Financial Software Engineering
Lecture 4

Co-Pierre Georg
AIFMRM

July 15, 2018

Today

1. Introduction to front-end development
2. HTML
3. CSS
4. Bootstrap
5. Intro to JavaScript

Front-end development

The development continuum

- Earlier in the course we spoke about distinguishing between programming languages
- We did this on fundamental level: functional vs object-oriented and compiled vs executed
- Programming languages and paradigms exist to develop software, programs and/or applications
- In this broader development context we are able to distinguish between different types of development
- Specifically, we can consider development as having two components: front-end and back-end development

The development continuum

- Front-end $\rightarrow$ design and development of the interface users interact with
- Back-end $\rightarrow$ functionality of the platform, specifically the execution of action between user input and a database
- Specifically, when we speak of front-end development, we speak of a suite of three languages: CSS, HTML and JavaScript
- Back-end development refers to a broader suite of languages, typically including PHP, .NET, Java, Python etc.

Front-end development

- We'll focus on the tools for front-end development in the remainder of the course
- We'll use:
- HyperText Markup Language (HTML) $\rightarrow$ structure the content of our web pages
- Cascading Style Sheets (CSS) $\rightarrow$ provide the design and style of our web pages
- JavaScript (JS) $\rightarrow$ provides interactive elements of a web page
- Bootstrap $\rightarrow$ HTML, CSS, and JS framework
- jQuery $\rightarrow$ JS library to interact with HTML

Front-end development

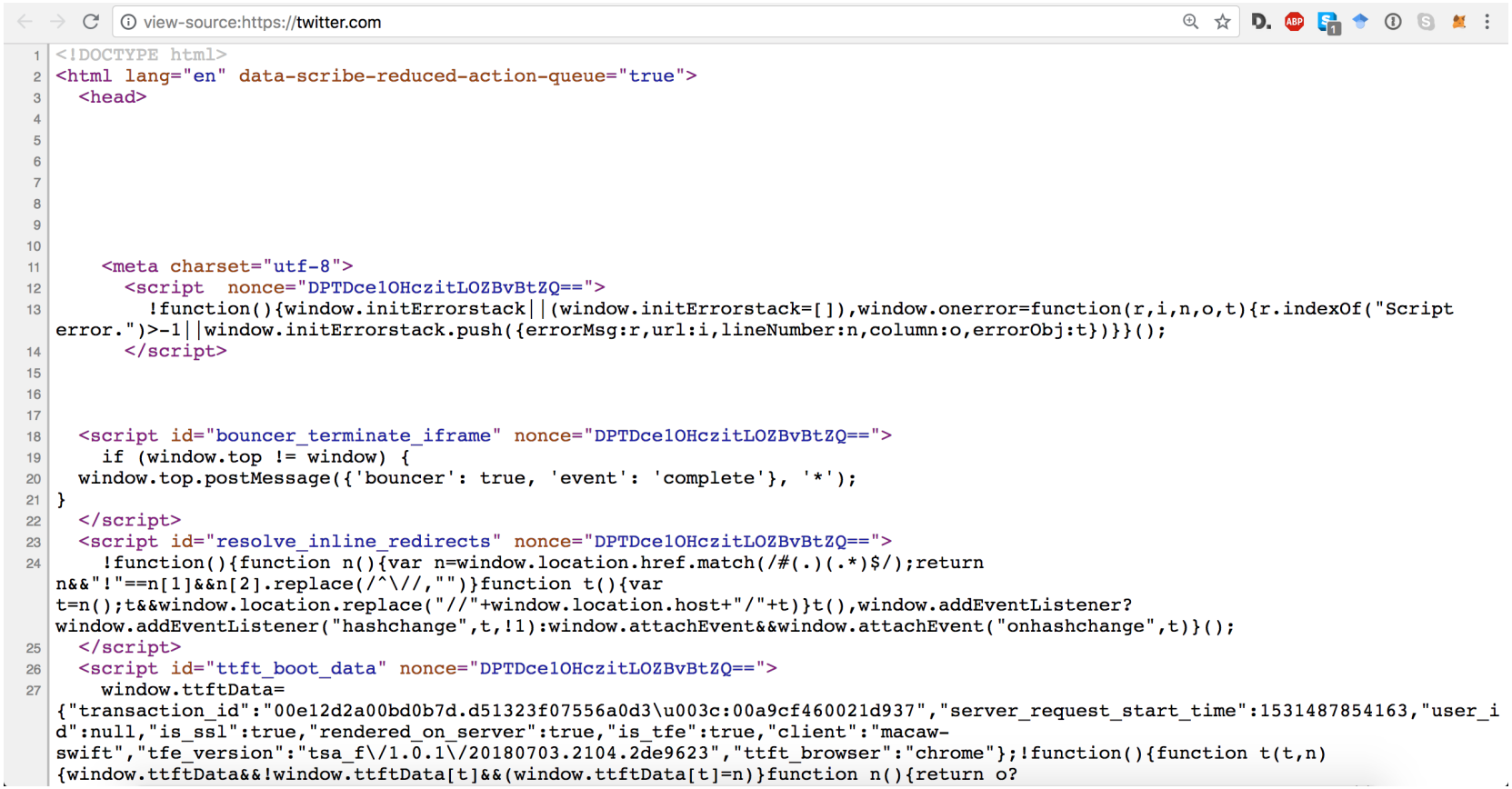
- All of the front-end tools we'll be using have many free online resources
- The most well-known $\rightarrow$ W3Schools
- You'll be able to find examples of everything we cover on W3Schools
- $\rightarrow$ your de facto resource for HTML, CSS and JS
- Another good front-end resource: Mozilla

