

Responsive Design & Introduction to JavaScript

CISC-2350-R01 | Fall 2017 | Week 11-2

Ruta Kruliauskaite

Today's Agenda

- Attendance
- Homework Show & Tell (CSS animations)
- CSS Responsive Design
- Introduction to JavaScript
 - Overview
 - Properties, Events, Methods
 - Writing script
 - Syntax
 - Variables
- Homework Assignment

Homework show & tell

<http://bit.ly/2xzLHvt>

Intro to Responsive Design

What is responsive design?

- It makes your website look good on all the devices
- It can be done using only HTML and CSS, without JavaScript



iPhone 4

Size: 3.5 inches

Resolution: 960 x 640 pixels



iPad 2

Size: 9.7 inches

Resolution: 1024 x 768 pixels



13" MacBook

Size: 13.3 inches

Resolution: 1280 x 800 pixels



27" iMac

Size: 27 inches

Resolution: 2560 x 1440 pixels

What is responsive design?

- It adapts information to fit any device



Desktop



Tablet



Phone

The Viewport

- The viewport is the user's visible area of a web page (remember the “above the fold”?)
- You can set it in the `<meta>` tag in the `<head>` section of an HTML document
 - `<meta>` gives the browser instructions on how to control the page's dimensions and scaling
 - `width=device-width` sets the width of the page to follow the screen-width of the device (varies)
 - `initial-scale=1.0` sets the initial zoom level when the page is first loaded by the browser

`<meta name="viewport" content="width=device-width, initial-scale=1.0">`

Example 1
Example 2

Grids

- We can also build responsive grids
- A responsive grid-view often has 12 columns, and has a total width of 100%, and will shrink and expand as you resize the browser window
- Notice using of “%” is important, so it’s not a fixed layout

Building a responsive grid

1. border-sizing should be set to border-box
2. Entire width of the grid equals to 100%
3. One column equals to 8.33% (100 divided by 12)
4. All columns should float left
5. All rows should be wrapped in a `<div>`

Example 1
Example 2

Media Query

1. Uses the @media rule to include a block of CSS properties only if a certain condition is true
2. For example, change the background color to blue if the screen is smaller than 400px:

```
@media only screen and (max-width: 400px) {  
  body {  
    background-color: lightblue;  
  }  
}
```

Example

Adding breakpoints with media queries

1. Using media queries we can add breakpoints when the width of columns should change (for example, become 100% when browser width becomes small)
2. You can add as many breakpoints as you want (if you want to add tablets, etc.) - then you'd have to add multiple classes to the same HTML element, so it knows what to do at each breakpoint

```
@media only screen and (max-width: 768px) {  
  [class*="col-"] {  
    width: 100%;  
  }  
}
```

Example

Mobile first

1. Mobile first means designing for mobile devices first and only later for larger screens

```
@media only screen and (min-width: 768px) {  
  .col-1 {width: 8.33%;}  
  .col-2 {width: 16.66%;}  
  .col-3 {width: 25%;}  
  .col-4 {width: 33.33%;}  
  .col-5 {width: 41.66%;}  
  .col-6 {width: 50%;}  
  .col-7 {width: 58.33%;}  
  .col-8 {width: 66.66%;}  
  .col-9 {width: 75%;}  
  .col-10 {width: 83.33%;}  
  .col-11 {width: 91.66%;}  
  .col-12 {width: 100%;}  
}
```

Example

Bootstrap

Bootstrap

- Bootstrap is the main piece you should aim to get to with your web development
- It's also just a library like jQuery or 960.gs, but it uses jQuery/JavaScript and CSS to make websites work for mobile without major issues
- You can do everything Bootstrap does yourself, but Bootstrap makes it easier to think about responsive and mobile first sites
- You need to have the jQuery code linked in your website
- Their website is pretty good about simple templates
- You'll need to start using meta tags, and making sure all CSS and JS files are linked to appropriately!!!

