
LET'S GET EVERYTHING SET UP!

1. In Schoology, go to: **Courses(in the top menu) > FEWD CHI 1: Section 1**
2. Then go to the **Class Materials** folder — it's the pink one!
3. Navigate to the **Week 4 (It's the yellow folder) > Lesson 7 folder**
4. There you'll find all the materials for today's class
5. Download `starter_code_lesson_7.zip`
6. Move it from your Downloads folder to your Desktop
7. Double-click on `starter_code_lesson_7.zip` to unzip it
8. After you've unzipped, delete the original .zip to avoid confusion and make sure you don't unzip it again later!!!

GA GENERAL ASSEMBLY

JS BASICS

Sarah Holden

JAVASCRIPT

WHERE ARE WE GOING?



YOUR RESPONSIBILITIES

Don't feel like you have to sit down and memorize the syntax!

It's important that you:

- Focus on understanding the key concepts
- Are resourceful — we'll work on honing your Google-ing skills
- Get as much practice in as possible

THE NEXT COUPLE OF WEEKS

JS BASICS — LESSONS 1 AND 2

- The first two JavaScript classes we'll be focusing on JS fundamentals — variables, data types, conditionals and functions.
- These classes will be more 'theory' focused.
- Labs will include building a thermostat and a Rock Paper Scissors game

JS BASICS — LESSONS 3 AND 4

- The second two classes we'll be learning more JS fundamentals — but this time we'll be getting into the DOM more as we introduce jQuery.
- The focus will be on understanding how we can use the theory/foundations we've learned to add interactions and behavior to our pages.
- Labs will include an interactive todo list and an image gallery slider.

JS REVIEW & REFACTOR — LESSONS 5 AND 6

- Lesson 5 will be an all-class lab — a chance to put your new skills to use!
- Lesson 6 will be focused on refactoring — learning how to keep our code clean and fix code

LEARNING OBJECTIVES

- Define variables and identify best cases to use them.
- Differentiate between strings, integers and floats.
- Apply conditionals to change the program's control flow

AGENDA

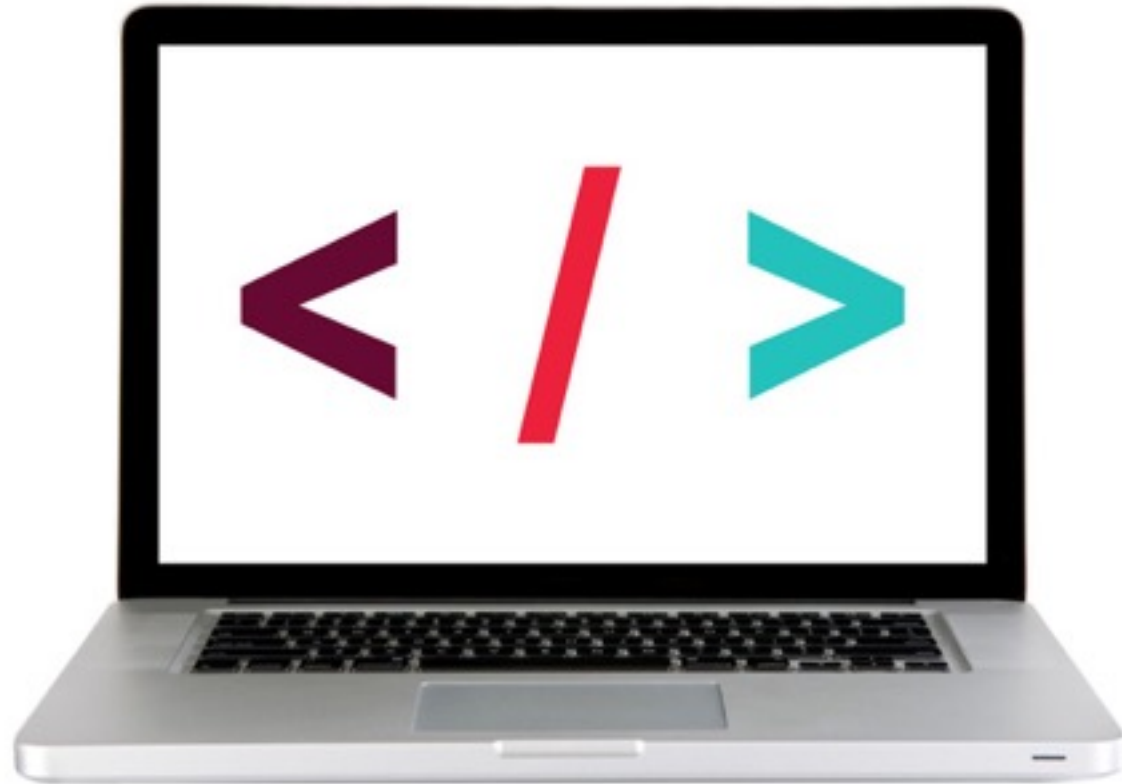


- JS Basics
- Adding JavaScript to a Project
- Variables/Data Types
- Conditionals
- Lab — Temperature Converter

JS BASICS

JS BASICS

LET'S TAKE A CLOSER LOOK



View the code in [CodePen](#)

STATEMENTS

- Last lesson we chatted about how scripts are a series of instructions that are executed one-by-one
- Each individual step is called a **statement**

```
var name = "John";
```

Keep it clean! Each statement should:

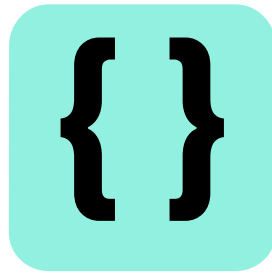


1. Begin on a new line
2. End with a **semicolon**

JS SYNTAX

Syntax: Spelling and grammar rules of a programming language.

- Like any language, there are formal rules around how to write Javascript. This is the syntax.



