IDV460

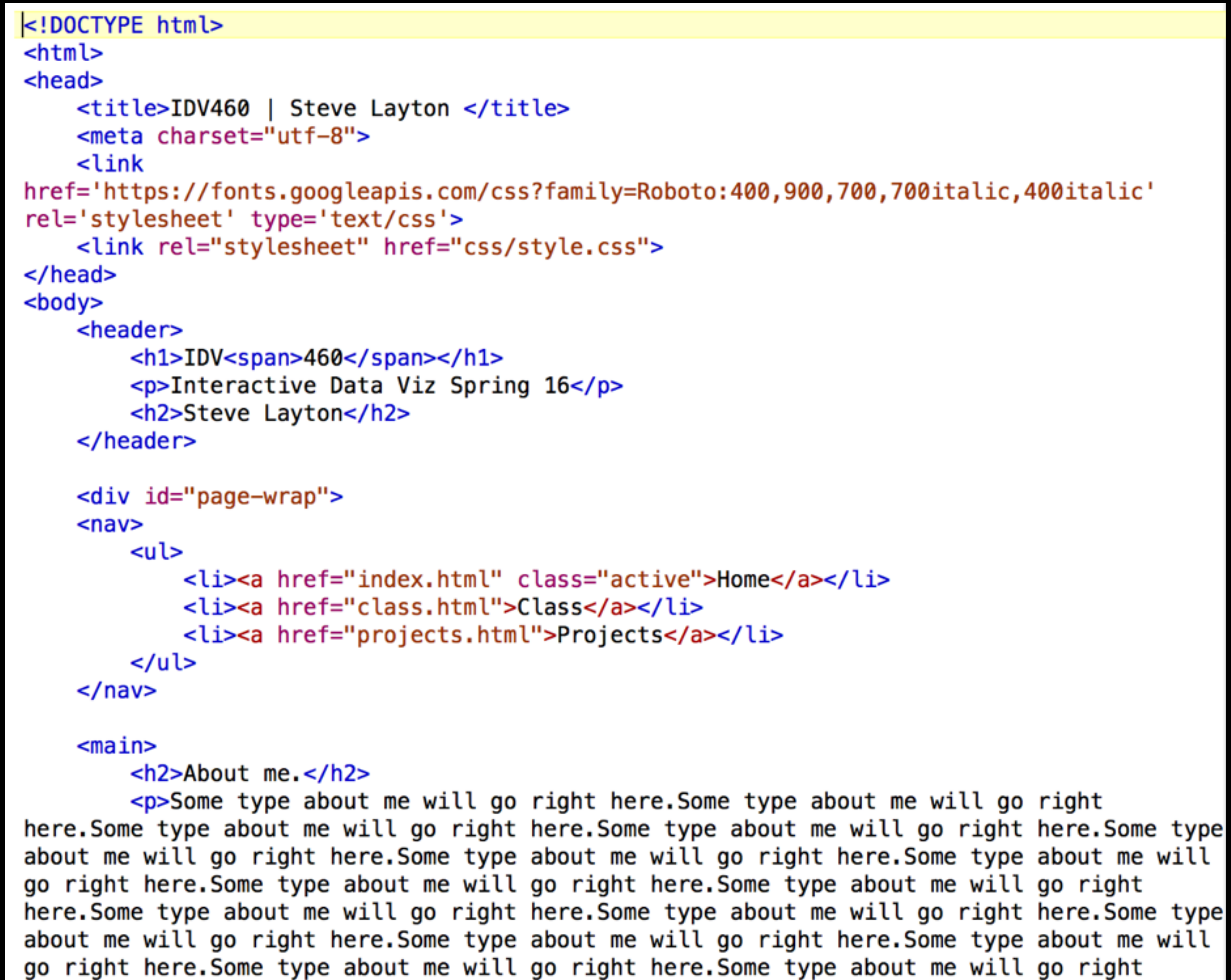
INTERACTIVE DATA VIZ SPRING 16

JAVASCRIPT INTRODUCTION

WHAT IS JAVASCRIPT?

Javascript is one of the three core languages of the web.

HTML is used to define content (markup)



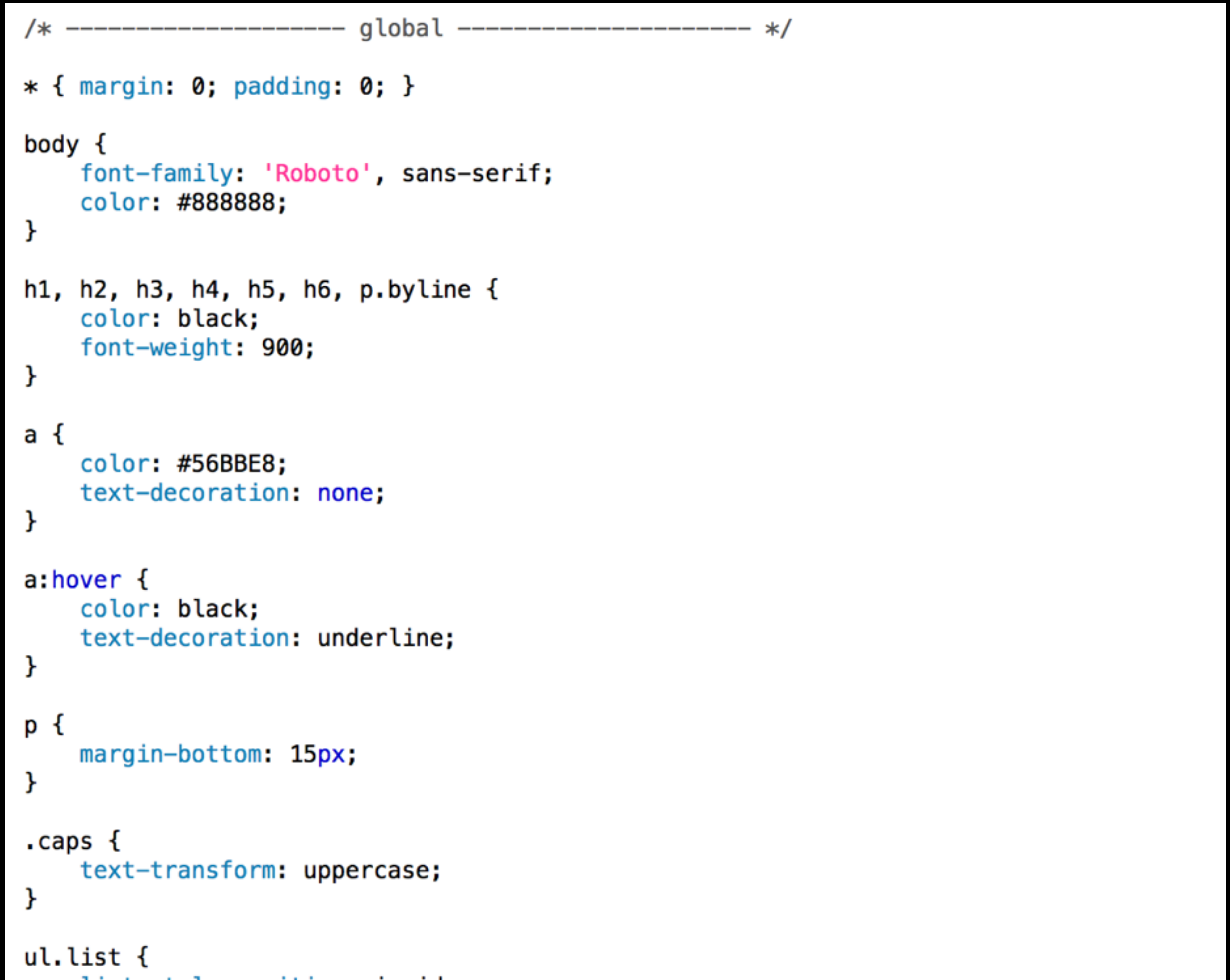
```
<!DOCTYPE html>
<html>
<head>
    <title>IDV460 | Steve Layton </title>
    <meta charset="utf-8">
    <link
href='https://fonts.googleapis.com/css?family=Roboto:400,900,700,700italic,400italic'
rel='stylesheet' type='text/css'>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <header>
        <h1>IDV<span>460</span></h1>
        <p>Interactive Data Viz Spring 16</p>
        <h2>Steve Layton</h2>
    </header>

    <div id="page-wrap">
    <nav>
        <ul>
            <li><a href="index.html" class="active">Home</a></li>
            <li><a href="class.html">Class</a></li>
            <li><a href="projects.html">Projects</a></li>
        </ul>
    </nav>

    <main>
        <h2>About me.</h2>
        <p>Some type about me will go right here.Some type about me will go right
here.Some type about me will go right here.Some type about me will go right here.Some type
about me will go right here.Some type about me will go right here.Some type about me will
go right here.Some type about me will go right here.Some type about me will go right
here.Some type about me will go right here.Some type about me will go right here.Some type
about me will go right here.Some type about me will go right here.Some type about me will
go right here.Some type about me will go right here.Some type about me will go right
```