To use Bootstrap

- You can download it from getbootstrap.com
- Or link it from CDN (must include jquery as well): <https://cdnjs.com/libraries/>

```
<!-- Latest compiled and minified CSS -->  
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/  
bootstrap/3.3.7/css/bootstrap.min.css">
```

```
<!-- jQuery library -->  
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/  
jquery.min.js"></script>
```

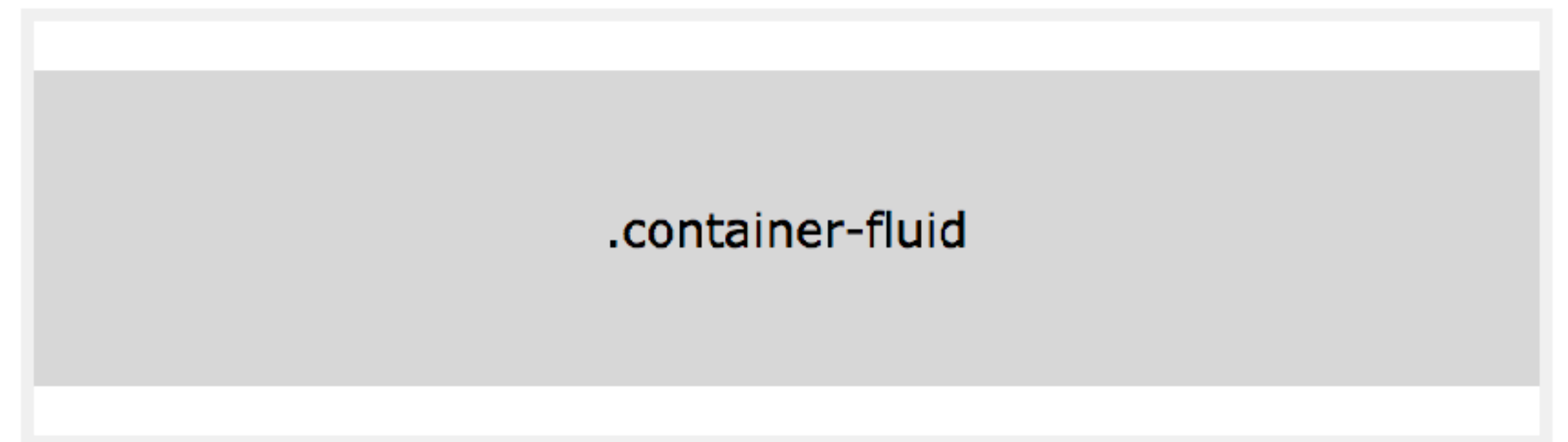
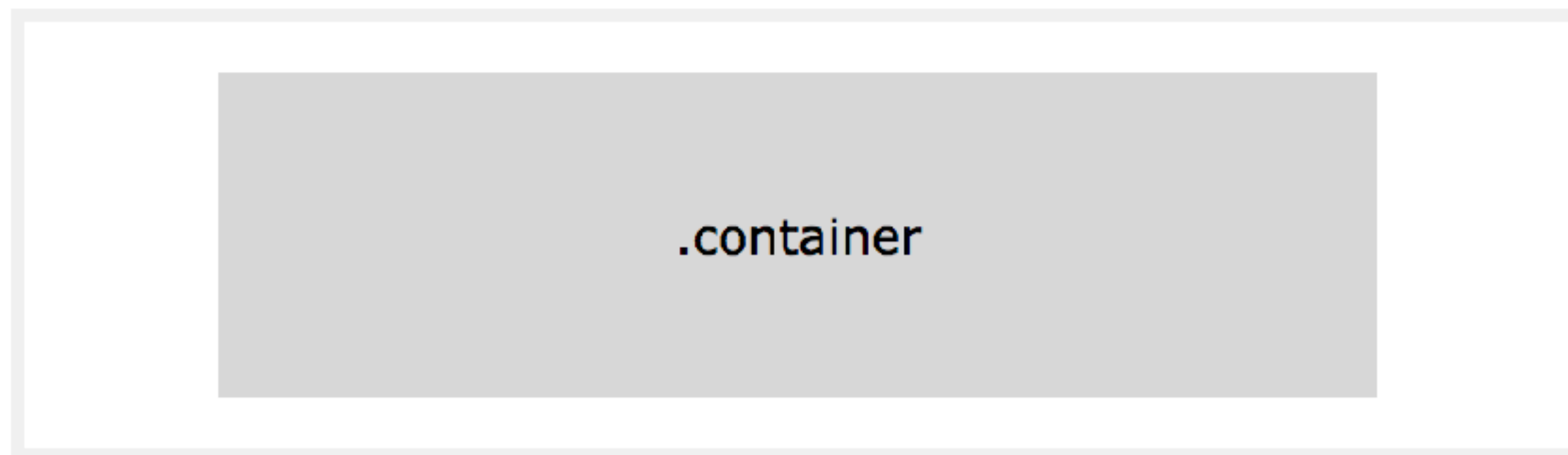
```
<!-- Latest compiled JavaScript -->  
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/  
bootstrap.min.js"></script>
```