COMMENTS

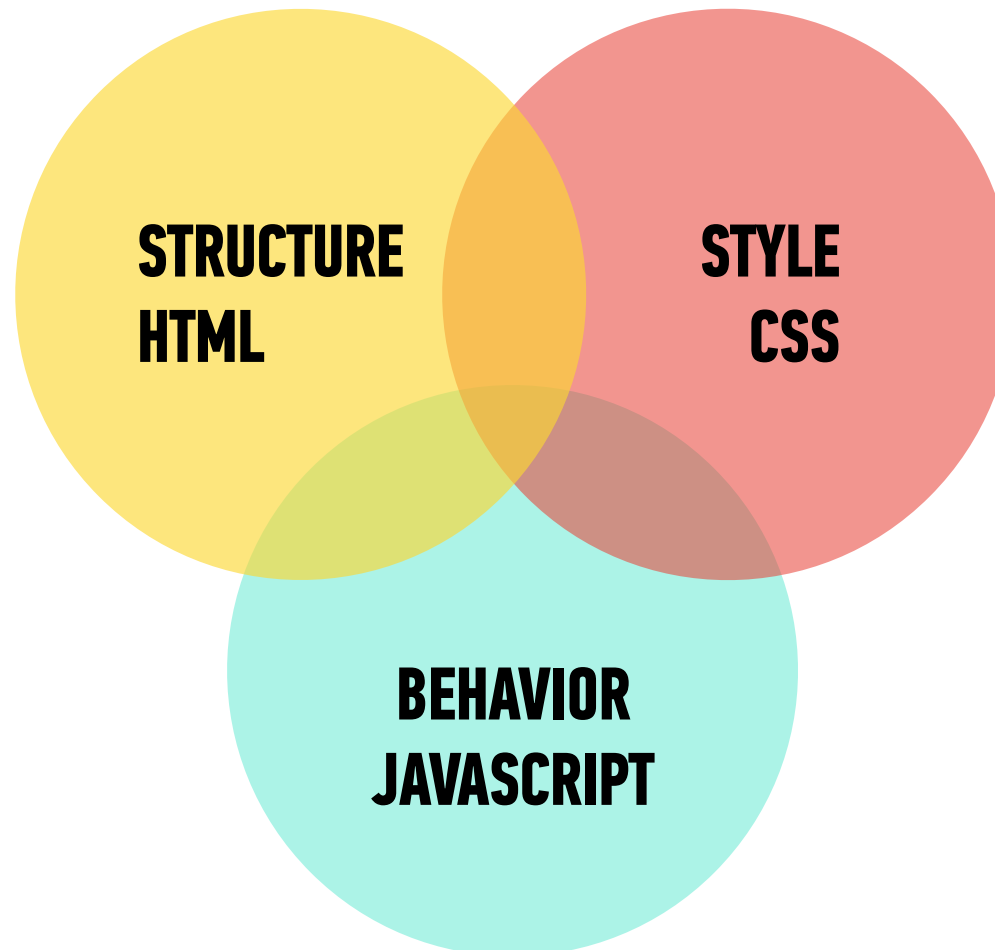
```
// this is a single line comment
```

```
/*  
this  
is  
a  
multiline comment  
*/
```

SEPARATION OF CONCERNS

THE THREE AMIGOS: STRUCTURE, STYLE, BEHAVIOR

- HTML = Noun
- CSS = Adjective
- Javascript = Verb



SEPARATION OF CONCERNS

- Our JavaScript should focus on *behavior* and not on *presentation* (that's what our CSS is for!)
- How could we refactor our Color Switch from last week to follow this guideline?

WORKING WITH THE DOM — AN OVERVIEW

WORKING WITH THE DOM TREE

TO ACCESS AND MODIFY THE DOM:

1

Select an element/elements

2

Work with those elements

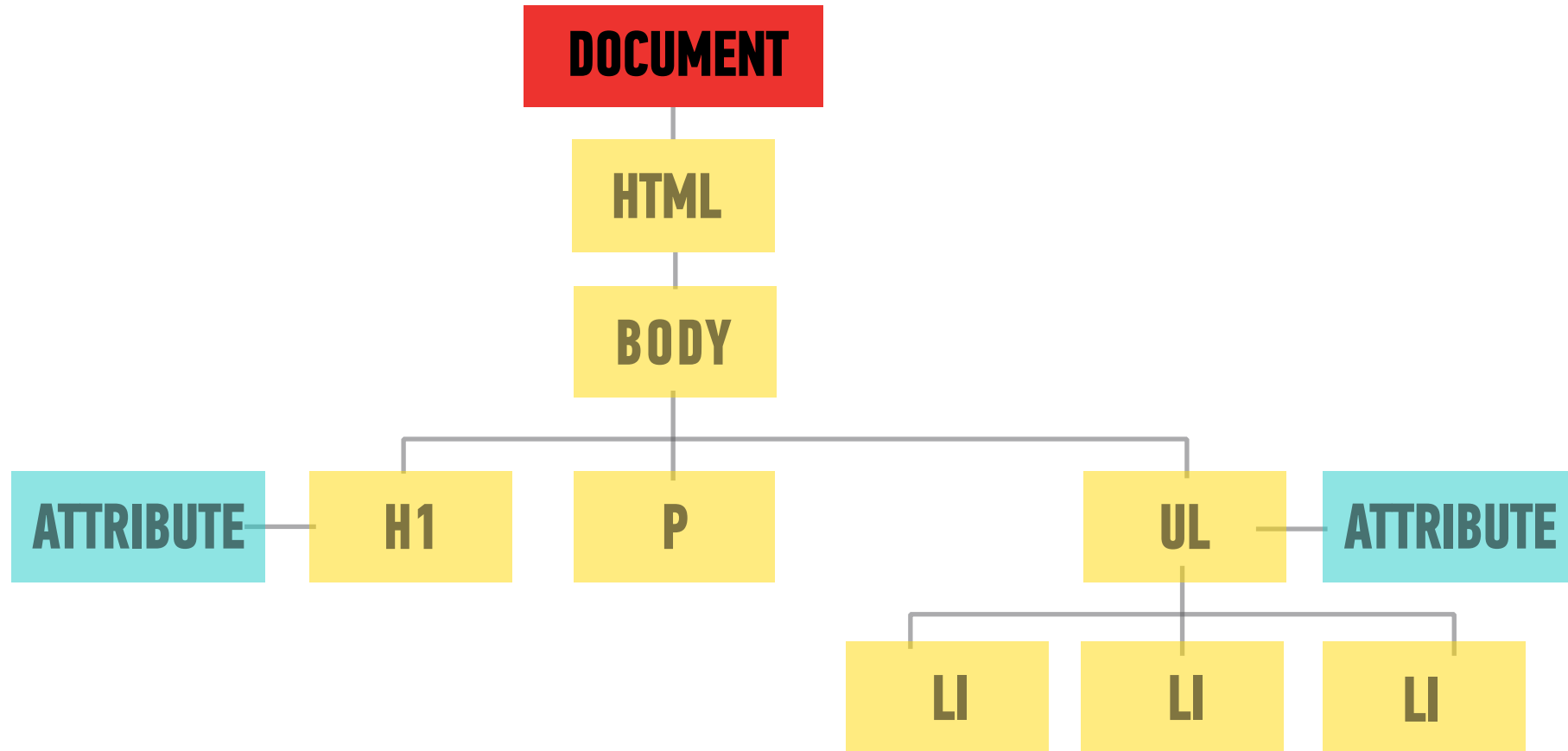
SYNTAX — USING OBJECTS AND METHODS

Object
document.

