate spikes in orientation data.

The Generic Sensor API provides access to the gyroscope with local angular measurements around the X, Y, and Z axis (shown in Figure 8.1) without the need for a reference point and which take account of screen orientation. A downside of this approach are spikes in energy consumption, as the gyroscope has to

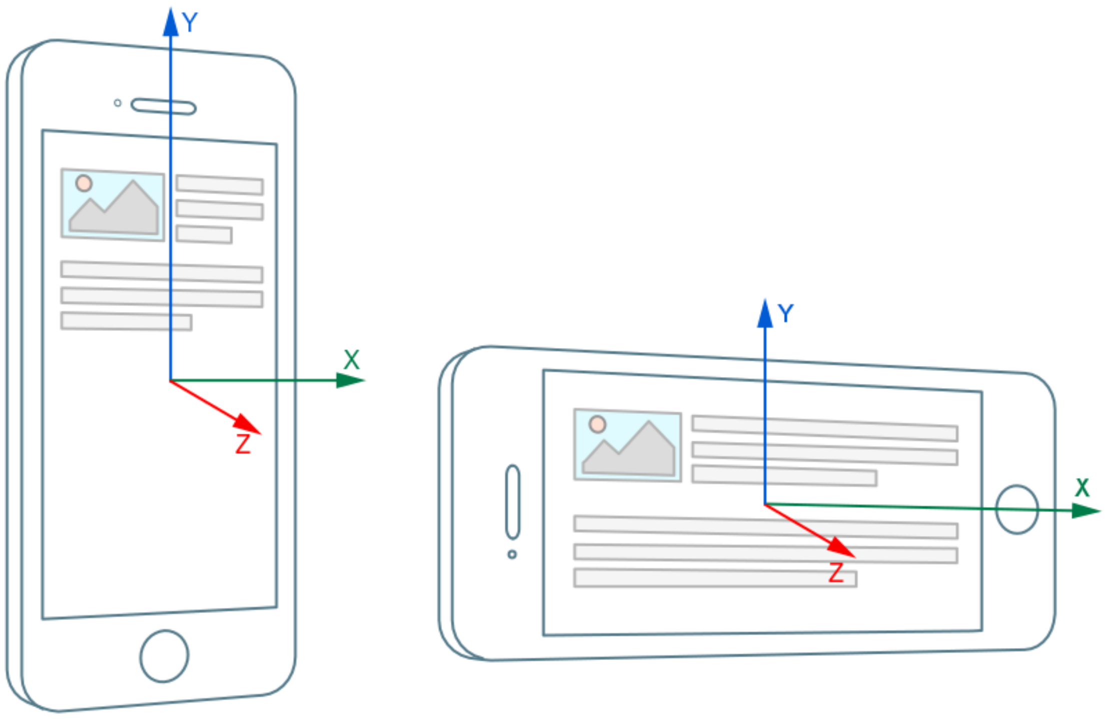


Figure 8.1: The Generic Sensor API provides access to gyroscope sensor readings taking account of screen orientation. [Illustration taken from Shalamov et al. [2021] under the terms of a Creative Commons Attribution 4.0 License.]

oscillate at a relatively high frequency in order to be accurate. There are also privacy concerns associated with access to sensor data. An optimal solution to this problem would be a standardised Gesture API that provides boolean functions for common motion and orientation gestures and thus abstracts the sensor data in such a way that privacy concerns are no longer warranted. The Generic Sensor API is currently not supported by Firefox and Safari.

Chapter 9

Concluding Remarks

Web-based slide decks have become widespread. There are a variety of applications to choose from, each with their own advantages and disadvantages. The trend nowadays is moving towards hosted approaches, where cloud computing and AI can be used to assist the slide creator and personalise the experience via analytics. However, standalone minimalist approaches like Rslidy still have their place and can outperform hosted approaches in certain aspects, such as runtime performance, responsiveness, and usability.

Web browsers are gaining more and more features and can almost rival the capabilities of entire operating systems. A huge effort is being made to make the web as a whole more accessible, as internet access becomes more widespread in developing countries. Wearable technology, such as smart watches, in combination with voice controls may play a bigger role in the future, and slide decks will possibly have to adapt to even smaller screens. Head-mounted displays and virtual meetings could also heavily impact the way presentations are held.

Another aspect which may gain more traction in the near future is the ability to collaborate on slide decks as a group of slide creators. This could be important not only for class projects in a school setting, but also for business meetings in an office setting. Hosted approaches seem to be the primary candidate for this, because they can display a live update in a "what you see is what you get" preview window, but version-control systems such as git can also make this possible with minimalist standalone approaches like Rslidy.

In an office or work settings, presenting is also as much about collecting feedback as it is about sharing your work. So far, this thesis has mostly delved into the technical aspects of slide decks and has not touched upon the importance of interacting with the audience. Donna Spencer [2020] goes over some key points about optimising the presentation workflow in order to maximise concise and relevant feedback. Her five main points are:

1. It is vital to know the audience and give them a definitive role or problem to focus on.
2. Skip the intermediate steps and start with the end result.
3. Show tangible use cases from the user's perspective rather than explaining isolated features.
4. Provide a concrete process for the audience to give their feedback.
5. Follow up on the collected feedback and explain how it was handled.

Combining this approach with the general slide guidelines shown in Section 1.2 will pave the way for a fruitful presentation.

Appendix A

User Guide

Rslidy is a lightweight open-source package supporting accessible and responsive HTML5 slide decks for the web. It is available on GitHub [Hipp and Andrews 2021].

Navigating an Rslidy slide deck on the web should feel familiar to anyone who has used presentation software before, but there are some key differences. In contrast to offline slide decks, which typically have fixed slide dimensions, Rslidy responsively adapts the slide content to the user's screen size. Slides can be scrolled like any regular web page, should the content not fully fit the screen. On devices with a touch screen, an initial long press may be required to initiate horizontal scrolling in cases where horizontal scrolling is unavoidable. In addition to the slide content, the user interface is also adjusted to different screen sizes to ensure that all buttons fit the toolbar. When first visiting an online Rslidy slide deck, a loading spinner might appear briefly. Keyboard shortcuts are provided for devices with a keyboard. Touch and motion gestures are provided for devices which support them. First-time users are advised to peruse the Rslidy Help Panel by pressing the H key or the ? button on the Toolbar.

A.1 Touch Gestures

Various touch gestures can be used to navigate Rslidy. Slides can be navigated either by swiping left or right, or by tapping the left or right slide margin. Multi-touch gestures such as zooming in or out and scrolling are supported as well. However, touch gestures may not be fully supported by every device.

A.2 Keyboard and Mouse Controls

All of Rslidy's on screen buttons can either be clicked or toggled via keyboard shortcuts. In addition, the J key followed by the slide number and Return ↩ can be used to quickly jump to a specific slide. Rslidy and its panels can be navigated with the tabulator ⇆ and Return ↩ keys. The keyboard shortcut Esc can be used to close all open panels. Slide font sizes can be decreased with the - key, reset with the R key, and increased with the + key. If the Shift key is pressed simultaneously, the user interface font sizes are modified instead.

Slides can be navigated by clicking the on-screen buttons, by clicking on the left or right slide margin, or by pressing the ← or → keys. Extra mouse buttons can also be used to navigate Rslidy, if they are mapped to navigate the browser's history.

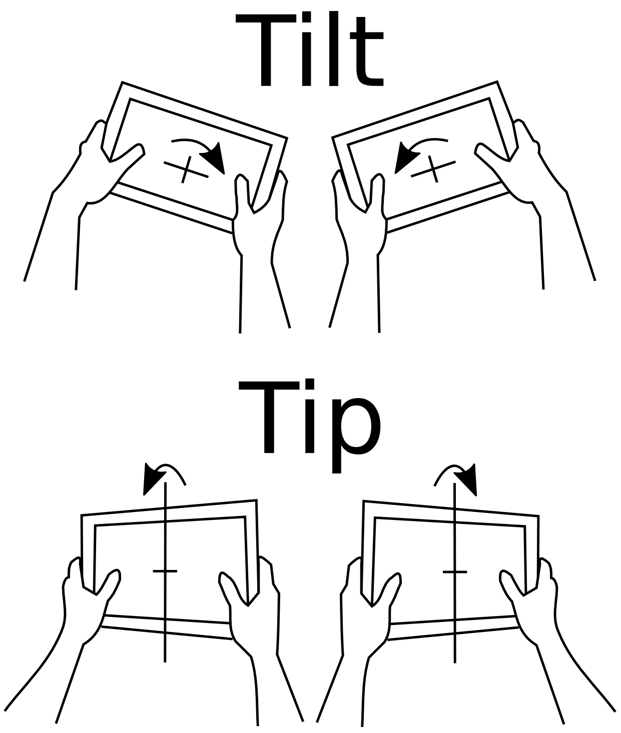


Figure A.1: Tilt and tip gestures can be performed by holding the device with both hands and either tilting or tipping the device towards the left or right side. [Illustration made by the author using Inkscape.]

