Web Dev 101

Frontend vs Backend, Client vs Server

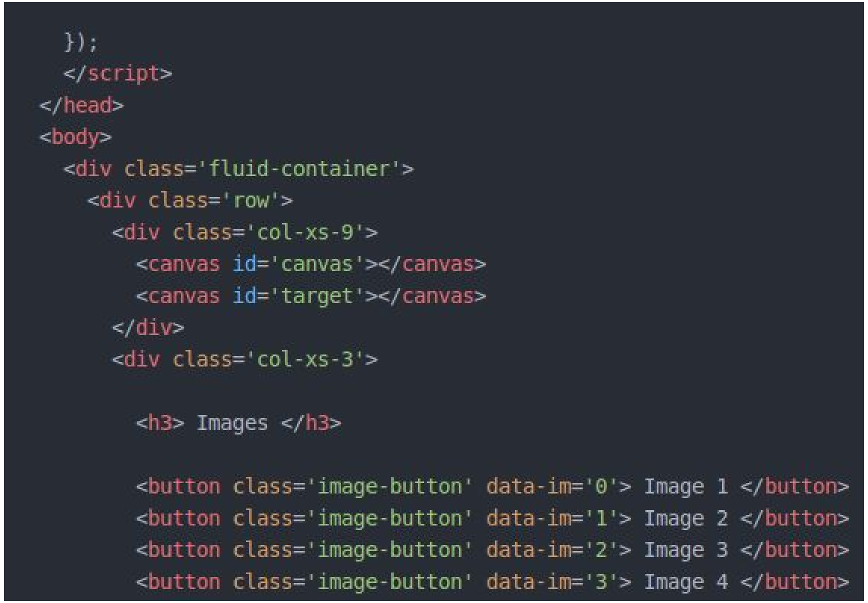
- The client is the browser visiting the webpage
- The client receives all of the frontend code
- The frontend code defines the user interface for the client and how the page is displayed
- The frontend code is sent to the client via the backend code running on the server
- The backend code isn’t visible to the client, and can use completely different technologies than the frontend code
    - (Hence why we can have Frontend, Backend, and Full Stack Developers)

Frontend Technologies

- HTML, the layout
- CSS, the styling
- Javascript, the scripting/program

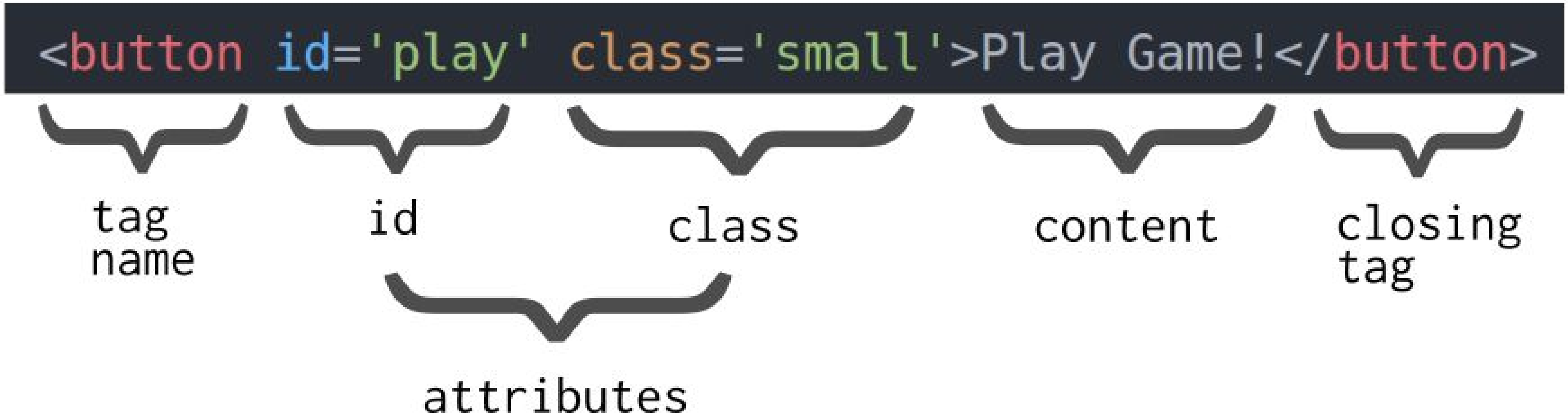
HTML, the “Scaffolding”