Method

Method

DOM TREE REFRESHER



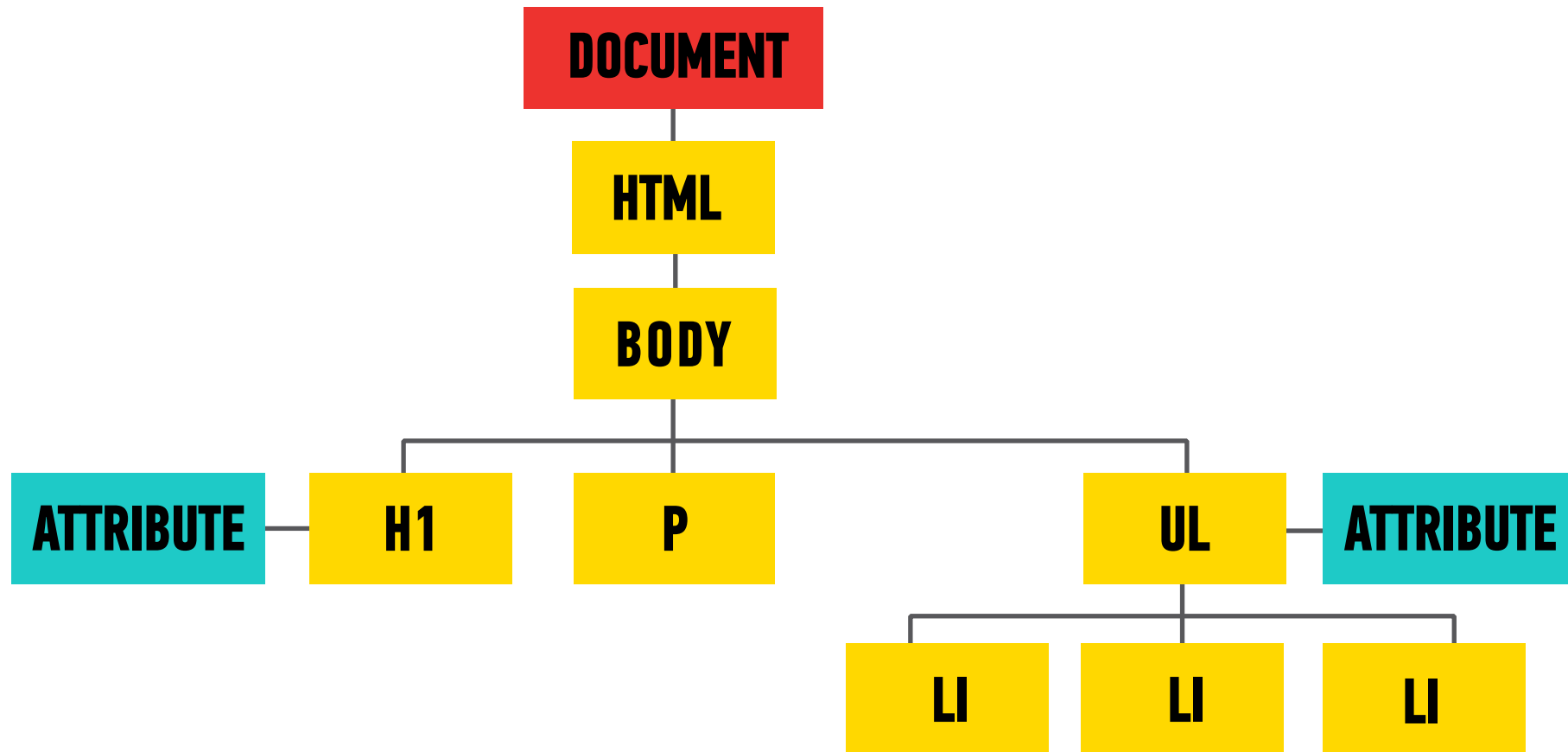
SYNTAX USING OBJECTS AND METHODS

The diagram illustrates the components of the JavaScript code `document.getElementById('about');`. The code is color-coded and annotated with labels and brackets:

- Object:** The word `document` is colored light blue. A bracket above it points to the label "Object".
- Method:** The text `.getElementById` is colored yellow. A bracket below it points to the label "Method".
- Parameter:** The string `"about"` is colored pink. A bracket above it points to the label "Parameter".

The code is followed by a semicolon `;`.

DOM TREE REFRESHER



STEP 1 — SELECT AN ELEMENT/ELEMENTS

SELECT AN INDIVIDUAL ELEMENT NODE:



```
document.getElementById('id');
```



```
document.getElementById('about');
```

Selects the element with the id "about"
(`<h1 id="about"> </h1>`)

SELECT MULTIPLE ELEMENTS (NODELIST):

```
document.getElementsByTagName('tag');
```

```
document.getElementsByTagName('p');
```

Selects all `<p>` elements

```
document.getElementsByClassName('class');
```

```
document.getElementsByClassName('myClass');
```

Selects all elements with the class 'myClass'
`<p class="myClass"> </p>`

WORKING WITH THE DOM TREE

TO ACCESS AND MODIFY THE DOM:

1

Select an element/elements

2

Work with those elements

STEP 2 — WORK WITH THOSE ELEMENTS

| PROPERTY: | DESCRIPTION: | |
|-----------|--------------|--------------------------------|
| | innerHTML | Gets/sets text & markup |
| | value | Gets the value of a text input |
| | className | Gets/sets the class name |

JS BASICS

ADDING JAVASCRIPT TO A PROJECT

ADDING A JAVASCRIPT FILE TO A PROJECT

1. Create a Javascript file. This process will be similar to creating an HTML or CSS file, but this time the file should have a .js extension (example: main.js)
2. Link to the Javascript file from your HTML page using the `<script>` element. We'll almost always want to add this script element **right before the closing body tag**.

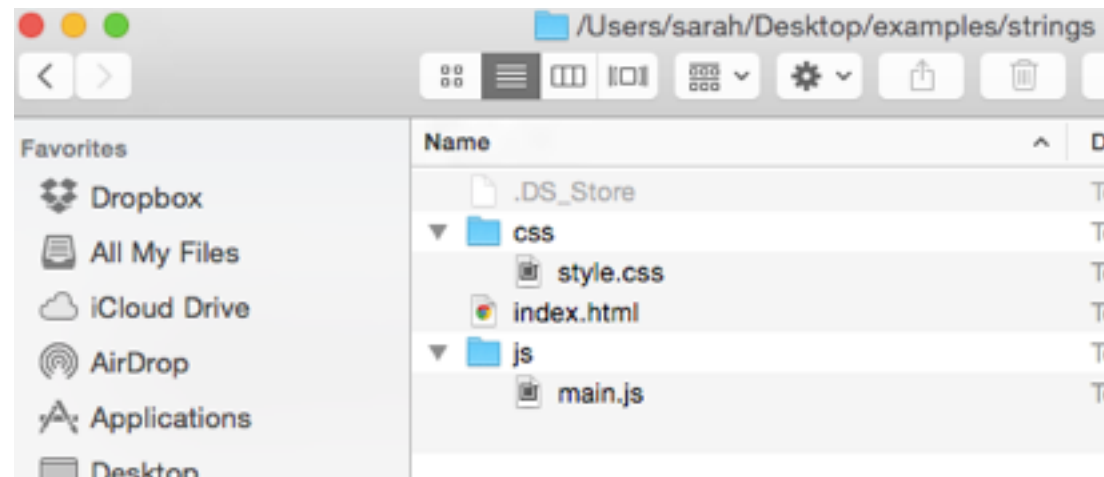
```
<body>
  <!-- Content here -->

  <script src="js/main.js"></script>
</body>
```