A.3 Motion and Orientation Gestures

Tilt and tip gestures, illustrated in Figure A.1, can also be used to navigate in Rslidy. On some devices, these gestures may be locked due to potential security concerns. If they are not working properly, it might be necessary to give permissions manually. This can be done by opening Rslidy's Settings menu. Additionally, the shake gesture can be used to navigate to the first slide. To prevent accidental activation, the device needs to be shaken three times consecutively with adequate force in a relatively short time frame.

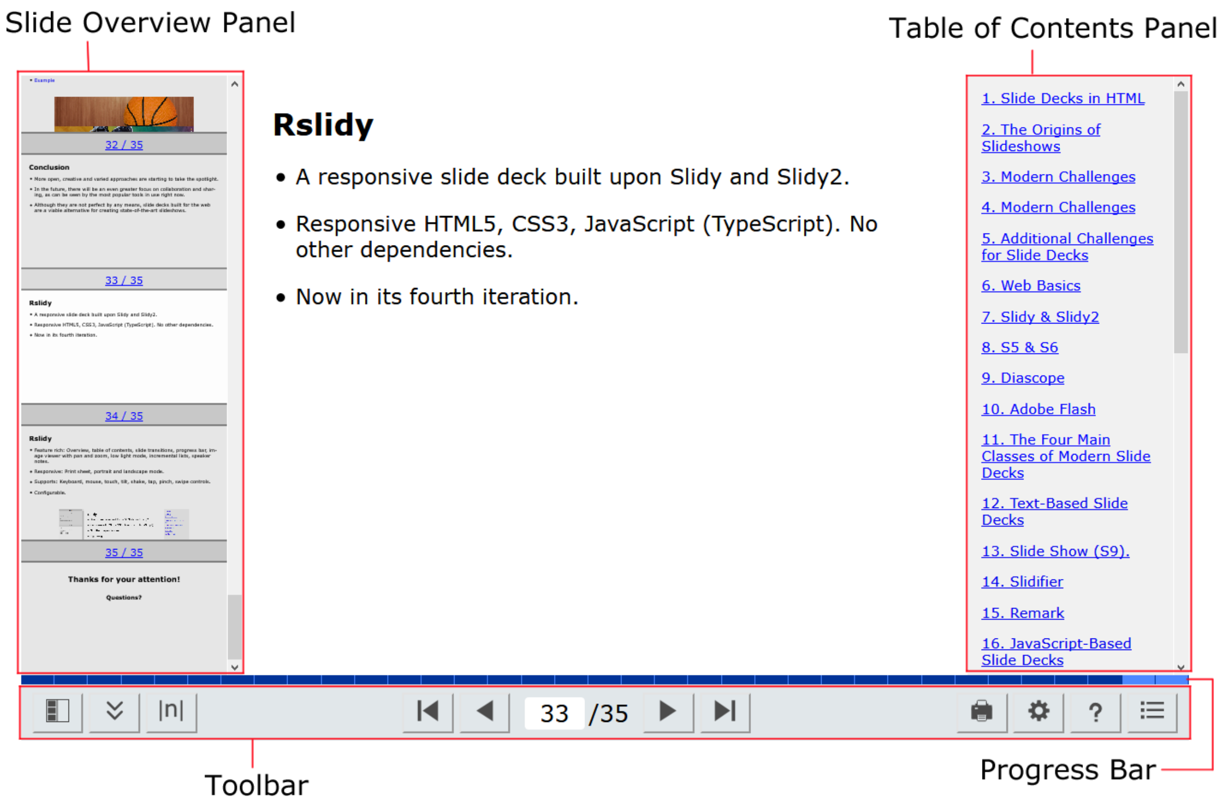
Rslidy adjusts to changes in device orientation automatically, as long as the orientation is not locked by the operating system. For most devices, an upright portrait mode and a sideways landscape mode are supported. The landscape mode can be entered by rotating the device 90 degrees in either left or right direction from the upright position. Additionally, some devices might support an upside down portrait mode.

A.4 Progress Bar

The Progress Bar shows the current slide progression in segments. Previous slides appear in dark blue, while upcoming slides are coloured in light blue. Hovering over a section on the Progress Bar with the mouse pointer shows a slide preview that can be jumped to by clicking it.

A.5 Toolbar

The Toolbar at the bottom of the Rslidy window contains 12 controls, shown in Figure A.2. The following subsections describe them from left to right.



• Example

32 / 35

Conclusion

- More open, creative and varied approaches are starting to take the spotlight.
- In the future, there will be an even greater focus on collaboration and sharing, as can be seen by the most popular tools in use right now.
- Although they are not perfect by any means, slide decks built for the web are a viable alternative for creating state-of-the-art slideshows.

33 / 35

Rslidy

- A responsive slide deck built upon Slidy and Slidy2.
- Responsive HTML5, CSS3, JavaScript (TypeScript). No other dependencies.
- Now in its fourth iteration.

34 / 35

Rslidy

- Feature rich: Overview, table of contents, slide transitions, progress bar, image viewer with pan and zoom, low light mode, incremental lists, speaker notes.
- Responsive: Print sheet, portrait and landscape mode.
- Supports: Keyboard, mouse, touch, tilt, shake, tap, pinch, swipe controls.
- Configurable.

35 / 35

Thanks for your attention!

Questions?

Rslidy

- A responsive slide deck built upon Slidy and Slidy2.
- Responsive HTML5, CSS3, JavaScript (TypeScript). No other dependencies.
- Now in its fourth iteration.

1. Slide Decks in HTML
2. The Origins of Slideshows
3. Modern Challenges
4. Modern Challenges
5. Additional Challenges for Slide Decks
6. Web Basics
7. Slidy & Slidy2
8. S5 & S6
9. Diascope
10. Adobe Flash
11. The Four Main Classes of Modern Slide Decks
12. Text-Based Slide Decks
13. Slide Show (S9).
14. Slidifier
15. Remark
16. JavaScript-Based Slide Decks

Figure A.2: Rslidy features a variety of on-screen buttons in its Toolbar at the bottom of the screen, but can also be controlled through keyboard shortcuts and touch gestures. [Screenshot made by the author from Rslidy.]

A.5.1 Slide Overview Button ◧

The Slide Overview Button ◧ toggles the Slide Overview Panel, which appears on the left-hand side of the slide content to display a sequence of slide thumbnails. The Slide Overview Panel can also be toggled with the O key.

A.5.2 Hide Toolbar Button ⌄

The Hide Toolbar Button ⌄ toggles the visibility of the Toolbar. The Progress Bar is still visible when the Toolbar is hidden. When the Toolbar is hidden, the Show Toolbar Button ⌃ is displayed at the bottom left of the screen (immediately above the Progress Bar) in order to show the Toolbar again. The visibility of the Toolbar can also be toggled with the T key.

A.5.3 Display All Slides Button |n|

The Display All Slides Button |n| toggles the display of all slides as a single web page which can be scrolled through. A frame is drawn around each slide and the slide number is shown in the bottom right corner of each slide. The browser's "Find in page" functionality can be used in order to search for keywords contained in specific slides. The Display Single Slides Button ⊡ is provided to revert to the original single slide layout. This display can also be toggled with the A key.

A.5.4 First Slide Button ⏮

The First Slide Button ⏮ navigates to the first slide of the slideshow. This can also be done by pressing the Home key or with the shake gesture.

A.5.5 Previous Slide Button ◀

The Previous Slide Button ◀ navigates to the previous slide. Four gestures are also provided for this function: swipe right, tap or click the left slide margin, tilt left, and tip left. The corresponding keyboard shortcuts are the ← and Page ↑.

A.5.6 Slide Input Field

The Slide Input Field shows the current slide number, but can also be used to jump to a specific slide by entering its number and pressing Return ↩. Keyboard input is set to the Slide Input Field with the J key or upon pressing any of the number keys.

A.5.7 Next Slide Button ▶

The Next Slide Button ▶ advances to the next slide. Four gestures are also provided for this function: swipe left, tap or click the right slide margin, tilt right, and tip right. The corresponding keyboard shortcuts are the →, Page ↑ and Space (if the option to advance with Space is enabled).

A.5.8 Last Slide Button ⏭

The Last Slide Button ⏭ navigates to the last slide. This can also be done by pressing the End key.

A.5.9 Print Settings Button 🖶

The Print Settings Button 🖶 toggles the Print Settings Panel, shown in Figure A.3a, which can be used to adjust various print-specific options. The Print Settings Panel can also be toggled with the P key.

