hakimel/reveal.js

Originalartikel

Backup

<html> <h3>

```
    <svg aria-hidden="true" class="octicon octicon-book" height="16"
version="1.1" viewbox="0 0 16 16" width="16"/>
    README.md
  </h3>
    <article class="markdown-body entry-content" itemprop="text">
```

A framework for easily creating beautiful presentations using HTML. Check out the live demo. reveal.js comes with a broad range of features including nested slides, Markdown contents, PDF export, speaker notes and a JavaScript API. There's also a fully featured visual editor and platform for sharing reveal.js presentations at slides.com. <h2><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Table of contents</h2> Online Editor Instructions Markup Markdown Element Attributes Slide Attributes Configuration Presentation Size Dependencies Ready Event Auto-sliding Keyboard Bindings Touch Navigation Lazy Loading API Slide Changed Event Presentation State Slide States Slide Backgrounds Parallax Background Slide Transitions Internal links Fragments <a

href=„https://github.com/hakimel/reveal.js#fragment-events“>Fragment events Code syntax highlighting Slide number Overview mode Fullscreen mode Embedded media Stretching elements postMessage API PDF Export Theming Speaker Notes Share and Print Speaker Notes Server Side Speaker Notes Multiplexing Master presentation Client presentation Socket.io server MathJax Installation Basic setup Full setup Folder Structure License <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>More reading</h4> Changelog: Up-to-date version history. Examples: Presentations created with reveal.js, add your own! Browser Support: Explanation of browser support and fallbacks. Plugins: A list of plugins that can be used to extend reveal.js. <h2><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Online Editor</h2> Presentations are written using HTML or Markdown but there's also an online editor for those of you who prefer a graphical interface. Give it a try at https://slides.com. <h2><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Instructions</h2> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Markup</h3> Here's a barebones example of a fully working reveal.js presentation: <div class=„highlight highlight-text-html-basic“><pre><html>

```
&lt;head&gt;
```

```
    &lt;link rel="stylesheet" href="css/reveal.css"&gt;
    &lt;link rel="stylesheet" href="css/theme/white.css"&gt;
&lt;/head&gt;
&lt;body&gt;
    &lt;div class="reveal"&gt;
        &lt;div class="slides"&gt;
            &lt;section&gt;Slide 1&lt;/section&gt;
            &lt;section&gt;Slide 2&lt;/section&gt;
        &lt;/div&gt;
    &lt;/div&gt;
    &lt;script src="js/reveal.js"&gt;&lt;/script&gt;
    &lt;script&gt;
        Reveal.initialize();
    &lt;/script&gt;
&lt;/body&gt;
```

</html></pre> The presentation markup hierarchy needs to be

```
.reveal &gt; .slides &gt; section
```

where the

```
section
```

represents one slide and can be repeated indefinitely. If you place multiple

```
section
```

elements inside of another

```
section
```

they will be shown as vertical slides. The first of the vertical slides is the „root“ of the others (at the top), and will be included in the horizontal sequence. For example: <div class=„highlight highlight-text-html-basic“><pre><div class=„reveal“>

```
&lt;div class="slides"&gt;
    &lt;section&gt;Single Horizontal Slide&lt;/section&gt;
    &lt;section&gt;
        &lt;section&gt;Vertical Slide 1&lt;/section&gt;
        &lt;section&gt;Vertical Slide 2&lt;/section&gt;
    &lt;/section&gt;
&lt;/div&gt;
```

</div></pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Markdown</h3> It's possible to write your slides using Markdown. To enable Markdown, add the

```
data-markdown
```

attribute to your

```
&lt;section&gt;
```

elements and wrap the contents in a

```
&lt;textarea data-template&gt;
```

like the example below. This is based on data-markdown from Paul Irish modified to use marked to support GitHub Flavored Markdown. Sensitive to indentation (avoid mixing tabs and spaces) and line breaks (avoid consecutive breaks). <div class=„highlight highlight-text-html-basic“><pre><section data-markdown>

```
&lt;textarea data-template&gt;
    ## Page title
    A paragraph with some text and a [link](http://hakim.se).
&lt;/textarea&gt;
```

</section></pre> <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>External Markdown</h4> You can write your content as a separate file and have reveal.js load it at runtime. Note the separator arguments which determine how slides are delimited in the external file: the

```
data-separator
```

attribute defines a regular expression for horizontal slides (defaults to

```
^\r?\n---\r?\n$
```

, a newline-bounded horizontal rule) and

```
data-separator-vertical
```

defines vertical slides (disabled by default). The

```
data-separator-notes
```

attribute is a regular expression for specifying the beginning of the current slide's speaker notes (defaults to

```
note:
```

). The

```
data-charset
```

attribute is optional and specifies which charset to use when loading the external file. When used locally, this feature requires that reveal.js runs from a local web server. The following example customises all available options: <div class=„highlight highlight-text-html-basic“><pre><section data-markdown=„example.md“ data-separator=„^\n\n\n“ data-separator-vertical=„^\n\n“ data-separator-notes=„^Note:“ data-charset=„iso-8859-15“> </section></pre> <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Element Attributes</h4> Special syntax (in html comment) is available for adding attributes to Markdown elements. This is useful for fragments, amongst other things. <div class=„highlight highlight-text-html-basic“><pre><section data-markdown>

```
&lt;script type="text/template"&gt;
    - Item 1 &lt;!-- .element: class="fragment" data-fragment-index="2" --
&gt;
    - Item 2 &lt;!-- .element: class="fragment" data-fragment-index="1" --
&gt;
&lt;/script&gt;
```

</section></pre> <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Slide Attributes</h4> Special syntax (in html comment) is available for adding attributes to the slide

```
&lt;section&gt;
```

elements generated by your Markdown. <div class=„highlight highlight-text-html-basic“><pre><section data-markdown>

```
&lt;script type="text/template"&gt;
&lt;!-- .slide: data-background="#ff0000" --&gt;
    Markdown content
&lt;/script&gt;
```

