Introduction to JavaScript

CISC-2350-R01 | Fall 2017 | Week 12-1

Ruta Kruliauskaite

Today’s Agenda

- Attendance
- Introduction to JavaScript
    - Overview
    - Properties, Events, Methods
    - Writing script
    - Syntax
    - Variables
    - Arrays
    - Expressions & Operators
- Homework Show & Tell (Responsive Design)
- Web Review Presentations
- Homework Assignment

Introduction to JavaScript

HTML CSS JS

Web Page

HTML
Content & Structure

Headings,
Paragraphs
Lists

CSS
Presentation

Font
Color
Background color
Border

JavaScript
Behavior

dynamic display
widgets
user iteraction
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)

```
var car = {
   color: green,
   size: large,
   price: $40k,
   speed: fast
}
```

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

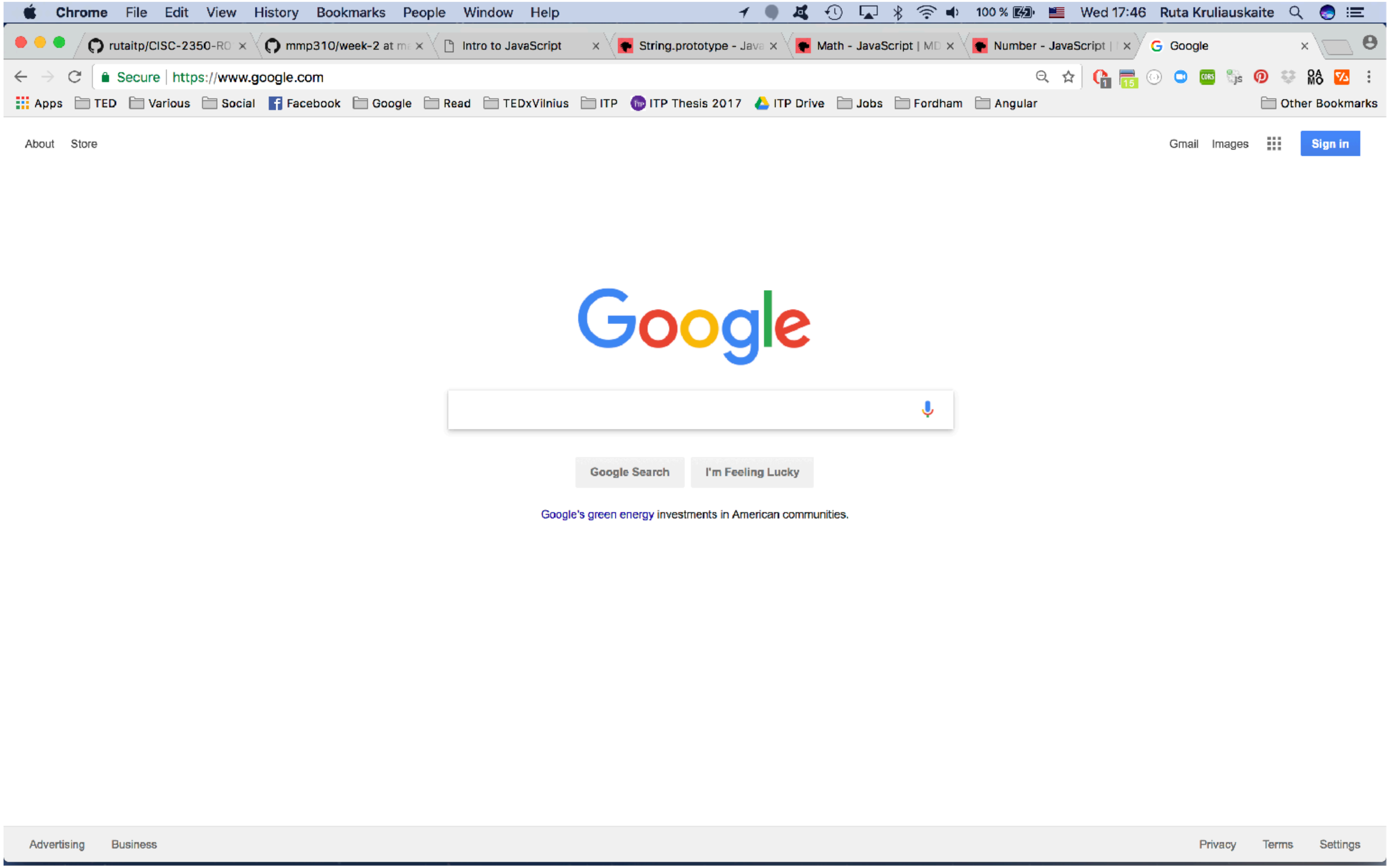
For example: change background color to blue when this button is clicked!