A.5.10 Settings Button ⚙

The Settings Button ⚙ toggles the Settings Panel, shown in Figure A.3b, which can be used to turn on or off specific aspects of Rslidy. The Settings Panel can also be toggled with the S key.

A.5.11 Help Button ?

The Help Button ? toggles the Help Panel, shown in Figure A.4. The Help Panel displays an overview of the interface controls. The Help Panel can also be toggled with the H key.

A.5.12 Table of Contents Button ☰

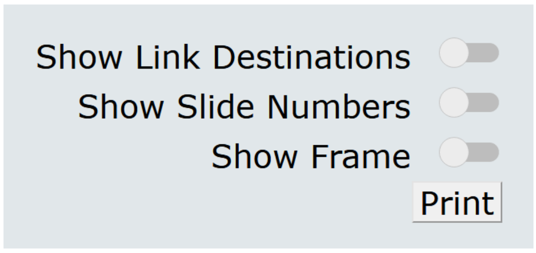
The Table of Contents Button ☰ toggles the Table of Contents Panel, which appears on the right-hand side of the slide content showing a list of slide headings. The Table of Contents Panel can also be toggled with the C key.

A.6 Panels

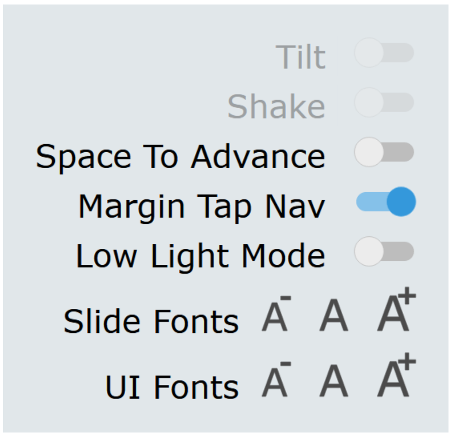
The Rslidy interface has five panels which appear in response to user interaction.

A.6.1 Slide Overview Panel

The Slide Overview Panel, shown in Figure A.2, opens on the left-hand side of the slide content. It displays a sequence of slide thumbnails. Clicking on a slide thumbnail navigates to the corresponding slide. The current slide is highlighted in the overview. For smaller screens, the Slide Overview Panel is limited to occupy at most half of the horizontal screen size.


Show Link Destinations
Show Slide Numbers
Show Frame
Print

(a) Print Settings Panel


Tilt
Shake
Space To Advance
Margin Tap Nav
Low Light Mode
Slide Fonts A- A A+
UI Fonts A- A A+

(b) Settings Panel

Figure A.3: Rslidy's two settings panels. [Screenshots made by the author from Rslidy.]

A.6.2 Table of Contents Panel

The Table of Contents Panel, shown in Figure A.2, opens on the right-hand side of the slide content. It displays a list of slide headings. Clicking on a slide heading navigates to the corresponding slide. Should a slide have no heading, the slide number is shown instead. For smaller screens, the Table of Contents Panel is limited to occupy at most half of the horizontal screen size.

A.6.3 Print Settings Panel

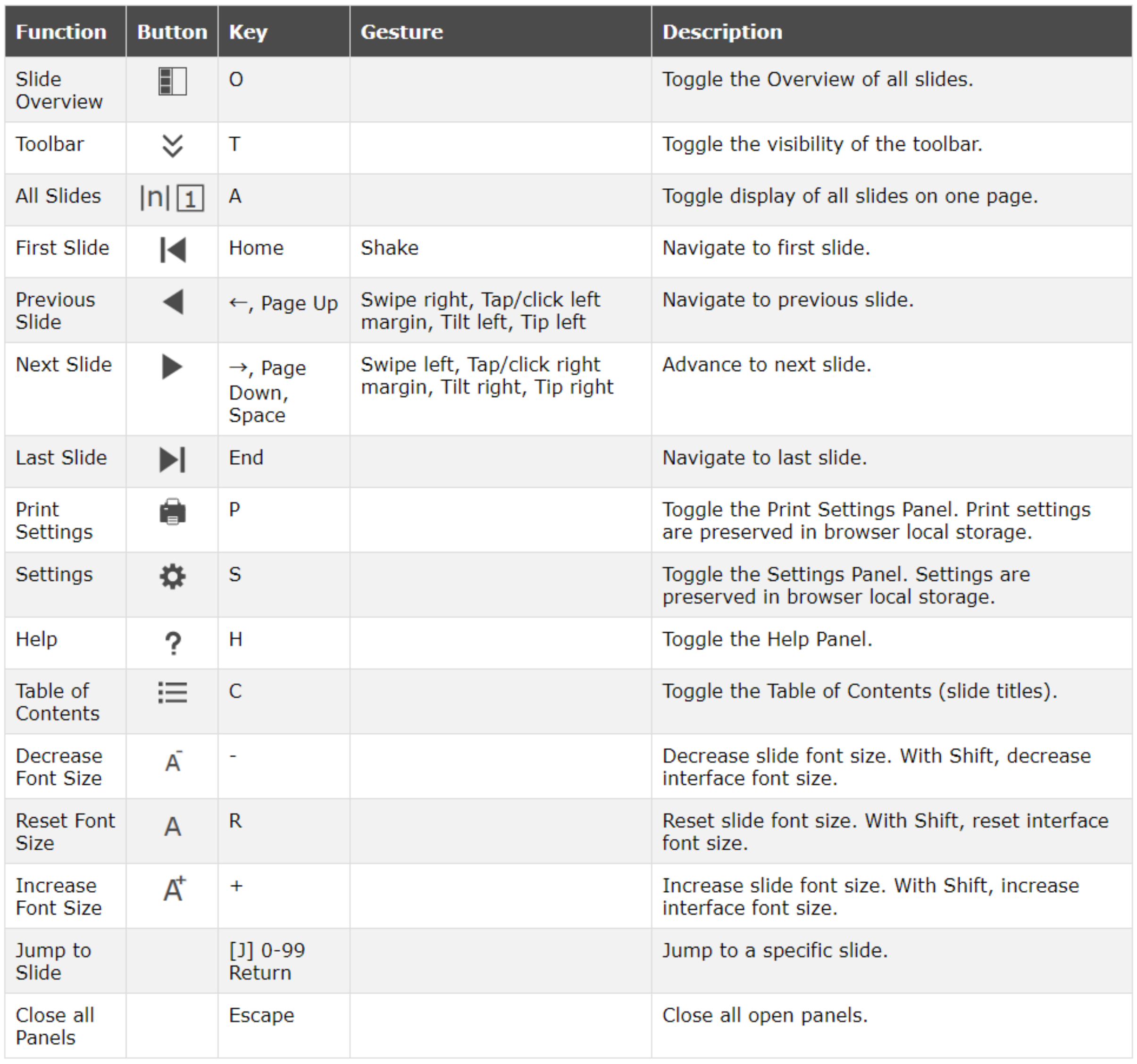
The Print Settings Panel, shown in Figure A.3a, can be used to adjust various print-specific options like the display of link destinations, slide numbers, or the presence of a frame around each slide. Further options, like printing in portrait or landscape mode, pages per sheet, or print margins, are controlled by the browser's print dialogue (as of Firefox Version 85). Print settings are persisted in local storage, if allowed by the browser.

A.6.4 Settings Panel

The Settings Panel, shown in Figure A.3b, can be used to turn on or off specific aspects of Rslidy as well as adjust the slide and interface font sizes. The sliders for tilt and shake controls are disabled unless there is device support for those gestures and special permissions might need to be granted as explained in Section A.3. Navigation aids such as using the space key to advance or margin tap navigation (which allows Rslidy to be navigated like an e-reader) can also be disabled in the Settings Panel. Both of these are enabled by default. Lastly, there is an option for low light mode, which is disabled by default. Settings are persisted in local storage, if allowed by the browser.

A.6.5 Help Panel

The Help Panel, shown in Figure A.4, appears when the Help Button ? or H key is pressed. It displays an overview of Rslidy's interface controls.