</section></pre> <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Configuring marked</h4> We use marked to parse Markdown. To customise marked's rendering, you can pass in options when configuring Reveal: <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
// Options which are passed into marked
// See https://github.com/chjj/marked#options-1
markdown: {
```

```
    smartypants: true
}
```

});</pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Configuration</h3> At the end of your page you need to initialize reveal by running the following code. Note that all config values are optional and will default as specified below. <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
// Display controls in the bottom right corner
controls: true,
// Display a presentation progress bar
progress: true,
// Set default timing of 2 minutes per slide
defaultTiming: 120,
// Display the page number of the current slide
slideNumber: false,
// Push each slide change to the browser history
history: false,
// Enable keyboard shortcuts for navigation
keyboard: true,
// Enable the slide overview mode
overview: true,
// Vertical centering of slides
center: true,
// Enables touch navigation on devices with touch input
touch: true,
// Loop the presentation
loop: false,
// Change the presentation direction to be RTL
rtl: false,
// Randomizes the order of slides each time the presentation loads
shuffle: false,
// Turns fragments on and off globally
fragments: true,
// Flags if the presentation is running in an embedded mode,
// i.e. contained within a limited portion of the screen
embedded: false,
// Flags if we should show a help overlay when the questionmark
// key is pressed
help: true,
// Flags if speaker notes should be visible to all viewers
showNotes: false,
// Global override for autolaying embedded media (video/audio/iframe)
// - null: Media will only autoplay if data-autoplay is present
// - true: All media will autoplay, regardless of individual setting
// - false: No media will autoplay, regardless of individual setting
autoPlayMedia: null,
// Number of milliseconds between automatically proceeding to the
// next slide, disabled when set to 0, this value can be overwritten
```

```
// by using a data-autoslide attribute on your slides
autoSlide: 0,
// Stop auto-sliding after user input
autoSlideStoppable: true,
// Use this method for navigation when auto-sliding
autoSlideMethod: Reveal.navigateNext,
// Enable slide navigation via mouse wheel
mouseWheel: false,
// Hides the address bar on mobile devices
hideAddressBar: true,
// Opens links in an iframe preview overlay
previewLinks: false,
// Transition style
transition: 'slide', // none/fade/slide/convex/concave/zoom
// Transition speed
transitionSpeed: 'default', // default/fast/slow
// Transition style for full page slide backgrounds
backgroundTransition: 'fade', // none/fade/slide/convex/concave/zoom
// Number of slides away from the current that are visible
viewDistance: 3,
// Parallax background image
parallaxBackgroundImage: '', // e.g.
"'https://s3.amazonaws.com/hakim-static/reveal-js/reveal-parallax-1.jpg'"
// Parallax background size
parallaxBackgroundSize: '', // CSS syntax, e.g. "2100px 900px"
// Number of pixels to move the parallax background per slide
// - Calculated automatically unless specified
// - Set to 0 to disable movement along an axis
parallaxBackgroundHorizontal: null,
parallaxBackgroundVertical: null,
// The display mode that will be used to show slides
display: 'block'
```

});</pre> The configuration can be updated after initialization using the

```
configure
```

method: <div class=„highlight highlight-source-js“><pre> *Turn autoSlide off Reveal.configure({ autoSlide: 0 });* Start auto-sliding every 5s Reveal.configure({ autoSlide: 5000 });</pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Presentation Size</h3> All presentations have a normal size, that is the resolution at which they are authored. The framework will automatically scale presentations uniformly based on this size to ensure that everything fits on any given display or viewport. See below for a list of configuration options related to sizing, including default values: <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
...
// The "normal" size of the presentation, aspect ratio will be preserved
// when the presentation is scaled to fit different resolutions. Can be
```

```
// specified using percentage units.
width: 960,
height: 700,
// Factor of the display size that should remain empty around the content
margin: 0.1,
// Bounds for smallest/largest possible scale to apply to content
minScale: 0.2,
maxScale: 1.5
```

});</pre> If you wish to disable this behavior and do your own scaling (e.g. using media queries), try these settings: <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
...
width: "100%",
height: "100%",
margin: 0,
minScale: 1,
maxScale: 1
```

});</pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Dependencies</h3> Reveal.js doesn't rely on any third party scripts to work but a few optional libraries are included by default. These libraries are loaded as dependencies in the order they appear, for example: <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
dependencies: [
    // Cross-browser shim that fully implements classList -
https://github.com/eligrey/classList.js/
    { src: 'lib/js/classList.js', condition: function() { return
!document.body.classList; } },
    // Interpret Markdown in &lt;section&gt; elements
    { src: 'plugin/markdown/marked.js', condition: function() { return
!!document.querySelector( '[data-markdown]' ); } },
    { src: 'plugin/markdown/markdown.js', condition: function() { return
!!document.querySelector( '[data-markdown]' ); } },
    // Syntax highlight for &lt;code&gt; elements
    { src: 'plugin/highlight/highlight.js', async: true, callback:
function() { hljs.initHighlightingOnLoad(); } },
    // Zoom in and out with Alt+click
    { src: 'plugin/zoom-js/zoom.js', async: true },
    // Speaker notes
    { src: 'plugin/notes/notes.js', async: true },
    // MathJax
    { src: 'plugin/math/math.js', async: true }
]
```

});</pre> You can add your own extensions using the same syntax. The following properties are available for each dependency object: src: Path to the script to load async: [optional] Flags if the script should load after

reveal.js has started, defaults to false callback: [optional] Function to execute when the script has loaded condition: [optional] Function which must return true for the script to be loaded To load these dependencies, reveal.js requires head.js (a script loading library) to be loaded before reveal.js. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Ready Event</h3> A 'ready' event is fired when reveal.js has loaded all non-async dependencies and is ready to start navigating. To check if reveal.js is already 'ready' you can call