HTML

HTML

- HyperText Markup Language or **HTML** ...
- Importantly, HTML represents a markup language as opposed to a programming language
- Markup language $\rightarrow$ used to structure and present data
- Specifically, HTML is used to structure web pages
- Every web page is the product of underlying HTML which can easily be viewed
- Google Chrome $\rightarrow$ open web page $\rightarrow$ right-click $\rightarrow$ view page source

HTML



Source: Twitter's HTML

HTML

- All HTML documents share the same structure

```
<!DOCTYPE html>
<!--This is a comment-->
<html> <!--Represents the root element of the page-->

        <head> <!--Metadata-->
                <title>This is the title</title>
        </head>

        <body> <!--Page content-->
                <h1>I am a header</h1>
                <h2>I am smaller header</h2>
                <p>I am a paragraph</p>
                <p><em>I am a paragraph in italics</em></p>
        <ul>
        <li>List item one</li>
        <li>List item two</li>
        <li>List item three</li>
        </ul>
        </body>

</html>
```

HTML

- Which renders this in a web browser

I am a header

I am smaller header

I am a paragraph

I am a paragraph in italics

- List item one
- List item two
- List item three

HTML

- In HTML, everything we type is always encased in elements called tags $\rightarrow$ tell HTML what to do with the text contained in the tag
- Perhaps, make the text a header <h1></h1> or a paragraph <p><p/> etc.
- We won't spend anymore time on HTML since everything you'll need can be found on W3Schools

CSS

CSS

- Much like HTML, Cascading Style Sheets or **CSS**, represents a markdown language
- While HTML is responsible for representing and structuring data on a web page, CSS is used to describe how these HTML elements are displayed
- CSS allows us to control fonts, colours, borders and other properties of HTML elements
- While HTML types, CSS *typesets*
- As a convention, you'll have a seperate .css file and link it to your .html file
- Just like with HTML, we'll rely on W3School

CSS

- CSS works through styling specific HTML tags
- As a language, CSS therefore takes the following structure

```
tag {
        property : value;
}
```

- To change the color of our paragraphs to blue, and list items to red in our previous example, we'll do the following

```
p {
    color: blue;
}

li {
    color: red;
}
```

