An introduction to programming for the web.

Part One: HTML, CSS and JavaScript

What we’re going to be doing

1. We will introduce some concepts about web requests
2. Everyone will set up a “Hello, World!!” app
3. We will cover some basic parts of HTML, CSS and JavaScript
4. Everyone will build a small web app that we will set as a task.

Things you need for this session

1. A source code editor, we suggest Visual Studio Code (http://code.visualstudio.com/)

2. An http server, we suggest the NodeJS http-server package
    - Download and install NodeJS (https://nodejs.org/)
    - Install http-server (run `npm install -g http-server` in your terminal)

Web Requests

What is a web request?

When you open a webpage in your browser, it is fetching that content from a server somewhere via a protocol called HTTP.

The browser sends a request to the server

- The request has a URL, e.g. “http://www.someapp.com/login”
- And some headers

And the server replies with a response

- The response has also got headers
- And a body - usually the contents of a file.

  One of the headers: “Content-Type” tells the browser what to do with the contents of the body

What program is running on the server?

On our client we’re using a web browser. But the program on the server could be anything, so long as it replies using HTTP. (You *could* use almost any programming language)

In our case, we want a typical static file webserver. (our http-server module)

We want it to map URLs to files. E.g.

localhost:8080
localhost:8080/index.html } >> C:\Users\JohnDoe\Documents\App\index.html

localhost:8080/an-image.png >> C:\Users\JohnDoe\Documents\App\an-image.png

(If you write your own program to run on the server, it’s called a “backend”. Many languages have libraries for serving web content, such as: Express for NodeJS, Flask and Django for Python, ASP.NET for C#, Rails and Sinatra for Ruby)

URLs aren’t necessarily file paths

For our static file webserver, we don’t want it to do anything besides send our HTML, CSS, JavaScript and other content, like fonts and images back to the browser. So we have chosen a webserver program that maps URLs to files in our project folder.

However, there’s no reason why “localhost:8080/image.png” has to route to a file called image.png, or even return an image. It’s up to the program running on the server to decide what to do. It could interpret that URL to be asking for anything, and send that back to the browser.

Ports

The standard port for HTTP is port 80 (and 443 for HTTPS)

When a server is using the standard ports, your browser hides them from the URL.

So when you type “www.somewebsite.com/dashboard” into your web browser, you’re telling it to connect to a program listening on port 80 at that address.

To send requests to a program on a different port, you use a colon, e.g. “www.somewebsite.com:4000/dashboard” would send the request to port 4000.

The http-server module we will be using listens on port 8080

Hello, World!!

Hello, World!!

Create a new folder on your computer for the project.

In the folder create a new file called “index.html”.

Put the following code into the file and save it:

```
<html>
    <head>
        <title>Guestbook App</title>
    </head>
    <body>
        Hello, World!!
    </body>
</html>
```

Then, open a terminal in your project folder and run:

```
http-server
```

You should see something like the following:

```
Starting up http-server, serving ./
Available on:
  Http:138.38.187.189:8080
  http:127.0.0.1:8080

Hit CTRL-C to stop the server
```

Now open up “localhost:8080” in your browser!

HTML, CSS and JavaScript basics

HTML basics

(inside the `<body>` tag)

Image

```
<img src="image.png" />
```

Headings

```
<h1>Heading</h1>
<h2>Sub-heading</h2>
<h3>Sub-sub-heading<h3>
```

Paragraph

```
<p>Paragraph</p>
```

Generic

```
<div>A full width box</div>
<section>Another one</section>
<span>An inline box used for
text</span>
```

Lists

```
<ul><li>Bulleted item</li></ul>
<ol><li>Numbered item</li></ol>
```

Links

```
<a href="/page2.html">
    A hyperlink to page2
</a>
```

CSS basics

Example HTML being styled:

```
<div class="box">
     <h2 id="subheading1">
          A heading
     </h2>
     <p>
          Some text
     </p>
</div>
```

In the `<head>` tag:

```
<style>
.box {
     background: black;
}

#subheading1 {
     color: blue;
}

.box p {
     font-size: 30px;
}
</style>
```

JavaScript basics

Write code like this to be evaluated immediately

```
alert("This pops up in the middle of the
screen");

console.log("This appears in the dev
console");

var newElem = document.createElement("p");
newElem.innerHTML = "some text";
document.body.appendChild(newElem);
```

Or put it into a function and call it when needed

JS:

```
function doTheThing() {
     alert("The button was pressed");
}
```

HTML:

```
<button onclick=”doTheThing();”>
     Press Me!
</button>
```

What we are going to build

What we are going to build

A “Guestbook” web app:

- A single HTML page, containing:
    - A form
    - A list of previous entries
- We want the form submit button to trigger some JavaScript
- We want to validate the form, if it is ok then:
- We want the list of previous entries to be updated with a new item, and
- We want the form to reset the field.

Next session: Backend Magic.

- Server side code - A replacement for the http-server module
- Server side rendering vs. Client side rendering (AJAX)
- Working with databases
- User authentication and management