```
Reveal.isReady()
```

. <div class=„highlight highlight-source-js“><pre>Reveal.addEventListener('ready', function(event) {

```
// event.currentSlide, event.indexh, event.indexv
```

});</pre> Note that we also add a

```
.ready
```

class to the

```
.reveal
```

element so that you can hook into this with CSS. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Auto-sliding</h3> Presentations can be configured to progress through slides automatically, without any user input. To enable this you will need to tell the framework how many milliseconds it should wait between slides: <div class=„highlight highlight-source-js“><pre> *Slide every five seconds Reveal.configure({ autoSlide: 5000 });</pre> When this is turned on a control element will appear that enables users to pause and resume auto-sliding. Alternatively, sliding can be paused or resumed by pressing »a« on the keyboard. Sliding is paused automatically as soon as the user starts navigating. You can disable these controls by specifying <code>autoSlideStoppable: false</code> in your reveal.js config. You can also override the slide duration for individual slides and fragments by using the <code>data-autoslide</code> attribute: <div class=„highlight highlight-text-html-basic“><pre><section data-autoslide=„2000“> <p>After 2 seconds the first fragment will be shown.</p> <p class=„fragment“ data-autoslide=„10000“>After 10 seconds the next fragment will be shown.</p> <p class=„fragment“>Now, the fragment is displayed for 2 seconds before the next slide is shown.</p> </section></pre> To override the method used for navigation when auto-sliding, you can specify the <code>autoSlideMethod</code> setting. To only navigate along the top layer and ignore vertical slides, set this to <code>Reveal.navigateRight</code>. Whenever the auto-slide mode is resumed or paused the <code>autoslideresumed</code> and <code>autoslidepaused</code> events are fired. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“*

width=„16“/>Keyboard Bindings</h3> If you're unhappy with any of the default keyboard bindings you can override them using the <code>keyboard</code> config option: <div class=„highlight highlight-source-js“><pre>Reveal.configure({ keyboard: { 13: 'next', go to the next slide when the ENTER key is pressed

```
  27: function() {}, // do something custom when ESC is pressed
  32: null // don't do anything when SPACE is pressed (i.e. disable a
reveal.js default binding)
}
```

});</pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Touch Navigation</h3> You can swipe to navigate through a presentation on any touch-enabled device. Horizontal swipes change between horizontal slides, vertical swipes change between vertical slides. If you wish to disable this you can set the

```
touch
```

config option to false when initializing reveal.js. If there's some part of your content that needs to remain accessible to touch events you'll need to highlight this by adding a

```
data-prevent-swipe
```

attribute to the element. One common example where this is useful is elements that need to be scrolled. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Lazy Loading</h3> When working on presentation with a lot of media or iframe content it's important to load lazily. Lazy loading means that reveal.js will only load content for the few slides nearest to the current slide. The number of slides that are preloaded is determined by the

```
viewDistance
```

configuration option. To enable lazy loading all you need to do is change your „src“ attributes to „data-src“ as shown below. This is supported for image, video, audio and iframe elements. Lazy loaded iframes will also unload when the containing slide is no longer visible. <div class=„highlight highlight-text-html-basic“><pre><section>

```
&lt;img data-src="image.png"&gt;
&lt;iframe data-src="http://hakim.se"&gt;&lt;/iframe&gt;
&lt;video&gt;
  &lt;source data-src="video.webm" type="video/webm" /&gt;
  &lt;source data-src="video.mp4" type="video/mp4" /&gt;
&lt;/video&gt;
```

</section></pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“

width=„16“/>API</h3> The

```
Reveal
```

object exposes a JavaScript API for controlling navigation and reading state: <div class=„highlight highlight-source-js“><pre> *Navigation Reveal.slide(indexh, indexv, indexf); Reveal.left(); Reveal.right(); Reveal.up(); Reveal.down(); Reveal.prev(); Reveal.next(); Reveal.prevFragment(); Reveal.nextFragment();* Randomize the order of slides Reveal.shuffle(); *Toggle presentation states, optionally pass true/false to force on/off Reveal.toggleOverview(); Reveal.togglePause(); Reveal.toggleAutoSlide();* Shows a help overlay with keyboard shortcuts, optionally pass true/false *to force on/off Reveal.toggleHelp();* Change a config value at runtime Reveal.configure({ controls: true }); *Returns the present configuration options Reveal.getConfig();* Fetch the current scale of the presentation Reveal.getScale(); *Retrieves the previous and current slide elements Reveal.getPreviousSlide(); Reveal.getCurrentSlide(); Reveal.getIndices();* { h: 0, v: 0 } } Reveal.getPastSlideCount(); Reveal.getProgress(); *(0 == first slide, 1 == last slide) Reveal.getSlides();* Array of all slides Reveal.getTotalSlides(); *total number of slides* Returns the speaker notes for the current slide Reveal.getSlideNotes(); *State checks Reveal.isFirstSlide(); Reveal.isLastSlide(); Reveal.isOverview(); Reveal.isPaused(); Reveal.isAutoSliding();</pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Slide Changed Event</h3> A 'slidechanged' event is fired each time the slide is changed (regardless of state). The event object holds the index values of the current slide as well as a reference to the previous and current slide HTML nodes. Some libraries, like MathJax (see #226), get confused by the transforms and display states of slides. Often times, this can be fixed by calling their update or render function from this callback. <div class=„highlight highlight-source-js“><pre>Reveal.addEventListener('slidechanged', function(event) {* event.previousSlide, event.currentSlide, event.indexh, event.indexv });</pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Presentation State</h3> The presentation's current state can be fetched by using the

```
getState
```

method. A state object contains all of the information required to put the presentation back as it was when