WHAT IS JAVASCRIPT?

Javascript is one of the three core languages of the web.

HTML is used to define content (markup)

CSS is used for presentation (style)



```
/* -------------------- global --------------------- */

* { margin: 0; padding: 0; }

body {
    font-family: 'Roboto', sans-serif;
    color: #888888;
}

h1, h2, h3, h4, h5, h6, p.byline {
    color: black;
    font-weight: 900;
}

a {
    color: #56BBE8;
    text-decoration: none;
}

a:hover {
    color: black;
    text-decoration: underline;
}

p {
    margin-bottom: 15px;
}

.caps {
    text-transform: uppercase;
}

ul.list {
```

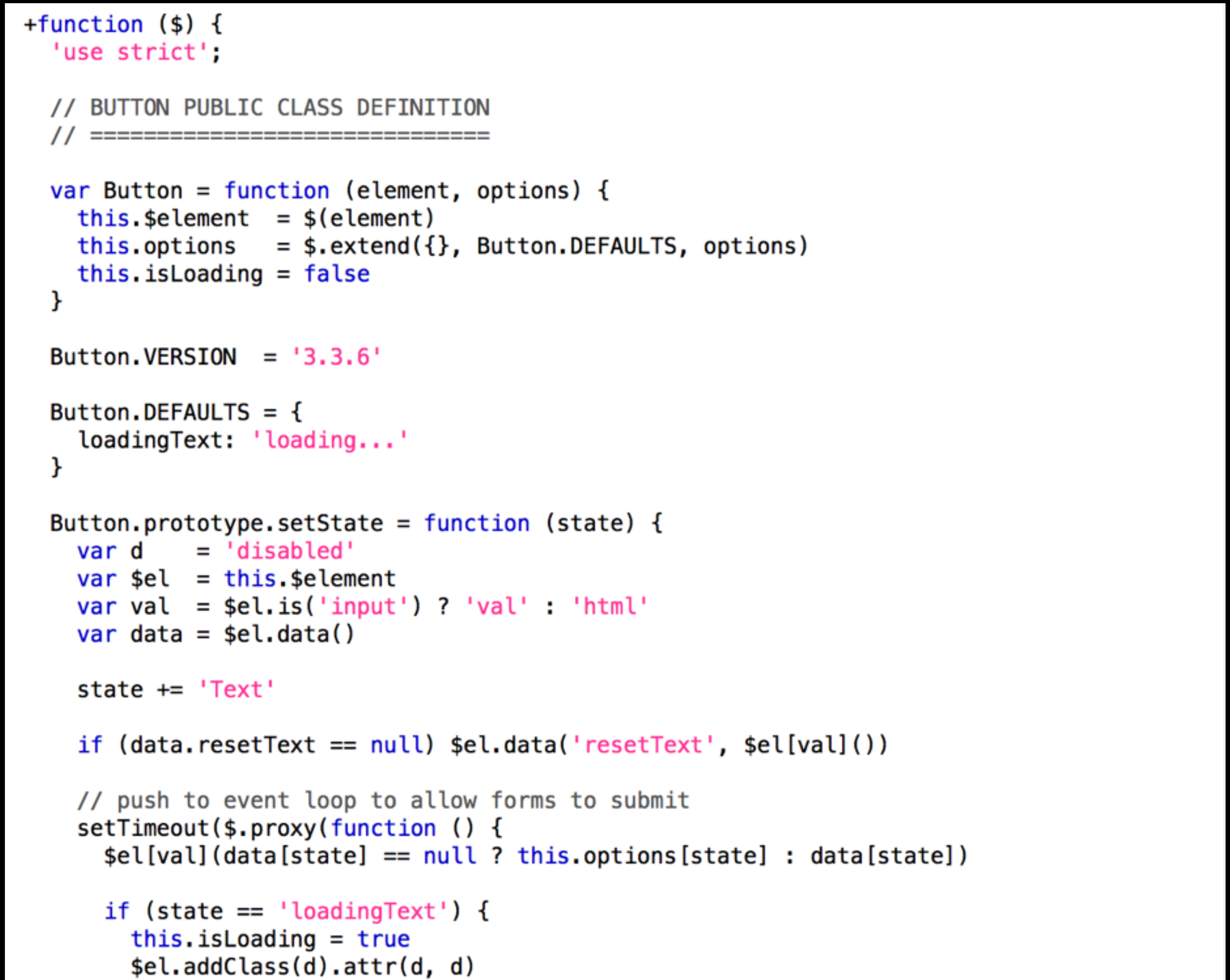
WHAT IS JAVASCRIPT?