Methods

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

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;


Chrome File Edit View History Bookmarks People Window Help
100 % Wed 17:46 Ruta Kruliauskaite
rutaitp/CISC-2350-RO
mmp310/week-2 at m
Intro to JavaScript
String.prototype - Java
Math - JavaScript | MD
Number - JavaScript |
Google
Secure https://www.google.com
Apps TED Various Social Facebook Google Read TEDxVilnius ITP ITP Thesis 2017 ITP Drive Jobs Fordham Angular Other Bookmarks
About Store
Gmail Images Sign in
Google
Google Search
I'm Feeling Lucky
Advertising Business
Privacy Terms Settings

Google's green energy investments in American communities.

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)

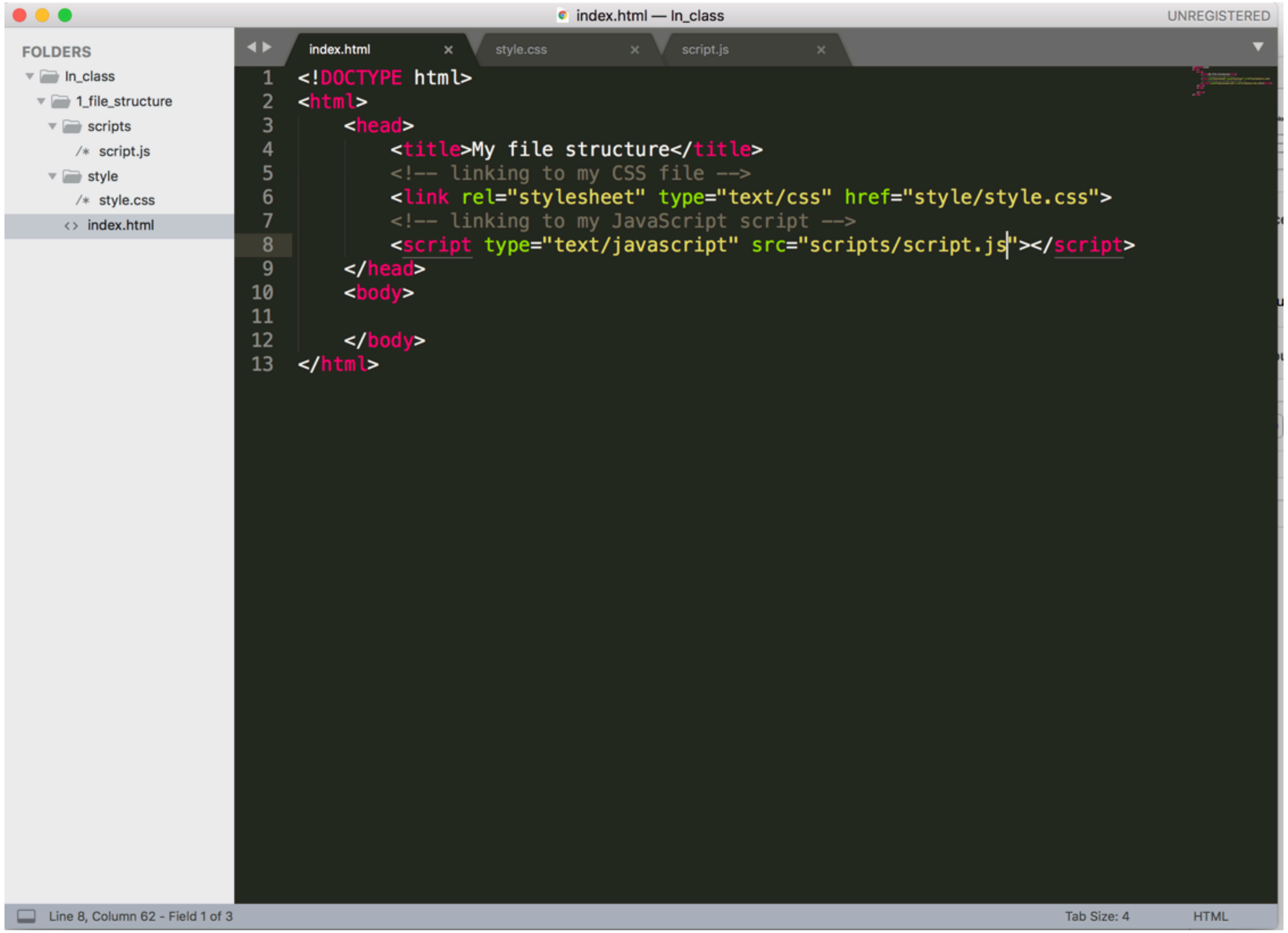
Try these in your inspect window

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?