```
getState
```

was first called. Sort of like a snapshot. It's a simple object that can easily be stringified and persisted or sent over the wire. <div class=„highlight highlight-source-js“><pre>Reveal.slide(1); *we're on slide 1 var state = Reveal.getState(); Reveal.slide(3);* we're on slide 3 Reveal.setState(state); *we're back on slide 1</pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Slide States</h3> If you set <code>data-state=„somestate“</code> on a slide <code><section></code>, „somestate“ will be applied as a class on the document element when that slide is opened. This allows you to apply broad style changes to the page based on*

the active slide. Furthermore you can also listen to these changes in state via JavaScript: <div class=„highlight highlight-source-js“><pre>Reveal.addEventListener('somestate', function() { TODO: Sprinkle magic }, false);</pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Slide Backgrounds</h3> Slides are contained within a limited portion of the screen by default to allow them to fit any display and scale uniformly. You can apply full page backgrounds outside of the slide area by adding a

```
data-background
```

attribute to your

```
&lt;section&gt;
```

elements. Four different types of backgrounds are supported: color, image, video and iframe. <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Color Backgrounds</h4> All CSS color formats are supported, like rgba() or hsl(). <div class=„highlight highlight-text-html-basic“><pre><section data-background-color=„#ff0000“>

```
&lt;h2&gt;Color&lt;/h2&gt;
```

</section></pre> <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Image Backgrounds</h4> By default, background images are resized to cover the full page. Available options: <table><thead><tr><th align=„left“>Attribute</th> <th align=„left“>Default</th> <th align=„left“>Description</th> </tr></thead><tbody><tr><td align=„left“>data-background-image</td> <td align=„left“/> <td align=„left“>URL of the image to show. GIFs restart when the slide opens.</td> </tr><tr><td align=„left“>data-background-size</td> <td align=„left“>cover</td> <td align=„left“>See background-size on MDN.</td> </tr><tr><td align=„left“>data-background-position</td> <td align=„left“>center</td> <td align=„left“>See background-position on MDN.</td> </tr><tr><td align=„left“>data-background-repeat</td> <td align=„left“>no-repeat</td> <td align=„left“>See background-repeat on MDN.</td> </tr></tbody></table><div class=„highlight highlight-text-html-basic“><pre><section data-background-image=„http://example.com/image.png“>

```
&lt;h2&gt;Image&lt;/h2&gt;
```

</section> <section data-background-image=„http://example.com/image.png“ data-background-size=„100px“ data-background-repeat=„repeat“>

```
&lt;h2&gt;This background image will be sized to 100px and
```

```
repeated&lt;/h2&gt;
```

</section></pre> <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Video Backgrounds</h4> Automatically plays a full size video behind the slide. <table><thead><tr><th align=„left“>Attribute</th> <th align=„left“>Default</th> <th align=„left“>Description</th> </tr></thead><tbody><tr><td align=„left“>data-background-video</td> <td align=„left“/> <td align=„left“>A single video source, or a comma separated list of video sources.</td> </tr><tr><td align=„left“>data-background-video-loop</td> <td align=„left“>false</td> <td align=„left“>Flags if the video should play repeatedly.</td> </tr><tr><td align=„left“>data-background-video-muted</td> <td align=„left“>false</td> <td align=„left“>Flags if the audio should be muted.</td> </tr><tr><td align=„left“>data-background-size</td> <td align=„left“>cover</td> <td align=„left“>Use

```
cover
```

for full screen and some cropping or

```
contain
```

for letterboxing.</td> </tr></tbody></table><div class=„highlight highlight-text-html-basic“><pre><section
data-
background-video=„https://s3.amazonaws.com/static.slid.es/site/homepage/v1/homepage-video-editor.mp4,https://s3.amazonaws.com/static.slid.es/site/homepage/v1/homepage-video-editor.webm“ data-background-video-loop data-background-video-muted>

```
&lt;h2&gt;Video&lt;/h2&gt;
```

</section></pre> <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Iframe Backgrounds</h4> Embeds a web page as a slide background that covers 100% of the reveal.js width and height. The iframe is in the background layer, behind your slides, and as such it's not possible to interact with it by default. To make your background interactive, you can add the

```
data-background-interactive
```

attribute. <div class=„highlight highlight-text-html-basic“><pre><section data-background-iframe=„https://slides.com“ data-background-interactive>

```
&lt;h2&gt;Iframe&lt;/h2&gt;
```

</section></pre> <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Background Transitions</h4> Backgrounds transition using a fade animation by default. This can be changed to a linear sliding transition by passing

```
backgroundTransition: 'slide'
```

to the

```
Reveal.initialize()
```

call. Alternatively you can set

```
data-background-transition
```

on any section with a background to override that specific transition. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Parallax Background</h3> If you want to use a parallax scrolling background, set the first two config properties below when initializing reveal.js (the other two are optional). <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
// Parallax background image
parallaxBackgroundImage: '', // e.g.
"https://s3.amazonaws.com/hakim-static/reveal-js/reveal-parallax-1.jpg"
// Parallax background size
parallaxBackgroundSize: '', // CSS syntax, e.g. "2100px 900px" - currently
only pixels are supported (don't use % or auto)
// Number of pixels to move the parallax background per slide
// - Calculated automatically unless specified
// - Set to 0 to disable movement along an axis
parallaxBackgroundHorizontal: 200,
parallaxBackgroundVertical: 50
```

});</pre> Make sure that the background size is much bigger than screen size to allow for some scrolling. View example. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Slide Transitions</h3> The global presentation transition is set using the

```
transition
```

config value. You can override the global transition for a specific slide by using the

```
data-transition
```

attribute: <div class=„highlight highlight-text-html-basic“><pre><section data-transition=„zoom“>

```
&lt;h2&gt;This slide will override the presentation transition and
```

```
zoom!&lt;/h2&gt;
```

</section> <section data-transition-speed=„fast“>

```
&lt;h2&gt;Choose from three transition speeds: default, fast or
slow!&lt;/h2&gt;
```