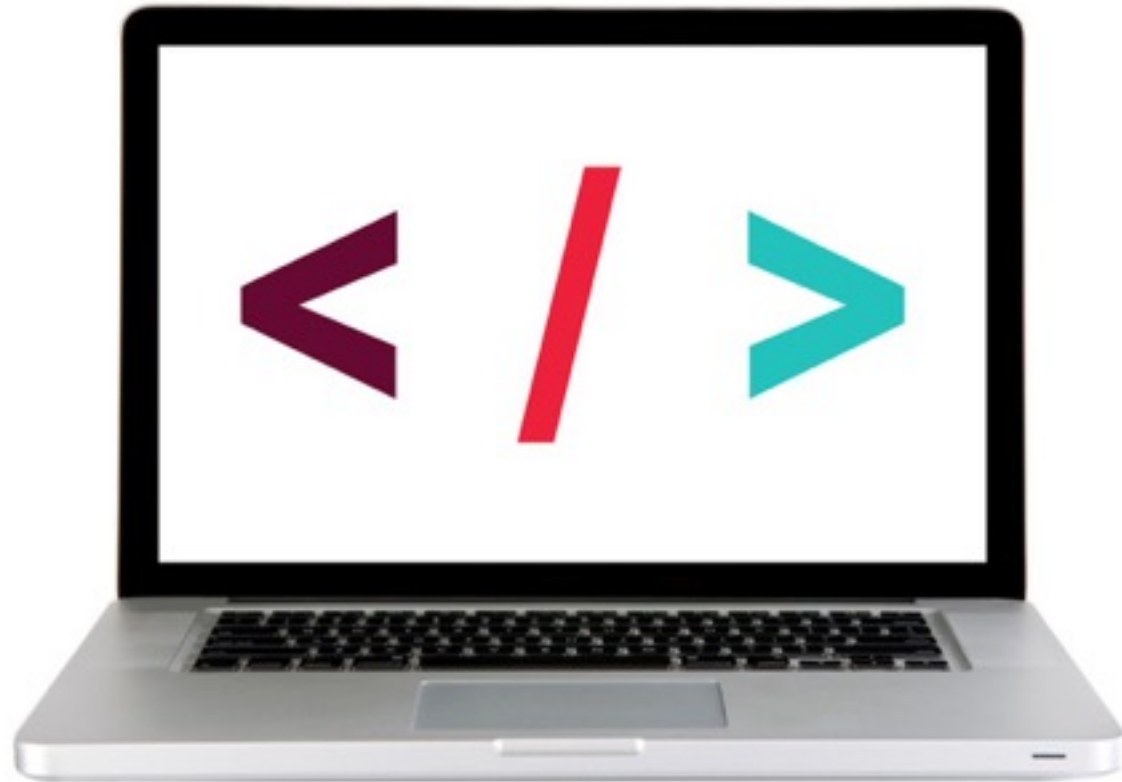
KEEP IT ON THE UP AND UP!

- ▶ Similar to when we save CSS files, it is considered **best practice** to keep Javascript files organized in one folder (which can have subfolders for larger sites).
- ▶ Usually people name this folder 'scripts', 'js', or 'javascript'. For this class we'll usually be calling this folder 'js'
- ▶ Remember, when naming files and folders, it's best to use an underscore or dash between words instead of a space. And try to avoid characters/symbols in file names (*really_cool_page.html* or *really-cool-page.html*).

sample structure:



LET'S TAKE A CLOSER LOOK



PRO TIPS

MAKING SURE YOUR JS FILES ARE LINKED PROPERLY:

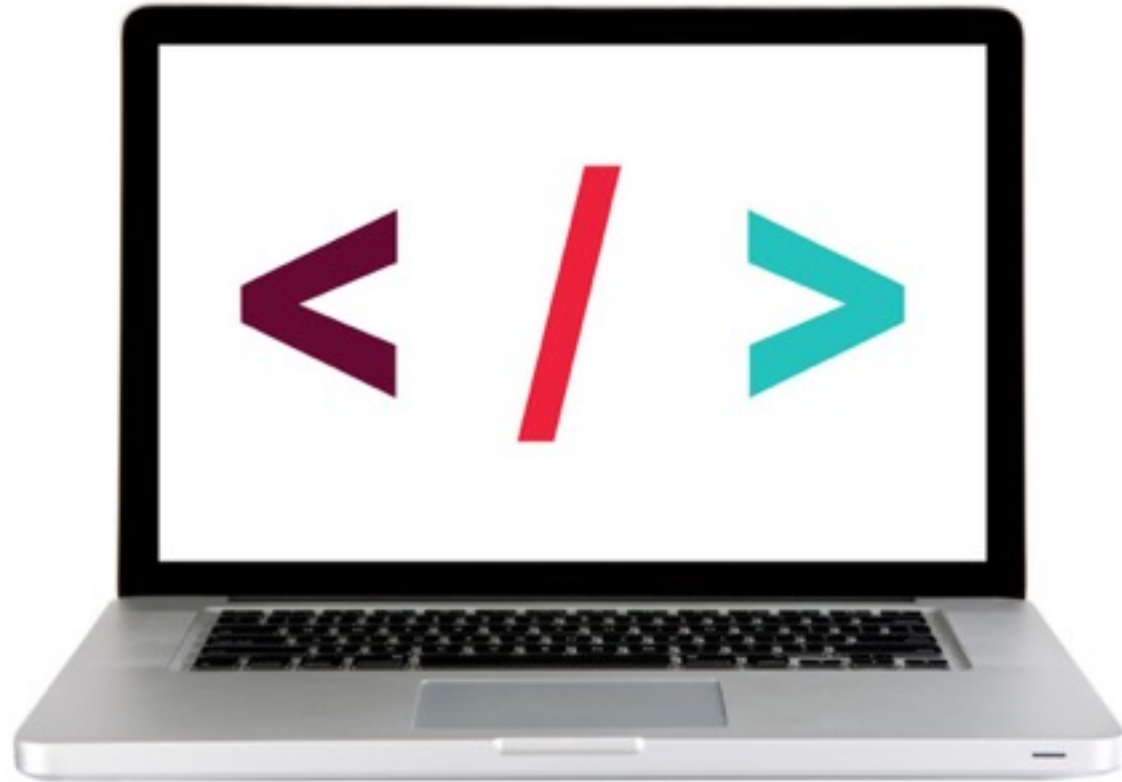
- To test and make sure that your js files are linked properly to your html, add an alert to your js and load the html page in the browser.
- If you don't see an alert pop up when you load the page, you know you have linked your files incorrectly.

```
alert();
```

- Remember, `console.log()` can be used to display variable values. This is useful for debugging.

```
var x = 5;  
var y = 3;  
var z = x + y;  
console.log(z); // this will print 8 to the console
```