- “Markup” language, defines “elements” which make up the page
- Makes up the user interface to a web application or website



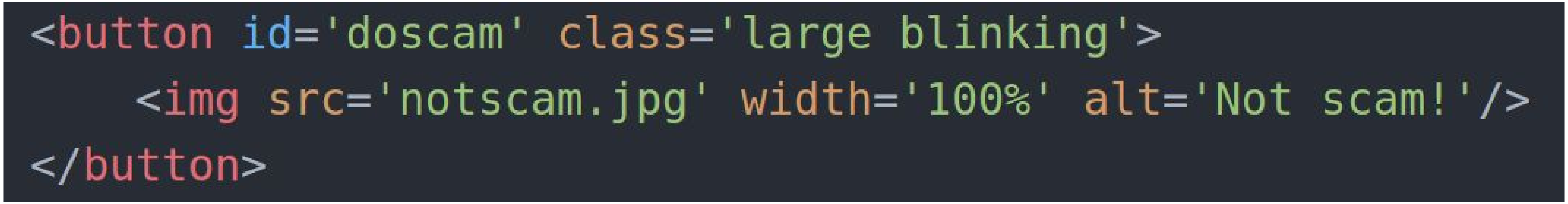
```
  });
  </script>
</head>
<body>
  <div class='fluid-container'>
    <div class='row'>
      <div class='col-xs-9'>
        <canvas id='canvas'></canvas>
        <canvas id='target'></canvas>
      </div>
      <div class='col-xs-3'>

        <h3> Images </h3>

        <button class='image-button' data-im='0'> Image 1 </button>
        <button class='image-button' data-im='1'> Image 2 </button>
        <button class='image-button' data-im='2'> Image 3 </button>
        <button class='image-button' data-im='3'> Image 4 </button>
```

