

# 造轮子之从填坑到挖坑

## reveal.js (HTML 演示文稿) 中文文档

📅 2016-09-03 | 📁 [manual](#) | 📄 127

📖 8,835 | ⌚ 36

最近发现了一个可以用 HTML+CSS3 就能做出一份优美的 PPT 的 JavaScript 框架 (reveal.js)，但是我并没有找到该框架的中文帮助文档，所以在这里将提供中文帮助文档（本人的英文水平并不是很好，所以客官们凑合着看，顺便添加一些相关内容）

### 英文帮助文档地址

 [reveal.js 英文帮助文档](#)

## reveal.js 介绍

### 是什么？

- 一个使用 HTML 轻松创建精美的演示文稿框架，你只要有一个支持 CSS 3D 切换的浏览器。。[点击查看 demo](#)
- reveal.js 配备了广泛的功能，包括嵌套幻灯片，Markdown 内容，PDF 导出，演讲笔记和 JavaScript API。还有一个全功能的可视化编辑器和平台：[slides.com](#)。

### 特点

- 支持标签来区分每一页幻灯片
- 可以使用 Markdown 来写内容
- 支持 PDF 的导出
- 支持演说注释
- 提供 JavaScript API 来控制页面
- 提供了多个默认主题和切换方式

### 幻灯片实现步骤

1. 从 [reveal.js](#) 上下载压缩包，并解压
2. 进入 reveal.js 文件夹，直接修改 index.html 文件就可以

### 在线编辑

演示文档是使用 HTML 或者 Markdown 编写的，如果你们更喜欢图形界面的在线编辑器，点击 [slides.com](https://slides.com) 尝试一下。

## 写法说明

### HTML 实现

幻灯片的内容需要包含在 `<div class="reveal"> <div class="slides">` 的标签中。

一个 section 是一页幻灯片。当 section 包含在 section 中时，是一个纵向的幻灯片。实际上就是 `.reveal > .slides > section` 这样子结构的。

怎么理解呢？可以这样理解：横向的幻灯片代表一章，纵向的幻灯片代表一章中的一节。那么横向的幻灯片在播放时是左右切换的，而纵向的幻灯片是上下切换的。

For Example:

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

LANG-HTML | COPY

### HTML 实现内容

#### 标题和正文

section 中的内容就是幻灯片的内容，你可以使用 `h2` 标签表示标题，`p` 表示内容。需要改变颜色的只需 `style="color: ”`。

当某一页需要特殊背景色，可以使用 `data-background` 在 section 上设置，`data-background-transition` 表示背景过渡效果。

For Example:

```
<section data-background-transition="zoom" data-background="#dddddd">
<section>
```

LANG-HTML | COPY

如果需要正文一段一段出现。可以使用 `fragment`。

For Example:

```
<p class="fragment"></p>
```

LANG-HTML | COPY

## 代码

reveal.js 使用 highlight.js 来支持代码高亮。可以直接写 `code` 标签来实现, `data-trim` 表示去除多余的空格。

For Example:

```
<pre>
  <code data-trim>
    console.log('hello reveal.js!');
  </code>
</pre>
```

LANG-HTML | COPY

## 注释

在演示文稿里可能会用到注释, 对于注释, 可以通过 `<aside class="notes">` 来实现。

For Example:

```
<aside class="notes">
  <!-- 这里是注释。 -->
</aside>
```

LANG-HTML | COPY

在幻灯片页面, 按下 `s` 键, 就可以调出注释页面, 注释页面包含了当前幻灯片, 下一章幻灯片, 注释, 以及幻灯片播放时间。

## Markdown 实现

reveal.js 不仅支持 html 表示来实现内容, 还可以通过 Markdown 来实现内容。使用 Markdown 实现内容时, 需要对 `section` 标示添加 `data-markdown` 属性, 然后将 Markdown 内容写到一个 `text/template` 脚本中。

For Example:

```
<section data-markdown>
  <script type="text/template">
    ## Page title

    A paragraph with some text and a [link](//hakim.se).
  </script>
</section>
```

## 背景色

fragment 功能的实现，可以通过注释来实现。这是一个特殊的语法（在 HTML 中注释）。

For Example:

```
<section data-markdown>
  <script type="text/template">
    <!-- .slide: data-background="#ff0000" -->
    - Item 1 <!-- .element: class="fragment" data-fragment-index="2" -->
    - Item 2<!-- .element: class="fragment" data-fragment-index="1" -->
  </script>
</section>
```

## 外置 Markdown 文件

reveal.js 可以引用一个外置的 Markdown 文件来解析。data-charset 属性是可选的，它指定加载外部文件时使用的字符集。

当在本地使用，此功能要求 reveal.js 从本地 Web 服务器中运行。本地 Web 服务器推荐使用 Node.js。

For Example:

```
<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>
```

## 分页实现

一个 Markdown 文件中可以连续包含多个章内容。可以在 section 中通过属性 data-separator, data-separator-vertical 来划分章节。

For Example:

```
<section data-separator="---" data-separator-vertical="--" >
  <script type="text/template">
    # 主题1
    - 主题1-内容1
    - 主题1-内容2
    --
    ## 主题1-内容1
    内容1-细节1
    --
    ## 主题1-内容2
    内容1-细节2
    ---
    # 主题2
  </script>
</section>
```

## 注释

对 section 添加 data-separator-notes="^Note:" 属性, 就可以指定 Note: 后面的内容为当前幻灯片的注释。

For Example:

```
# Title
```

LANG-MARKDOWN | COPY

```
## Sub-title
```

```
Here is some content...
```

```
Note:
```

```
This will only display in the notes window.
```