Javascript is one of the three core languages of the web.

HTML is used to define content (markup)

CSS is used for presentation (style)

… and **Javascript** is used for behavior (programming)



```
+function ($) {
  'use strict';

  // BUTTON PUBLIC CLASS DEFINITION
  // ==============================

  var Button = function (element, options) {
    this.$element  = $(element)
    this.options   = $.extend({}, Button.DEFAULTS, options)
    this.isLoading = false
  }

  Button.VERSION  = '3.3.6'

  Button.DEFAULTS = {
    loadingText: 'loading...'
  }

  Button.prototype.setState = function (state) {
    var d    = 'disabled'
    var $el  = this.$element
    var val  = $el.is('input') ? 'val' : 'html'
    var data = $el.data()

    state += 'Text'

    if (data.resetText == null) $el.data('resetText', $el[val]())

    // push to event loop to allow forms to submit
    setTimeout($.proxy(function () {
      $el[val](data[state] == null ? this.options[state] : data[state])

      if (state == 'loadingText') {
        this.isLoading = true
        $el.addClass(d).attr(d, d)
```

HISTORY

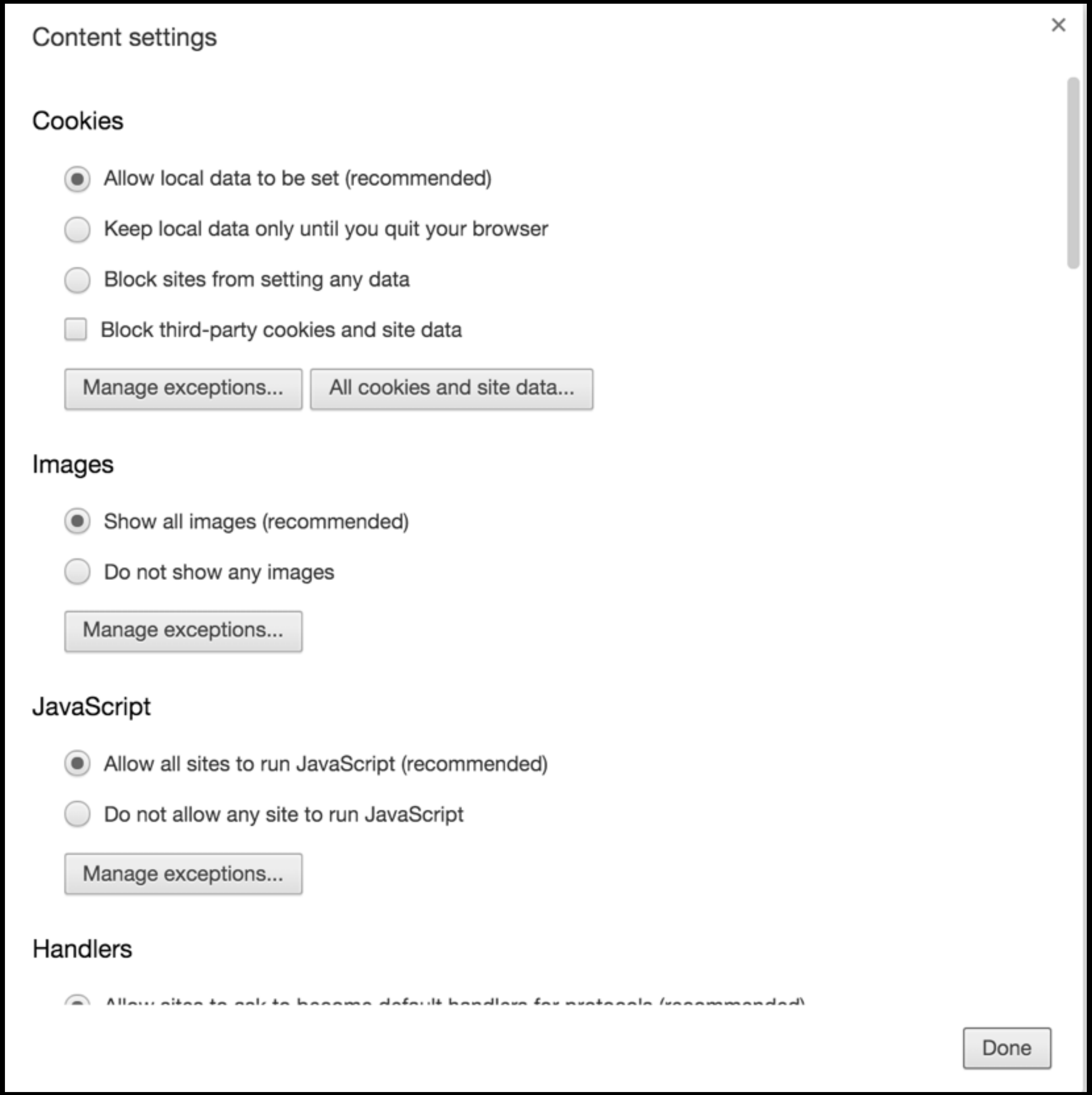
Javascript is a scripting language. It was written in 1995 by Brendan Eich (for Netscape). JavaScript only works inside another application: the web browser. All (leading) browsers have a JavaScript engine inside them. The operating system runs the web browser, the web browser contains a page, and the page contains the JavaScript.

HISTORY

Javascript's syntax is derived from programming language called **C**, which first appeared in 1972. Despite the name similarity, Javascript has almost nothing to do with **Java**, which is a different programming language altogether.

RUNS IN BROWSER

That means that for Javascript to work on your page, the user must have Javascript enabled in her browser settings. Because of security issues, many users used to do this, though it has become much more rare. You can disable your own Javascript by going to Chrome' Settings, then Advanced Settings, then to Privacy, and finally to Content Settings.



Content settings

Cookies

- [✓ Allow local data to be set (recommended)]
- Keep local data only until you quit your browser
- Block sites from setting any data
- Block third-party cookies and site data

Images

- [✓ Show all images (recommended)]
- Do not show any images

JavaScript

- [✓ Allow all sites to run JavaScript (recommended)]
- Do not allow any site to run JavaScript

Handlers

- Allow sites to ask to become default handlers for protocols (recommended)

PROGRESSIVE ENHANCEMENT

The concept of **progressive enhancement**, which grew out of a web-design strategy called “graceful degradation” holds that your page should “work” and be accessible to the highest possible number of users.