Breakdown of an HTML Element



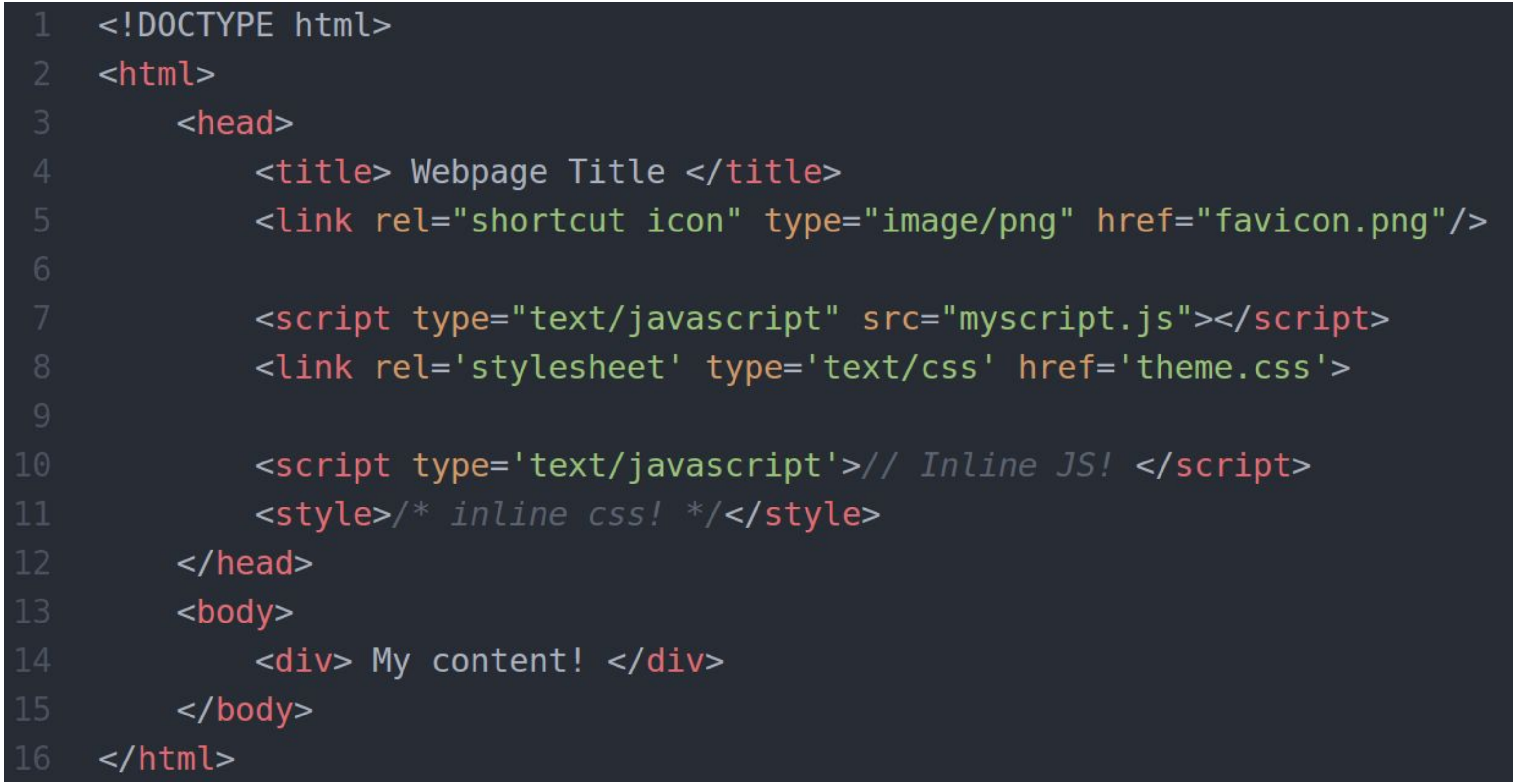
```
<button id='play' class='small'>Play Game!</button>
```

More Complex Elements



```html
<button id='doscam' class='large blinking'>
    <img src='notscam.jpg' width='100%' alt='Not scam!'/>
</button>
```

Basic HTML Elements



```html
<!DOCTYPE html>
<html>
    <head>
        <title> Webpage Title </title>
        <link rel="shortcut icon" type="image/png" href="favicon.png"/>

        <script type="text/javascript" src="myscript.js"></script>
        <link rel='stylesheet' type='text/css' href='theme.css'>

        <script type='text/javascript'>// Inline JS! </script>
        <style>/* inline css! */</style>
    </head>
    <body>
        <div> My content! </div>
    </body>
</html>
```

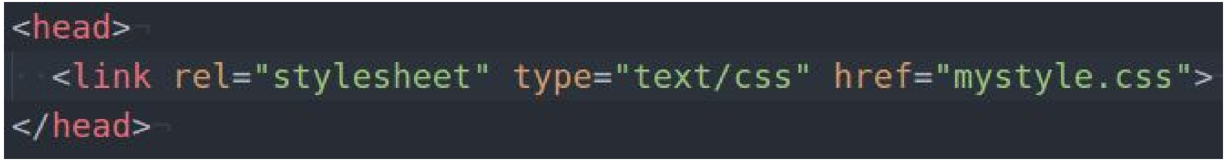
Basic HTML Elements (cont.)

- **<!DOCTYPE html>** : Tells browser you’re using HTML5
- **<html>** : Main content
- **<head>** : Contains meta-information, page resources (sometimes)
- **<title>** : What shows up in a tab
- **<body>** : All displayed elements are contained here
- **<link>** : Link to styling resources, CSS or favicon
- **<script>** : Link or inline script
- **<style>** : Inline styling (CSS)
- **<div>**, **<span>, <p>** : Content containers (each has unique properties)