Creating Bootstrap website

- Do not forget !doctype at the top of HTML page
- Also lang="en" in the opening <html> tag & charset="utf-8" in the <meta> tag in <head> section
- Set a viewport:
 - `<meta name="viewport" content="width=device-width, initial-scale=1">`

Choosing Bootstrap container

- You have two container classes to choose from:
 - `.container` - a fixed width container
 - `.container-fluid` - full width container that spans the entire width of the viewport



Creating Bootstrap grid system

- Bootstrap's grid system is responsive and will rearrange automatically depending on a screen size
- Grid classes:
 - `xs` - for phones, screens less than 768 px
 - `sm` - for tablets, ≥ 768 px
 - `md` - for small laptops, ≥ 992 px
 - `lg` - for laptops and desktops, ≥ 1200 px
- Classes can be combined to create more dynamic and flexible layouts
- Columns in the same row should be wrapped by `<div>` with a class "row"
- Columns in the same row should add up to 12

Bootstrap grid example

Introduction to JavaScript

HTML



CSS



JS



Web Page

```
graph TD; WP[Web Page] --- HTML[HTML]; WP --- CSS[CSS]; WP --- JS[JavaScript]; HTML --- HTML_Sub[Content & Structure]; CSS --- CSS_Sub[Presentation]; JS --- JS_Sub[Behavior]; HTML_Sub --- HTML_Examples[Headings, Paragraphs, Lists]; CSS_Sub --- CSS_Examples[Font, Color, Background color, Border]; JS_Sub --- JS_Examples[dynamic display, widgets, user interaction, click to open a popup];
```

HTML

Content & Structure

Headings,
Paragraphs
Lists

CSS

Presentation

Font
Color
Background color
Border

JavaScript

Behavior

dynamic display
widgets
user interaction
click to open a popup

**JavaScript is a scripting language,
and is optimal for creating dynamic
content on the web.**

What JavaScript can do

1. **Access content:** you can use JavaScript to select any element, attribute or text from an HTML page.
2. **Modify content:** you can use JavaScript to add elements, attributes, and text to the page or remove them.
3. **Program rules:** you can specify a set of steps for the browser to follow, which allows it to access or change the content of a page.
4. **React to events:** you can specify that a script should run when a specific event has occurred.

How do I write JavaScript code?

1. **Script** is a series of interactions that a computer can follow to achieve a goal (just like a recipe, a handbook or manual).
2. You need to start with the big picture of what you want to achieve, and break that down into smaller steps:
 1. Define the goal
 2. Design the script
 3. Code each step
3. Use a **language** that computer can understand and follow in a **syntax** that it can read.

Example: find the tallest person in a room

1. Find the height of the first person
2. Assume s/he is the tallest person
3. Look at the height of the remaining people one by one and compare their height to the first “tallest person”
4. At each step, if you found someone taller, s/he becomes the new “tallest person”
5. Once you checked all the people, define which one is the tallest

**Computers create models of the
world using data.**

Object-oriented programming

1. Models include many objects (e.g. tree, car, etc.)
2. In object-oriented programming, objects can be repeated over and over and consist of:
 1. Properties (characteristics)
 2. Events
 3. Methods

Properties

1. Characteristics that you can tell about the object (e.g. color, speed)
2. Programmers call these characteristics **properties**
3. Each property has a pair of **name** and **value** (e.g. color = blue)