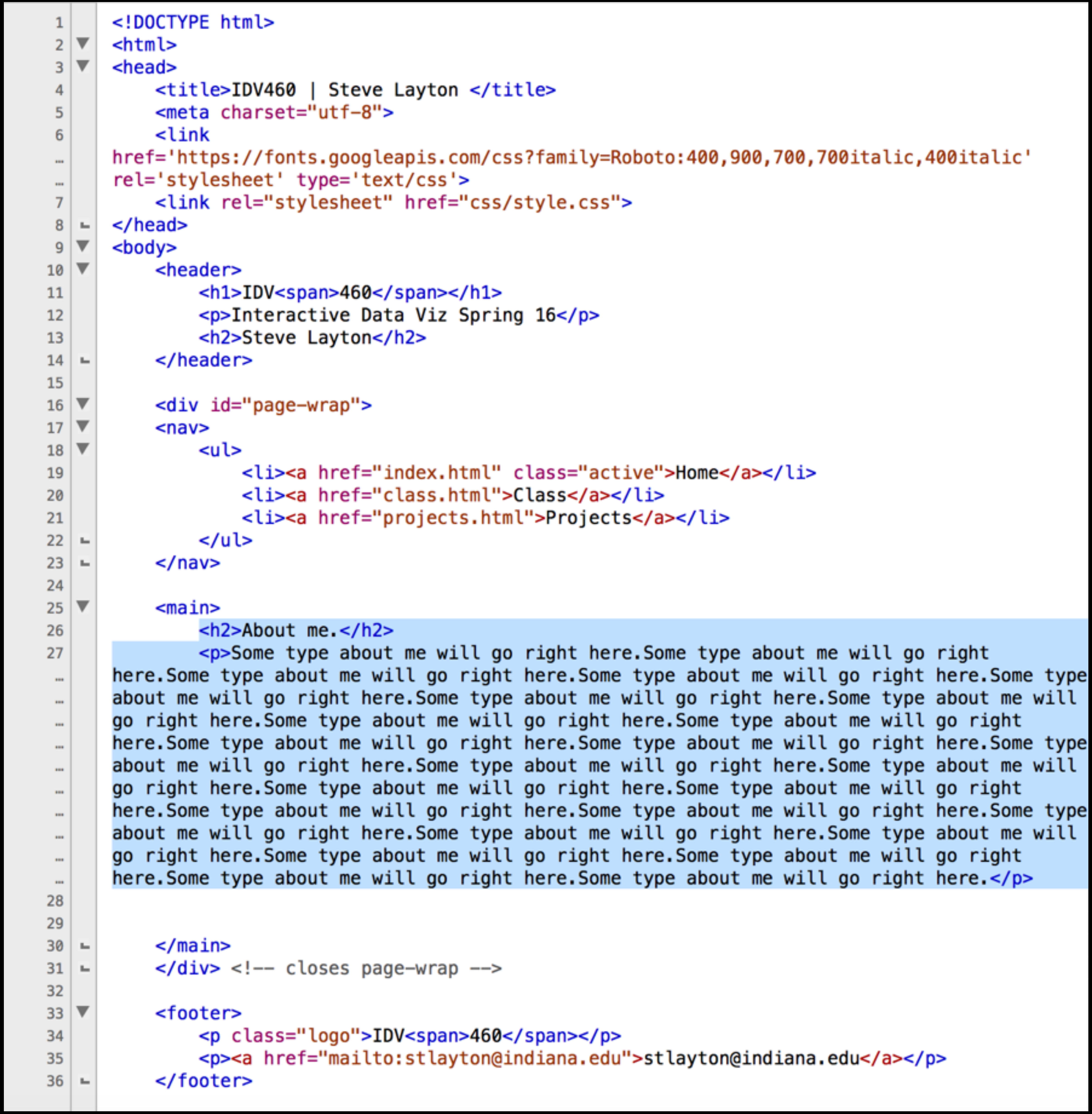
PROGRESSIVE ENHANCEMENT

Core principles:

- Content should be accessible to all browsers
- Semantic markup contains all content
- Enhanced layout is provided by externally linked CSS
- Enhanced behavior is provided by unobtrusive, externally linked JavaScript