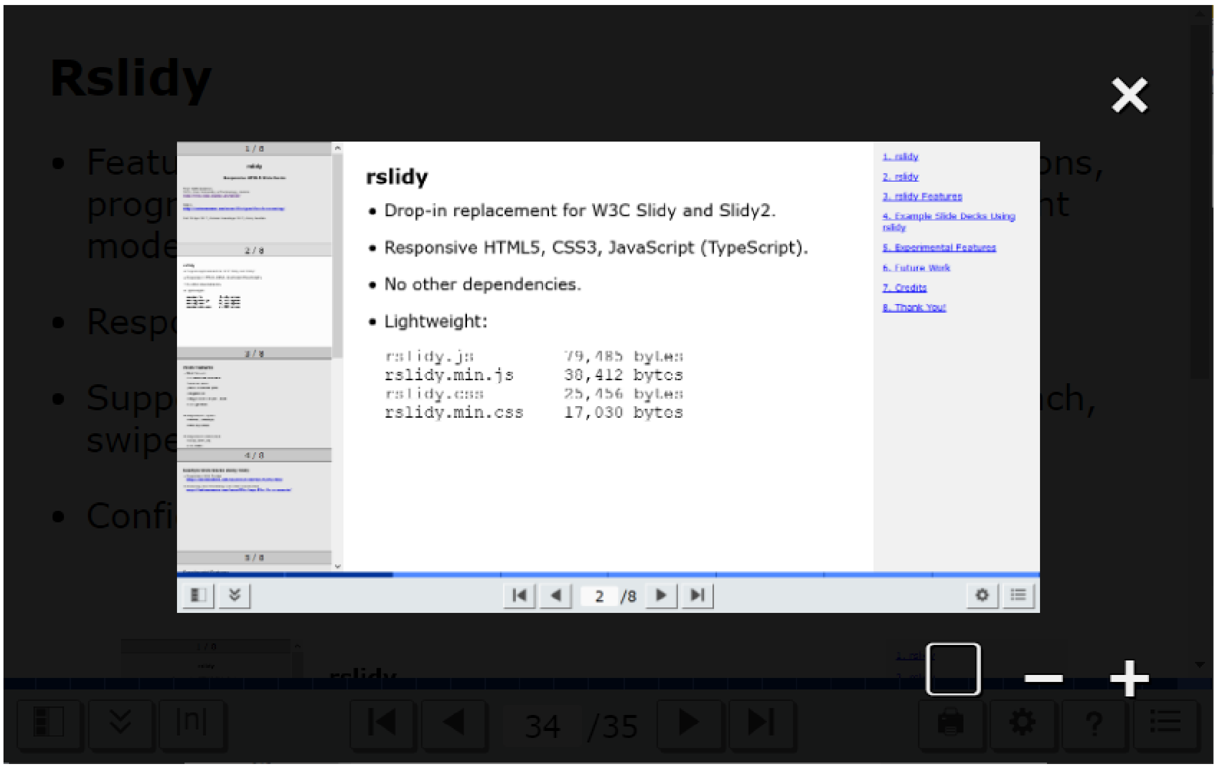
Function	Button	Key	Gesture	Description
Slide Overview	◧	O		Toggle the Overview of all slides.
Toolbar	≚	T		Toggle the visibility of the toolbar.
All Slides	|n| 1	A		Toggle display of all slides on one page.
First Slide	⏮	Home	Shake	Navigate to first slide.
Previous Slide	◀	←, Page Up	Swipe right, Tap/click left margin, Tilt left, Tip left	Navigate to previous slide.
Next Slide	▶	→, Page Down, Space	Swipe left, Tap/click right margin, Tilt right, Tip right	Advance to next slide.
Last Slide	⏭	End		Navigate to last slide.
Print Settings	🖶	P		Toggle the Print Settings Panel. Print settings are preserved in browser local storage.
Settings	⚙	S		Toggle the Settings Panel. Settings are preserved in browser local storage.
Help	?	H		Toggle the Help Panel.
Table of Contents	☰	C		Toggle the Table of Contents (slide titles).
Decrease Font Size	A⁻	-		Decrease slide font size. With Shift, decrease interface font size.
Reset Font Size	A	R		Reset slide font size. With Shift, reset interface font size.
Increase Font Size	A⁺	+		Increase slide font size. With Shift, increase interface font size.
Jump to Slide		[J] 0-99 Return		Jump to a specific slide.
Close all Panels		Escape		Close all open panels.

Figure A.4: The Help Panel displays an overview of the various Rslidy controls and is opened by pressing the H key or the ? button. [Screenshot made by the author from Rslidy.]

A.7 Image Viewer

Multi-touch gestures have become a standard feature of modern operating systems as well as web browsers. For example, pinch gestures can be used to enlarge or shrink an image, which can then be panned by dragging. Since these gestures cannot be used without a touch screen, Rslidy includes an Image Viewer component which replicates this functionality using mouse and keyboard controls. The Image Viewer is automatically attached to every slide image during start-up and is activated when clicking on an image. Its interface can be seen in Figure A.5.

Images can be zoomed in or out with the mouse wheel, via the on-screen + and − buttons, or via the + and − keyboard shortcuts. When using the mouse wheel to zoom, the image is zoomed towards the mouse cursor allowing for easy magnification of specific regions. Images can be panned by dragging with the left mouse button depressed. To reset the position of an image, the □ button or the R key can be used. The Image Viewer can be closed by pressing the × button, the browser's back button, or the Esc key.



rslidy

- Drop-in replacement for W3C Slidy and Slidy2.
- Responsive HTML5, CSS3, JavaScript (TypeScript).
- No other dependencies.
- Lightweight:

```
rslidy.js         79,485 bytes
rslidy.min.js     30,412 bytes
rslidy.css        25,456 bytes
rslidy.min.css    17,030 bytes
```

1. rslidy
2. rslidy
3. rslidy Features
4. Example Slide Decks Using rslidy
5. Experimental Features
6. Future Work
7. Credits
8. Thank You!

Figure A.5: Rslidy's built-in image viewer allows for close-up viewing of images with zooming and panning and is automatically attached to all slide images. [Screenshot made by the author from Rslidy.]

A.8 Accessibility

Rslidy and all of its panels can be navigated with the tabulator key ⇆. The Return ↩ key can be used in place of a mouse click, making Rslidy fully navigable using only a keyboard. The font sizes of both slide content and UI elements are freely adjustable. In addition, ARIA [W3C 2017] roles and properties are used whenever possible. These properties are dynamic and change as the state of Rslidy's elements changes. For example, every slider in the Settings Panel has its current state mapped to an ARIA property. Assistive technology, such as screen readers, can read and interpret the states accordingly. Additionally, special HTML5 elements are used to discern main slide content from navigational elements, so screen reader users can quickly navigate to a specific context.

Appendix B

Slide Creator Guide

Rslidy is a lightweight open-source package supporting accessible and responsive HTML5 slide decks for the web. It is available on GitHub [Hipp and Andrews 2021].

Rslidy is deployed in the form of a JavaScript and a CSS file. These two files need to be included in the slide deck of your choice. Slide creators can choose whether they want to use the minified versions or not. It might be easier to debug potential errors with the unminified versions, but these have larger file sizes. Rslidy's file sizes are shown in Table B.1.

B.1 Overall Structure

To create a presentation in Rslidy, the Rslidy JavaScript and CSS files need to be included in the head of the HTML file:

```
<link rel="stylesheet" href="rslidy.min.css"/>
<script src="rslidy.min.js"/>
```

An example of a basic Rslidy slide deck comprising two slides can be seen in Listing B.1. The HTML content of each slide is placed within either a `<section>` or a `<div class="slide">` element.

Regular	
`rslidy.js`	133,306 bytes
`rslidy.css`	26,086 bytes
Minified	
`rslidy.min.js`	70,223 bytes
`rslidy.min.css`	17,270 bytes
Minified and Gzipped	
`rslidy.min.js.gz`	15,069 bytes
`rslidy.min.css.gz`	4,362 bytes

Table B.1: Rslidy's file sizes for regular, minified, and gzipped minified files.

```html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

<head>
<meta charset="UTF-8"/>

<!-- for css3 media queries to work properly on mobile devices -->
<meta name="viewport" content="width=device-width, initial-scale=1"/>

<title>Rslidy deck</title>

<link rel="stylesheet" href="rslidy.min.css"/>
<script src="rslidy.min.js"></script>
</head>

<body>

<section>
  <h1>Welcome!</h1>
</section>

<section>
  <h1>Slide Heading</h1>
  <ul class="incremental">
    <li>First point</li>
    <li>Second point</li>
    <li>Third point</li>
  </ul>
</section>

</body>
</html>
```

Listing B.1: A basic Rslidy slide deck containing two slides. Slide content can be defined using any standard HTML elements, such as <p> for paragraphs of text, `<ul>` or `<ol>` for lists, `<table>` for tables, and so forth.

```html
<div class="slide">
  <h1>Slide Heading</h1>
  <ul class="incremental">
    <li>First point</li>
    <li>Second point</li>
    <li>Third point</li>
  </ul>
</div>
```

Listing B.2: By adding the class `incremental` to a list, its items will be displayed one at a time.

B.2 Incremental Display of List Items

To make list elements appear one at a time, rather than all at once, the class `incremental` can be applied to the `<ul>` or `<ol>` element, as shown in Listing B.2. To skip over incremental lists, hold the Shift key while navigating.