Events

1. Events (interactions) with objects can change the values of the properties
2. When a specific event happens, that event can be used to trigger a specific section of the code
3. Use different events to trigger different types of functionality

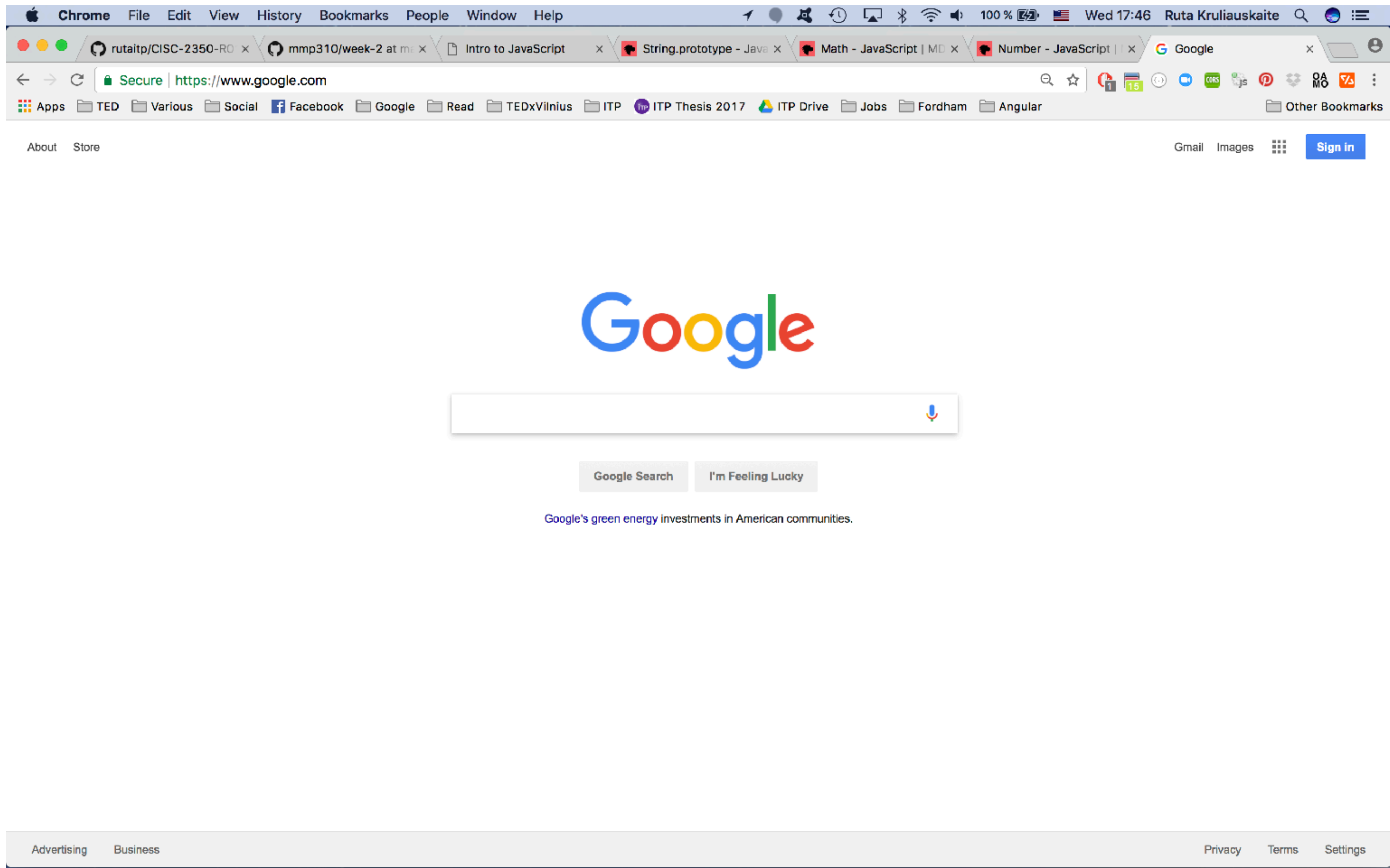
For example: change background color to blue when this button is clicked!