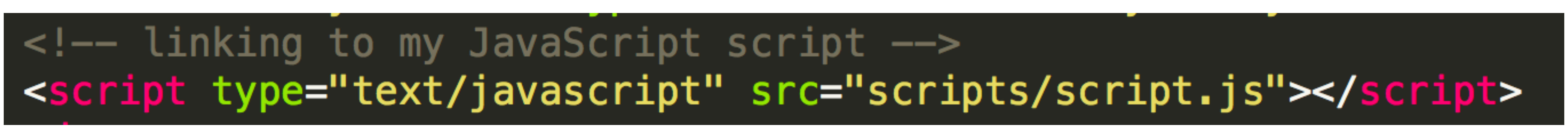


```
<!DOCTYPE html>
<html>
    <head>
        <title>My file structure</title>
        <!-- linking to my CSS file -->
        <link rel="stylesheet" type="text/css" href="style/style.css">
        <!-- linking to my JavaScript script -->
        <script type="text/javascript" src="scripts/script.js"></script>
    </head>
    <body>

    </body>
</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)**

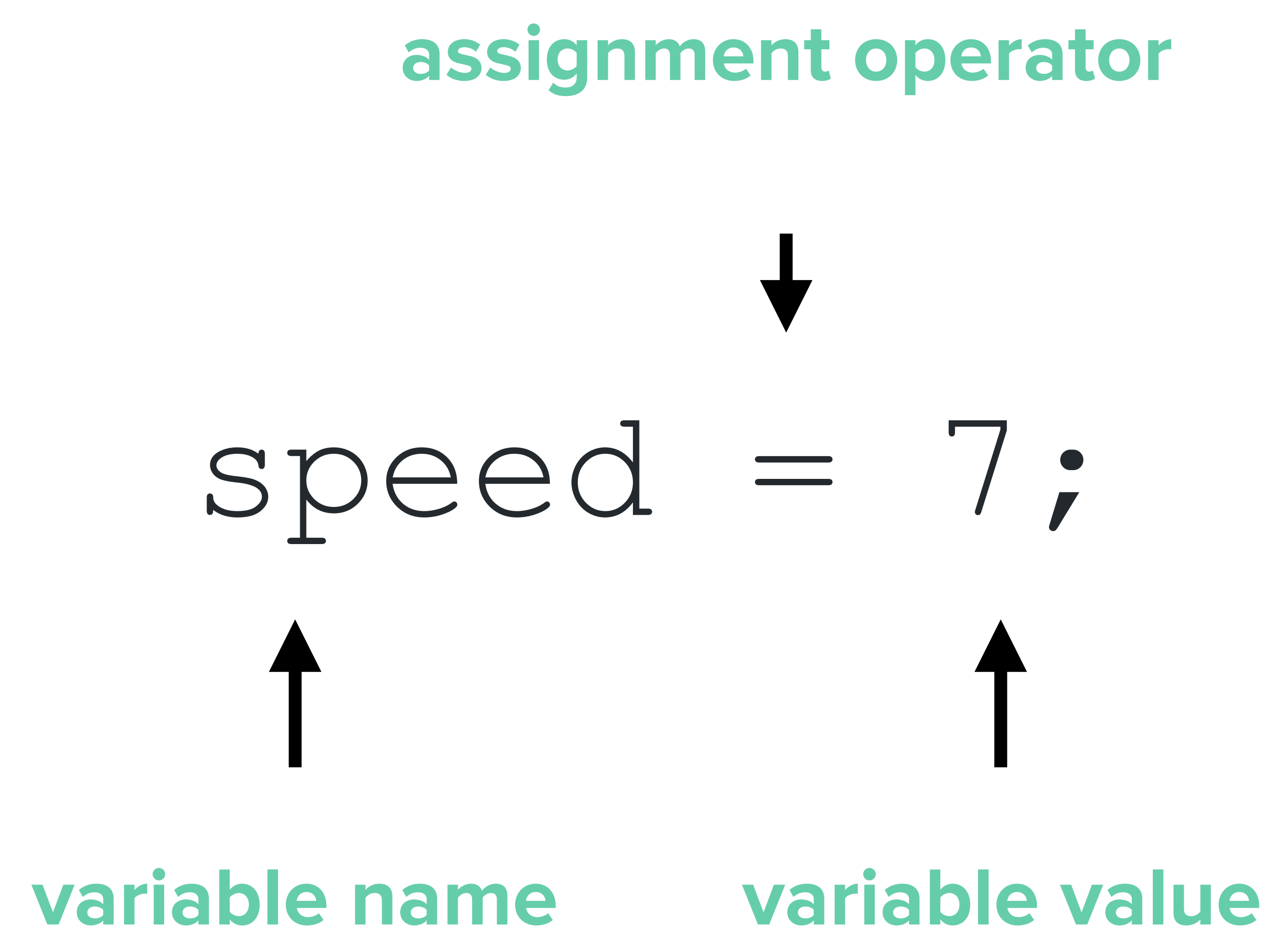
```
var speed;
```

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. Until you have assigned a value, it's **undefined**



```
speed = 7;
```

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

Arrays

An array is a special type of variable. It doesn’t just store one value; it stores a list of values.

When to use arrays

1. You should use an array whenever you’re working with a list of values that are related to each other
2. e.g. items in a shopping list, colors, prices, etc.

Creating an array

1. Create an array and give it a name just like any other variable:

```
var movies = [];
```

2. Create an array using **array literal** technique:

```
var movies = [“The Lobster”, “Get Out”, “Blade Runner”];
```

3. Create an array using **array constructor** technique:

```
var movies = new Array (“The Lobster”, “Get Out”, “Blade
 Runner”);
```

Note: values in an array do not have to be the same data type (could be string, number, boolean in one array).

Note2: array literal is preferred way to create an array.

Example

Values in arrays

1. Values in an array are accessed through their numbers they are assigned in the list.
2. The number is called **index** and starts from **0**.