LET'S TAKE A CLOSER LOOK

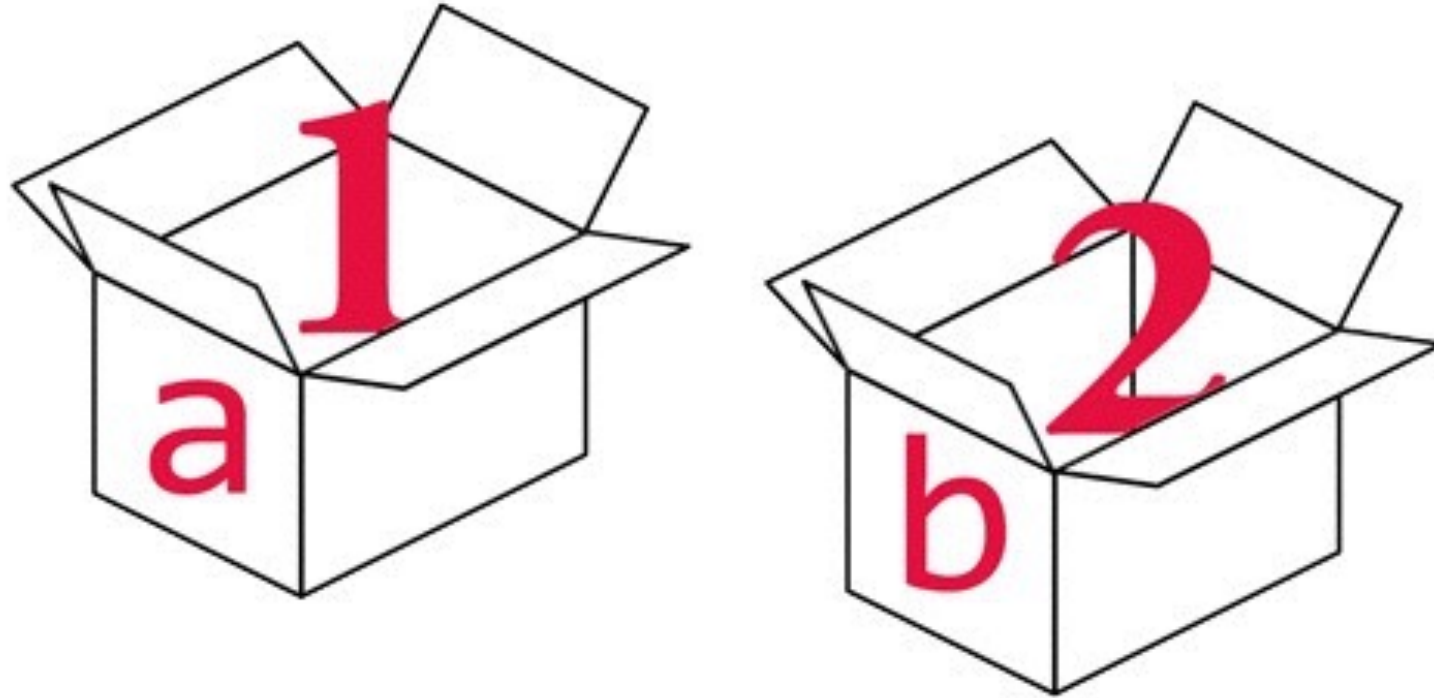


JS BASICS

VARIABLES

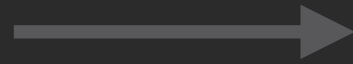
WHAT ARE VARIABLES?

- We can tell our program to remember (store) values for us to use later on.
- The entity we use to store the value is called a **variable**



JAVASCRIPT — VARIABLES

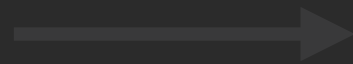
Declaring a variable



var **age**;

Keyword Name

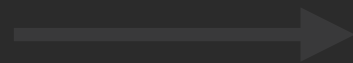
Assigning



age

Name Value

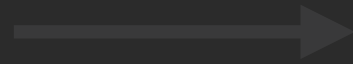
Both in one step



var

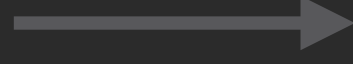
JAVASCRIPT — VARIABLES

Declaring



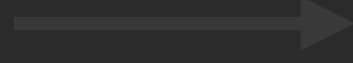
var
[] []
Keyword Name

Assigning a variable



age = **29**;
[] []
Name Value

Both in one step



var

JAVASCRIPT — VARIABLES

Declaring

→ **var**
Keyword Name

Assigning

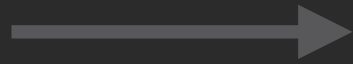
→ **age**
Name Value

Both in one step

→ **var** **age** = 29;

JAVASCRIPT — VARIABLES

Declaring a variable

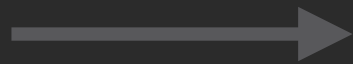


var age;



Semicolon!

Assigning a variable

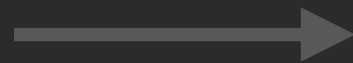


age = 29;



Semicolon!

Both in one step



var age = 29;



Semicolon!

JAVASCRIPT — VARIABLE RE-ASSIGNMENT

```
var name = "Matt";
```

```
name = "Ana";
```

VARIABLE CONVENTIONS

1. Variables start with a **lowercase** letter
2. If they contain multiple words, subsequent words start with an upper case letter.
3. Names can only contain: letters, numbers, \$ and _ (no dashes - or periods .)
4. Variables cannot start with a number
5. Case sensitive - numberofstudents is not the same as numberOfStudents
6. Names should be descriptive

```
var numberOfStudents = 10;
```



WHAT CAN BE STORED IN VARIABLES?

DATA TYPES:

STRINGS

"Today is Monday"

Letters and other characters enclosed in quotes

NUMBERS

10

22.75

- ▶ Positive numbers
- ▶ Negative numbers
- ▶ Decimals

BOOLEANS

true

false

Can have one of two values:

- ▶ True
- ▶ False

** Note: we'll meet some more data types later on down the road, too!*

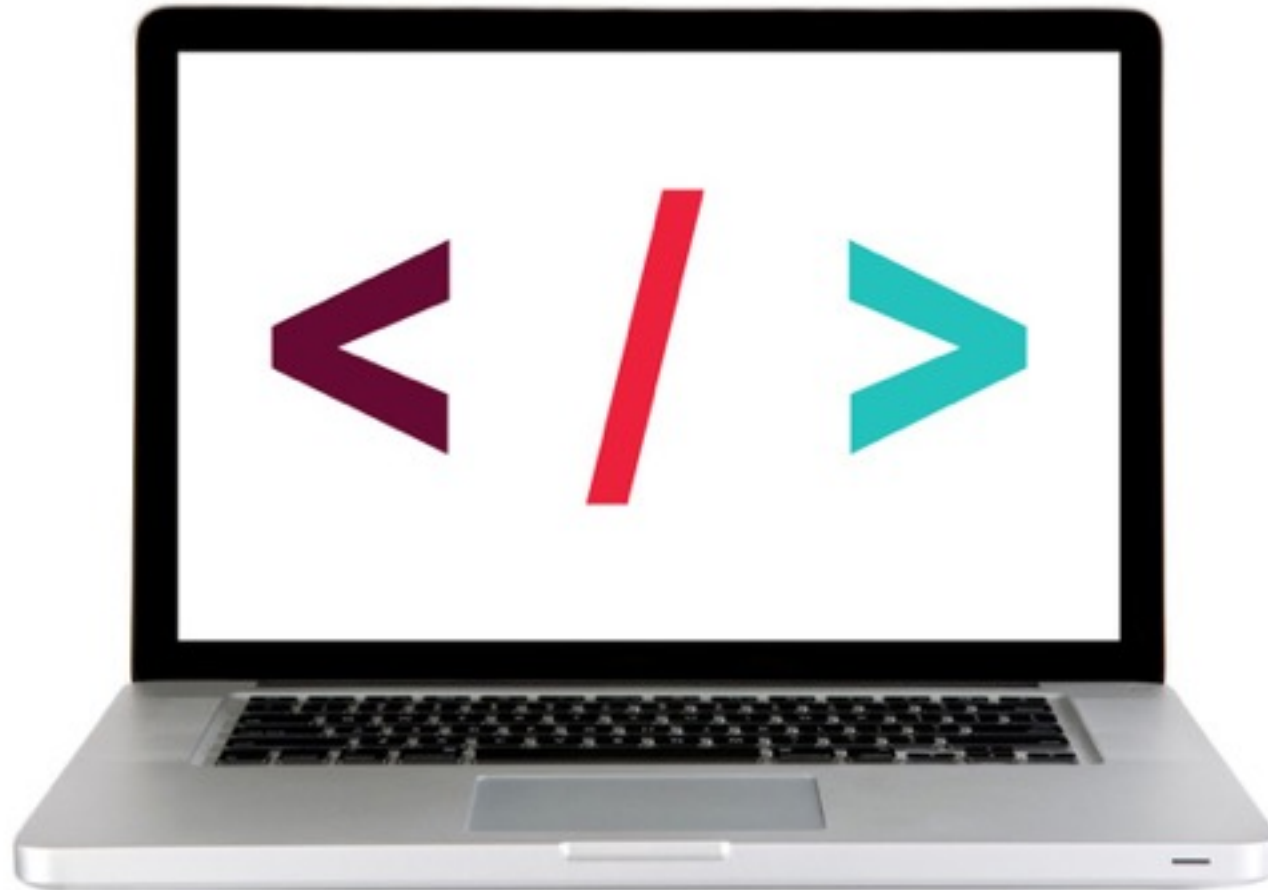
TO SUMMARIZE

1. A variable has both a “name” and a “value”
2. That value can change
3. A variable can be used multiple times throughout the code

DATA TYPES

NUMBERS

LET'S TAKE A CLOSER LOOK



View in [Codepen](#)

MORE ABOUT NUMBERS

INTEGERS:

Integers are whole numbers

10

FLOATS:

Number that uses a decimal to represent a fraction

22.75

**Can perform arithmetic on number data types*

ARITHMETIC OPERATORS

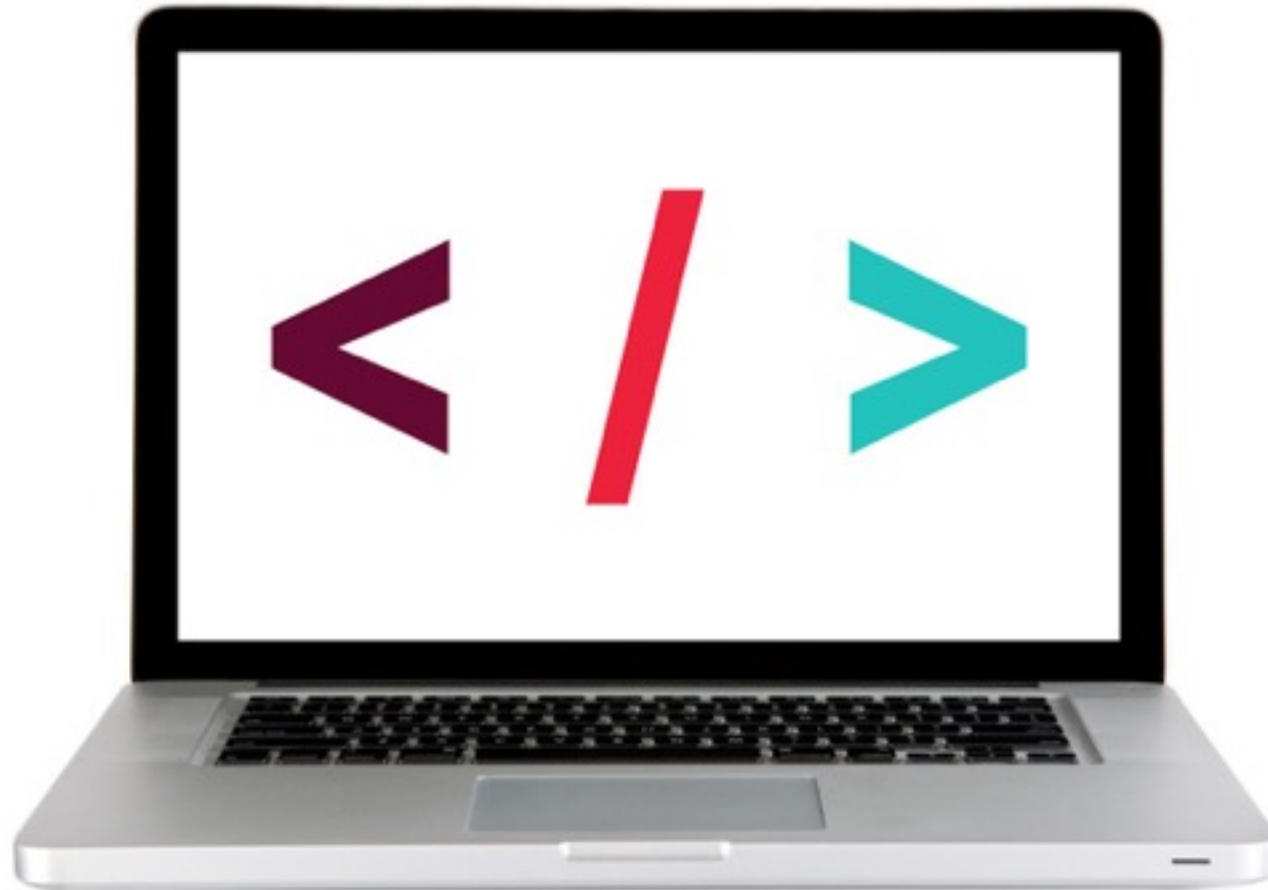
NAME:

| | OPERATOR: | | EXAMPLE: | | RESULT: |
|----------------|-----------|--|----------|--|---------|
| ADDITION | + | | 2 + 4 | | 6 |
| SUBTRACTION | - | | 8 - 1 | | 7 |
| MULTIPLICATION | * | | 2 * 3 | | 6 |
| DIVISION | / | | 4 / 2 | | 2 |
| MODULUS | % | | 12 % 5 | | 2 |

DATA TYPES

STRINGS

LET'S TAKE A CLOSER LOOK



View in [Codepen](#)

MORE ABOUT STRINGS

A STRING:

- Stores textual information
- Is surrounded by quotes


"How is the weather today?"