Methods

1. The code for a method can contain a lot of instructions that together represent one task (e.g. `changeSpeed()`);

Events trigger methods -> methods retrieve or update an object's properties

1. Event: button clicked;
2. Calls method `changeBackgroundColor()`;
3. Updates property `backgroundColor` to blue;



window, document

1. window object is each window or tab open in a browser
 1. try writing window.location in the window inspect (it will return the current url)
2. document object is the current webpage loaded in the window
 1. try writing document.title (it will return the <title> in the current HTML document)

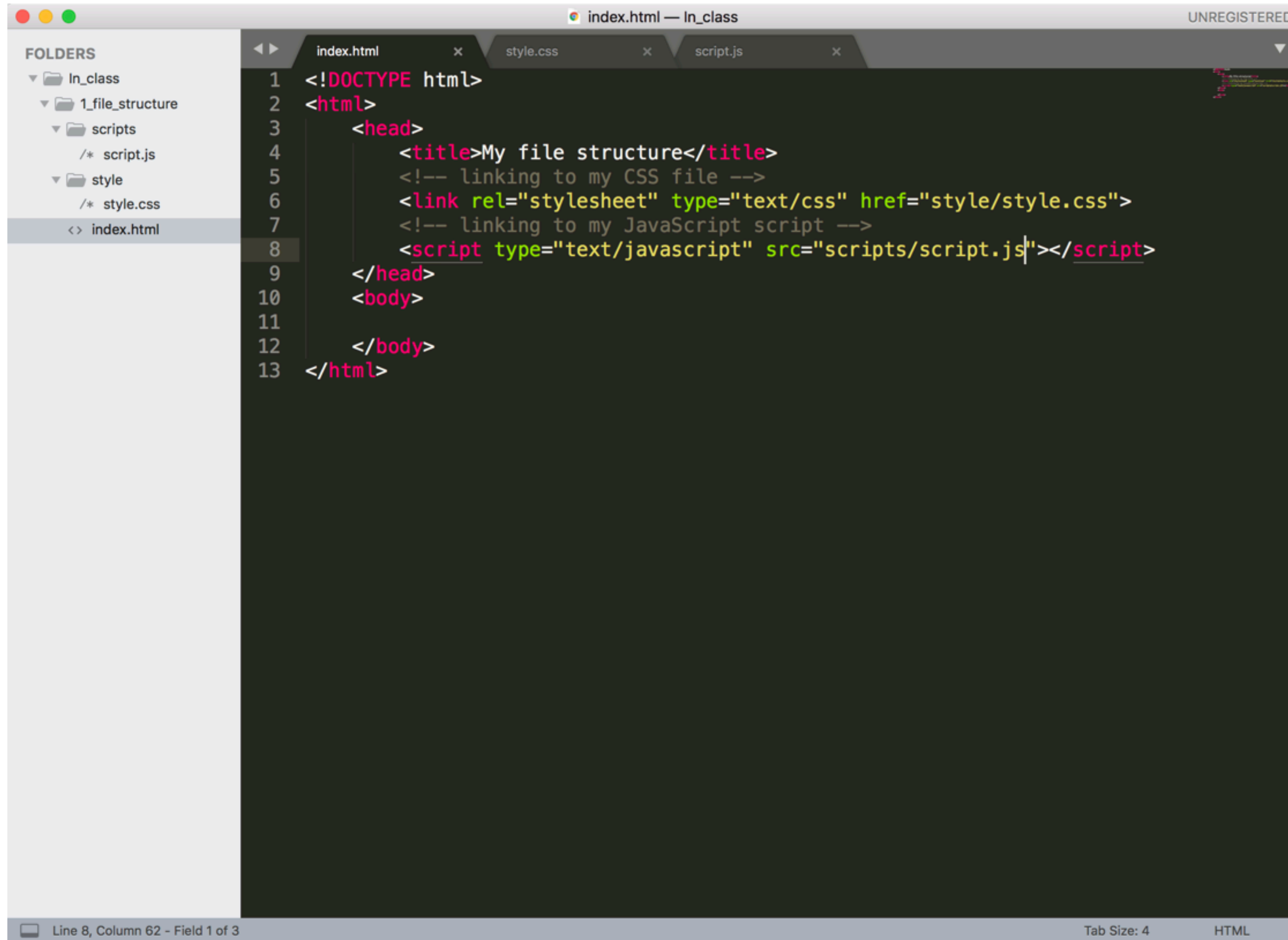
Try these in your inspect window

1. `console.log("Hello World");`
2. `window.location;`
3. `document;`
4. `document.body;`
5. `document.body.style;`
6. `document.body.style.color = "pink";`
7. `document.write("New Website!");`