```
var movies = ["The Lobster", "Get Out", "Blade Runner"];
```

index [0] **index [1]** **index [2]**

3. You can check the number of items in an array using `length` property

```
var moviesLength = movies.length;
```

Example

Changing values in the array

1. Let’s say we want to update the value of the third item (change “Blade Runner” to something else)
2. To access the current third value, we have to call the name of the array followed by the index number for that value inside the square brackets:

```
var movies = [“The Lobster”, “Get Out”, “Blade Runner”];
movies[2]
```

3. After we select a value, we can assign a new value to it:

```
movies[2] = “Gone Girl”;
```

4. When we log the the updated array, we see updated values:

```
movies = [“The Lobster”, “Get Out”, “Gone Girl”];
```

Example

Adding and removing values from the array

1. You can add values to the array using .push() method:

```
var movies = [“The Lobster”, “Get Out”, “Blade Runner”];
movies.push(“The Shining”);
```

2. You can remove values from the array using .splice() method:

```
var movies = [“The Lobster”, “Get Out”, “Blade Runner”, “The Shining”];
movies.splice(0,1);
```
(will remove “The Lobster” movie)

Where 0 is an index at what position an item should be removed, 1 how many items should be removed (in this case only one movie)

Example

Array example

More array documentation

1. https://www.w3schools.com/js/js_arrays.asp
2. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Expressions

An expression results in a single value (produces a value and is written whenever a value is expected).

Types of expressions

1. Expressions that just assign a value to a variable

```
var movie = “Blade Runner”;
```

2. Expressions that use two or more values to return a single value

```
var height = 50 * 3;
var sentence = “My name is “ + “Ruta”;
```

Operators

Operators allow us to create single values from one or more values.

Types of operators

1. Assignment operators (assign values to variables):
   `var movie = “Blade Runner”;`

2. Arithmetic operators (perform basic math):
   `var height = 50 * 3;`
3. String operators (combine two strings):
   `var sentence = “My name is “ + “Ruta”;`
4. Comparison operators (compare two values and return `true` or `false`):
   `height = 50 > 3` (will return false)
5. Logical operators (combine expressions and return `true` or `false`:
   `height = (50 < 3) && (3 > 2)` (will return true)

1. Arithmetic operators

From John Duckett book

NAME	OPERATOR	PURPOSE & NOTES	EXAMPLE	RESULT
ADDITION	+	Adds one value to another	10+5	15
SUBTRACTION	-	Substracts one value from another	10-5	5
DIVISION	/	Divides two values	10/5	2
MULTIPLICATION	*	Multiplies two values	10*5	50

INCREMENT	++	Adds one to the current number	i=10;
			i++;
				11

DECREMENT	- -	Subtracts one from the current number	i=10; i- -;	9
MODULUS	%	Divides two values and returns the remainder	10%3;	1

2. String operator

String operator

1. There is only one string operator **+**
2. It’s used to join the strings together

```
var firstName = “Ruta ”;
var lastName = “Kruliauskaite”;

var fullName = firstName + lastName;
```

4. Process of joining two or more strings together into a new one is called **concatenation**.
5. If you’ll try to add other arithmetic operators on a string, it will return **NaN** (meaning **not a number**):

```
var fullName = firstName * lastName;
```

String operator example

Homework show & tell

http://bit.ly/2xzLHvt

Web Review Presentations

Homework assignment

Homework

By Wednesday, November 15, 6pm

1. **Review the class slides**
2. **Go through more of JavaScript documentation and tutorials:**
    1. Codecademy JS class: https://www.codecademy.com/learn/introduction-to-javascript
        1. Do the Control Flow and Functions classes
        2. Post on #general channel on Slack what new you learnt or what got more clarified
    2. Documentation on arrays:
        1. https://www.w3schools.com/js/js_arrays.asp
        2. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
    3. JavaScript tutorial on W3SCHOOLS:
        1. https://www.w3schools.com/js/
3. **Take one of your old homework and add some interactivity using JavaScript. It could be:**
        - Some styling properties changing when the button is clicked
        - Data displayed when the user enters data into your form, and so on
    - Make sure to clearly comment your code and process, so it's easy to follow
    - Use variables and other things we've learned so far in class
    - Make sure you have proper file structure
    - The website should be responsive (using Bootstrap or other framework we used)
    - 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)