'Cold'

STRINGS

DOUBLE QUOTES VS. SINGLE QUOTES

"It's a beautiful day"



'They "purchased" it'



ESCAPING

'It\'s a beautiful day'

"They \"purchased\" it"

STRING METHODS/PROPERTIES

Continued...

| Method | Description |
|---------------------------------|--|
| charAt() | Returns the character at the specified index |
| charCodeAt() | Returns the Unicode of the character at the specified index |
| concat() | Joins two or more strings, and returns a copy of the joined strings |
| fromCharCode() | Converts Unicode values to characters |
| indexOf() | Returns the position of the first found occurrence of a specified value in a string |
| lastIndexOf() | Returns the position of the last found occurrence of a specified value in a string |
| localeCompare() | Compares two strings in the current locale |
| match() | Searches for a match between a regular expression and a string, and returns the matches |
| replace() | Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring |

| | |
|-------------------------------------|---|
| search() | Searches for a match between a regular expression and a string, and returns the position of the match |
| slice() | Extracts a part of a string and returns a new string |
| split() | Splits a string into an array of substrings |
| substr() | Extracts the characters from a string, beginning at a specified start position, and through the specified number of character |
| substring() | Extracts the characters from a string, between two specified indices |
| toLocaleLowerCase() | Converts a string to lowercase letters, according to the host's locale |
| toLocaleUpperCase() | Converts a string to uppercase letters, according to the host's locale |
| toLowerCase() | Converts a string to lowercase letters |
| toString() | Returns the value of a String object |
| toUpperCase() | Converts a string to uppercase letters |
| trim() | Removes whitespace from both ends of a string |
| valueOf() | Returns the primitive value of a String object |

- ▶ `string.length` is really handy, too, to find the length of a string.
- ▶ *Note this is not a function. It is a property (no () after length).*

```
var x = "My String";  
console.log(x.length);  
// prints 9 to the console
```

***From W3Schools: [Strings](#)*

STRING CONCATENATION

- ▶ Note when we use the + for numbers, we are doing math.
- ▶ When we use the plus for letters and words (called strings in programming), we are performing concatenation, which takes two strings and sticks them together.

```
var x = 5;  
var y = 7;  
var z = x + y; // 12
```

```
var book = "Happy";  
var summary = "Best book ever.";  
var review = book + ": " + summary; // Happy: Best book ever.
```

DATA TYPES

BOOLEANS

BOOLEANS

Can have one of two values:

- True
- False

true

false

- Booleans will become very important when we are making comparisons and using logic (if statements and loops).

View [Codepen](#)

CONVERTING DATA TYPES

CONVERSION: STRING TO NUMBER

```
var intString = "4";  
var intNumber = parseInt(intString, 10);  
var floatString = "3.14159";  
var floatNumber = parseFloat(floatString);
```

CONVERSION: STRING TO NUMBER

```
var number = 4;  
number.toString(); => "4";
```

or

```
number + ""; => "4";
```

CODE ALONG — SCORE KEEPER



Let's code! [Score Keeper](#) (Codepen)

CONDITIONALS

CONDITIONAL LOGIC

If something is true, do one thing. If it is not, do something else. This type of logic or statement is a condition.

In JavaScript (and coding in general) you'll need to make comparisons all the time:

- Is a user logged in?
- Has the user chosen three or more colors?
- Is the password correct?
- Does a user have enough money in their bank account?
- etc.

JAVASCRIPT — COMPARISON OPERATORS

== Equal to

Greater than **>**

=== Strict equal to

Less than **<**

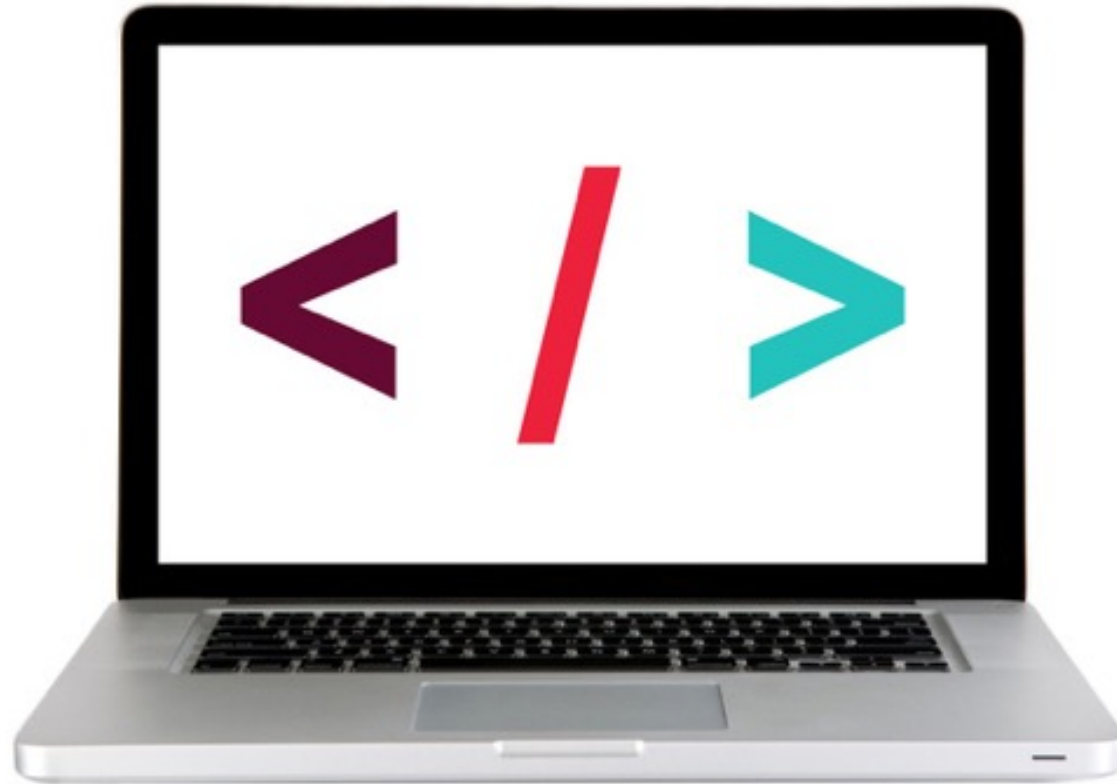
!= Not equal to

Greater than or equal to **>=**

!== Strict not equal to

Less than or equal to **<=**

LET'S TAKE A CLOSER LOOK — EQUALITY



[Comparison Operators on Codepen](#)

MAKING DECISIONS

```
if (age > 18){  
    document.write("You are an adult");  
}
```

JAVASCRIPT — IF STATEMENT

Condition

```
if (answer == 38) {  
    // Do something if true  
}
```

IF STATEMENTS

- Like booleans, either true or false
- If you are greater than 18, you are an adult

```
if (age > 18){  
    document.write("You are an adult");  
}
```