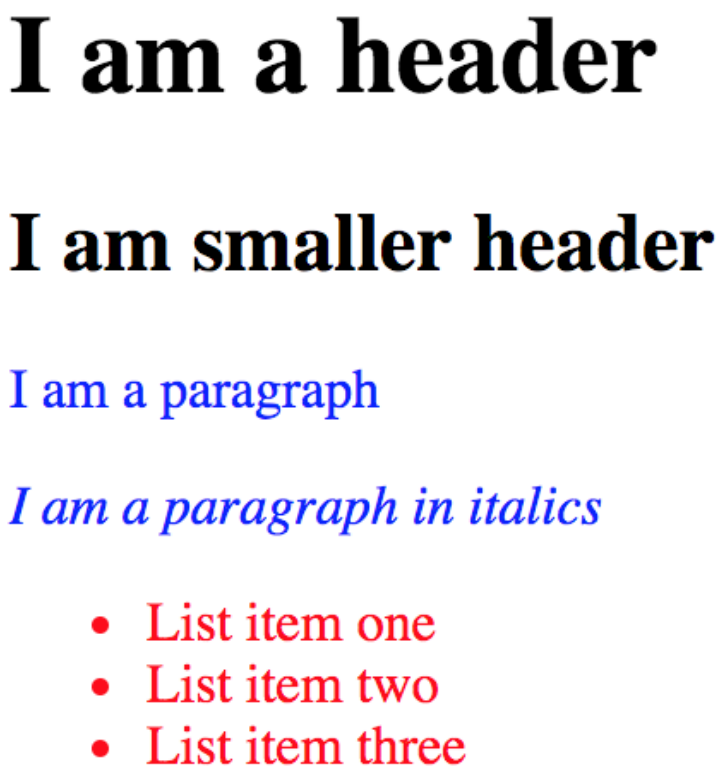
Linking CSS to HTML

- Before the changes take effect, we need to link the CSS file to our HTML file
- We do so in the <head> </head> of the file

```
<head> <!--Metadata-->
        <title>This is the title</title>
        <link rel="stylesheet" href="my_css.css">
</head>
```

CSS

- Which renders this in a web browser



HTML & CSS

- Using HTML and CSS we are able to build basic web pages quite easily
- Nonetheless, HTML and CSS alone have two major limitations
- Firstly, creating a web page with HTML and CSS from scratch can be a tedious process
- Secondly, HTML and CSS lack the tools to create interactive elements on our web pages
- We'll address the second limitation when we cover JavaScript
- The first limitation can be overcome using a framework which provides us with ready-to-go HTML and CSS templates $\rightarrow$ speeds up the process of designing a front-end

Bootstrap

Framework

- In our context, a framework provides a way to import ready-made templates to aid the development process
- Specifically, frameworks differ from say a package, library or add-in due to one feature, the **inversion of control**
- $\rightarrow$ you are able edit certain parts of template in clearly defined ways
- In this way, you give up the full freedom to edit every aspect of a code base in return for templates of code which can be used easily
- We'll explore one such front-end development framework $\rightarrow$ Bootstrap

Bootstrap

- Free and open-source framework for front-end web development
- Effectively a collection of CSS components or templates
- Avoids having to spend hours manually adjusting CSS
- Allows us to easily add lots of functionality through simple templates, such as forms and buttons
- Easily imported JavaScript functionality such as drop-downs and navigation bars
- Thorough and easy to use documentation

Bootstrap grid system

- The most fundamental feature of Bootstrap is the grid system
- $\rightarrow$ allows Bootstrap sites to render on different devices of different sizes
- $\rightarrow$ splits the screen into 12 columns
- We place objects into these columns or combinations of columns
- We then tell bootstrap how to arrange the web pages data across these columns, for different screen sizes

Arrangemet of grids must always add up to 12

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1
span 4			span 4			span 4					
span 4			span 8								
span 6						span 6					
span 12											