## 基础配置项

在页面的最后, 你需要初始化一些配置项, 请注意, 所有的配置值是可选的。以下指定的内容, 是默认的配置值, 可以根据需求修改。

Example Configuration:

```
Reveal.initialize({
  // 是否在右下角展示控制条
  controls: true,

  // 是否显示演示的进度条
  progress: true,

  // 是否显示当前幻灯片的页数编号, 也可以使用代码 "s/t" , 表示当前页/总页数。
  slideNumber: false,
```

LANG-JAVASCRIPT | COPY

```
// 是否将每个幻灯片改变加入到浏览器的历史记录中去
history: false,

// 是否启用键盘快捷键来导航
keyboard: true,

// 是否启用幻灯片的概览模式, 可使用 "Esc" 或 "o" 键来切换概览模式
overview: true,

// 是否将幻灯片垂直居中
center: true,

// 是否在触屏设备上启用触摸滑动切换
touch: true,

// 是否循环演示
loop: false,

// 是否将演示的方向变成 RTL, 即从右往左
rtl: false,

// 是否每次演示的时候, 随机幻灯片的顺序
shuffle: false,

// 全局开启和关闭碎片。
fragments: true,

// 标识演示文稿是否在嵌入模式中运行, 即包含在屏幕的有限部分中的
embedded: false,

// 标识当问号键被点击的时候是否应该显示一个帮助的覆盖层
help: true,

// 标识演讲者备注标志是否让所有观看者可见
showNotes: false,

// 两个幻灯片之间自动切换的时间间隔 (毫秒)
// 当设置成 0 的时候则禁止自动切换
// 该值可以被幻灯片上的 "data-autoslide" 属性覆盖
autoSlide: 0,

// 当遇到用户输入的时候停止自动切换
autoSlideStoppable: true,

// 当自动滑动时, 使用此方法进行导航。
autoSlideMethod: Reveal.navigateNext,

// 是否启用通过鼠标滚轮来导航幻灯片
mouseWheel: false,

// 是否在移动设备上隐藏地址栏
hideAddressBar: true,
```

```
// 是否在一个弹出的 iframe 中打开幻灯片中的链接
previewLinks: false,

// 切换过渡效果
transition: "default", // none/fade/slide/convex/concave/zoom

// 过渡速度
transitionSpeed: "default", // default/fast/slow

// 全屏幻灯片背景的过渡效果
backgroundTransition: "default", // none/fade/slide/convex/concave/zoom

// 加载除当前可见的幻灯片之外的幻灯片数量
viewDistance: 3,

// 视差背景图片
parallaxBackgroundImage: "",
// e.g. '///s3.amazonaws.com/hakim-static/reveal-js/reveal-parallax-1.jpg'

// 视差背景尺寸
parallaxBackgroundSize: "", // CSS syntax, e.g. "2100px 900px"

// 移动视差背景 (水平和垂直) 滑动变化的数量, 例如 100
// - 除了指定自动计算
// - 设置为 0 时, 禁止沿轴运动
parallaxBackgroundHorizontal: null,
parallaxBackgroundVertical: null
});
```

这些配置使用 `configure` 方法将会被更新。

For Example:

```
// 关闭自动切换
Reveal.configure({ autoSlide: 0 });

// 开启每 5 秒自动切换一次
Reveal.configure({ autoSlide: 5000 });
```

LANG-JAVASCRIPT | COPY

## 演示文稿的大小

所有的演示稿都有一个正常大小, 这是他们所撰写的分辨率。该框架统一在此基础上自动缩放并规模演示, 确保一切都适合任何于给定的屏幕上。

见下文的相关的大小配置选项, 包括默认值的列表。

Example Configuration:

```
Reveal.initialize({
  ...

  // 演示稿的“正常”的大小。
  // 当演示稿被缩放时，将会适应不同的分辨率，而宽高比也将被保留。
  // 可以用百分比单位指定。
  width: 960,
  height: 700,

  // 显示大小应该考虑到在内容的外围保留一些空白。
  margin: 0.1,

  // 应用到内容的最大最小的设置项
  minScale: 0.2,
  maxScale: 1.5
});
```

## 依赖

Reveal.js 不依赖于任何第三方的脚本，但一些可选库默认被包含。这些库在它们被加载时出现。

For Example:

```
Reveal.initialize({
  dependencies: [
    // 完全实现跨浏览器: classList
    // https://github.com/eligrey/classList.js/
    {
      src: "lib/js/classList.js",
      condition: function() {
        return !document.body.classList;
      }
    },

    // 在 section 标签中解读 Markdown
    {
      src: "plugin/markdown/marked.js",
      condition: function() {
        return !!document.querySelector("[data-markdown]");
      }
    },
    {
      src: "plugin/markdown/markdown.js",
      condition: function() {
        return !!document.querySelector("[data-markdown]");
      }
    },

    // 在 code 标签中高亮代码
    {
```



```
    src: "plugin/highlight/highlight.js",
    async: true,
    callback: function() {
        hljs.initHighlightingOnLoad();
    }
},

// 按住 Alt+点击 实现放大和缩小
{
    src: "plugin/zoom-js/zoom.js",
    async: true
},

// 演讲者备注
{
    src: "plugin/notes/notes.js",
    async: true
},

// MathJax
{
    src: "plugin/math/math.js",
    async: true
}
]
});
```

你可以使用相同的语法添加自己的扩展。以下属性可用于每个依赖的对象当中：

- **src:** 脚本加载的路径。
- **async:** 是否异步，可选标志。本是否在 reveal.js 之后开始加载，默认值为 false。
- **callback:** 回调方法，可选方法。当脚本加载是执行。
- **condition:** 返回条件，可选方法。为要加载的脚本返回 true。