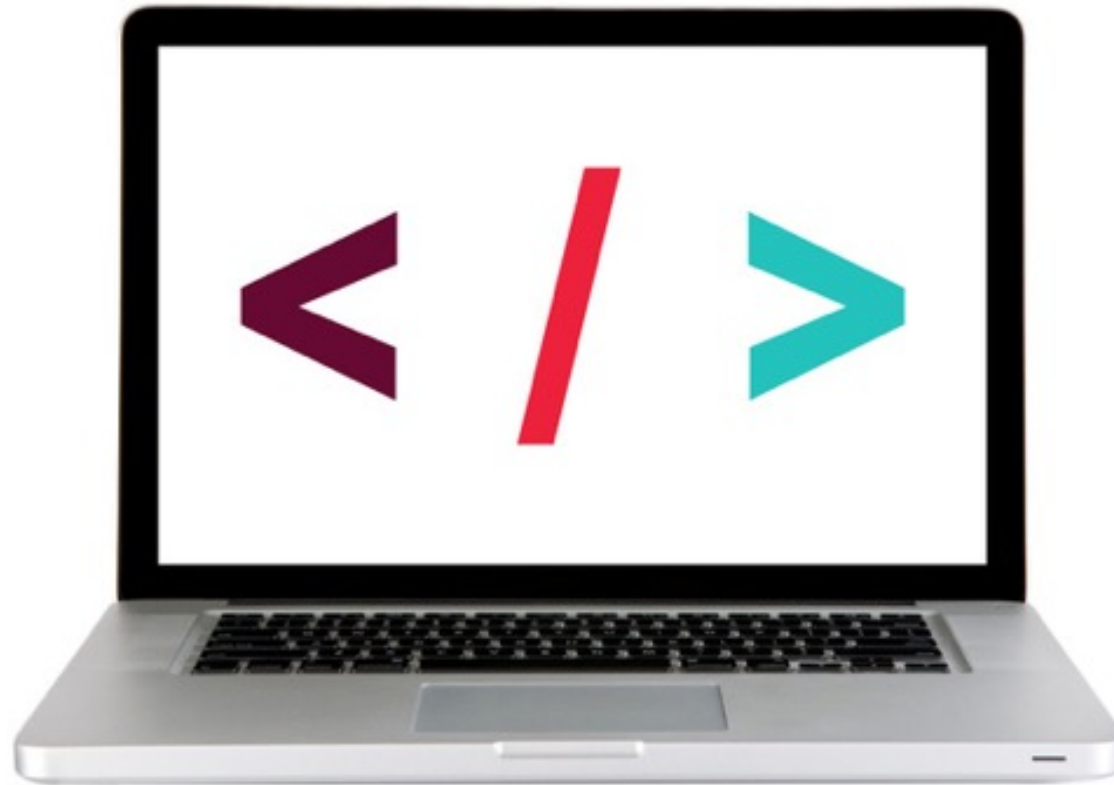
JAVASCRIPT — IF/ELSE STATEMENT

```
if (answer == 38) {  
    // Do something if true  
} else {  
    // Do something if false  
}
```

JAVASCRIPT — IF/ELSE IF/ELSE

```
if (answer == 38) {  
    // Do something if first condition is true  
} else if (answer == 30) {  
    // Do something second condition is true  
} else {  
    // Do something if all above conditions are false  
}
```

LET'S TAKE A CLOSER LOOK



View in [Codepen](#)

JAVASCRIPT — LOGICAL OPERATORS

&& and

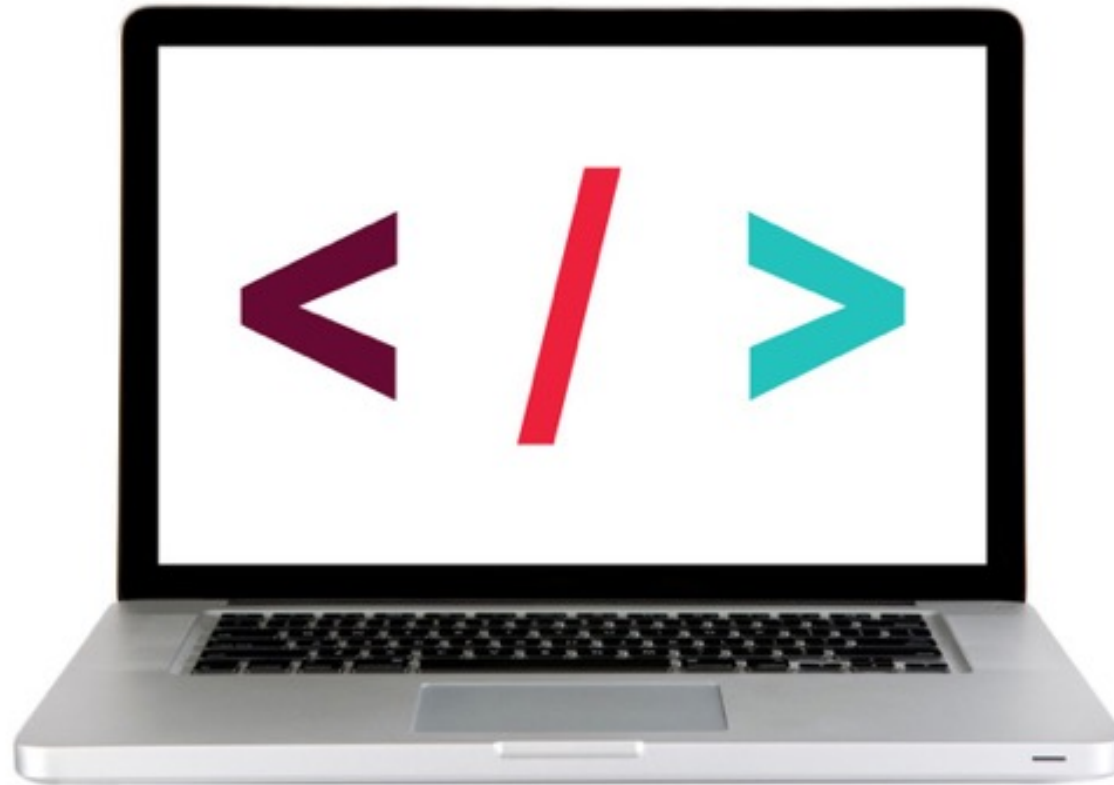
|| or

! not

MULTIPLE CONDITIONS

```
if (name == "GA" && password == "YellowPencil"){  
    //Allow access to internet  
}
```

LET'S TAKE A CLOSER LOOK



View in [Codepen](#)

CODE ALONG — COMPARE TWO NUMBERS



Let's code! [Compare Two Numbers](#) (Codepen)

JS BASICS

LAB

LAB — TEMP CONVERTER



LAB — TEMP CONVERTER



EXERCISE

KEY OBJECTIVE

- Build an application using HTML/CSS and JS that converts a temperature from Fahrenheit to Celsius

TYPE OF EXERCISE

- Groups, then pairs

SMALL GROUP PLANNING

8 min

1. In groups of 3-4 write pseudo code to convert a temperature from Fahrenheit to Celsius

EXECUTION

Until 8:50

1. Write some HTML markup
2. Write your JavaScript in the main.js file
3. Style it up!

LEARNING OBJECTIVES

- Define variables and identify best cases to use them.
- Differentiate between strings, integers and floats.
- Apply conditionals to change the program's control flow.

JS BASICS

HOMEWORK

HOMEWORK

This week's homework will be to finish the thermostat lab from today

REQUIRED READING:

If you purchased the textbook - Javascript & jQuery by Jon Duckett

- Read pages 88-99 (Functions), pages 186-189 (DOM), pages 295 - 309

Otherwise:

- Read [Eloquent JavaScript](#) Chapter 3
- Read [jQuery Basics](#)

OPTIONAL READING:

- Chapter 3, Chapter 5, Chapter 7 from the textbook

JS BASICS

EXIT TICKETS