Source: W3Schools

Bootstrap

- Let's take a look at some Bootstrap examples

Intro to JavaScript

Javascript

- Up to now we've covered three front-end technologies
- **HTML** $\rightarrow$ Content
- **CSS** $\rightarrow$ Style
- **Bootstrap** $\rightarrow$ Template
- While this allows us to create front-ends easily, we lack one component, namely reactivity or interaction
- Currently, we have no way of interacting with user input via our web pages or no way to have content that reacts depending on user behavior

Javascript

- Introducing **JavaScript** (JS) $\rightarrow$ programming language for HTML and web pages
- Importantly, we can use JS to interact with HTML
- Represents a complete programming language with all the functionality we are used to
- Allows us to use the OOP principles we learnt with Python
- In fact, you'll notice how easy the transition from Python to JS is now that you have a basic understanding of OOP
- Note, that while JavaScript syntax tends to resemble that of Java, they are two different languages

Javascript

- Built into (most) web browsers $\rightarrow$ can run JS directly from browser console
- In addition to the browser console, we can also connect a JS file to a HTML file, in the same way we did with CSS
- Our ultimate goal $\rightarrow$ use JS to directly change HTML or CSS on a page

Variables

- Integers

```
myInteger = 2;
```

- Strings

```
a = "Hello, World";

a[0] // indexing
>>> "H"

"Hello" + "World" // Concatenation
>>> "Hello, World"

a.length // Attributes
>>> 12

a.includes(",")// Methods
>>> true
```

Control flow: If/else

```
var name = "John"; // Note the syntax here

if (name == "Adam") {
    console.log("Hello Adam")
} else if (name == "John") {
    console.log("Hello John")
} else {
    console.log("Hello [insert name]")
}
```

Control flow: for

- We could replace `i = i + 1` with `i++`
- We could also use say for example `i = i + 3` if we wanted to iterate by 3 in our loop

```
for (i = 0; i < 5; i = i + 1) {
    console.log(i); // we use console.log to print
}
```

Control flow: while

```
var sum_of_series = 0;

while (sum_of_series < 10) {
    console.log('sum_of_series is currently: ', sum_of_series)
    sum_of_series += 1
}
```

Other useful functionality

- Print to the console

```
console.log("Hello world")
```

- Alert / pop-up message on the screen

```
alert("hello world")
```

- User input

```
prompt("What is your name")
```

- User input + alert

```
name = prompt("What is your name")
alert("Welcome " + name)
```

Functions

- Just like before in Python, we can create our own function

```
function hello() {
    console.log("Hello, World");
}

hello()
>>> Hello world

function squareMe(num1) {
    return num1 ** 2;
}

squareMe(5)
>>> 25
```

Data structures

- Unlike Python with it's range of data structures, JS has two primary data structures, **arrays** and **objects**
- Arrays $\rightarrow$ a collection of data, similar to lists in Python
- Objects $\rightarrow$ store data using a key-value pair similar to a dictionary in Python
- In JS, we can think of the object key as the equivalent of the object attribute in Python
- Like Python, we can also create methods for our objects and call these methods on their respective object types

Arrays

- We can perform many of the same actions we did with lists in Python with arrays in JS

```
var my_array = ["John", 5, true, "Sally"];
my_array[0]
>>> "John"

my_array.length // Array attributes
>>> 3

my_array.includes("John") // Array methods

for (var i = 0; i < my_array.length; i++) { //Loop over array
    console.log(my_array[i])
}
```

Objects

- We create objects in JS in the same way we created dictionaries in Python

```
var my_cat = {name: "Garfield", color: "Orange"};
my_cat.name
>>> "Garfield"
```

- We can add methods to these objects too

```
var my_cat = {name: "Garfield",
              color: "Orange",
              meow: function() {
              console.log("Meow")
              }};

my_cat.meow()
>>> Meow
```

Objects

- We can also access and edit attributes of our objects
- Where we used `self` in Python, we use `this` in JS
- Note that unlike Python, we do not pass this as a parameter to our function

```
var my_cat = {name: "Garfield",
              color: "Orange",
              meow: function() {
              console.log("Meow, my name is " + this.name)
              }};

my_cat.meow()
>>> Meow, my name is Garfield
```