</section></pre> You can also use different in and out transitions for the same slide: <div class=„highlight highlight-text-html-basic“><pre><section data-transition=„slide“>

```
  The train goes on &#8230;
```

</section> <section data-transition=„slide“>

```
  and on &#8230;
```

</section> <section data-transition=„slide-in fade-out“>

```
  and stops.
```

</section> <section data-transition=„fade-in slide-out“>

```
  (Passengers entering and leaving)
```

</section> <section data-transition=„slide“>

```
  And it starts again.
```

</section></pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Internal links</h3> It's easy to link between slides. The first example below targets the index of another slide whereas the second targets a slide with an ID attribute (

```
&lt;section id="some-slide"&gt;
```

): <div class=„highlight highlight-text-html-basic“><pre>Link Link</pre> You can also add relative navigation links, similar to the built in reveal.js controls, by appending one of the following classes on any element. Note that each element is automatically given an

```
enabled
```

class when it's a valid navigation route based on the current slide. <div class=„highlight highlight-text-html-basic“><pre>   <!– Previous vertical or horizontal slide –> <!– Next vertical or horizontal slide –></pre> <h3><a id=„user-content-fragments“ class=„anchor“

href=„https://github.com/hakimel/reveal.js#fragments“ aria-hidden=„true“><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Fragments</h3> Fragments are used to highlight individual elements on a slide. Every element with the class

```
fragment
```

will be stepped through before moving on to the next slide. Here's an example: http://lab.hakim.se/reveal-js/#/fragments The default fragment style is to start out invisible and fade in. This style can be changed by appending a different class to the fragment: <div class=„highlight highlight-text-html-basic“><pre><section>

```
&lt;p class="fragment grow"&gt;grow&lt;/p&gt;
&lt;p class="fragment shrink"&gt;shrink&lt;/p&gt;
&lt;p class="fragment fade-out"&gt;fade-out&lt;/p&gt;
&lt;p class="fragment fade-up"&gt;fade-up (also down, left and
right!)&lt;/p&gt;
&lt;p class="fragment current-visible"&gt;visible only once&lt;/p&gt;
&lt;p class="fragment highlight-current-blue"&gt;blue only once&lt;/p&gt;
&lt;p class="fragment highlight-red"&gt;highlight-red&lt;/p&gt;
&lt;p class="fragment highlight-green"&gt;highlight-green&lt;/p&gt;
&lt;p class="fragment highlight-blue"&gt;highlight-blue&lt;/p&gt;
```

</section></pre> Multiple fragments can be applied to the same element sequentially by wrapping it, this will fade in the text on the first step and fade it back out on the second. <div class=„highlight highlight-text-html-basic“><pre><section>

```
&lt;span class="fragment fade-in"&gt;
    &lt;span class="fragment fade-out"&gt;I'll fade in, then
out&lt;/span&gt;
&lt;/span&gt;
```

</section></pre> The display order of fragments can be controlled using the

```
data-fragment-index
```

attribute. <div class=„highlight highlight-text-html-basic“><pre><section>

```
&lt;p class="fragment" data-fragment-index="3"&gt;Appears last&lt;/p&gt;
&lt;p class="fragment" data-fragment-index="1"&gt;Appears first&lt;/p&gt;
&lt;p class="fragment" data-fragment-index="2"&gt;Appears second&lt;/p&gt;
```

</section></pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Fragment events</h3> When a slide fragment is either shown or hidden reveal.js will dispatch an event. Some libraries, like MathJax (see #505), get confused by the initially hidden fragment elements. Often times this can be fixed by calling their update or render function from this callback. <div class=„highlight highlight-source-

js“><pre>Reveal.addEventListener('fragmentshown', function(event) {

```
// event.fragment = the fragment DOM element
```

}); Reveal.addEventListener('fragmenthidden', function(event) {

```
// event.fragment = the fragment DOM element
```

});</pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Code syntax highlighting</h3> By default, Reveal is configured with highlight.js for code syntax highlighting. Below is an example with clojure code that will be syntax highlighted. When the

```
data-trim
```

attribute is present, surrounding whitespace is automatically removed. HTML will be escaped by default. To avoid this, for example if you are using

```
&lt;mark&gt;
```

to call out a line of code, add the

```
data-noescape
```

attribute to the

```
&lt;code&gt;
```

element. <div class=„highlight highlight-text-html-basic“><pre><section>

```
&lt;pre&gt;&lt;code data-trim data-noescape&gt;
```

(def lazy-fib

```
(concat
 [0 1]
 &lt;mark&gt;((fn rfib [a b]&lt;/mark&gt;
      (lazy-cons (+ a b) (rfib b (+ a b)))) 0 1)))
&lt;/code&gt;&lt;/pre&gt;
```

</section></pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Slide number</h3> If you would like to display the page number of the current slide you can do so using the

```
slideNumber
```