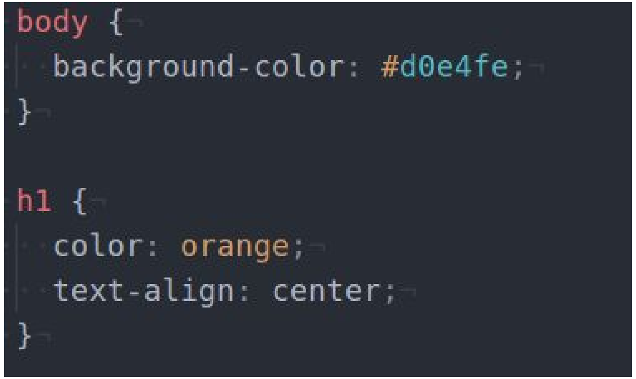
Quick Demo: Let’s make facebook

Styling, CSS

- CSS stands for Cascading Style Sheets.
- It’s used to define styles for your web pages, including design, layout and variations in display for different devices and screen sizes



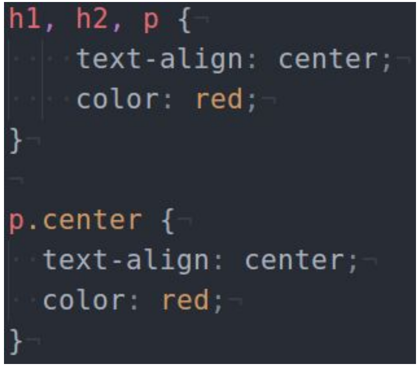
```
<head>
  <link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```



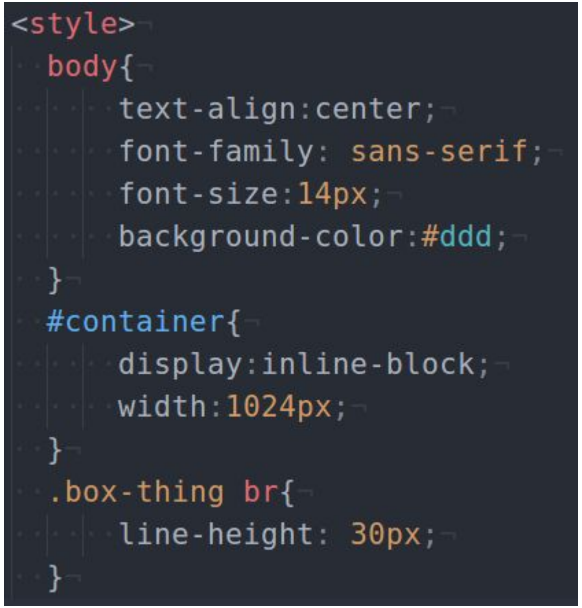
```
body {
  background-color: #d0e4fe;
}

h1 {
  color: orange;
  text-align: center;
}
```

CSS Selector

- It allows you to select and manipulate HTML elements
- They are used to “find” an element based on their
  - id
  - class
  - attribute
- Grouping Selectors



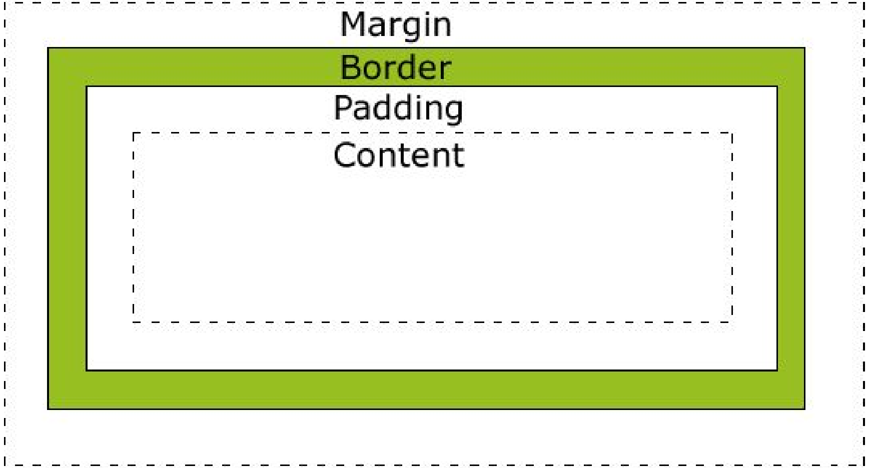
```
h1, h2, p {
    text-align: center;
    color: red;
}