## Ready 事件

当 reveal.js 已加载了所有非异步的依赖，并准备开始导航时被 ready 事件被触发。要检查 reveal.js 是否已经 ready，你可以调用 Reveal.isReady() 方法。

For Example:

```
Reveal.addEventListener("ready", function(event) {
    // event.currentSlide, event.indexh, event.indexv
});
```

LANG-JAVASCRIPT | COPY

## 自动滑动

演示文稿可以配置为通过自动滑动，而无需用户的任何输入。为了启用这个，你需要告诉框架幻灯片之间应该间隔多少毫秒进行切换。

For Example:

```
// 5 秒切换间隔
Reveal.configure({
  autoSlide: 5000
});
```

LANG-JAVASCRIPT | COPY

当启用时，会出现让用户可以暂停和恢复自动滑动的控件。另外，滑动可以按键盘上 A 键进行恢复和暂停。用户开始手动导航时就会自动暂停滑动。你可以通过 reveal.js 的配置项指定禁用控件：autoSlideStoppable: false。

你也可以在幻灯片上设置 data-autoslide 属性进行覆盖。

For Example:

```
<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>
```

LANG-HTML | COPY

要覆盖用于当自动滑动导航的方法，你可以指定 autoSlideMethod 的设置项。默认只能沿着头层导航而忽略垂直滑动，这里可以设置为：Reveal.navigateRight。

每当自动幻灯片模式恢复或者暂停时，autoslideresumed 和 autoslidepaused 事件会被释放。

## 键盘绑定

如果你不满意任何默认键盘绑定的，你可以使用 keyboard 配置选项来覆盖它们。

For Example:

```
Reveal.configure({
  keyboard: {
    13: "next", // 按下 Enter 键切入下一张幻灯片
    27: function() {}, // 当 Esc 键被按下时执行的方法
    32: null // 当按下 SPACE 键时没有做任何事情 (即禁用 reveal.js 默认绑定)
  }
});
```

LANG-JAVASCRIPT | COPY

## 触摸导航

你可以通过在任何支持触摸的设备上滑动来演示导航。支持水平和垂直滑动实现幻灯片之间切换。如果你希望禁用这个功能，可以在初始化 reveal.js 设置：touch: false。

如果有你的内容的某些部分仍然可以访问触摸事件，你需要在元素中添加 data-prevent-swipe 属性来突出这个。一个非常有用的元素需要被滚动是个常见的例子。

## 延迟加载

在演示文稿时，大量的媒体文件或者 iframe 里的内容，被延迟加载是很重要的。延迟加载意味着 reveal.js 只会加载当前幻灯片最接近的几张幻灯片内容。被预加载的幻灯片是由配置项中的 viewDistance 的值所决定的。

要启用延迟加载，你需要在 data-src 中改变 src 属性。这是支持 image, video, audio 和 iframe 元素。当包含不再长时间可见的幻灯片，延时加载的 iframes 将会被卸载。

For Example:

LANG-HTML | COPY

```
<section>
  
  <iframe data-src="//hakim.se"></iframe>
  <video>
    <source data-src="video.webm" type="video/webm">
    <source data-src="video.mp4" type="video/mp4">
  </video>
</section>
```

## API

Reveal 对象公布了 JavaScript API 用来控制导航和阅读状态。

JavaScript API

LANG-JAVASCRIPT | COPY

```
// 导航
Reveal.slide(indexh, indexv, indexf);
Reveal.left();
Reveal.right();
Reveal.up();
Reveal.down();
Reveal.prev();
Reveal.next();
Reveal.prevFragment();
Reveal.nextFragment();

// 随机幻灯片的顺序
```

```
Reveal.shuffle();

// 切换演示状态, 通过 true/false 来控制 开/关
Reveal.toggleOverview();
Reveal.togglePause();
Reveal.toggleAutoSlide();

// 在运行时改变一个配置值
Reveal.configure({ controls: true });

// 返回当前的配置值
Reveal.getConfig();

// 获取当前演示文稿的规模, 即大小
Reveal.getScale();

// 检索之前和当前幻灯片元素
Reveal.getPreviousSlide();
Reveal.getCurrentSlide();

Reveal.getIndices(); // { h: 0, v: 0 } }
Reveal.getProgress(); // 0-1
Reveal.getTotalSlides();

// 返回演讲者当前幻灯片的备注
Reveal.getSlideNotes();

// 返回当前状态
Reveal.isFirstSlide();
Reveal.isLastSlide();
Reveal.isOverview();
Reveal.isPaused();
Reveal.isAutoSliding();
```

## 幻灯片更改事件

'slidechanged' 事件在每次幻灯片滑动时被释放 (无论什么状态)。这个事件对象包含当前幻灯片的索引值, 以及之前和当前幻灯片的 HTML 节点作为参考。

一些库, 如 MathJax (见 [#226](#)), 可以获取幻灯片的变化和显示状态带来的困惑。很多时候, 可以在回调中通过调用更新或者给予方法来确立。

For Example:

```
Reveal.addEventListener("slidechanged", function(event) {                                LANG-JAVASCRIPT | COPY
    // event.previousSlide, event.currentSlide, event.indexh, event.indexv
});
```

## 演示状态

演示文稿的当前状态可以通过使用的 `getState` 方法获取。 `state` 对象包含着后面所有演示所需的信息，因此它第一次被称为 `getSate` 。这有一点像快照。它可以很容易地字符串化和持久化或通过网络发送一个简单的对象。

For Example:

```
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
```

LANG-JAVASCRIPT | COPY

## 幻灯片状态

如果你在幻灯片的 `section` 中设置 `data-state="somestate"` ，当幻灯片被打开时， `"somestate"` 将会应用一个 `class` 在文档元素上。这允许你使用广泛的样式变化应用到基于活动的幻灯片页面。