and

```
showSlideNumber
```

configuration values. <div class=„highlight highlight-source-js“><pre> *Shows the slide number using default formatting Reveal.configure({ slideNumber: true });* Slide number formatting can be configured using these variables: *„h.v“: horizontal . vertical slide number (default)* „h/v“: horizontal / vertical slide number *„c“: flattened slide number* „c/t“: flattened slide number / total slides Reveal.configure({ slideNumber: 'c/t' }); *Control which views the slide number displays on using the „showSlideNumber“ value:* „all“: show on all views (default) *„speaker“: only show slide numbers on speaker notes view* „print“: only show slide numbers when printing to PDF Reveal.configure({ showSlideNumber: 'speaker' }); </pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Overview mode</h3> Press „Esc“ or „o“ keys to toggle the overview mode on and off. While you're in this mode, you can still navigate between slides, as if you were at 1,000 feet above your presentation. The overview mode comes with a few API hooks: <div class=„highlight highlight-source-js“><pre>Reveal.addEventListener('overviewshown', function(event) { /* … */ }); Reveal.addEventListener('overviewhidden', function(event) { /* … */ }); *Toggle the overview mode programmatically Reveal.toggleOverview();</pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Fullscreen mode</h3> Just press »F« on your keyboard to show your presentation in fullscreen mode. Press the »ESC« key to exit fullscreen mode. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Embedded media</h3> Add <code>data-autoplay</code> to your media element if you want it to automatically start playing when the slide is shown: <div class=„highlight highlight-text-html-basic“><pre><video data-autoplay src=„http://clips.vorwaerts-gmbh.de/big_buck_bunny.mp4“></video></pre> If you want to enable or disable autoplay globally, for all embedded media, you can use the <code>autoPlayMedia</code> configuration option. If you set this to <code>true</code> ALL media will autoplay regardless of individual <code>data-autoplay</code> attributes. If you initialize with <code>autoPlayMedia: false</code> NO media will autoplay. Note that embedded HTML5 <code><video></code>/<code><audio></code> and YouTube/Vimeo iframes are automatically paused when you navigate away from a slide. This can be disabled by decorating your element with a <code>data-ignore</code> attribute. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Embedded iframes</h3> reveal.js automatically pushes two post messages to embedded iframes. <code>slide:start</code> when the slide containing the iframe is made visible and <code>slide:stop</code> when it is hidden. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Stretching elements</h3> Sometimes it's desirable to have an element, like an image or video, stretch to consume as much space as possible within a given slide. This can be done by adding the <code>.stretch</code> class to an element as seen below: <div*

class=„highlight highlight-text-html-basic“><pre><section> <h2>This video will use up the remaining space on the slide</h2> <video class=„stretch“ src=„http://clips.vorwaerts-gmbh.de/big_buck_bunny.mp4“></video> </section></pre> Limitations: Only direct descendants of a slide section can be stretched Only one descendant per slide section can be stretched <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>postMessage API</h3> The framework has a built-in postMessage API that can be used when communicating with a presentation inside of another window. Here's an example showing how you'd make a reveal.js instance in the given window proceed to slide 2: <div class=„highlight highlight-source-js“><pre><window>.postMessage(JSON.stringify({ method: 'slide', args: [2] }), '');</pre> When reveal.js runs inside of an iframe it can optionally bubble all of its events to the parent. Bubbled events are stringified JSON with three fields: namespace, eventName and state. Here's how you subscribe to them from the parent window: <div class=„highlight highlight-source-js“><pre>window.addEventListener('message', function(event) { var data = JSON.parse(event.data); if(data.namespace === 'reveal' && data.eventName ==='slidechanged') {* Slide changed, see data.state for slide number

```
}
```

});</pre> This cross-window messaging can be toggled on or off using configuration flags. <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
...,
// Exposes the reveal.js API through window.postMessage
postMessage: true,
// Dispatches all reveal.js events to the parent window through postMessage
postMessageEvents: false
```

});</pre> <h2><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>PDF Export</h2> Presentations can be exported to PDF via a special print stylesheet. This feature requires that you use Google Chrome or Chromium and to be serving the presention from a webserver. Here's an example of an exported presentation that's been uploaded to SlideShare: http://www.slideshare.net/hakimel/revealjs-300. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Page size</h3> Export dimensions are inferred from the configured presentation size. Slides that are too tall to fit within a single page will expand onto multiple pages. You can limit how many pages a slide may expand onto using the

```
pdfMaxPagesPerSlide
```

config option, for example

```
Reveal.configure({ pdfMaxPagesPerSlide: 1 })
```

ensures that no slide ever grows to more than one printed page. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Print stylesheet</h3> To enable the PDF print capability in your presentation, the special print stylesheet at /css/print/pdf.css must be loaded. The default index.html file handles this for you when

```
print-pdf
```

is included in the query string. If you're using a different HTML template, you can add this to your HEAD: <div class=„highlight highlight-text-html-basic“><pre><script>

```
var link = document.createElement( 'link' );
link.rel = 'stylesheet';
link.type = 'text/css';
link.href = window.location.search.match( /print-pdf/gi ) ?
'css/print/pdf.css' : 'css/print/paper.css';
document.getElementsByTagName( 'head' )[0].appendChild( link );
```

</script></pre> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Instructions</h3> Open your presentation with

```
print-pdf
```

included in the query string i.e. http://localhost:8000/?print-pdf. You can test this with lab.hakim.se/reveal-js?print-pdf. If you want to include speaker notes in your export, you can append

```
showNotes=true
```

to the query string: http://localhost:8000/?print-pdf&showNotes=true Open the in-browser print dialog (CTRL/CMD+P). Change the Destination setting to Save as PDF. Change the Layout to Landscape. Change the Margins to None. Enable the Background graphics option. Click Save. <img src=„https://camo.githubusercontent.com/e3b3088a2dd7a53caf72de529b3ce41465dd99f0/687474700733a2f2f73332e616d617a6f6e6177732e636f6d2f68616b696d2d7374617469632f72657665616c2d6a732f7064662d7072696e742d73657474696e67732d322e706e67“ alt=„Chrome Print Settings“ data-

canonical-src=„https://s3.amazonaws.com/hakim-static/reveal-js/pdf-print-settings-2.png“/> Alternatively you can use the decktape project. <h2><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Theming</h2> The framework comes with a few different themes included: black: Black background, white text, blue links (default theme) white: White background, black text, blue links league: Gray background, white text, blue links (default theme for reveal.js < 3.0.0) beige: Beige background, dark text, brown links sky: Blue background, thin dark text, blue links night: Black background, thick white text, orange links serif: Cappuccino background, gray text, brown links simple: White background, black text, blue links solarized: Cream-colored background, dark green text, blue links Each theme is available as a separate stylesheet. To change theme you will need to replace black below with your desired theme name in index.html: <div class=„highlight highlight-text-html-basic“><pre><link rel=„stylesheet“ href=„css/theme/black.css“ id=„theme“></pre> If you want to add a theme of your own see the instructions here: /css/theme/README.md. <h2><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Speaker Notes</h2> reveal.js comes with a speaker notes plugin which can be used to present per-slide notes in a separate browser window. The notes window also gives you a preview of the next upcoming slide so it may be helpful even if you haven't written any notes. Press the 's' key on your keyboard to open the notes window. A speaker timer starts as soon as the speaker view is opened. You can reset it to 00:00:00 at any time by simply clicking/tapping on it. Notes are defined by appending an