p.center {
  text-align: center;
  color: red;
}
```

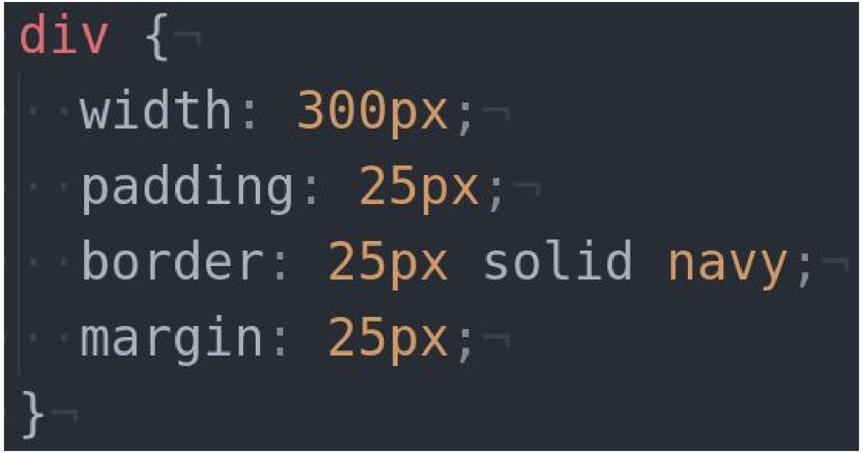


```
<style>
  body{
        text-align:center;
        font-family: sans-serif;
        font-size:14px;
        background-color:#ddd;
  }
  #container{
        display:inline-block;
        width:1024px;
  }
  .box-thing br{
        line-height: 30px;
  }
```

Box Model

- Content - The contents of the box, where text and images appear
- Padding - Clears an area around the content. The padding is transparent
- Border - A border that goes around the padding and content
- Margin - Clears an area outside the border. The margin is transparent



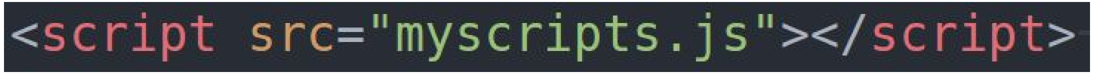


```css
div {
  width: 300px;
  padding: 25px;
  border: 25px solid navy;
  margin: 25px;
}
```

CSS Example

Scripting, Javascript

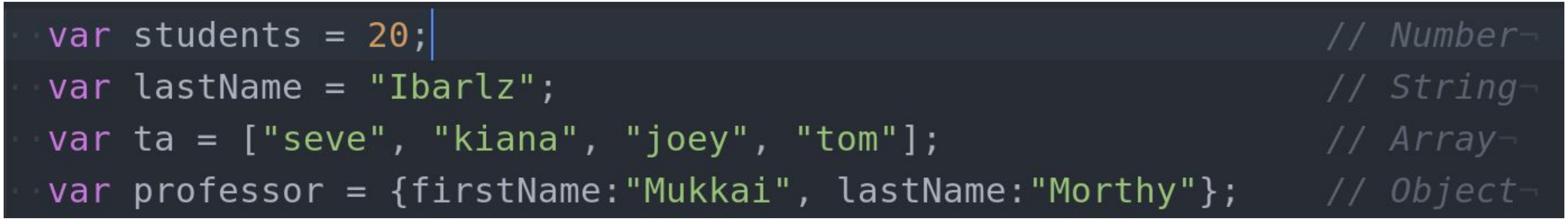
- A dynamic, untyped, and interpreted programming language
- Used to program the “behavior” of web pages
  - Provides dynamic interactivity on websites ie. buttons, pop-ups, ...