此外，你还可以通过 `JavaScript` 监听这些变化状态。

For Example:

```
Reveal.addEventListener(  
  "somestate",  
  function() {  
    // TODO: Sprinkle magic  
  },  
  false  
);
```

LANG-JAVASCRIPT | COPY

## 幻灯片背景

幻灯片默认包含在屏幕的有限范围之内，这允许其适应任何的显示屏并且均匀的缩放。你可以在 `section` 元素上添加 `data-background` 属性，定义幻灯片区域之外的整页背景。拥有四种不同类型的背景支持：`color` , `image` , `video` 和 `iframe` 。

## 颜色背景

所有 `CSS` 颜色都将支持，如 `rgba()` 或者 `hsl()` 。

For Example:

```
<section data-background-color="#ff0000">
  <h2>Color</h2>
</section>
```

LANG-HTML | COPY

## 图片背景

默认情况下，背景图片将会缩放至覆盖整个页面。可用选项：

Attribute	Default	Description
data-background-image		显示图像的 URL 地址。当幻灯片打开时，GIF 重新启动。
data-background-size	cover	在 MDN 上查看 <a href="#">background-size</a>
data-background-position	center	在 MDN 上查看 <a href="#">background-position</a>
data-background-repeat	no-repeat	在 MDN 上查看 <a href="#">background-repeat</a>

For Example:

```
<section data-background-image="//example.com/image.png">
  <h2>Image</h2>
</section>
<section data-background-image="//example.com/image.png"
  data-background-size="100px"
  data-background-repeat="repeat">
  <h2>This background image will be sized to 100px and repeated</h2>
</section>
```

LANG-HTML | COPY

## 视频背景

在幻灯片背后上自动播放全尺寸的视频。

属性	默认值	描述
----	-----	----

属性	默认值	描述
data-background-video		单个视频源，或者使用逗号分隔的视频源列表
data-background-video-loop	false	设置视频是否重复播放
data-background-video-muted	false	设置视频是否静音

For Example:

```
<section data-background-video="//s3.amazonaws.com/static.slid.es/site/homepage/v1/homepage-video-edit" data-background-video-loop data-background-video-muted>
  <h2>Video</h2>
</section>
```

LANG-HTML | COPY

## Iframe 背景

嵌入了网页作为背景。请注意，由于 `iframe` 是在背景层，即幻灯片的后面，我们是不可能与嵌入式网页交互。

For Example:

```
<section data-background-iframe="//slides.com">
  <h2>Iframe</h2>
</section>
```

LANG-HTML | COPY

## 背景切换

默认情况下，使用一个渐变的动画背景进行过渡。这可以在 `Reveal.initialize()` 申明 `backgroundTransition`：'slide' 改变滑动过渡效果，另外，你可以在 `section` 中指定 `data-background-transition`，衣服该指定的背景过渡。

## 视差背景

如果你想使用视差滚动背景，可以在 `reveal.js` 初始化设置下面两项（另外两个是可选的）。

Example Configuration:

```
Reveal.initialize({
  // 视差背景图片地址
```

LANG-JAVASCRIPT | COPY

```
parallaxBackgroundImage: "",
// e.g. "///s3.amazonaws.com/hakim-static/reveal-js/reveal-parallax-1.jpg"

// 视差背景图片大小
parallaxBackgroundSize: "",
// CSS syntax, e.g. "2100px 900px"
// 目前只有像素支持 (不要使用百分比或者 auto)

// 每张幻灯片的背景视差移动像素数
// - 除了指定自动计算
// - 设置为 0 时, 禁止沿轴运动
parallaxBackgroundHorizontal: null,
parallaxBackgroundVertical: null
});
```

## 幻灯片切换

全局使用 `transition` 配置值设置演示文稿的切换方法。你可以通过添加 `data-transition` 属性来覆盖指定的全局幻灯片切换效果。

For Example:

```
<section data-transition="zoom">
  <h2>This slide will override the presentation transition and zoom!</h2>
</section>

<section data-transition-speed="fast">
  <h2>Choose from three transition speeds: default, fast or slow!</h2>
</section>
```

LANG-HTML | COPY

你也可以为进出同一张幻灯片使用不同的切换效果。

For Example:

```
<section data-transition="slide">
  The train goes on ...
</section>
<section data-transition="slide">
  and on ...
</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">
```

LANG-HTML | COPY



```
And it starts again.  
</section>
```

## 内部链接

很容易就可以实现幻灯片之间的链接。下面第一个例子是针对某一张幻灯片进行链接，而第二个，则用 ID 属性进行链接至另一张幻灯片。( <section id="some-slide"> )

For Example:

```
<a href="#/2/2">Link</a>                                LANG-HTML | COPY  
<a href="#/some-slide">Link</a>
```

你还可以添加相对导航链接，类似于内置在 reveal.js 的控件, 可以在任何元素上附加以下其中一个 class。需要注意的是，在它的基础上，当前的幻灯片一个有效的导航路线的每个元素将会被自动给予 enabled 类。

For Example:

```
<a href="#" class="navigate-left">                                LANG-HTML | COPY  
<a href="#" class="navigate-right">  
<a href="#" class="navigate-up">  
<a href="#" class="navigate-down">  
<a href="#" class="navigate-prev"> <!-- 之前幻灯片垂直或水平滑动 -->  
<a href="#" class="navigate-next"> <!-- 下一个幻灯片垂直或水平滑动 -->
```