B.3 Slide Transitions

Rslidy supports four different animations for slide transitions: `slidein` (default), `zoom`, `fade`, and `unanimated`. To enable the desired transition type, the respective class is added to the HTML body:

```
    <body class="unanimated">
```

The specified transition type is applied to all slides in the deck.

B.4 Images

Images can be included like in any other web page. Rslidy's Image Viewer is automatically enabled for each image. This can be disabled by the slide creator, along with numerous other custom settings, as explained in Section B.8.

B.5 Executable Content

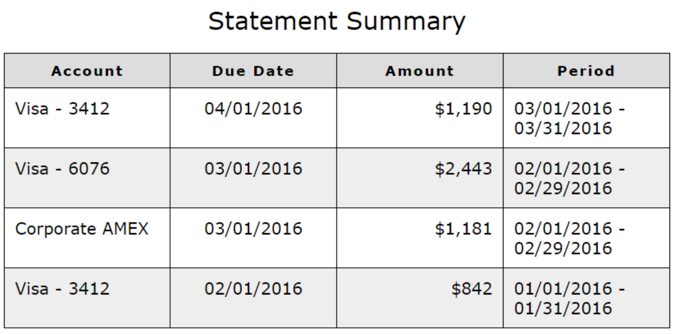
Since Rslidy runs in the browser, it is possible to execute (JavaScript) code at runtime. This can be used, for example, to add support for syntax highlighting in code examples, or to embed other dynamically changing content such as interactive charts or graphics, although these might not show correctly in the slide thumbnails or when printing. An example of syntax highlighting using the library highlight.js [Sagalaev 2019] can be seen in Listing B.3. This library automatically formats code inside nested `<pre>` and `<code>` elements, and provides CSS classes to specify a particular programming language.

```html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

<head>
<meta charset="UTF-8"/>

<!-- for css3 media queries to work properly on mobile devices-->
<meta name="viewport" content="width=device-width, initial-scale=1"/>

<title>Syntax Highlighting</title>

<link rel="stylesheet" href="rslidy.min.css"/>
<script src="rslidy.min.js"></script>

<link rel="stylesheet" href="highlight/default.min.css">
<script src="highlight/highlight.min.js"></script>
<script>hljs.initHighlightingOnLoad();</script>

</head>

<body>

<section>
<h1>Source Code Highlighting</h1>
<pre><code class="html">&lt;body&gt;
  &lt;section&gt;
    &lt;h1&gt;Source Code Highlighting!&lt;/h1&gt;
    &lt;h2&gt;Or through third-party tools.&lt;/h2&gt;
  &lt;/section&gt;
&lt;/body&gt;</code></pre>
</section>

</body>
</html>
```

Listing B.3: Rslidy supports external libraries and runtime (JavaScript) code execution. One application for this is syntax highlighting, achieved here with the highlight.js library.

B.6 Responsive Content

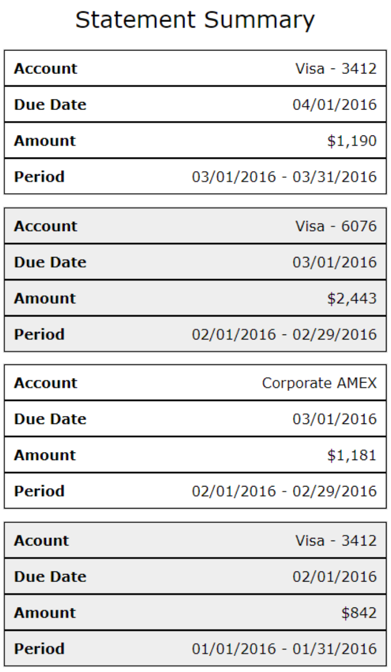
It is the responsibility of the slide creator to ensure that individual slide content is responsive. This might include responsive data tables, responsive charts and figures, and responsive listings. Some best practice examples, like the responsive table shown in Figure B.1, can be found in Rslidy's `examples/slide-types` folder.



Statement Summary

Account	Due Date	Amount	Period
Visa - 3412	04/01/2016	$1,190	03/01/2016 - 03/31/2016
Visa - 6076	03/01/2016	$2,443	02/01/2016 - 02/29/2016
Corporate AMEX	03/01/2016	$1,181	02/01/2016 - 02/29/2016
Visa - 3412	02/01/2016	$842	01/01/2016 - 01/31/2016

(a) Wide Screen



Statement Summary

Account	Visa - 3412
Due Date	04/01/2016
Amount	$1,190
Period	03/01/2016 - 03/31/2016

Account	Visa - 6076
Due Date	03/01/2016
Amount	$2,443
Period	02/01/2016 - 02/29/2016

Account	Corporate AMEX
Due Date	03/01/2016
Amount	$1,181
Period	02/01/2016 - 02/29/2016

Acount	Visa - 3412
Due Date	02/01/2016
Amount	$842
Period	01/01/2016 - 01/31/2016

(b) Narrow Screen

Figure B.1: A responsive data table rendered for two different screen widths. [Screenshots made by the author from Rslidy.]

B.7 Accessible Content

The slide creator is also responsible for ensuring that the content of individual slides is accessible. This can be done in many different ways, a simple example to boost accessibility could include providing descriptive text for charts and figures. Static HTML5 elements, which make up the majority of slide content, should be handled well by most assistive technology. ARIA [W3C 2017] roles and properties can be used to provide additional information wherever needed.

B.8 Custom Settings

Custom settings offer a way of tweaking Rslidy's internal settings. They need to be overridden using JavaScript, as shown in Listing B.4. The custom settings are:

- `image_viewer`: If set to false, disables the image viewer component. The default value is true.
- `close_menu_on_selection`: If set to true, will close menus upon making a selection. The default value is false.
- `close_menu_on_blur`: If set to true, will close menus when losing focus. The default value is true.
- `close_navigation_on_selection`: If set to true, will close the Slide Overview Panel and the Table of Contents Panel upon making a selection. The default value is false.
- `show_slide_dividers`: If set to true, shows slide dividers in the Progress Bar. The default value is true.
- `start_in_low_light_mode`: If set to true, starts Rslidy in low light mode. The default value is false.
- `start_with_toolbar_minimized`: If set to true, starts Rslidy with the Toolbar minimised. The default value is false.

- `block_slide_text_selection`: If set to true, blocks selection of text on slides. The default value is false.

B.9 Advanced Custom Settings

Similarly, advanced custom settings can be used to tweak some of Rslidy's internal variables directly. These include:

- `custom_aspect_ratio`: Specifies a custom aspect ratio for slides in the slide thumbnails. By default, it is set to 4/3 to provide a uniform experience. If set to 0, Rslidy will try to calculate the aspect ratio dynamically. This can cause scaling issues on some screens. If set to a value greater than zero, uses the parameter `custom_width` for scaling.
- `overview_slide_zoom`: Applies a zoom factor to the slide thumbnails in the Slide Overview Panel. Set to 1 to disable.
- `doubletap_delay`: Double tap delay in milliseconds. The default value is 200.
- `min_slide_font`: Sets the minimum font size that can be achieved with Rslidy's internal font scaling commands. The default is 0.1em.
- `max_slide_font`: Sets the maximum font size that can be achieved with Rslidy's internal font scaling commands. The default is 5em.
- `font_step`: Sets the step size with which Rslidy's internal font scaling commands operate. The default is 0.1em.
- `swipe_max_duration`: The maximum duration for the swipe gesture in milliseconds. The default is 400.
- `swipe_threshold`: The distance threshold in rem that needs to be exceeded in order for the gesture to count as a swipe. The default is 3.75rem.
- `swipe_y_limiter`: Sets how many times larger the swiped x distance needs to be than the y distance. If the value is 1, x has to be bigger than y. If the value is 2, x has to be twice as big as y. The default value is 1.
- `shake_sensitivity`: The smaller the number, the tougher it is to trigger the shake event. The default is 0.8.
- `required_shakes`: The number of consecutive shakes required to trigger the shake event. The default is 4.
- `print_frame`: The CSS string for the slide frame when printing. The default value is `0.05rem solid black`.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

<head>
  <meta charset="UTF-8" />

  <!-- for css3 media queries to work properly on mobile devices-->
  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Overrides</title>

  <link rel="stylesheet" href="rslidy.min.css" />
  <script src="rslidy.min.js"></script>
  <script>
    window.rslidy.close_menu_on_selection = true;
    window.rslidy.close_navigation_on_selection = true;
    window.rslidy.start_with_status_bar_minimized = true;
    window.rslidy.image_viewer = false;
    window.rslidy.start_in_low_light_mode = true;
    window.rslidy.block_slide_text_selection = true;
    window.rslidy.show_slide_dividers = false;
  </script>