Manipulating content on the existing site

1. Go to: <https://p5js.org/>
2. Open inspect window
3. We'll change background color of the entire homepage:
 1. `var background = document.getElementById("home-page");`
 2. `background.style.backgroundColor = "lightblue";`
 3. `background.innerHTML = "New text";`

Where do I put my JS files?



The screenshot shows a code editor window titled "index.html — In_class" with a status bar indicating "UNREGISTERED". The editor has three tabs: "index.html", "style.css", and "script.js". The "index.html" tab is active, displaying the following HTML code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>My file structure</title>
5     <!-- linking to my CSS file -->
6     <link rel="stylesheet" type="text/css" href="style/style.css">
7     <!-- linking to my JavaScript script -->
8     <script type="text/javascript" src="scripts/script.js"></script>
9   </head>
10  <body>
11
12  </body>
13 </html>
```

The left sidebar shows a "FOLDERS" panel with the following structure:

- In_class
 - 1_file_structure
 - scripts
 - /* script.js
 - style
 - /* style.css
 - index.html

The status bar at the bottom indicates "Line 8, Column 62 - Field 1 of 3", "Tab Size: 4", and "HTML".

Example

How do I link them?

```
<!-- linking to my JavaScript script -->  
<script type="text/javascript" src="scripts/script.js"></script>
```



<script> instead of <link>



src="" instead of href=""

Console.log()

1. `console.log()` is your best friend to debug code;
2. You can use it right in the inspect window or from `script.js`
3. **Log every single step throughout your code to see if things are working!**

JavaScript Syntax

JavaScript Syntax

1. A script is a series of instructions that a computer can follow one-by-one.
2. Each individual instruction or step is known as a **statement**.
3. Each statement must end with semicolon ;

JavaScript Syntax 2

1. Semicolon tells when a step is over and JavaScript interpreter should move to the next step
2. Each statement starts **on a new line**
3. JavaScript is **case sensitive**: myName is not equal to myname or MYNAME
4. Some statements can be organised into **code blocks**:
if (value > x) {
 //do this;
} (notice no ; after the code block)

JavaScript Comments

1. You should **ALWAYS** write comments to explain what your code does (especially JavaScript logic)
2. There can be single-line and multi-line comments
3. Single-line comments are written with two forward slashes `//`
 1. Often used for short descriptions of what the code is doing
4. Multi-line comments are written using `/*` and `*/` characters
 1. Often used for descriptions of how the script works, or to prevent a section of the script from running when testing it

Variables

What is a variable?

1. **Variable** is a container for storing data values
2. The data stored in a variable can change (or vary) overtime a script runs
 1. It's good for reusing it in multiple places in a script
 2. You also don't have to remember the value each time

Declaring variables

1. Before you can use a variable, you need to announce that you want to use it
2. To announce it you need to create a variable and give it a name (**declare it**)

```
var speed;
```

3. If a name is a few words, it should come in **camelCase**:

```
var mySpeed;
```

Assigning a value to a variable

1. Once you created a variable, you need to tell what information it should store (**assign a value**)
2. To announce it you need to create a variable and give it a name (**declare it**)
3. Until you have assigned a value, it's **undefined**

assignment operator
↓
speed = 7;
↑ ↑
variable name variable value

Data types: numbers

1. Numeric data type handles numbers
2. Use `typeof` to log what kind of data type you're using

```
var price;  
var quantity;  
var total;  
  
price = 5;  
quantity = 14;  
total = price * quantity;  
  
console.log(typeof total);
```

Example

Data types: strings

1. The string data type consists of letters and other characters
2. They have to be enclosed with a pair of quotes (single or double)
 1. Opening quote must match the closing quote
3. They can be used working with any kind of text