## 片段

片段被用于突出幻灯片上单个元素。这类片段的每个元素都将在下一个幻灯片之前显示。这里有个例子：

<http://lab.hakim.se/reveal-js/#/fragments>。

默认片段风格是启动了淡入淡出效果。这种风格可以通过在片段中添加不同类而被改变不同效果。

For Example:

```
<section>                                                    LANG-HTML | COPY  
  <p class="fragment grow">grow</p>  
  <p class="fragment shrink">shrink</p>  
  <p class="fragment fade-out">fade-out</p>  
  <p class="fragment fade-up">fade-up (also down, left and right!)</p>  
  <p class="fragment current-visible">visible only once</p>  
  <p class="fragment highlight-current-blue">blue only once</p>  
  <p class="fragment highlight-red">highlight-red</p>  
  <p class="fragment highlight-green">highlight-green</p>  
  <p class="fragment highlight-blue">highlight-blue</p>  
</section>
```

多个片段可以通过包裹它而被应用到相同的元件顺序之中，这将在第一步骤中的文本淡入之后淡出第二个。

For Example:

LANG-HTML | COPY

```
<section>
  <span class="fragment fade-in">
    <span class="fragment fade-out">I'll fade in, then out</span>
  </span>
</section>
```

片段的显示顺序可以通过 `data-fragment-index` 属性来控制。

For Example:

LANG-HTML | COPY

```
<section>
  <p class="fragment" data-fragment-index="3">Appears last</p>
  <p class="fragment" data-fragment-index="1">Appears first</p>
  <p class="fragment" data-fragment-index="2">Appears second</p>
</section>
```

## 片段事件

当任一幻灯片片段显示或隐藏时 `reveal.js` 将派遣一个事件。

一些库，如 `MathJax` (见 [#505](#))，可以获取到最初隐藏幻灯片片段的困惑。很多时候，可以在回调中通过调用更新或者给予方法来确立。

For Example:

LANG-JAVASCRIPT | COPY

```
Reveal.addEventListener("fragmentshown", function(event) {
  // event.fragment = the fragment DOM element
});
Reveal.addEventListener("fragmenthidden", function(event) {
  // event.fragment = the fragment DOM element
});
```

## 代码语法高亮

默认情况下，`Reveal` 配置了 `highlight.js` 来支持代码语法高亮。下面的例子中，`clojure` 代码就使用了代码语法高亮。当 `data-trim` 属性存在时，周围的空格被自动删除。HTML 将被默认逃脱。为了避免这种情况，举个例子，如果你想要用 `mark` 标签高亮代码行，你得在 `code` 标签上指定 `data-noescape` 属性。

For Example:

```
<section>
  <pre>
    <code data-trim data-noescape>
      (def lazy-fib
        (concat
          [0 1]
          <mark>((fn rfib [a b]</mark>
            (lazy-cons (+ a b) (rfib b (+ a b)))) 0 1)))
    </code>
  </pre>
</section>
```

## 幻灯片页码

如果你想要显示当前的幻灯片页码，你得在配置中配置 `slideNumber` 的值。

Example Configuration:

```
// 显示默认格式的幻灯片编号
Reveal.configure({ slideNumber: true });

// 幻灯片编号格式可以用这些变量进行配置:
// "h.v": 水平.上下滑动数 (默认)
// "h/v": 水平/垂直幻灯片编号
// "c": 当前幻灯片编号
// "c/t": 当前幻灯片编号/总幻灯片数
Reveal.configure({ slideNumber: "c/t" });
```

LANG-JAVASCRIPT | COPY

## 概览模式

按“ESC”或“O”键来打开和关闭概览模式。当你在这种模式下，你仍然可以在幻灯片之间导航，因此你的演示文稿需要在千分尺以上。

For Example:

```
Reveal.addEventListener("overviewshown", function(event) {
  /* ... */
});
Reveal.addEventListener("overviewhidden", function(event) {
  /* ... */
});

// 编程切换概览模式
Reveal.toggleOverview();
```

LANG-JAVASCRIPT | COPY

## 全屏模式

只要按下键盘上的 `F` 键来全屏显示你的演示文稿。按 `Esc` 键退出全屏模式。

## 嵌入式媒体

嵌入式 HTML5 的 `video` 标签和引入的 `iframe` 会在你离开幻灯片时自动暂停。这可以通过在元素上用一个 `data-ignore` 属性来禁用它。

如果你想要在幻灯片显示时自动开始播放，你得在你的媒体元素上添加 `data-autoplay`。

For Example:

```
<video data-autoplay src="//clips.vorwaerts-gmbh.de/big_buck_bunny.mp4">
</video>
```

LANG-HTML | COPY

此外，该框架会自动得将两次 `post` 信息传递给所有的 `iframes`，当 `iframe` 里的幻灯片内容可见时使用 `slide:start`，当要隐藏时使用 `slide:stop`。

## 延伸元素

有时候，它需要一个元素，像 `image` 或者 `video`，延伸到一个给定的幻灯片中消耗尽可能多的空间。这可以通过在元素上添加 `.stretch` 类来完成。：

For Example:

```
<section>
  <h2>This video will use up the remaining space on the slide</h2>
  <video class="stretch" src="//clips.vorwaerts-gmbh.de/big_buck_bunny.mp4"></video>
</section>
```

LANG-HTML | COPY

限制：

- 只有幻灯片部分的直接后代可以被拉伸
- 只有每张幻灯片部分的一个后代可以被拉伸

## postMessage API

当与另一个窗口的演示文稿通信时，该框架有 `built-in` 的 `postMessage` API 来帮助你。这里是展示了，你要怎样使用 `reveal.js` 在给定的窗口进行滑动的例子。

For Example:

```
window.postMessage(JSON.stringify({ method: "slide", args: [2] })), "*")
```