GETTING STARTED

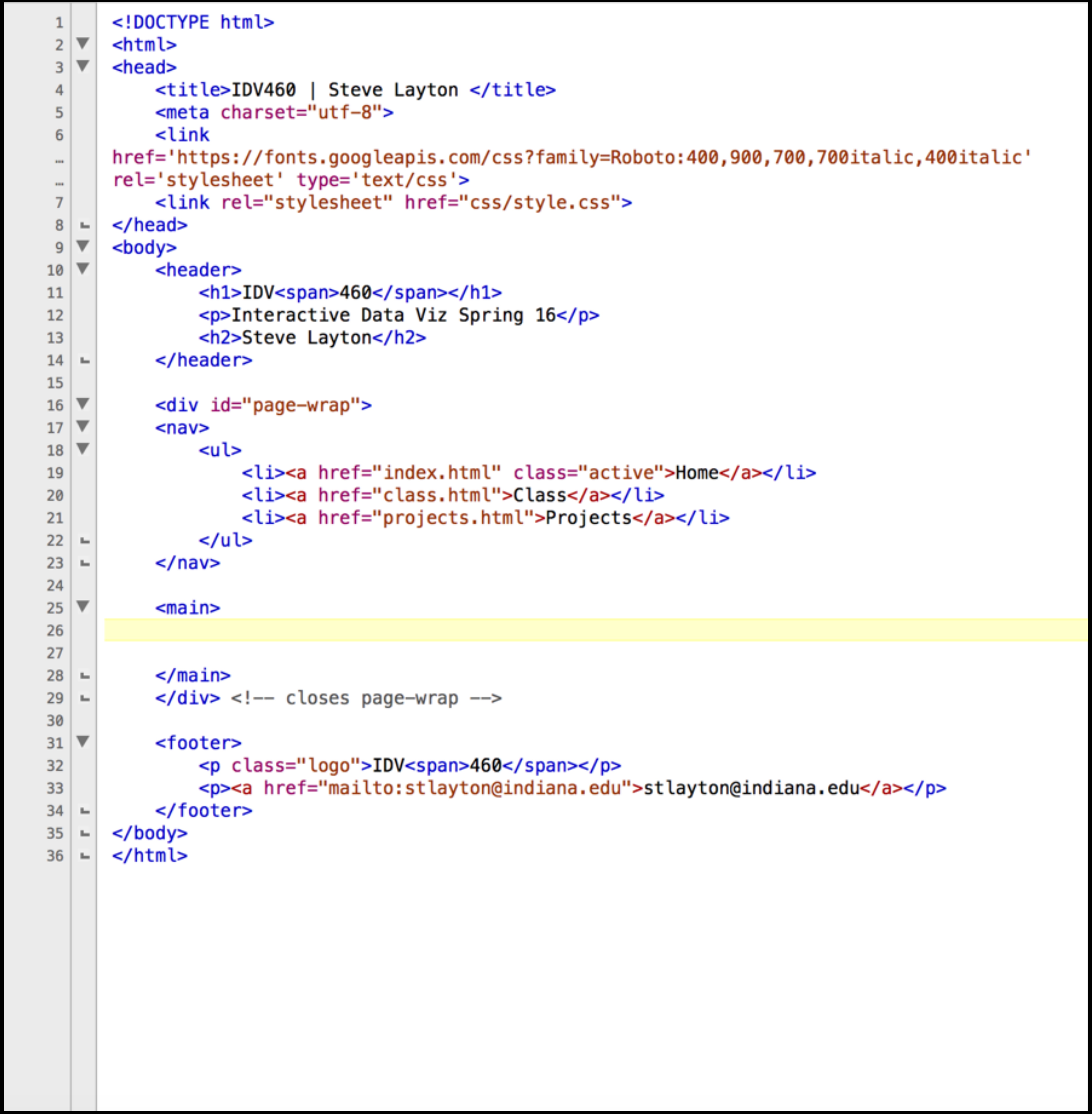
Because we already have a working site with working pages, we can use the basic template to build new pages without having to re-invent our basic structure or CSS.
We will start today by opening up your index.html page, and removing the changeable content. Save this as template.html. We won't be adding this to our live site, but we can use it to start any "blank" exercises we undertake from this point forward.



```html
<!DOCTYPE html>
<html>
<head>
    <title>IDV460 | Steve Layton </title>
    <meta charset="utf-8">
    <link
href='https://fonts.googleapis.com/css?family=Roboto:400,900,700,700italic,400italic'
rel='stylesheet' type='text/css'>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <header>
        <h1>IDV<span>460</span></h1>
        <p>Interactive Data Viz Spring 16</p>
        <h2>Steve Layton</h2>
    </header>

    <div id="page-wrap">
    <nav>
        <ul>
            <li><a href="index.html" class="active">Home</a></li>
            <li><a href="class.html">Class</a></li>
            <li><a href="projects.html">Projects</a></li>
        </ul>
    </nav>

    <main>
        <h2>About me.</h2>
        <p>Some type about me will go right here.Some type about me will go right
here.Some type about me will go right here.Some type about me will go right here.Some type
about me will go right here.Some type about me will go right here.Some type about me will
go right here.Some type about me will go right here.Some type about me will go right
here.Some type about me will go right here.Some type about me will go right here.Some type
about me will go right here.Some type about me will go right here.Some type about me will
go right here.Some type about me will go right here.Some type about me will go right
here.Some type about me will go right here.Some type about me will go right here.Some type
about me will go right here.Some type about me will go right here.Some type about me will
go right here.Some type about me will go right here.Some type about me will go right
here.Some type about me will go right here.Some type about me will go right here.</p>


    </main>
    </div> <!-- closes page-wrap -->

    <footer>
        <p class="logo">IDV<span>460</span></p>
        <p><a href="mailto:stlayton@indiana.edu">stlayton@indiana.edu</a></p>
    </footer>
```

GETTING STARTED

Because we already have a working site with working pages, we can use the basic template to build new pages without having to re-invent our basic structure or CSS.
We will start today by opening up your index.html page, and removing the changeable content. Save this as template.html. We won't be adding this to our live site, but we can use it to start any "blank" exercises we undertake from this point forward.



```html
<!DOCTYPE html>
<html>
<head>
    <title>IDV460 | Steve Layton </title>
    <meta charset="utf-8">
    <link
href='https://fonts.googleapis.com/css?family=Roboto:400,900,700,700italic,400italic'
rel='stylesheet' type='text/css'>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <header>
        <h1>IDV<span>460</span></h1>
        <p>Interactive Data Viz Spring 16</p>
        <h2>Steve Layton</h2>
    </header>

    <div id="page-wrap">
    <nav>
        <ul>
            <li><a href="index.html" class="active">Home</a></li>
            <li><a href="class.html">Class</a></li>
            <li><a href="projects.html">Projects</a></li>
        </ul>
    </nav>

    <main>


    </main>
    </div> <!-- closes page-wrap -->

    <footer>
        <p class="logo">IDV<span>460</span></p>
        <p><a href="mailto:stlayton@indiana.edu">stlayton@indiana.edu</a></p>
    </footer>
</body>
</html>
```

STRUCTURE

Before we actually enter in some Javascript to this page, a few things to bear in mind:

1. Javascript is CASE SENSITIVE

```
getElementById("id");
```

~~`getelementbyID("id");`~~

STRUCTURE

Before we actually enter in some Javascript to this page, a few things to bear in mind:

1. Javascript is CASE SENSITIVE

2. Like other programming languages, Javascript is written as STATEMENTS — lines of code that say piece by piece what you want to do. Each statement should end with a semicolon.

```
alert("Hello, world!");
```

STRUCTURE

Before we actually enter in some Javascript to this page, a few things to bear in mind:

1. Javascript is CASE SENSITIVE

2. Like other programming languages, Javascript is written as STATEMENTS — lines of code that say piece by piece what you want to do. Each statement should end with a semicolon.