</head>

<body>
...
</body>

</html>
```

Listing B.4: Some of Rslidy's internal settings can be directly overridden via JavaScript. This should be done in a `<script>` element after including `rslidy.js`.

Appendix C

Developer Guide

Rslidy is a lightweight open-source package supporting accessible and responsive HTML5 slide decks for the web. It is available on GitHub [Hipp and Andrews 2021].

To install and build Rslidy, first install Node [Dahl 2019] and npm [NPM 2018], then run the following commands in the Rslidy folder:

```
npm install
npx gulp
```

The default gulp task `build` will automatically produce minified files and place them into every slide deck found in `src/examples`. The resulting files can be found in the folder `build`. During development, it is advised to use the gulp task `watch`, which will automatically rebuild the project every time a source file is changed and synchronise the output to the browser. A particular slide deck from the `tests` directory can be specified using:

```
npx gulp watch --slide <file>
```

To bundle Rslidy's icon definitions, the gulp task `icons` can be used. The build will also produce a file `rslidy.js.map`, which can be used for debugging in the browser's developer tools in combination with the unminified JavaScript file.

C.1 List of Gulp Tasks

While using the gulp tasks `build` and `watch` may be sufficient for a developer, they themselves are comprised of many smaller tasks and functions. The full list of gulp tasks is as follows:

- `clean`: Deletes temporary build files.
- `transpile`: Transpiles the TypeScript files using the configuration in `tsconfig.json`.
- `minifyjs`: Produces the minified JavaScript files.
- `minifycss`: Produces the minified CSS files.
- `compress`: Compresses the minified files with gzip.
- `svgo`: Optimises the SVG files in the icons folder.
- `icon-definitions`: Takes the optimised SVG files, converts them into strings, and saves them in the file `icon-definitions.ts`. This process also removes stroke and fill colours, since they are set from CSS to save space.

- `icons`: Runs the task `svgo`, then the task `icon-definitions`.
- `webpack`: Bundles the main JavaScript file. Depends on the task `transpile`.
- `scss`: Creates the single CSS file `rslidy.css` from all the SCSS source files.
- `html`: Copies the examples into the `build` folder.
- `copy`: Copies the minified JavaScript file `rslidy.min.js` and the minified CSS file `rslidy.min.css` into every example folder.
- `s9`: Creates the S9 template.
- `minify`: Runs the tasks `minifyjs`, `minifycss`, and `compress`.
- `assemble`: Runs the tasks `webpack`, `scss`, and `html`.
- `build`: Runs the tasks `clean`, `assemble`, `minify`, `copy`, and `s9`.
- `watch`: Runs the task `build`, then runs `build` without clean whenever the source code is modified.

C.2 Development Dependencies

Rslidy has no external runtime dependencies. It is completely self-contained in the two files `rslidy.js` and `rslidy.css`. When developing or building Rslidy, there are numerous development dependencies, listed in the file `package.json` along with their version numbers. These are:

- `Node.js,NPM,andgulp`: Used for the build system.
- `typescript,gulp-typescript`: Used to transpile TypeScript into JavaScript.
- `gulp-sass`: Used to convert SCSS to CSS.
- `webpackandwebpack-stream`: Used to unify the transpiled JavaScript components into a single JavaScript file.
- `browser-sync`: Used for the gulp watch task to update the browser with a newly built version whenever source files have been modified.
- `file-system,del,merge2,gulp-concatandgulp-rename`: Used for file operations.
- `gulp-uglifyes,gulp-uglifycss`: Used to create minified JavaScript and CSS files.
- `gulp-svgo`: Used to optimise SVG files.
- `gulp-gzip`: Used to gzip files, and for checking gzipped file sizes.
- `gulp-util`: Supports logging in the gulp file.
- `yargs`: Used to pass arguments to gulp from the command line.

Bibliography

Adobe [2020]. *Adobe Flash Player EOL General Information Page*. 16 Jun 2020. https://adobe.com/products/flashplayer/end-of-life.html (cited on page 14).

Alley, Michael [2013]. *The Craft of Scientific Presentations*. 2nd Edition. Springer, 30 Apr 2013. 286 pages. ISBN 1441982787 (cited on page 3).

Andrews, Keith [2019]. *Writing a Thesis: Guidelines for Writing a Master's Thesis in Computer Science*. Graz University of Technology, Austria. 24 Jan 2019. http://ftp.iicm.edu/pub/keith/thesis/ (cited on page xiii).

Andrews, Keith [2020]. *Information Architecture and Web Usability*. 21 Oct 2020. https://courses.isds.tugraz.at/iaweb/ (cited on pages 39–40, 43–44, 60).

Apple [2003]. *Apple Unveils Keynote*. Press Release. 07 Jan 2003. https://apple.com/newsroom/2003/01/07Apple-Unveils-Keynote/ (cited on page 1).

Apple [2018]. *Keynote*. 18 Oct 2018. https://apple.com/lae/keynote/ (cited on page 1).

Bang, Ole Petter [2018a]. *Remark Example*. 07 May 2018. https://remarkjs.com (cited on page 19).

Bang, Ole Petter [2018b]. *Remark: A Simple, In-Browser, Markdown-Driven Slideshow Tool*. GitHub. 07 May 2018. https://github.com/gnab/remark (cited on page 18).

Barrett, Brian [2017]. *Adobe Finally Kills Flash Dead*. Wired. 25 Jul 2017. https://wired.com/story/adobe-finally-kills-flash-dead (cited on page 14).

Bauer, Gerald [2011]. *Slide Show (S9) Guide*. 09 Jul 2011. https://slideshow-s9.github.io/ (cited on page 15).

Bauer, Gerald [2013]. *S6 – Slide Show Templates Using HTML5, CSS3 'n' JavaScript*. 01 Jul 2013. https://slidekit.github.io/ (cited on page 13).

Beautiful.ai [2018a]. *Beautiful.ai - Pricing*. 06 Feb 2018. https://beautiful.ai/pricing/ (cited on page 33).

Beautiful.ai [2018b]. *Beautiful.ai - The First Presentation Maker That Designs for You*. 06 Feb 2018. https://beautiful.ai/ (cited on pages 33–34).

Belevich, Kirill [2019]. *SVG Optimizer*. GitHub. 30 Oct 2019. https://github.com/svg/svgo (cited on page 49).

Bostock, Mike [2015a]. *Stack*. GitHub. 19 Mar 2015. https://github.com/mbostock/stack (cited on page 35).

Bostock, Mike [2015b]. *Stack Example*. GitHub. 19 Mar 2015. https://mbostock.github.io/stack (cited on page 35).

Brock, David C. [2017]. *The Improbable Origins of Powerpoint*. IEEE Spectrum 54.11 (02 Nov 2017), pages 42–49. doi:10.1109/mspec.2017.8093800 (cited on page 1).

Cederholm, Dan [2013]. *Sass For Web Designers*. A Book Apart, 13 Nov 2013. 97 pages. ISBN 193755712X (cited on page 9).

Cederholm, Dan [2015]. *CSS3 for Web Designers*. 2nd Edition. A Book Apart, 24 Feb 2015. 139 pages. ISBN 1937557200 (cited on pages 7, 41).

Chun, Russell [2014]. *Adobe Flash Professional CC*. Classroom in a Book. 11 Aug 2014. 384 pages. ISBN 0133927105 (cited on page 14).

Clover, Juli [2017]. *Apple Makes iMovie, GarageBand, and iWork Apps for Mac and iOS Free for All Users*. MacRumors. 18 Apr 2017. https://macrumors.com/2017/04/18/apple-imovie-garageband-iwork-free-for-all-users/ (cited on page 1).

Dahl, Ryan [2019]. *Node.js*. 15 Aug 2019. https://nodejs.org/ (cited on pages 9, 85).

Dalgleish, Mark [2015]. *Fathom.js*. GitHub. 06 Aug 2015. https://github.com/markdalgleish/fathom (cited on page 20).