```
&lt;aside&gt;
```

element to a slide as seen below. You can add the

```
data-markdown
```

attribute to the aside element if you prefer writing notes using Markdown. Alternatively you can add your notes in a

```
data-notes
```

attribute on the slide. Like

```
&lt;section data-notes="Something important"&gt;&lt;/section&gt;
```

. When used locally, this feature requires that reveal.js runs from a local web server. <div class=„highlight highlight-text-html-basic“><pre><section>

```
&lt;h2&gt;Some Slide&lt;/h2&gt;
&lt;aside class="notes"&gt;
    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).
&lt;/aside&gt;
```

</section></pre> If you're using the external Markdown plugin, you can add notes with the help of a special delimiter: <div class=„highlight highlight-text-html-basic“><pre><section data-markdown=„example.md“ data-separator=„^\n\n\n“ data-separator-vertical=„^\n\n“ data-separator-notes=„^Note:“></section> # Title ## Sub-title Here is some content… Note: This will only display in the notes window.</pre> <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Share and Print Speaker Notes</h4> Notes are only visible to the speaker inside of the speaker view. If you wish to share your notes with others you can initialize reveal.js with the

```
showNotes
```

config value set to

```
true
```

. Notes will appear along the bottom of the presentations. When

```
showNotes
```

is enabled notes are also included when you export to PDF. By default, notes are printed in a semi-transparent box on top of the slide. If you'd rather print them on a separate page after the slide, set

```
showNotes: "separate-page"
```

. <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Speaker notes clock and timers</h4> The speaker notes window will also show: Time elapsed since the beginning of the presentation. If you hover the mouse above this section, a timer reset button will appear. Current wall-clock time (Optionally) a pacing timer which indicates whether the current pace of the presentation is on track for the right timing (shown in green), and if not, whether the presenter should speed up (shown in red) or has the luxury of slowing down (blue). The pacing timer can be enabled by configuring by the

```
defaultTiming
```

parameter in the

```
Reveal
```

configuration block, which specifies the number of seconds per slide. 120 can be a reasonable rule of thumb. Timings can also be given per slide

```
&lt;section&gt;
```

by setting the

```
data-timing
```

attribute. Both values are in numbers of seconds. <h2><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Server Side Speaker Notes</h2> In some cases it can be desirable to run notes on a separate device from the one you're presenting on. The Node.js-based notes plugin lets you do this using the same note definitions as its client side counterpart. Include the required scripts by adding the following dependencies: <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
...
dependencies: [
    { src: 'socket.io/socket.io.js', async: true },
    { src: 'plugin/notes-server/client.js', async: true }
]
```

});</pre> Then: Install Node.js (4.0.0 or later) Run

```
npm install
```

 Run

```
node plugin/notes-server
```

 <h2><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Multiplexing</h2> The multiplex plugin allows your audience to view the slides of the presentation you are controlling on their own phone, tablet or laptop. As the master presentation navigates the slides, all client presentations will update in real time. See a demo at https://reveal-js-multiplex-ccjbegmaii.now.sh/. The multiplex plugin needs the following 3 things to operate: Master presentation that has control Client presentations that follow the master Socket.io server to broadcast events from the master to the clients More details: <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Master presentation</h4> Served from a static file server accessible (preferably) only to the presenter. This need only be on your (the presenter's) computer. (It's safer to run the master presentation from your own computer, so if the venue's Internet goes down it doesn't stop the show.) An example would be to execute the following commands in the directory of your

master presentation:

```
npm install node-static
```



```
static
```

  If you want to use the speaker notes plugin with your master presentation then make sure you have the speaker notes plugin configured correctly along with the configuration shown below, then execute

```
node plugin/notes-server
```

in the directory of your master presentation. The configuration below will cause it to connect to the socket.io server as a master, as well as launch your speaker-notes/static-file server. You can then access your master presentation at

```
http://localhost:1947
```

 Example configuration: <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
// other options...
multiplex: {
    // Example values. To generate your own, see the socket.io server
instructions.
    secret: '13652805320794272084', // Obtained from the socket.io server.
Gives this (the master) control of the presentation
    id: '1ea875674b17ca76', // Obtained from socket.io server
    url: 'https://reveal-js-multiplex-ccjbegmaii.now.sh' // Location of
socket.io server
},
// Don't forget to add the dependencies
dependencies: [
    { src: '//cdn.socket.io/socket.io-1.3.5.js', async: true },
    { src: 'plugin/multiplex/master.js', async: true },
    // and if you want speaker notes
    { src: 'plugin/notes-server/client.js', async: true }
    // other dependencies...
]
```

});</pre> <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Client presentation</h4> Served from a publicly accessible static file server. Examples include: GitHub Pages, Amazon S3, Dreamhost, Akamai, etc. The more reliable, the better. Your audience can then access the client presentation via

```
http://example.com/path/to/presentation/client/index.html
```

, with the configuration below causing them to connect to the socket.io server as clients. Example configuration: <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
// other options...
multiplex: {
    // Example values. To generate your own, see the socket.io server
instructions.
    secret: null, // null so the clients do not have control of the master
presentation
    id: '1ea875674b17ca76', // id, obtained from socket.io server
    url: 'https://reveal-js-multiplex-ccjbegmaii.now.sh' // Location of
socket.io server
},
// Don't forget to add the dependencies
dependencies: [
    { src: '//cdn.socket.io/socket.io-1.3.5.js', async: true },
    { src: 'plugin/multiplex/client.js', async: true }
    // other dependencies...
]
```

