HackSoc: Intro to Web

Amy Dickens - PhD Student Computer Science

Intro to Web: Session Schedule

Week One 7/3/17: Online (follow these slides)

Week Two 14/3/17: 6.30pm - 8.30pm, The Hub, School of Computer Science

Week Three 21/3/17: 6.30pm - 8.30pm, The Hub, School of Computer Science

Week Four 28/3/17: 6.30pm - 8.30pm, The Hub, School of Computer Science

Requirements:

Please bring a laptop with you to the sessions in The Hub, if you do not have your own device please get in touch at info@hacksocnotts.co.uk and we will try to arrange a loan for you for the sessions.

If you don't already have one, please sign up for GitHub account here. We will use GitHub in these workshops for version control of your projects and FREE hosting on GitHub Pages. Also install GitHub desktop.

We will be using a Text Editor to write our web code, personally I use Sublime Text but others (atom, notepad ++ or vim etc.) are fine - use what you are comfortable with.

What will we cover today?

In this second session we will cover the basics of developing a website, this will include:

- ➔ Separating your CSS from your HTML
- ➔ Learning basic JavaScript for the web
- ➔ Creating a JavaScript Slider element

Got it?

Ok let’s start . . .

What makes a web page?

STRUCTURE - our structure is built using HTML (HyperText Markup Language), the standard markup language for websites and web applications.

STYLING - styling is achieved using CSS (Cascading Style Sheets) a style sheet language that describes the presentation of documents.

LOGIC - interactivity is created using JavaScript, a language used to program the behaviour of a web page.

Let’s take a look at your sites!

Here’s a recap of what we did last week:

- ➔ Structuring a web page with HTML
- ➔ Styling our page with CSS
- ➔ Hosting our web page on GitHub

CSS can be written in an external file

It looks like this:

```
h1 {
        color: white;
        background: RGB(129, 219, 209);
        padding:30px;
        margin: 40px;
        text-align: center;
        font-weight: bold;
    }
```

Create your style sheet!

1. Open a new document in your text editor.
2. Copy & paste the information from your index.html within the <styles> tag.
3. Save the document as styles.css in the folder for your project.
4. Now we need to add a reference to this in our index.html

styles.css

```
 body, html {
                   background-color: white;
                   padding: 0px;
                   margin: 0px;
             }
             h1 {
                   color: white;
                   background: RGB(129, 219, 209);
                   padding:30px;
 ...
```

index.html

```
<head>
   <link rel="stylesheet" type="text/css" href="styles.css">
</head>

<body>
...
```

Ok so it’s a bit basic looking… let’s fix that

1. Use fonts.google.com to find a web font that you like for your site
2. Embed this font into your index.html file
3. Now update this in the styles.css file
4. Save both files and refresh your page to see your changes!

index.html

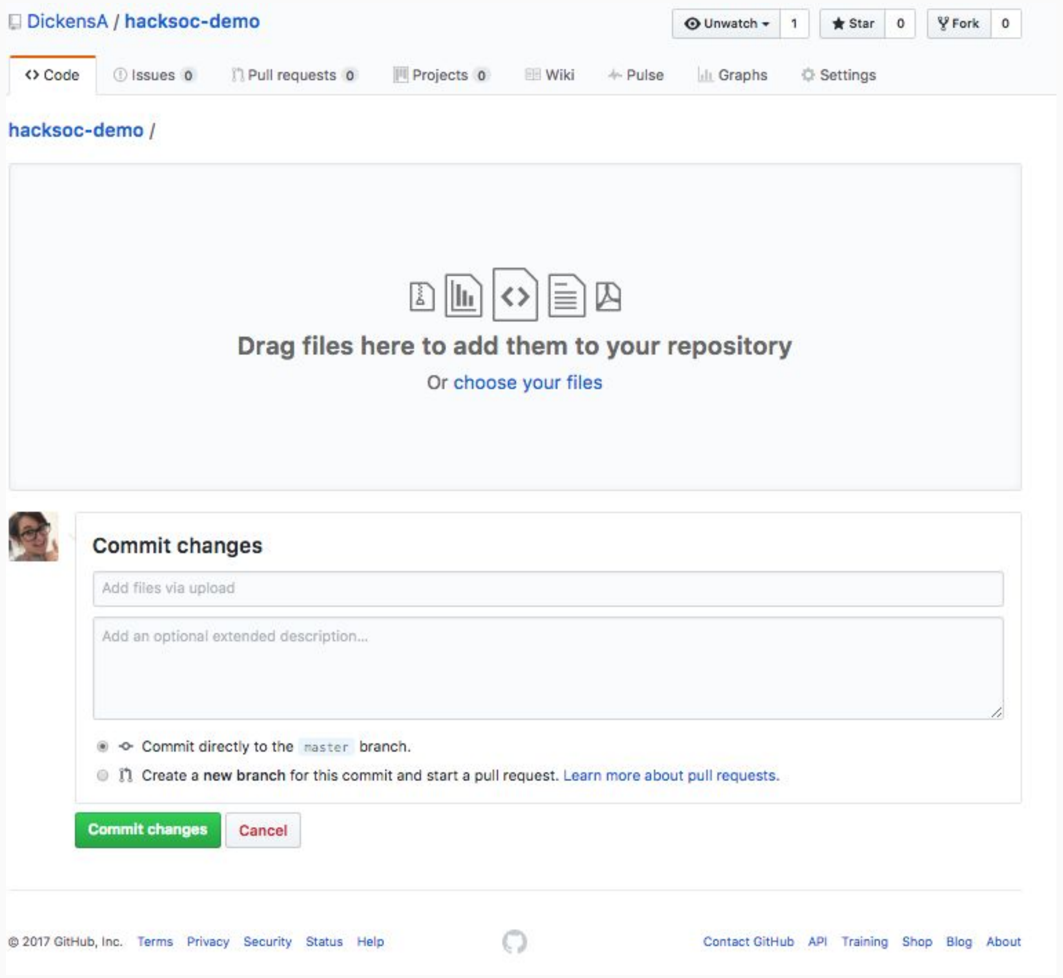
```
<head>
   <link rel="stylesheet" type="text/css" href="styles.css">
   <link
    href="https://fonts.googleapis.com/css?family=Raleway"
    rel="stylesheet">
</head>

<body>
...
```