LANG-JAVASCRIPT | COPY

当 `reveal.js` 在 `iframe` 内部运行时它可以选择性地在所有的事件的父级冒泡。冒泡事件字符串化 `JSON` 的三个领域：`namespace`，`eventName` 和 `state`。这里是你可以如何从父窗口订阅它们。

For Example:

```
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  
    }  
});
```

LANG-JAVASCRIPT | COPY

`cross-window` 可以在配置标志中开启或者关闭信息。

Example Configuration:

```
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  
});
```

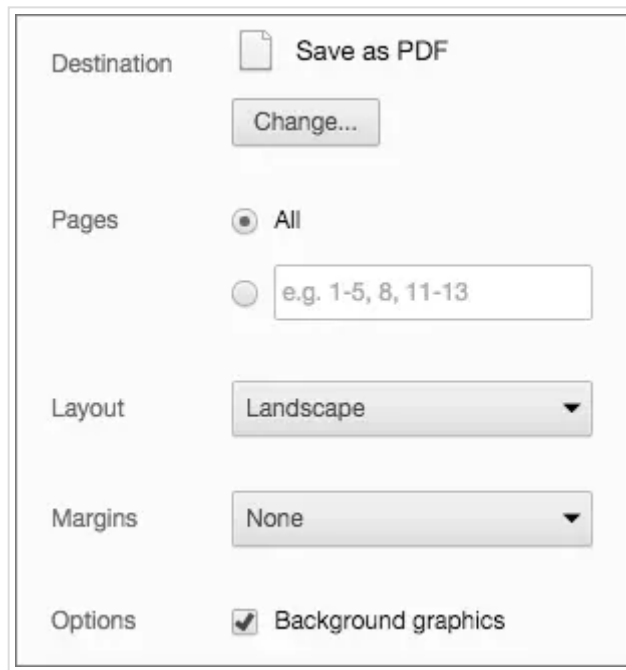
LANG-JAVASCRIPT | COPY

## PDF 导出

演示文稿可以通过特殊的打印样式表导出为 PDF。你需要在 [Google Chrome](#) 或者 [Chromium](#) 发送请求。这里是一个已经上传到 [SlideShare](#) 上的呈现的一个例子 <http://www.slideshare.net/hakimel/revealjs-300>

1. 打开你的演示文稿，在地址栏的查询字符串中添加 `print-pdf` 参数。这会默认的触发 `HTML` 加载 `PDF` 样式表 (`css/print/pdf.css`)。你可以在 [lab.hakim.se/reveal-js?print-pdf](#) 这里测试。
2. 在浏览器中打开打印对话框 (`CTRL/CMD+P`)。
3. 更改 **Destination** 设置为 **Save as PDF**。
4. 在 **Layout** 中更改为 **Landscape**。
5. 在 **Margins** 中更改为 **None**。
6. 启用 **Background graphics** 选项。
7. 点击 **Save**。

另外，你可以使用 `decktape` 选项。



The image shows a 'Save as PDF' dialog box with the following settings:

- Destination:** Save as PDF (with a 'Change...' button)
- Pages:** All (radio button selected, with a text input field showing 'e.g. 1-5, 8, 11-13')
- Layout:** Landscape (dropdown menu)
- Margins:** None (dropdown menu)
- Options:** Background graphics (checkbox checked)

我试过保存 PDF 的功能，有内容会重叠，导出的内容也是怪怪的，怀疑是样式没有处理好。

## 主题

这个框架默认包含着一些默认的主题。

- **black:** 黑色背景，白色文本，蓝色链接（默认主题）
- **white:** 白色背景，黑色文本，蓝色链接
- **league:** 灰色背景，白色文本，蓝色链接（reveal.js 小于 3.0.0 默认的主题）
- **beige:** 米色背景，深色文本，棕色链接
- **sky:** 蓝色背景，瘦黑文本，蓝色链接
- **night:** 黑色背景，亮白文本，橙色链接
- **serif:** 卡布奇诺背景，灰色文本，棕色链接
- **simple:** 白色背景，黑色文本，蓝色的链接
- **solarized:** 米色背景，深绿色文本，蓝色链接

每个主题可以作为一个独立的样式表。要更改主题，你只要在 HTML 中将以下代码中的 **black** 修改就可以了。

Example Configuration:

```
<link rel="stylesheet" href="css/theme/black.css" id="theme">
```

LANG-HTML | COPY

如果你想添加自己的主题，看这里的帮助：</css/theme/README.md>

reveal.js 提供了多种样式。可以通过引用不同的主题来实现。具体主题查看 `reveal.js/css/theme` 下的 `css` 文件。

## 演讲者备注

reveal.js 自带演讲者备注插件，它可以在一个单独的浏览器窗口中呈现每个幻灯片备注。即使你没有写任何注释，注释窗口呈现的下一个即将到来的幻灯片预览也会给你带来帮助。按你的键盘上的 S 键来开启备注窗口。

如下所示，备注只要在幻灯片中追加一个 aside 元素。如果你想要通过 Markdown 格式来书写你的备注，你可以在 aside 元素上添加 data-markdown 属性。

另外，你可以在幻灯片中使用 data-notes 属性添加你的备注。如：<section data-notes="Something important"></section>。

当在本地使用，此功能要求 reveal.js 从本地 Web 服务器中运行。本地 Web 服务器推荐使用 Node.js。

For Example:

```
<section>
  <h2>Some Slide</h2>

  <aside class="notes">
    Oh hey, these are some notes. They'll be hidden in your presentation, but you can see them
  </aside>
</section>
```

LANG-HTML | COPY

如果你想要引入外部的 Markdown 文件，你可以在一个特殊分隔符的帮助下添加注释：

For Example:

```
<section data-markdown="example.md"
  data-separator="^\n\n\n"
  data-separator-vertical="^\n\n" data-separator-notes="^Note:"></section>
```