3. Javascript does not care about space — except inside the quotes.

```
alert("Hello, world!");

alert(  "Hello, world!"
)  ;
```

STRUCTURE

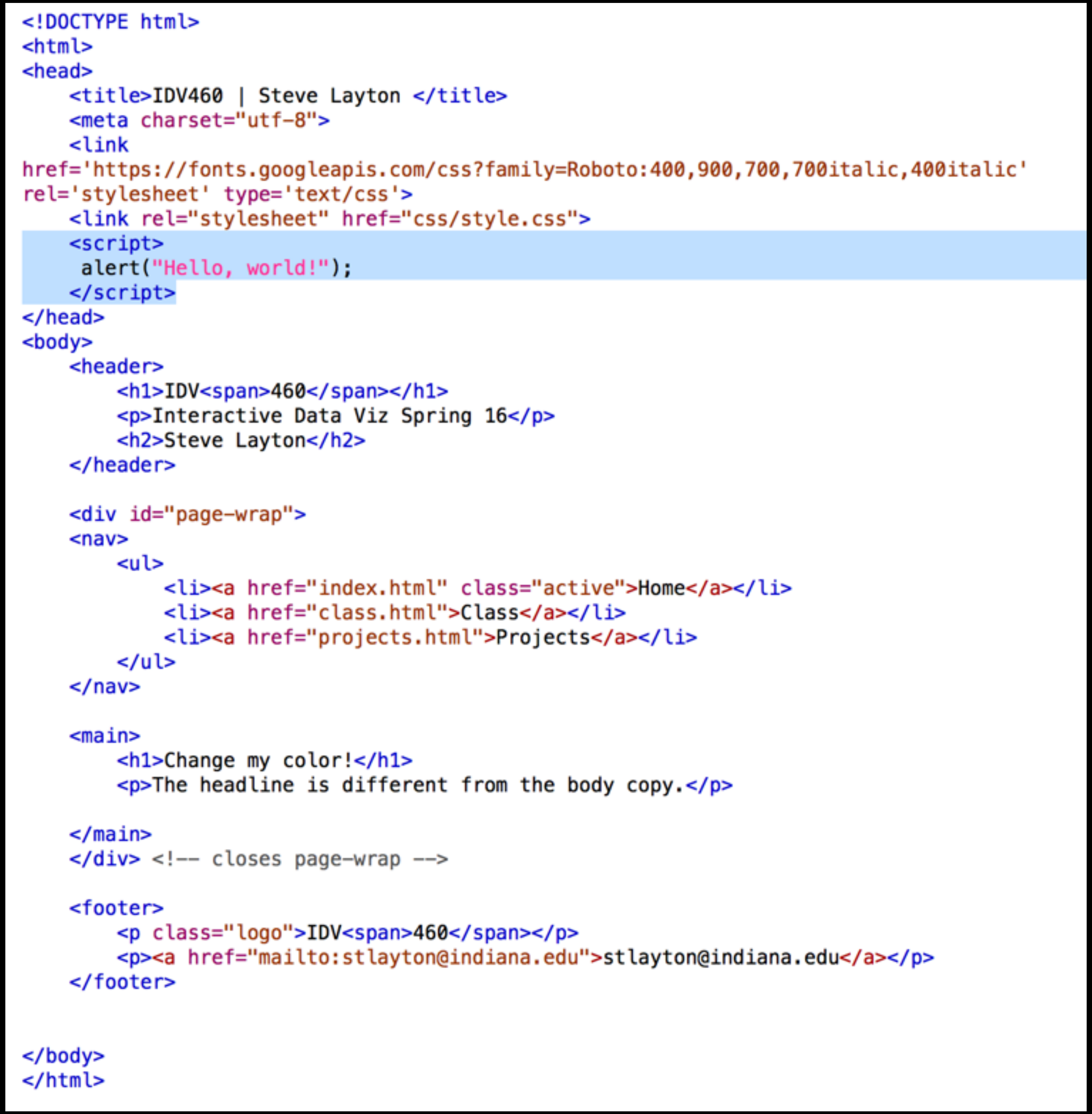
Like other languages, you can add comments to your Javascript — in fact, you will often find yourself doing this, especially early on.

You can write a comment starting the line with two slashes:

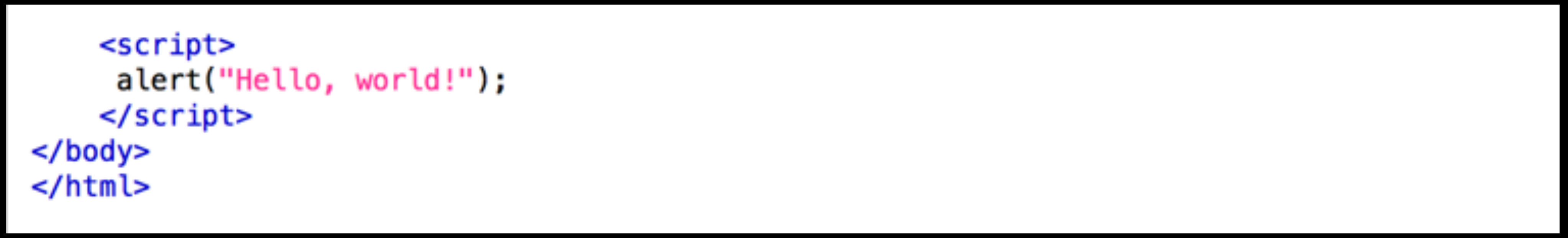
```
alert("Hello, world!");

//This is a comment.
```

WHERE TO PUT JS?

You call in the Javascript code through the <script> tag. This can occur anywhere in your HTML document, but you will primarily see script tags either in the <head> section, which will call in the code on page load, or else just above the closing body tag.