Dalgleish, Mark [2018a]. *Bespoke.js*. GitHub. 18 Jan 2018. https://github.com/bespokejs/bespoke (cited on page 21).

Dalgleish, Mark [2018b]. *Bespoke.js Example*. 18 Jan 2018. http://markdalgleish.com/projects/bespoke.js (cited on page 24).

Doumont, Jean-luc [2002]. *The Three Laws of Professional Communication*. IEEE Transactions on Professional Communication 45.4 (Dec 2002), pages 291–296. ISSN 0361-1434. doi:10.1109/TPC.2002.805164. http://todroberts.com/USF/3_laws_com.pdf (cited on page 3).

Doumont, Jean-luc [2005]. *The Cognitive Style of PowerPoint: Slides Are Not All Evil*. Technical Communication 52.1 (01 Feb 2005), pages 64–70. ISSN 0049-3155. https://web.mit.edu/5.95/readings/doumont-responds-to-tufte.pdf (cited on page 3).

Doumont, Jean-luc [2009]. *Trees, Maps, and Theorems*. Principiae, Jan 2009. 178 pages. ISBN 9081367706. http://treesmapsandtheorems.com/ (cited on page 3).

Droisner, Angelika and Ana Korotaj [2019]. *Thinking Aloud Test Report: Usability Test of rslidy Responsive HTML5 Slides*. Unpublished class project. 19 Jan 2019 (cited on pages xiii, 60–61).

Dzielak, Josh [2019]. *MDX, Authors and Richer JAMstack Content*. 13 Jun 2019. https://mdx-talk.developermode.com/ (cited on page 29).

Eibl, Thomas, Michael Glatzhofer, Christoph Heidenreich, and Verena Schiffer [2018]. *Rslidy: Responsive HTML5 Slide Decks*. Unpublished class project. 05 Feb 2018 (cited on pages 44–45).

Friend, James [2018]. *PCE.js – Emulates Mac Plus, PC, & Atari ST in asm.js*. GitHub. 24 Nov 2018. https://github.com/jsdf/pce (cited on page 2).

FS [2021]. *JAWS - Job Access With Speech*. Freedom Scientific, 19 Jan 2021. https://freedomscientific.com/products/software/jaws/ (cited on page 60).

Gabrielle, Bruce R. [2010]. *Speaking PowerPoint: The New Language of Business*. Insights Publishing, 10 Oct 2010. ISBN 098423604X (cited on page 3).

Garolla, Filippo, Sabine Lukas, Matthias Schlesinger, and Karin Wilding [2015]. *rSlidy2*. GitHub. 22 Jan 2015. https://github.com/GPhilo/rSlidy2 (cited on pages 40–41).

Gaskins, Robert [2012]. *Sweating Bullets: Notes about Inventing PowerPoint*. Vinland Books, 12 Apr 2012. ISBN 0985142421. https://robertgaskins.com/powerpoint-history/sweating-bullets/gaskins-sweating-bullets-webpdf-isbn-9780985142414.pdf (cited on page 1).

Google [2019]. *Google Slides*. 26 Nov 2019. https://google.com/slides/about (cited on pages 30, 51).

Google [2020]. *Material Components*. GitHub. 29 Dec 2020. `https://github.com/material-components` (cited on page 44).

Gruber, John [2004]. *Markdown*. 17 Dec 2004. `https://daringfireball.net/projects/markdown` (cited on page 15).

Grunt [2020]. *Grunt: The JavaScript Task Runner*. 03 Jul 2020. `https://gruntjs.com/` (cited on page 41).

GSMA [2021a]. *Apple iPad Mini 2 Specification*. GSM Arena. `https://gsmarena.com/apple_ipad_mini_2-5735.php` (cited on page 60).

GSMA [2021b]. *OnePlus 5T Specification*. GSM Arena. `https://gsmarena.com/oneplus_5t-8912.php` (cited on page 59).

GSMA [2021c]. *Samsung Galaxy S III Specification*. GSM Arena. `https://gsmarena.com/samsung_i9300_galaxy_s_iii-4238.php` (cited on page 60).

Halácsy, Péter, Peter Arvai, and Adam Somlai-Fischer [2019a]. *Prezi – Online Presentation Tools*. 18 Nov 2019. `https://prezi.com/` (cited on page 30).

Halácsy, Péter, Peter Arvai, and Adam Somlai-Fischer [2019b]. *Prezi – Pricing*. 18 Nov 2019. `https://prezi.com/pricing` (cited on page 32).

Hammer.js [2016]. *hammer.js: A JavaScript Library for Detecting Touch Gestures*. GitHub. 22 Apr 2016. `https://github.com/hammerjs/hammer.js` (cited on pages 39–40).

Hattab, Hakim El [2019a]. *Reveal.js*. GitHub. 01 Apr 2019. `https://github.com/hakimel/reveal.js` (cited on page 23).

Hattab, Hakim El [2019b]. *Reveal.js – The HTML Presentation Framework*. 01 Apr 2019. `https://revealjs.com` (cited on page 26).

Hipp, Patrick [2019]. *Slide Decks in HTML*. 25 Jan 2019. `https://ftp.isds.tugraz.at/pub/surveys/hipp-2019-01-25-survey-slide-decks.pdf` (cited on page 11).

Hipp, Patrick and Keith Andrews [2021]. *Rslidy Repository*. 26 Jul 2021. `https://github.com/tugraz-isds/rslidy` (cited on pages 5, 39, 47, 69, 77, 85).

Jackson, Brent [2020]. *MDX Deck*. GitHub. 10 Dec 2020. `https://github.com/jxnblk/mdx-deck` (cited on page 29).

Jehl, Scott [2014]. *Responsible Responsive Design*. A Book Apart, 19 Nov 2014. 188 pages. ISBN 1937557162 (cited on page 10).

Jobs, Steve [2010]. *Thoughts on Flash*. Apple. Apr 2010. `https://apple.com/hotnews/thoughts-on-flash` (cited on page 14).

JSON [1999]. *JavaScript Object Notation*. 01 Dec 1999. `https://json.org/json-en.html` (cited on page 54).

Kasper, Patrick, Clemens Meinhart, Karina Priebernig, and Elias Zeitfogel [2014]. *rSlidy*. Unpublished class project. 21 Jan 2014 (cited on pages 39–40).

Keith, Jeremy and Rachel Andrew [2016]. *HTML5 for Web Designers*. 2nd Edition. A Book Apart, 17 Feb 2016. 92 pages. ISBN 1937557243 (cited on page 7).

Kogovšek, Rok, Alexei Kruglov, Fernando Pulido Ruiz, and Helmut Zöhrer [2017]. *Animated rSlidy*. GitHub. 06 Feb 2017. `https://github.com/kogi18/IAWEB` (cited on pages 43–44).

LePage, Pete and Thomas Steiner [2021]. *Stay Awake with the Screen Wake Lock API*. 23 Feb 2021. `https://web.dev/wake-lock/` (cited on page 63).

Lindley, Cody [2013]. *DOM Enlightenment: Exploring JavaScript and the Modern DOM*. O'Reilly Media, 03 Mar 2013. 180 pages. ISBN 1449342841. http://domenlightenment.com/ (cited on pages 7, 51).

Lindley, Cody [2019]. *Front-End Developer Handbook 2019*. GitHub. 19 Aug 2019. https://github.com/FrontendMasters/front-end-handbook-2019 (cited on page 7).

Ludvigsen, Holger [2016a]. *Slidifier*. GitHub. 27 Jan 2016. https://github.com/holgerl/Slidifier (cited on page 18).

Ludvigsen, Holger [2016b]. *Slidifier Example*. 27 Jan 2016. http://slidifier.com/slidifier.html (cited on page 17).

Mahém, Luke, Marcin Wichary, and Eric Bidelman [2013]. *Google HTML5 Slides*. 08 Apr 2013. https://code.google.com/archive/p/io-2013-slides (cited on page 20).

Makeev, Vadim [2018]. *Shower Presentation Engine*. 08 Oct 2018. https://shwr.me (cited on page 27).

Makeev, Vadim [2019]. *Shower*. GitHub. 22 Nov 2019. https://github.com/shower/shower (cited on page 27).

Marcotte, Ethan [2014]. *Responsive Web Design*. 2nd Edition. A Book Apart, 02 Dec 2014. ISBN 1937557189 (cited on page 10).

Marcotte, Ethan [2015]. *Responsive Design: Patterns & Principles*. A Book Apart, 18 Nov 2015. 160 pages. ISBN 1937557332 (cited on page 10).

Marcotte, Ethan [2017]. *On Container Queries*. 01 Mar 2017. https://ethanmarcotte.com/wrote/on-container-queries/ (cited on page 63).

Marquis, Mat [2016]. *JavaScript For Web Designers*. A Book Apart, 28 Sep 2016. 135 pages. ISBN 1937557464 (cited on page 7).

Meyer, Eric [2005a]. *S5: A Simple Standards-Based Slide Show System*. 28 Jul 2005. https://meyerweb.com/eric/tools/s5 (cited on page 13).

Meyer, Eric [2005b]. *S5: An Introduction*. 28 Jul 2005. https://meyerweb.com/eric/tools/s5/s5-intro.html (cited on page 14).