```
<script src="myscripts.js"></script>
```

JavaScript Basics

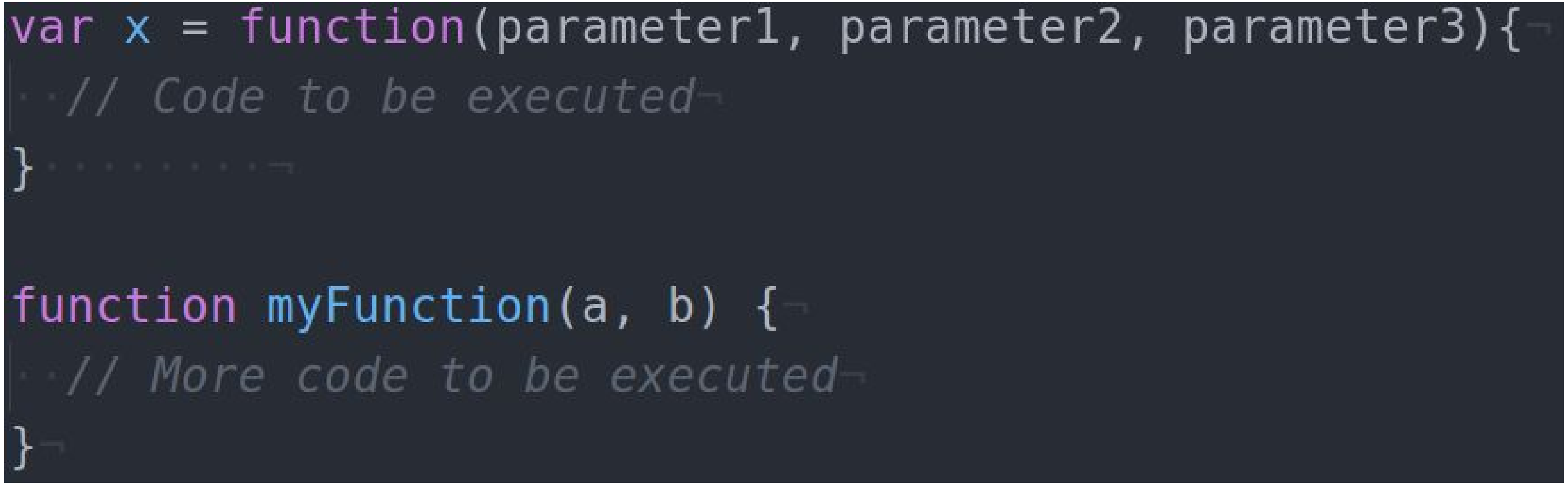
- Variables
  - Declared with the var keyword



```
  var students = 20;                                                  // Number
  var lastName = "Ibarlz";                                            // String
  var ta = ["seve", "kiana", "joey", "tom"];                          // Array
  var professor = {firstName:"Mukkai", lastName:"Morthy"};            // Object
```

JavaScript Basics (cont.)

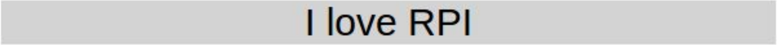
- Functions
  - Declared with the function keyword



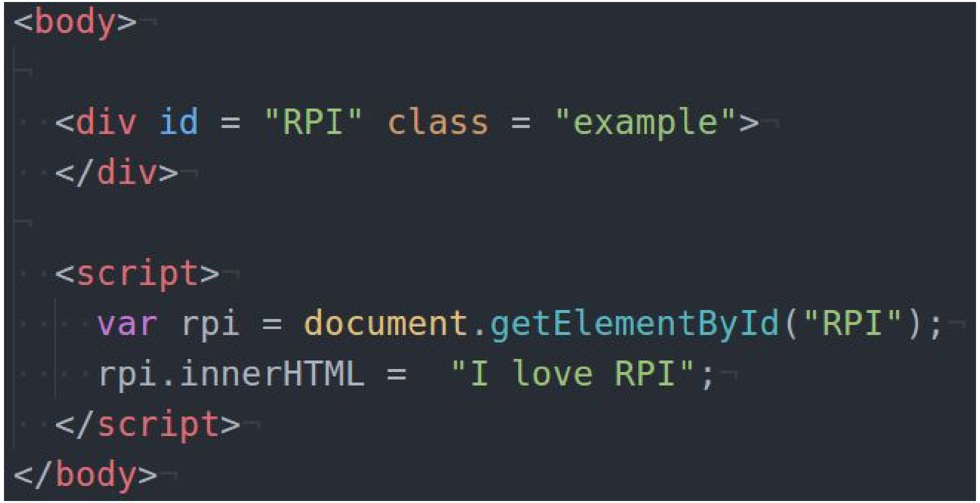
```
var x = function(parameter1, parameter2, parameter3){
  // Code to be executed
}