```
var myName;  
var myMessage;
```

```
myName = "Ruta";  
myMessage = "I am binge watching The Big Bang Theory!";
```

```
console.log(typeof myName);
```

Example

Using quotes inside a string

1. Sometimes you will want to use a double or single quote mark within a string
2. If you want to use double quotes inside a string, you should surround it by single quotes and vice versa.
3. You can also use backward slash before a quote to say that the following character is part of the string, rather than the end of it;

```
var myName;  
var myMessage;  
var myMessageTwo;
```

```
myName = "Ruta";  
myMessage = "I am binge watching 'The Big Bang Theory'!";  
myMessageTwo = "I am binge watching \'The Big Bang Theory\'!";
```

Example

Data types: boolean

1. Boolean data type can have one of two values: **true** or **false**
2. They are helpful when determining which part of a script should run (when code can take more than one path);

```
var todayIsCloudy;  
todayIsCloudy = true;  
  
if (todayIsCloudy == true) {  
    // if true, show this  
    console.log("Yes, today is cloudy!");  
    // otherwise show this  
} else {  
    console.log("No, it's actually sunny today!");  
}
```

Example

Shorthand for creating variables

1. You can use these shorthands to create variables:

1:

```
var speed = 5;  
var quantity = 14;  
var total = speed * quantity;
```

2:

```
var speed, quantity, total;  
speed = 5;  
quantity = 14;  
total = speed * quantity;
```

3:

```
var speed = 5, quantity = 14;  
var total = speed * quantity;
```

Changing the value of a variable

1. Once you have assigned a value to a variable, you can then change it later in a script

```
var speed = 5;  
var quantity = 14;  
var total = speed * quantity;
```

//maybe something changed here and the value has to change now

```
speed = 10;
```

```
var textToShow = document.getElementById("hello");  
textToShow.innerHTML = "Total cost is: " + "$" + total;
```

Rules for naming variables

1. There are **six rules** that you have to follow:
 1. The name must begin with a letter, \$ sign or an underscore _, it cannot start with a number (e.g. name, \$name, _name) - I would avoid \$, because it might confuse it with jquery
 2. You cannot use dash - or a dot . in a variable name (e.g. my-name, my.name)
 3. You cannot use keywords or reserved words as variable names (e.g. function, type, this, etc.) - it usually changes the color when you use it
 4. All variables are case sensitive (it is bad practice to create the same name using different cases (e.g. myName & Myname) - do not start with a capital letter in general
 5. Use a name that describes the kind of information that the variable stores (e.g. firstName, lastName)
 6. If a variable name is made up of more than one word use capital letter for the first letter of every word after the first one or underscore (e.g. myFirstName, myLastName, my_first_name, my_last_name)

Button click example

Homework assignment

Homework

By Sunday, November 12, 6pm

1. Review the class slides

2. Go through responsive design tutorials:

1. Codecademy class: <https://www.codecademy.com/learn/learn-responsive-design>
2. Bootstrap documentation:
 1. <http://getbootstrap.com/docs/4.0/getting-started/introduction/>
 2. <http://getbootstrap.com/docs/4.0/examples/grid/>
 3. <https://www.w3schools.com/bootstrap/>
3. Bootstrap tutorial:
 1. <https://www.youtube.com/watch?v=e5VQampkqdg>

3. Go through JavaScript tutorials:

1. Codecademy JS class: <https://www.codecademy.com/learn/introduction-to-javascript>
 1. Do the Introduction class (Introduction to JavaScript & Variables)
 2. Post on #general channel on Slack what new you learnt or what got more clarified

4. Create your first website using responsive Bootstrap grid:

- Make sure libraries are linked properly
- Do not forget to add columns to separate <div> under class="row"
- You can add any content (text or images), any column number, whatever you want, just make sure it makes sense
- Add some interactivity using JavaScript (a button, input, etc.). Make sure it displays in a browser
- Upload your code to a new folder on Github and post two links on #general channel on Slack:
 - Link to your project's repository (folder)
 - Working link to the Github page (yourusername.github.io/pathtofolder)