styles.css

```
.html {
     font-family: 'Raleway', sans-serif;
}
```

Let’s update this to your repo!

1. Open your GitHub Repository
2. Click upload files
3. Drag & Drop both the index.html and styles.css from your document folder into the browser
4. Remember to leave yourself a message about what you’ve done - and commit your changes!
5. When you load your page you’ll see nothing has changed!



Drag files here to add them to your repository

Or choose your files

Commit changes

Add files via upload

Add an optional extended description...

Commit directly to the `master` branch.

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Commit changes
Cancel
© 2017 GitHub, Inc. Terms Privacy Security Status Help
Contact GitHub API Training Shop Blog About

What have we covered so far?

- Recap on week one
- CSS as a separate file
- <link> tag like an image tag doesn’t need to be closed
- Adding web fonts & personalisation
- Updating your GitHub Repository using github.com

So what about JavaScript?

Right let’s try some JavaScript

Go to https://repl.it/languages/javascript

JavaScript Primer

1. Follow along on repl
2. Arithmatic
3. Variables - Creating, Using Reassigning
4. Functions - Making Sandwiches with JavaScript

So what about JavaScript?

Now we can put JavaScript into our page to create interactive elements!

We can create this in a separate .js file, like with our .css file

Let’s create our JavaScript file

1. Create a new file in your text editor and save it as scripts.js in your project folder

2. We need to create a link to this file in our index.html like we did with the CSS file

3. Once you’ve added the link save all files and update your GitHub repository - don’t forget to add your commit message to yourself!

Create a blank file save as *scripts.js*

index.html

```
<script type="text/javascript"
src="scripts.js"></script>
```

Creating a slider function

Now we can make our unordered list a bit more interesting with the use of a slider function.

So first let's remove our list from the index.html file and replace it with this:

```
<p id="switcher"></p>
```

Arrays & Variables

1. Open the scripts.js file
2. First we will create an array of our list items using a variable called ‘contents’
3. Then we will create a variable called ‘currentItem’ and set this to zero
4. Now we need to set up the variable which tells our script where this content will be , this is where the use of an id comes into play

```
var contentBox =
document.getElementById("switcher");

var currentItem = 0;

var contents = [
  "This is the first item",
  "This is the second item",
  "This is the third and last item"
];
```

Functions with functionalities

1. Now we need to load and interact with our content and we use functions to do this

2. Create a function called ‘initLoad’ and pass it the variables of contentBox and contents and currentItem following the example

3. Now we need to call the function initLoad for the content to display on loading the window

```
function initLoad(){
  contentBox.innerHTML =
contents[currentItem];
}

window.onload = initLoad();
```

More functionality!

1. Go to your index.html and add two buttons, one labelled ‘Previous’ and one labelled ‘Next’

2. Save and load your HTML to see we now have two buttons in our content box

3. Now let’s add their functionality, follow the example to add the increaseCounter and decreaseCounter functions to your scripts.js file

4. We need to call these within index.html for the functions to work , we do this using an onclick event

index.html

```
<button id="previous">Previous</button>
<p id="switcher"></p>
<button id="next">Next</button>
```

scripts.js

```
function increaseCounter() {
  currentItem++;
  changeContent();
} ...
```

index.html

```
<button id="previous"
onclick=”decreaseCounter()”>Previous</button>
```

Adding boundaries

1. Create the function changeContent

2. Follow the example to implement an if statement that uses the property *length* to prevent the slider from trying to call an undefined item

3. Save your file and reload the index.html and click through your list - you shouldn’t see the term undefined, as this will now cycle through your list array

4. This should work even if you continue to add items to your array!

```
function changeContent() {
  if(currentItem >= contents.length) {
    currentItem = 0;
  }
  if(currentItem < 0) {
    currentItem = contents.length-1;
  }

  contentBox.innerHTML = contents[currentItem];
}
```

Now back up your work to your repo!

Always commit your work!

Remember to leave yourself a comment that let’s you know what this commit was for <3

Well done <3

Thanks for coming along!