function myFunction(a, b) {
  // More code to be executed
}
```

JavaScript Basics (cont.)

- JavaScript HTML DOM Document
  - Grabs an HTML element by:
    - Id
    - Tag Name
    - Class Name



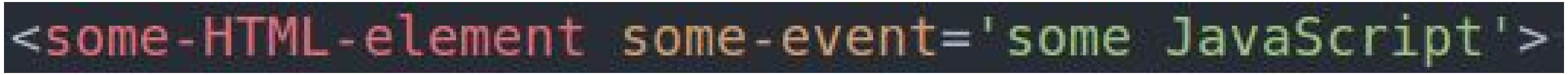
```
I love RPI
```



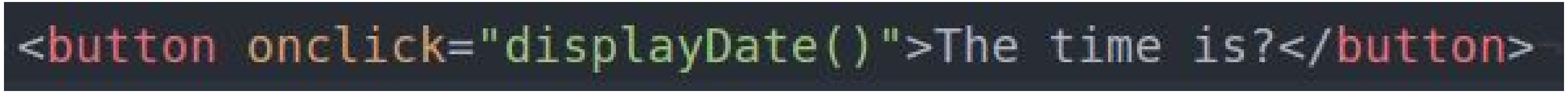
```
<body>

  <div id = "RPI" class = "example">
  </div>

  <script>
    var rpi = document.getElementById("RPI");
    rpi.innerHTML =  "I love RPI";
  </script>
</body>
```

JavaScript Basics (cont.)

- HTML events are “things” that happen to HTML elements
  - You can use JavaScript to “react” to these events
  - Some examples:
    - An HTML page finishes loading
    - An HTML button was clicked
    - An HTML input field was changed



```
<some-HTML-element some-event='some JavaScript'>
```



```
<button onclick="displayDate()">The time is?</button>
```

Quick Break: Let’s write a javascript program

Developer Tools

- Chrome/Firefox DevTools
  - Firebug for Firefox
  - Debugging tools built into the browser
- Postman
- Code linting

Clientside/Frontend Frameworks

- Bootstrap
- Foundation
- JQuery
- Semantic
- Angular
- React

Why do you need a backend?

- Serve files to the client
- Persistence & Accessing a database
- Private credentials
- Offload processing onto server
- Coordination among clients
- Security

Other options?

- Use someone else’s server - Github pages host static sites (no backend code)
- Use someone else’s api - Parse

Backend Frameworks

Many languages to choose from:

- PHP
- Javascript
  - NodeJS
  - Meteor
- C#
  - ASP.NET
- Ruby
  - Ruby on Rails
- Python
  - Django
  - Flask

Each provides routing, APIs, and interfacing with Databases

Github Pages

- User vs Project
- User
  - Hosted from repository username.github.io
  - Access at http://username.github.io
  - Can setup custom domain name
- Project
  - Hosted from gh-pages branch
  - Access at http://username.github.io/repository
  - Can setup custom domain name or access from subdirectory of a User Pages domain
- Uses Jekyll templating
- Can use any frontend framework that can be included via html

Create your own github page

https://pages.github.com

- Create a new repository named *username*.github.io
- Clone it :
- `git clone https://github.com/username/username.github.io`
- `echo "Hello World" > index.html`
- Commit and Push it
- View at http://*username*.github.io

Automatic generation:

- https://github.com/username/username.github.io > Settings

Other web dev things...

- Domain Registration
- Databases
- CSS preprocessing
- Responsive vs Adaptive Design
- Virtual Private Servers (VPS)/Hosting