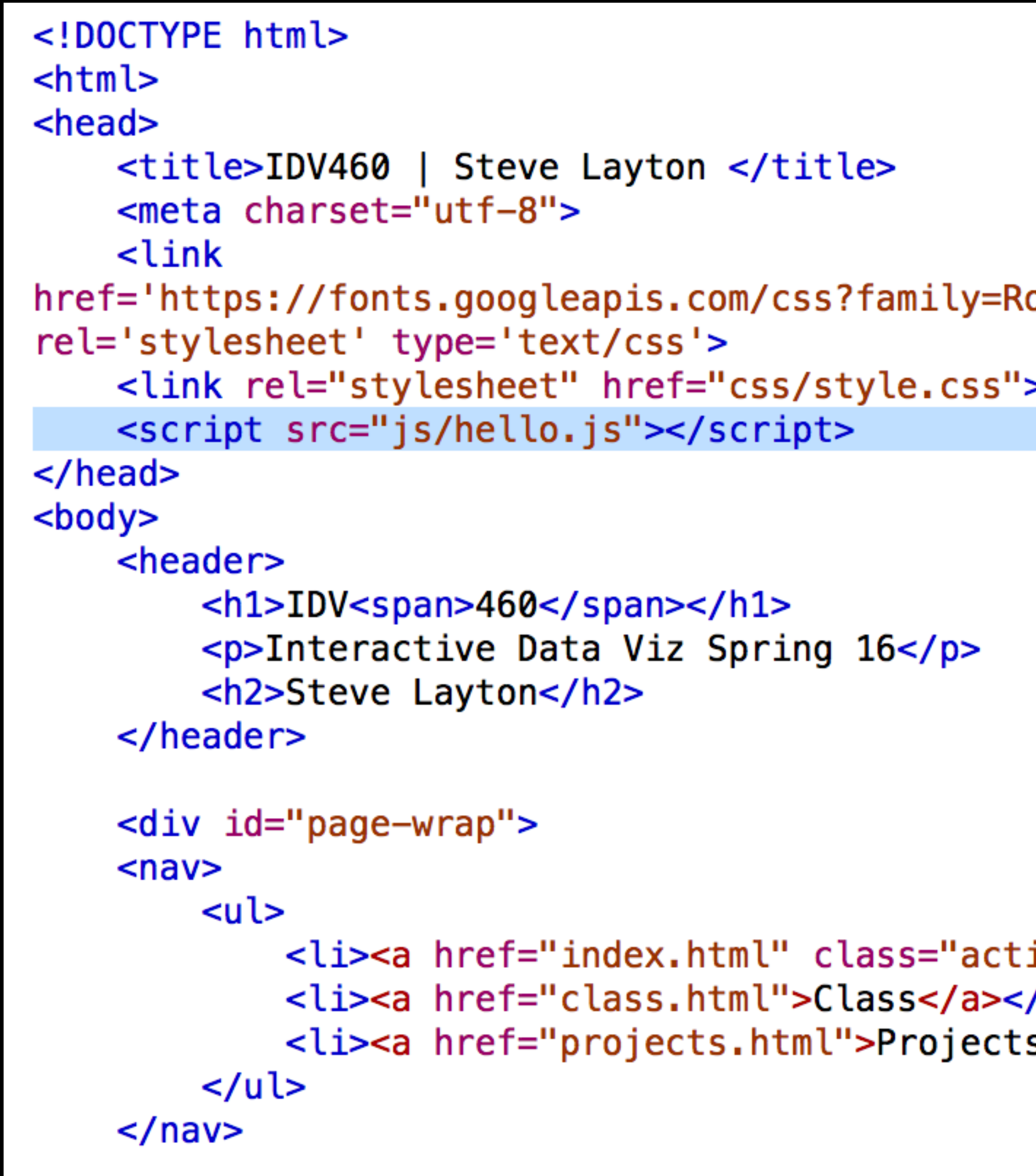
```
<!DOCTYPE html>
<html>
<head>
    <title>IDV460 | Steve Layton </title>
    <meta charset="utf-8">
    <link
href='https://fonts.googleapis.com/css?family=Roboto:400,900,700,700italic,400italic'
rel='stylesheet' type='text/css'>
    <link rel="stylesheet" href="css/style.css">
    <script>
     alert("Hello, world!");
    </script>
</head>
<body>
    <header>
        <h1>IDV<span>460</span></h1>
        <p>Interactive Data Viz Spring 16</p>
        <h2>Steve Layton</h2>
    </header>

    <div id="page-wrap">
    <nav>
        <ul>
            <li><a href="index.html" class="active">Home</a></li>
            <li><a href="class.html">Class</a></li>
            <li><a href="projects.html">Projects</a></li>
        </ul>
    </nav>

    <main>
        <h1>Change my color!</h1>
        <p>The headline is different from the body copy.</p>

    </main>
    </div> <!-- closes page-wrap -->

    <footer>
        <p class="logo">IDV<span>460</span></p>
        <p><a href="mailto:stlayton@indiana.edu">stlayton@indiana.edu</a></p>
    </footer>


</body>
</html>
```



```
    <script>
     alert("Hello, world!");
    </script>
</body>
</html>
```

WHERE TO PUT JS?

In either case, though, we will be using external Javascript rather than inline. That is, we will have a separate file for our Javascript code, which we will add to a folder called “**js**”. We still use the script tag to access this file, with the location of the file described in the **src** attribute.



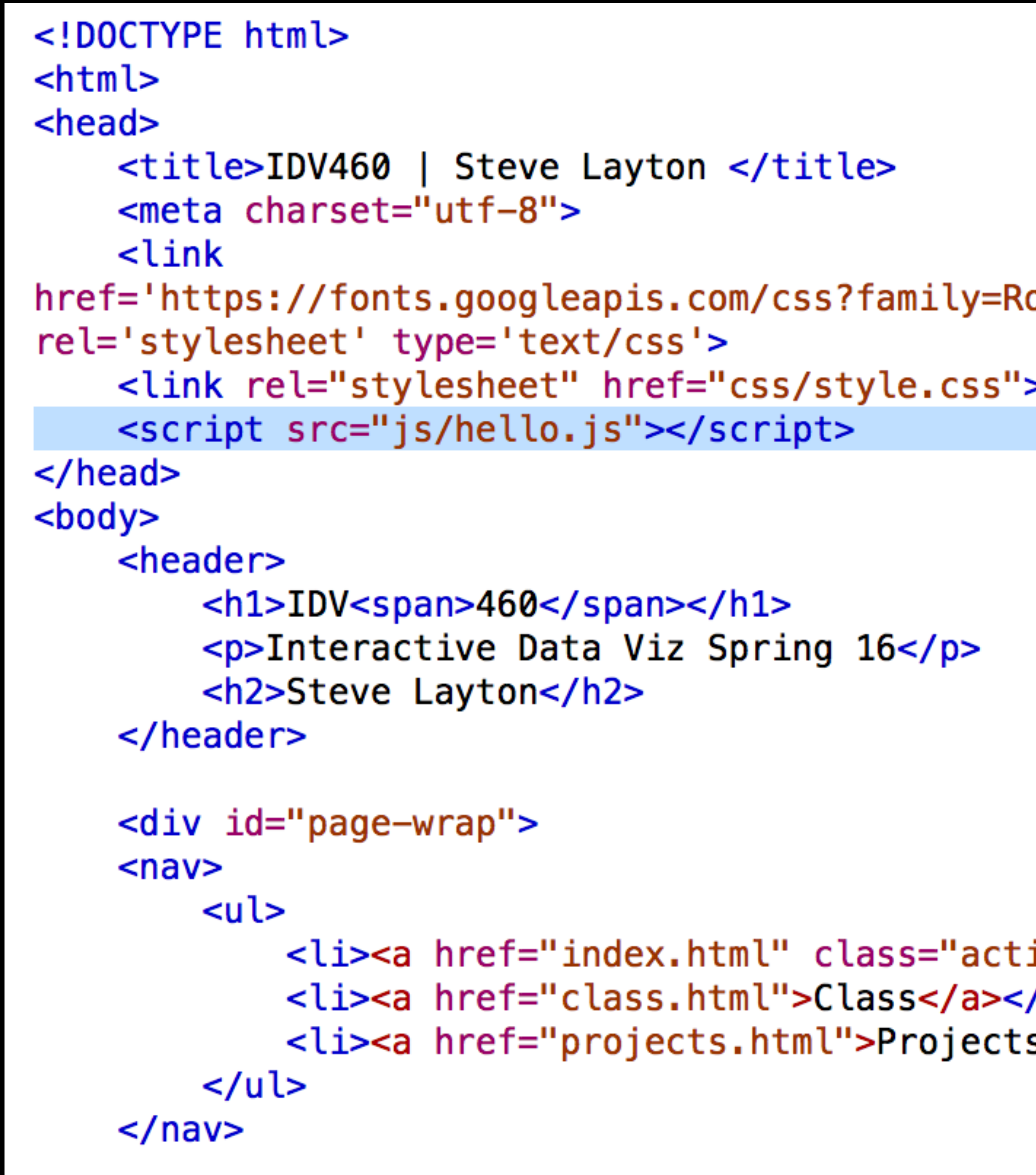
```
<!DOCTYPE html>
<html>
<head>
    <title>IDV460 | Steve Layton </title>
    <meta charset="utf-8">
    <link
href='https://fonts.googleapis.com/css?family=Ro
rel='stylesheet' type='text/css'>
    <link rel="stylesheet" href="css/style.css">
    <script src="js/hello.js"></script>
</head>
<body>
    <header>
        <h1>IDV<span>460</span></h1>
        <p>Interactive Data Viz Spring 16</p>
        <h2>Steve Layton</h2>
    </header>

    <div id="page-wrap">
    <nav>
        <ul>
            <li><a href="index.html" class="acti
            <li><a href="class.html">Class</a></
            <li><a href="projects.html">Projects
        </ul>
    </nav>
```