Microsoft [2019]. *TypeScript*. 06 Nov 2019. https://typescriptlang.org/ (cited on pages 9, 41).

Mozilla [2021]. *Using the Web Speech API*. 25 Jan 2021. https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API (cited on page 64).

Nakajima, Pat and Dan Croak [2012]. *Slidedown*. GitHub. 16 Feb 2012. https://github.com/nakajima/slidedown (cited on pages 16–17).

Netlify [2019]. *All-in-One Platform For Automating Modern Web Projects*. 22 Nov 2019. https://app.netlify.com/start/deploy?repository=https://github.com/shower/shower (cited on page 27).

Nielsen, Jakob [1994]. *10 Usability Heuristics for User Interface Design*. 24 Apr 1994. https://nngroup.com/articles/ten-usability-heuristics/ (cited on page 47).

NPM [2018]. *NPM Presentation Packages*. 29 Nov 2018. https://npmjs.com/search?q=presentation (cited on pages 9, 20, 85).

NVA [2019]. *NVDA*. NV Access, 01 Jan 2019. https://nvaccess.org/about-nvda/ (cited on page 60).

Pattis, Richard E. [2004]. *American Standard Code*. 31 Jul 2004. https://cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html (cited on page 15).

Phi, Jay [2018]. *The Origins of PowerPoint*. 30 Sep 2018. https://modernslide.com/the-origins-of-powerpoint/ (cited on page 1).

Pickering, Heydon [2016]. *Inclusive Design Patterns*. Smashing Magazine, 16 Oct 2016. 271 pages. ISBN 3945749433 (cited on page 10).

Polacek, John [2015a]. *jQuery Scrolldeck Parallax Plugin Demo*. GitHub. 23 Dec 2015. https://johnpolacek.github.io/scrolldeck.js/decks/parallax (cited on page 37).

Polacek, John [2015b]. *Scrolldeck.js*. GitHub. 23 Dec 2015. https://github.com/johnpolacek/scrolldeck.js (cited on page 35).

Prezibase [2015]. *Climb to Success – Prezi Template*. 13 Dec 2015. https://prezi.com/koluwfk7mcq_/climb-to-success-prezi-template (cited on page 31).

Puppetlabs [2019]. *Showoff*. GitHub. 23 Feb 2019. https://github.com/puppetlabs/showoff (cited on pages xiii, 32–33).

Raggett, Dave [2005]. *HTML Slidy: Slide Shows in XHTML*. Mar 2005. https://w3.org/2005/03/slideshow.html (cited on pages 11, 39).

Raggett, Dave [2006a]. *HTML Slidy: Slide Shows in HTML and XHTML*. 2006. https://w3.org/Talks/Tools/Slidy2 (cited on pages 11–12, 39).

Raggett, Dave [2006b]. *Slidy – A Web-Based Alternative to PowerPoint*. Slides from XTech, Amsterdam. 19 May 2006. https://w3.org/2006/05/Slidy-XTech (cited on page 11).

React [2020]. *Introducing JSX*. 04 Dec 2020. https://reactjs.org/docs/introducing-jsx.html (cited on page 29).

Resig, John [2006]. *jQuery*. 26 Aug 2006. https://jquery.com/ (cited on page 7).

Roemmele, Brian [2013]. *Did Apple Buy or Create the Keynote Software?* Quora. 21 Aug 2013. https://quora.com/Apple-company/Did-Apple-buy-or-create-the-Keynote-software/answers/2994640 (cited on page 1).

Rota, Andrew [2015]. *Google-IO-Slides-Fork*. GitHub. 22 Mar 2015. https://github.com/andrewrota/google-io-slides-fork (cited on page 20).

Rouget, Paul [2017a]. *DZSlides*. GitHub. 30 Oct 2017. https://github.com/paulrouget/dzslides (cited on page 21).

Rouget, Paul [2017b]. *DZSlides Example*. 30 Oct 2017. http://paulrouget.com/dzslides (cited on page 23).

Sagalaev, Ivan [2019]. *Highlight.js*. GitHub. 31 Jul 2019. https://github.com/highlightjs/highlight.js (cited on page 79).

Schoffstall, Eric [2019]. *Gulp.js - The Streaming Build System*. 13 Jul 2019. https://gulpjs.com/ (cited on pages 9, 21, 44).

Schofnegger, Markus [2015]. *rslidy: Responsive Presentation Slides in HTML5 and CSS3*. 02 Nov 2015. https://ftp.isds.tugraz.at/pub/theses/mschofnegger-2015-bsc.pdf (cited on pages 41–42).

Shalamov, Alexander, Mikhail Pozdnyakov, and Thomas Steiner [2021]. *Sensors For The Web*. 17 Feb 2021. https://developers.google.com/web/updates/2017/09/sensors-for-the-web (cited on pages xiii, 65).

Slidebean [2014a]. *Slidebean - Create Powerful Presentations*. 04 Jun 2014. https://slidebean.com/ (cited on page 32).

Slidebean [2014b]. *Slidebean - Plans*. 04 Jun 2014. https://slidebean.com/plans/ (cited on page 32).

Slidebean [2014c]. *Slidebean Pitch Deck Template*. 04 Jun 2014. https://slidebean.com/templates/slidebean-pitch-deck (cited on page 34).

Slides [2019]. *Slides – Create and Share Presentations online*. 01 Apr 2019. https://slides.com/ (cited on page 23).

Spencer, Donna [2020]. *Presenting Design Work*. A Book Apart, 20 Oct 2020. 52 pages. ISBN 1937557804 (cited on page 67).

StatCounter [2021]. *Browser Market Share Worldwide*. 01 Aug 2021. http://gs.statcounter.com/browser-market-share (cited on page 3).

Stoll, Martin [2016a]. *Diascope Example*. 30 Apr 2016. http://diascope.sourceforge.net/resources/template.txt (cited on page 15).

Stoll, Martin [2016b]. *Diascope Homepage*. 30 Apr 2016. http://diascope.sourceforge.net/ (cited on page 13).

Syed, Basarat Ali [2019]. *TypeScript Deep Dive*. GitHub. 23 Sep 2019. https://basarat.gitbooks.io/typescript/ (cited on page 9).

Szopka, Bartek and Henrik Ingo [2018a]. *Impress.js*. GitHub. 22 Oct 2018. https://github.com/impress/impress.js (cited on page 27).

Szopka, Bartek and Henrik Ingo [2018b]. *Impress.js Example*. 07 Mar 2018. https://impress.js.org/ (cited on page 28).

Tarr, Dominic [2012]. *BrowserStream*. GitHub. 26 Jun 2012. https://github.com/dominictarr/browser-stream (cited on page 44).

The Document Foundation [2019]. *LibreOffice Impress*. 31 Oct 2019. https://libreoffice.org/discover/impress/ (cited on page 1).

Tibbett, Rich [2015]. *Full Tilt*. GitHub. 02 Aug 2015. https://github.com/adtile/Full-Tilt (cited on page 40).

Troughton, Caleb [2016a]. *Deck.js*. GitHub. 04 May 2016. https://github.com/imakewebthings/deck.js (cited on page 21).

Troughton, Caleb [2016b]. *Deck.js Docs*. 04 May 2016. http://imakewebthings.com/deck.js/docs (cited on page 21).

Troughton, Caleb [2016c]. *Deck.js Example*. 04 May 2016. http://imakewebthings.com/deck.js (cited on page 22).

Troughton, Caleb [2016d]. *Deck.js Wiki*. GitHub. 04 May 2016. https://github.com/imakewebthings/deck.js/wiki (cited on page 21).

Unicode [2018]. *Unicode Version 11*. 05 Jun 2018. https://unicode.org/versions/Unicode11.0.0/ (cited on page 15).

Verou, Lea [2019a]. *Inspire.js*. GitHub. 19 Mar 2019. https://github.com/LeaVerou/inspire.js (cited on page 23).

Verou, Lea [2019b]. *Inspire.js: A Brief Introduction*. 19 Mar 2019. https://inspirejs.org (cited on page 25).

W3C [2017]. *Accessible Rich Internet Applications*. W3C Recommendation. 14 Dec 2017. https://w3.org/TR/wai-aria/ (cited on pages 10, 75, 81).

W3C [2021]. *Generic Sensor API*. W3C Candidate Recommendation Draft. 29 Jul 2021. `https://w3.org/TR/generic-sensor/` (cited on page 64).

W3Counter [2021]. *Browser & Platform Market Share*. W3Counter: Global Web Stats. 01 Aug 2021. `https://w3counter.com/globalstats.php?year=2021&month=8` (cited on page 4).

Webpack [2021]. *Webpack: Bundle Your Assets*. GitHub. 26 Jan 2021. `https://github.com/webpack/webpack` (cited on page 44).

WICG [2020]. *Accessibility Object Model*. GitHub. 24 Aug 2020. `https://github.com/WICG/aom/blob/gh-pages/explainer.md` (cited on page 64).

Wikipedia [2019]. *Plain Text*. 27 Oct 2019. `https://en.wikipedia.org/wiki/Plain_text` (cited on page 15).