Reveal.js Tutorial

by Kostas and Craig

March 9, 2016

In this tutorial we will create a presentation by constructing everything from scratch: HTML, CSS, Markdown and JavaScript. Before we start the tutorial it is useful to have a quickly go through the main elements of web-page building.

1 Crash-course

These *cheat-sheets* have been built from resources available on *CodeAcademy* and the *Mozilla Developer Network* which are also recommended for more in-depth training and resources. An excellent online book is also available on `http://eloquentjavascript.net/`.

We decided that the quickest way to approach this is to provide commented samples so we can quickly build a simple web-page.

1.1 HTML

HTML (*HyperText Markup Language* is used by browsers to communicate with each other and display data in an attractive fashion. It has a tree-like structure similar to that of XML. Each branch defines a section of a web page is started by some text in brackets `<branch>` and ends with a **backward** slash and the text `<\branch>`. Some objects are added in self-stopping branches by using a **forward** slash before closing the bracket `<branch/>`

```
<!doctype html>
<html>

<head>
        <title>Amazed</title>|
        <link rel="stylesheet" type="text/css"
                href="coolStyle.css"/>
        <script src=path/to/.js></script>
</head>

<body>

   <h1>Amazing Presentations Ltd.</h1>

   <div class="wow">

     <h2>Amazing discovery.</h2>

     <p>Something that changes the world.</p>1.2

     <a href="otherPageLoc.html">Lookie here.</a>

   </div>

   <p id = "footer">&copy;NGCM Ltd.</p>

<script>
document.getElementById('myID')
        .property = fun(input
</script>
</body>
</html>
```

Both do the same thing. Only the first is necessary in HTML v$\geqslant$ 4

The branch specifies information that is first read by the browser to correctly display the page. This is where JavaScript source files and CSS files are conventionally imported. It normally contains other information such as tags/titles/authors/etc used by search engines

This is the part where the data goes that is displayed to the user.

Display text using heading 1

Create a page division with special display properties.

Use paragraph formatting

Create a link

Usually after the body section some more scripts are added to process the HTML data

1.2 CSS

CSS or Cascading Style Sheets are commonly used in web applications to standardise formatting and styles. It is also used in reveal.js to set-up a presentation template.

```
body {
        height: 100%;
        margin: 0;
        text-align: center;
        width: 100%;
}
```

Specifies the general layout of the page.

```
p {
        font-size: 2rem;
}
```

Set the font size of text inside `<p></p>`

```
.wow {
  font-family: Helvetica, sans-serif;
  background-image: url("cool.jpg");
  background-size: cover;
  color: #ffffff
}
```

Specify properties of a page division, i.e. a `<div>` part of a page that should have special properties (e.g. a background image)

```
.wow a {
        color: #00FFAA;
        text-decoration: none;
        font-size: 1.25em;
}
```

Inside a division it might be necessary to want to change the way a link is formatted (the `<a>` inside a `<div>`.

```
#footer{
  font-size:0.75rem;
}
```

If a specific paragraph's has to be formatted differently (e.g. change only the colour) you can give it an ID and set it up here.

1.3 JavaScript

HTML contains static content and JavaScript is the way by which a web-page becomes dynamic. This is a scripting language that is interpreted by your browser and it is able to do what most programming languages do: mathematical operations, run loops, functions and create objects. The most important difference between JS and other languages is the way in which objects are defined (1), which is similar to the way in which any variable is defined (2). Functions are defined by calling a function 'constructor' that is assigned to a variable name with a set of inputs (2). The `for-loop` (4) and `if-statement` (3) syntax is similar to that of other languages. It is important to note that while a **';'** is not necessary in normal program flow, it is strongly recommended especially inside functions, loops and flow control statements.