SYNTAX

Today, we will look at various parts of Javascript **syntax** — this means the basic rules and principles upon which the language is based.

This includes:

- Variables
- Conditions
- Operators
- Functions



```
<!DOCTYPE html>
<html>
<head>
    <title>IDV460 | Steve Layton </title>
    <meta charset="utf-8">
    <link
href='https://fonts.googleapis.com/css?family=Ro
rel='stylesheet' type='text/css'>
    <link rel="stylesheet" href="css/style.css">
    <script src="js/hello.js"></script>
</head>
<body>
    <header>
        <h1>IDV<span>460</span></h1>
        <p>Interactive Data Viz Spring 16</p>
        <h2>Steve Layton</h2>
    </header>

    <div id="page-wrap">
    <nav>
        <ul>
            <li><a href="index.html" class="acti
            <li><a href="class.html">Class</a></
            <li><a href="projects.html">Projects
        </ul>
    </nav>
```

VARIABLES

A Javascript variable can be thought of as a container that stores data.

You create a variable by typing var, which is part of the Javascript vocabulary, then any word you want.

A variable name is up to you, BUT:

• It should be descriptive of the data

• Do not include any spaces

• It can be letters, numbers, can include underscore or dollar sign, but CANNOT begin with a number.

```
var year;
```

VARIABLES

Simply creating a variable does not assign a value to that variable — that must be done with additional code. So we can create a variable called year, but that data has no value until we assign one. In this case, the equal sign is what sets a value to the variable called “year.”

```
var year;
year = 2016;
```

VARIABLES

This can be combined into a single line to tighten the code. In fact, you can create multiple variables and assign multiple values within a single line if you wish, with each variable name and value instruction separated by a comma.

```
var year = 2016, month =
february, day = 22;
```

VARIABLES

Variables can be set to any kind of data — numbers, strings, which are letters and words within a set of quotes, or Boolean values, which are true/false. They can contain more complex data, including arrays, objects and even functions.

```
var myVariable = 100;
var myVariable = "Hi!"
var myVariable = true;
```

CONDITIONS

Beyond declaring variables, we need to start to set conditions — to execute certain code depending on certain situations. This involves various user actions, but can also be called in with a classic “if” statement, common to virtually all computer programming.

The format for this includes an “if” statement, followed by a condition within a set of parentheses, and a set of curly braces that tells the computer what to do if the condition is true.

```
if ( condition ) {
 //execute this code;
}
```

OPERATORS

Almost every statement we write is going to need some sort of operation to be performed on the data. Operators are symbols we use to manipulate values.

The simplest ones are arithmetical:

+ for addition

- for subtraction

* for multiplication

/ for division

```
var a = 100, b = 50;
resultX = a + b;
resultY = a - b;
resultZ = a * b;
resultZZ = a / b;
```

OPERATORS

Remember, the equal sign is used in programming to ASSIGN A VALUE, not to check a value. If you want to check a value — in other words, to check and see if a is equal to 100, you would write a double equal sign.

```
var a = 100, b = 50;
resultX = a + b;
resultY = a - b;
resultZ = a * b;
resultZZ = a / b;
if (resultZ == 5000) {
 alert("Wow!");
}
```

OPERATORS

You can also use operators to determine whether a value is more than or less than a certain value:

> for greater than

< for less than

>= for greater than or equal to

<= for less than or equal to

```
var a = 100, b = 50;
var resultX = a * b;
if (resultX >= 5000) {
 alert ("Attsa lot!");
}
```

OPERATORS

There are many other operators that make up Javascript; we will look into these in more detail in classes to come.

```
++

--

!==

**

>>

<<

||
```

FUNCTIONS

Functions consist of a series of statement that perform a specific task. A function can be called upon by a user action in order achieve a task. This is known as “calling” a function.

The steps that the function performs are packaged in a **code block**, contained within a set of curly braces. Unlike a statement, a code block does not end with a semicolon.

```
function sayWow() {
    var x = document.getEl
    x.innerHTML = "Wow!";
}
```

FUNCTIONS

To create a function, you declare it with the keyword function, followed by an identifier — the name for this is up to you, but it is better to be descriptive of what it will do.

To run the code within the function, you follow the function name with a pair of parentheses.

```
function sayWow() {
    var x = document.getEl
    x.innerHTML = "Wow!";
}
```

FUNCTIONS

Sometimes, a function needs information to perform its task. In such a case, you provide parameters inside the parentheses.

```
function getArea(width, height) {
    var area = width * height;
    return area;
}
var lotOne = getArea(100,200);
var lotTwo = getArea(50,125);
```

CHECKPOINT!

Your grade for your own site is 30 percent of your grade for this class. I will out your site through the first checkpoint on WEDNESDAY. Here is what you will need to earn 100 percent.

- **index.html** (home page) with image and bio
- **class.html** page with listing of our in-class activities as sections, along with working links to this pages:

  cong.html, deluna,html, quiz1.html, time.html, primary.html
- working pages and working css throughout