LANG-HTML | COPY

```
# Title
## Sub-title

Here is some content...

Note:
This will only display in the notes window.
```

## 共享和打印演讲者备注

备注并不是在所有的观看者的视图中可见。如果你想与他人分享你的笔记，你可在初始化 reveal.js 时，设置 showNotes: true。备注将在演示文稿的底部出现。

当 showNotes 启用，你导出为 PDF 的笔记也包括在内。

## 演讲者备注服务器端

在某些情况下，可以期望在一个单独的设备上呈现并运行的注释。Node.js-based 备注插件，让你做到使用相同的备注定义它对应的客户端。通过添加以下的依赖关系来包含所需的脚本。

Example Configuration:

```
Reveal.initialize({
  ...

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

LANG-JAVASCRIPT | COPY

然后:

1. 下载 [Node.js](#) (1.0.0 或者最新)
2. 运行 `npm install`
3. 运行 `node plugin/notes-server`

## 复用

复用插件可以让你的观众，观看你控制自己的手机，平板电脑或笔记本电脑来演示幻灯片。作为主导航演示幻灯片，所有的用户演示将会实时更新。这里有个 demo : <http://reveal-js-multiplex-ccjbegmaili.now.sh/>。

复用插件需要以下三件事来操作:

1. 拥有控制权的主演讲文稿。
2. 客户端演示文稿跟随者主机
3. Socket.io 服务器从主到客户端广播事件

更多细节:

## 主演示文稿

只有主持人从一个静态文件服务器访问 (最好) 服务。这个只需要在你 (演示者) 的计算机上。(它的安全性是从你自己的电脑运行主演示文稿，因此网络得不停的下載。) 在你的主呈现的目录执行以下命令:

1. `npm install node-static`
2. `static`

如果你想要在你想要在你的主演示文稿用演讲者备注插件，就得确保你有演讲者备注插件并与沿着下面所示的配置正确配置，然后在你的主演示文稿目录下执行 `node plugin/notes-server`。下面的配置是导致它连接到 `socket.io` 服务器，以及启



动 speaker-notes/static-file 服务为主。

你可以用 `//localhost:1947` 访问你的主演示文稿。

Example Configuration:

LANG-JAVASCRIPT | COPY

```
Reveal.initialize({
  // 其他选项...

  multiplex: {
    // 举例值。 生成自己的, 在 socket.io 服务器上查看说明。
    secret: "13652805320794272084",
    // 从 socket.io 服务器上获取。 给予 (主) 演示文稿控制权
    id: "1ea875674b17ca76", // id, 从 socket.io 服务器上获取
    url: "//reveal-js-multiplex-ccjbegmaii.now.sh" // socket.io 服务器地址
  },

  // 不要忘记添加依赖
  dependencies: [
    { src: "//cdn.socket.io/socket.io-1.3.5.js", async: true },
    { src: "plugin/multiplex/master.js", async: true },

    // 如果你想广播备注
    { src: "plugin/notes-server/client.js", async: true }

    // 其他依赖...
  ]
});
```

## 客户端演示文稿

服务器来自公共的无障碍的静态文件服务器。包含示例, GitHub Pages, Amazon S3, Dreamhost, Akamai, 等等。更可靠, 更好。你的观众则可以通过访问客户端的演示文稿:

`http://example.com/path/to/presentation/client/index.html`, 下面的配置是用来连接到 socket.io 服务器的客户端。

Example configuration:

LANG-JAVASCRIPT | COPY

```
Reveal.initialize({
  // 其他选项...

  multiplex: {
    // 举例值。 生成自己的, 在 socket.io 服务器上查看说明。
    secret: null, // 空, 因为客户端不必需要主演示文稿控制权
    id: "1ea875674b17ca76", // id, 在 socket.io 服务器获取
    url: "//reveal-js-multiplex-ccjbegmaii.now.sh" // socket.io 服务器地址
  },
```

```
// 不要忘记添加依赖
dependencies: [
  { src: "//cdn.socket.io/socket.io-1.3.5.js", async: true },
  { src: "plugin/multiplex/client.js", async: true }

  // 其他依赖..
]
});
```

## socket.io 服务器

从主演示文稿服务器上获取 `slideChanged` 事件，并在连接的客户端播放演示文稿。这需要公开访问。你可以在 `socket.io` 服务器上运行以下命令：

1. `npm install`
2. `node plugin/multiplex`

或者你在 `socket.io` 服务器上使用 <http://reveal-js-multiplex-ccjbegmaii.now.sh/>。

你需要在你的主机和客户端演示文稿中生成一个唯一的秘密和令牌。要做到这一点，访问 `//example.com/token`，而 `//example.com/` 是你的 `socket.io` 服务器链接地址。或者你在 `socket.io` 服务器去用 <http://reveal-js-multiplex-ccjbegmaii.now.sh/>，访问 <http://reveal-js-multiplex-ccjbegmaii.now.sh/token>。

你可以在你的 `socket.io` 服务器运行 <http://reveal-js-multiplex-ccjbegmaii.now.sh/>，但可用性和稳定性不能保证。对于任何关键任务，我建议你运行你自己的服务器。这是简单部署到 `nodejitsu`，`Heroku`，您自己的环境，等等。

## socket.io 服务器是文件静态服务器

该 `socket.io` 服务器可以作为播放静态文件服务器，为您的客户演示，这里有个例子：<http://reveal-js-multiplex-ccjbegmaii.now.sh/>。（在两个浏览器打开 <http://reveal-js-multiplex-ccjbegmaii.now.sh/>，通过在一个幻灯片浏览，并且其他的将匹配更新。）

Example configuration:

```
Reveal.initialize({
  // 其他选项...

  multiplex: {
    // 举例值。 生成自己的，在 socket.io 服务器上查看说明。
    secret: null, // 空，因为客户端不必需要主演示文稿控制权
    id: "1ea875674b17ca76", // id，在 socket.io 服务器获取
    url: "example.com:80" // socket.io 服务器地址
  },

  // 不要忘记添加依赖
```

LANG-JAVASCRIPT | COPY

```
dependencies: [
  { src: "//cdn.socket.io/socket.io-1.3.5.js", async: true },
  { src: "plugin/multiplex/client.js", async: true }

  // 其他依赖...
];
});
```

它也可以作为播放静态文件服务器，为您的主演示和客户端在同一时间演示（只要你不希望使用演讲者备注）。在两个浏览器打开 <http://reveal-js-multiplex-cjbegmaili.now.sh/>，通过在一个幻灯片浏览，并且其他的将匹配更新。）这可能是不可取的，因为你不希望你的听众与你呈现的幻灯片混淆。

Example configuration:

```
Reveal.initialize({
  // 其他选项...

  multiplex: {
    // 举例值。 生成自己的, 在 socket.io 服务器上查看说明。
    secret: "13652805320794272084",
    // 从 socket.io 服务器上获取。 给予 (主) 演示文稿控制权
    id: "1ea875674b17ca76", // id, 在 socket.io 服务器获取
    url: "example.com:80" // socket.io 服务器地址
  },

  // 不要忘记添加依赖
  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 }

    // 其他依赖...
  ]
});
```

LANG-JAVASCRIPT | COPY

## MathJax

如果你想在演示文稿中显示的数学方程，你可以通过包含这个插件。该插件是围绕 [MathJax](#) 库的一个非常薄的包装。要使用它，你需要将它作为一个 `reveal.js` 依赖，[这里有更多关于这个依赖的解释](#)。

该插件默认使用 [LaTeX](#)，但是你可以通过调整配置对象中的 `math`。注意，`MathJax` 是从远程服务器加载的。如果你想要离线使用，你需要下载并且复制到目录，然后调整 `mathjax` 的配置值。

下面是插件如何被配置的例子。如果你不打算改变这些值，你不必包含 `math` 配置对象里的所有内容。

```
Reveal.initialize({
  // 其他选项 ...
```

LANG-JAVASCRIPT | COPY

```
math: {  
  mathjax: "//cdn.mathjax.org/mathjax/latest/MathJax.js",  
  config: "TeX-AMS_HTML-full"  
  // //docs.mathjax.org/en/latest/config-files.html  
},  
  
dependencies: [{ src: "plugin/math/math.js", async: true }]  
});
```

如果你需要 [HTTPS delivery](#) 或者更加稳定的 [specific versions](#) , 请阅读 [MathJax](#) 的文档。

## 安装

基本设置仅用于创作演示文稿。你可以访问所有 `reveal.js` 功能和插件, 需要完整的配置, 如演讲者备注, 以及做出更改源所需的开发任务。

## 基本设置

`reveal.js` 的核心是非常容易安装。你只需要直接在浏览器下载复制该资源库, 并打开 `index.html` 文件。

1. 从 <https://github.com/hakimel/reveal.js/releases> 下载最新版本的 `reveal.js`
2. 解压缩并用自己的 `index.html` 替换例子中的内容
3. 在浏览器打开 `index.html` 并查看它

## 全部设置

一些 `reveal.js` 特征, 如外部的 Markdown 和演讲这备注, 要求演示文稿从本地 Web 服务器上运行。以下说明将设立这样的服务器, 以及所有的进行编辑的 `reveal.js` 源代码所需的开发服务。

1. 下载 [Node.js](#) (1.0.0 或者更新)
2. 克隆 `reveal.js` 目录

```
$ git clone https://github.com/hakimel/reveal.js.git
```

LANG-BASH | COPY

1. 进入 `reveal.js` 目录导航

```
$ cd reveal.js
```

LANG-BASH | COPY

1. 安装依赖

```
$ npm install
```

LANG-BASH | COPY

1. 演示文稿的服务器和监视源文件的修改

```
$ npm start
```

1. 打开 <<http://localhost:8000>> 来观看你的演示文稿

你可以使用 `npm start -- --port 8001` 修改端口。

## 目录结果

- **css/** 核心样式, 如果没有这些则项目不起作用
- **js/** 和上面一样, 但对 JavaScript 而言
- **plugin/** 已经作为开发扩展 reveal.js 部件
- **lib/** 所有第三方资源库(JavaScript, CSS, fonts)

## 总结

用 reveal.js 来实现幻灯片, 可以只关注内容, 忽略各种切换效果。而且可以使用 Markdown 来实现, 大大提高了编写效率。对于最后生成的 html 文件, 可以部署到服务器, 这样只需要网络就可以进行分享, 不需要使用 U 盘拷来拷去了。

亲!!! 听说给作者打赏一杯咖啡钱, 会给自己带来好运哦!

打赏

**本文作者:** 星火燎原@vxhly

**本文链接:** <https://vxhly.github.io/2016/09/reveal-js-cn-document/>

**版权声明:** 本博客所有文章除特别声明外, 均采用 [CC BY-NC-SA 3.0](#) 许可协议。转载请注明出处!

# JavaScript   # CSS   # Code   # HTML5   # Node.js

◀ 解读 npm 中的 package.json

Linux 学习笔记 (一): 目录结构说明 ▶

© 2018 星火燎原@vxhly | 143.7k

由 Hexo 强力驱动 | 主题 — NexT.Mist v5.1.4

3155 | 5060