});</pre> <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Socket.io server</h4> Server that receives the slideChanged events from the master presentation and broadcasts them out to the connected client presentations. This needs to be publicly accessible. You can run your own socket.io server with the commands:

```
npm install
```



```
node plugin/multiplex
```

  Or you can use the socket.io server at https://reveal-js-multiplex-ccjbegmaii.now.sh/. You'll need to generate a unique secret and token pair for your master and client presentations. To do so, visit

```
http://example.com/token
```

, where

```
http://example.com
```

is the location of your socket.io server. Or if you're going to use the socket.io server at https://reveal-js-multiplex-ccjbegmaii.now.sh/, visit https://reveal-js-multiplex-ccjbegmaii.now.sh/token. You are very welcome to point your presentations at the Socket.io server running at <a

href=„https://reveal-js-multiplex-ccjbegmaii.now.sh/“>https://reveal-js-multiplex-ccjbegmaii.now.sh/, but availability and stability are not guaranteed. For anything mission critical I recommend you run your own server. The easiest way to do this is by installing now. With that installed, deploying your own Multiplex server is as easy running the following command from the reveal.js folder:

```
now plugin/multiplex
```

. <h5><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>socket.io server as file static server</h5> The socket.io server can play the role of static file server for your client presentation, as in the example at https://reveal-js-multiplex-ccjbegmaii.now.sh/. (Open https://reveal-js-multiplex-ccjbegmaii.now.sh/ in two browsers. Navigate through the slides on one, and the other will update to match.) Example configuration: <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
// other options...
multiplex: {
    // Example values. To generate your own, see the socket.io server
instructions.
    secret: null, // null so the clients do not have control of the master
presentation
    id: '1ea875674b17ca76', // id, obtained from socket.io server
    url: 'example.com:80' // Location of your socket.io server
},
// Don't forget to add the dependencies
dependencies: [
    { src: '//cdn.socket.io/socket.io-1.3.5.js', async: true },
    { src: 'plugin/multiplex/client.js', async: true }
    // other dependencies...
]</pre></div>
```

It can also play the role of static file server for your master presentation and client presentations at the same time (as long as you don't want to use speaker notes). (Open https://reveal-js-multiplex-ccjbegmaii.now.sh/ in two browsers. Navigate through the slides on one, and the other will update to match. Navigate through the slides on the second, and the first will update to match.) This is probably not desirable, because you don't want your audience to mess with your slides while you're presenting. ;) Example configuration: <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
// other options...
multiplex: {
    // Example values. To generate your own, see the socket.io server
instructions.
    secret: '13652805320794272084', // Obtained from the socket.io server.
Gives this (the master) control of the presentation
```

```
    id: '1ea875674b17ca76', // Obtained from socket.io server
    url: 'example.com:80' // Location of your socket.io server
},
// Don't forget to add the dependencies
dependencies: [
    { src: '//cdn.socket.io/socket.io-1.3.5.js', async: true },
    { src: 'plugin/multiplex/master.js', async: true },
    { src: 'plugin/multiplex/client.js', async: true }
    // other dependencies...
]
```

});</pre> <h2><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>MathJax</h2> If you want to display math equations in your presentation you can easily do so by including this plugin. The plugin is a very thin wrapper around the MathJax library. To use it you'll need to include it as a reveal.js dependency, find our more about dependencies here. The plugin defaults to using LaTeX but that can be adjusted through the

```
math
```

configuration object. Note that MathJax is loaded from a remote server. If you want to use it offline you'll need to download a copy of the library and adjust the

```
mathjax
```

configuration value. Below is an example of how the plugin can be configured. If you don't intend to change these values you do not need to include the

```
math
```

config object at all. <div class=„highlight highlight-source-js“><pre>Reveal.initialize({

```
// other options ...
math: {
    mathjax:
'https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.0/MathJax.js',
    config: 'TeX-AMS_HTML-full'  // See
http://docs.mathjax.org/en/latest/config-files.html
},
dependencies: [
    { src: 'plugin/math/math.js', async: true }
]
```

});</pre> Read MathJax's documentation if you need HTTPS delivery or serving of specific

versions for stability. <h2><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Installation</h2> The basic setup is for authoring presentations only. The full setup gives you access to all reveal.js features and plugins such as speaker notes as well as the development tasks needed to make changes to the source. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Basic setup</h3> The core of reveal.js is very easy to install. You'll simply need to download a copy of this repository and open the index.html file directly in your browser. Download the latest version of reveal.js from https://github.com/hakimel/reveal.js/releases   Unzip and replace the example contents in index.html with your own   Open index.html in a browser to view it <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Full setup</h3> Some reveal.js features, like external Markdown and speaker notes, require that presentations run from a local web server. The following instructions will set up such a server as well as all of the development tasks needed to make edits to the reveal.js source code. Install Node.js (4.0.0 or later)   Clone the reveal.js repository <div class=„highlight highlight-source-shell“><pre>$ git clone https://github.com/hakimel/reveal.js.git</pre>   Navigate to the reveal.js folder <div class=„highlight highlight-source-shell“><pre>$ cd reveal.js</pre>   Install dependencies <div class=„highlight highlight-source-shell“><pre>$ npm install</pre>   Serve the presentation and monitor source files for changes <div class=„highlight highlight-source-shell“><pre>$ npm start</pre>   Open http://localhost:8000 to view your presentation You can change the port by using

```
npm start -- --port=8001
```

. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Folder Structure</h3> css/ Core styles without which the project does not function js/ Like above but for JavaScript plugin/ Components that have been developed as extensions to reveal.js lib/ All other third party assets (JavaScript, CSS, fonts) <h2><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>License</h2> MIT licensed Copyright (C) 2017 Hakim El Hattab, http://hakim.se </article> </html>