Listing 1: JS Object

```
var myObj = {
        myName: "Feeg",
        myNumbers: [2,3.5],
        myFun: function(someVar){
                return otherVar;
        }
}
```

Listing 2: Variables and function definition

```
var myNum = 999;
var myBool = false;
var myStr = "Good!";
var myFun = function(input){
        some actions on input;
        return ouput;
}
```

Listing 3: If statement

```
if (1 in [0,2,3]) {
        compute a thing;
}else{
        compute nothing;
}
```

Listing 4: For loop

```
for (var i=0, i<n, i++){
        change something;
        compute something;
}
for i in [element1, element2, ...]{...}
```

JavaScript is completely different from Java

1.4 Markdown

Markdown is very useful for writing content for web-pages books and other types of documents. The language is similar to LaTeXin the sense that it uses plain text to write the source for a nicely formatted document, but it has the advantage of having very human-readable source code. The best way to write markdown is to use a specialised editor such as Atom, but in this tutorial we will use Brackets with a plugin for MD live generation. Markdown is so simple that an actual tutorial is not required for the basics and the cheat-sheet below should prove sufficient (for a step-by-step 10 minute tutorial check `http://commonmark.org/help/tutorial/`):

Type	Or	... to Get
`*Italic*`	`_Italic_`	*Italic*
`**Bold**`	`__Bold__`	**Bold**
`# Heading 1`	`Heading 1` `=========`	Heading 1
`## Heading 2`	`Heading 2` `---------`	Heading 2
`[Link](http://a.com)`	`[Link][1]` ⋮ `[1]: http://b.org`	Link
`![Image](http://url/a.png)`	`![Image][1]` ⋮ `[1]: http://url/b.jpg`	
`> Blockquote`		Blockquote
`A paragraph.` `A paragraph after 1 blank line.`		A paragraph. A paragraph after 1 blank line.
`* List` `* List` `* List`	`- List` `- List` `- List`	• List • List • List
`1. One` `2. Two` `3. Three`	`1) One` `2) Two` `3) Three`	1. One 2. Two 3. Three
`Horizontal Rule` `---`	`Horizontal Rule` `***`	Horizontal Rule
`` `Inline code` with backticks ``		`Inline code` with backticks
`` ``` # code block print '3 backticks or' print 'indent 4 spaces' ``` ````	`    # code block` `    print '3 backticks or'` `    print 'indent 4 spaces'`	`# code block` `print '3 backticks or'` `print 'indent 4 spaces'`

2 Tutorial

The tutorial exercise is to create a presentation by yourself from the bottom-up. [1]

- Start with the basic HTML structure, i.e. the `<html>` tags with `<head>`, `<body>`, etc.
- Add the reveal.js stylesheet and your own in the ¡head¿ branch.
- Add the *reveal* `<div>` and inside it the *slides* `<div>`
- Add about five sections inside the slides `<div>`
- Create two JavaScript sections to call *reveal.js* and *head.min.js* (this is needed so that it works in all browsers),found in `./reveal.js/js/` and `./reveal.js/lib/js/`, respectively.

[1] The above structure does not have to be necessarily respected, but it is recommended for cross-compatibility among browsers and connection speed (this unlikely to be a problem), since HTML is normally loaded progressively, it is necessary to have all the elements required for the next processing steps)

- After the `<slides>` section add a new `<script>` section to initialise the presentation with the following add-ons:

```
Reveal.initialize({

    controls: true, // Display controls in the bottom right corer
    history: true, // Push each slide change to the browser history
    center: true, // Vertical centering of slides
    transition: 'convex', // none/fade/slide/convex/concave/zooms

    // Optional reveal.js plugins
    dependencies: [// Markdown Support
        {src: './reveal.js/plugin/markdown/marked.js'
            , condition: function () {
                return !!document.querySelector('[data-markdown]');
            }
        }, {src: './reveal.js/plugin/markdown/markdown.js'
            , condition: function () {
                return !!document.querySelector('[data-markdown]');
            }
        }       , // Slide Zooming        {
            src: './reveal.js/plugin/zoom-js/zoom.js'
            , async: true
        }]
});
```

- Now go back to the `<slides>` branch and start adding slides:
  - 1 HTML slide
  - 1 Markdown slide written in the HTML file
  - 1 Markdown slide that is imported from an .md slide.
  - a section of slides (so you can go downwards): one with an image and one with a video

For the brave:

- Create a slide with a table.
- Create a CSS file to look similar to the university `.ppt` template.
- Include plotfile.js in your presentation and modify it to create a smooth normal curve.