# 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 8 folder**
4. There you'll find all the materials for today's class
5. Download starter_code_lesson_8.zip
6. Move it from your Downloads folder to your Desktop
7. Double-click on starter_code_lesson_8.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!!!

GENERAL ASSEMBLY

# FUNCTIONS

*Sarah Holden*

# REVIEW

## 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>
```

## 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”;
```

# 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!*

## 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 | `<=` |

## 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
}
```

# JAVASCRIPT — LOGICAL OPERATORS

&& and

|| or

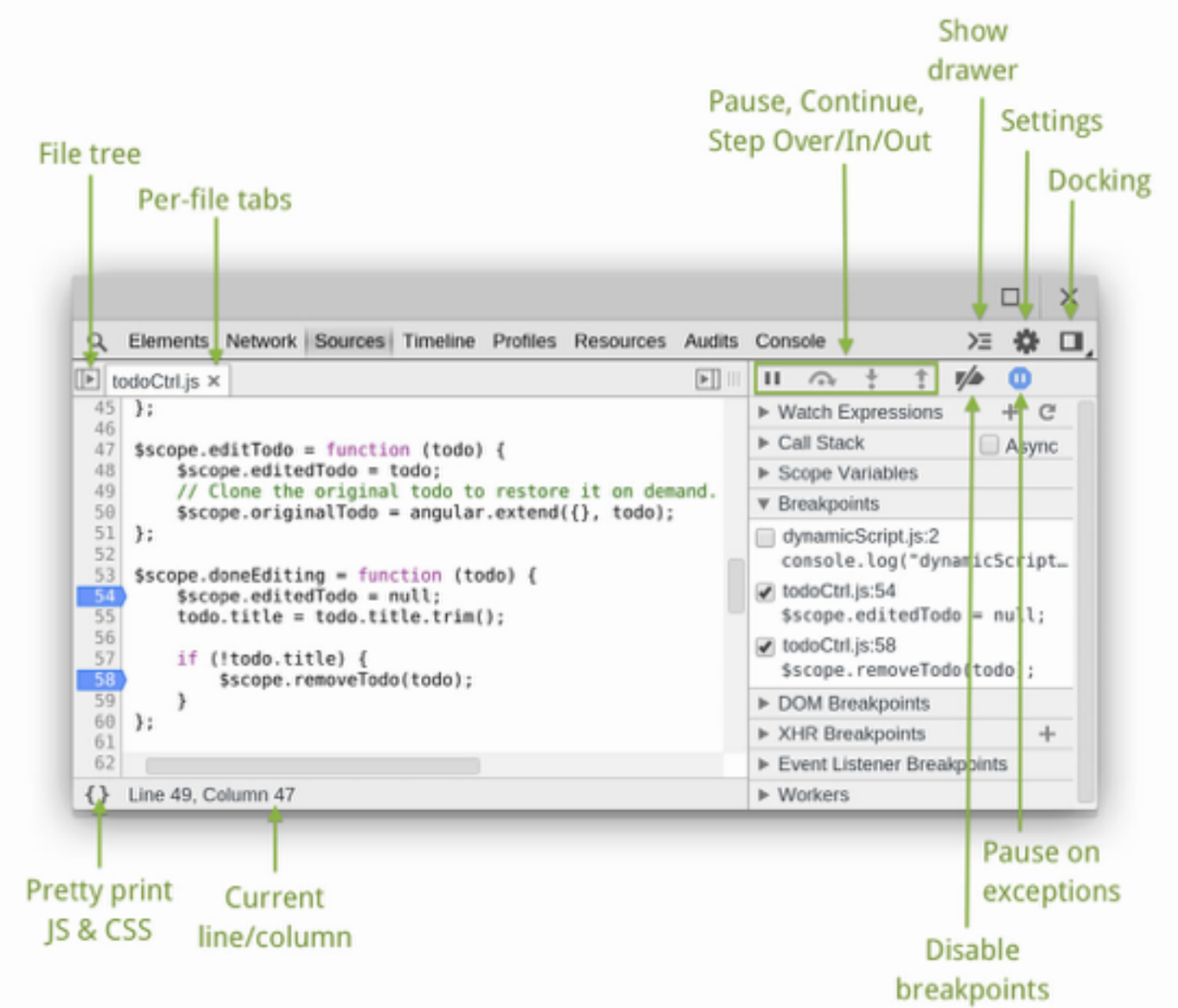
! not

# HOMEWORK

# HELP!

*Help! Something's not working and I don't know how to even begin to fix it. Where do I start?*

# CHROME DEV TOOLS – THEY'RE HERE TO HELP YOU!



# CODE ALONG — TEMPERATURE CONVERTER


Let's figure out what's wrong with our Temperature Converter

# LEARNING OBJECTIVES

- Describe arguments as they relate to functions.
- Predict values returned by a given function.
- Differentiate between named and anonymous functions

# AGENDA

- Functions
- Anonymous Functions
- Lab Time

# FUNCTIONS

# CODE ALONG — COMPARE TWO NUMBERS


Let's code! Cash Register (in your starter code folder)

## WHERE DO I START?

1. Are there any values we need to keep track of?
2. What are the major events that we need to listen for? A click on a button, hitting enter to submit a form?

## FUNCTIONS

- Functions allow us to group a series of statements together to perform a specific task
- We can use the same function multiple times in our script
- Functions are not always executed when a page loads, so they provide us with a way to 'store' the steps needed to achieve a task.

## SYNTAX — DECLARING A FUNCTION

Keyword

Name

```
function myFunction() {
    // Do something
}
```

Code block

## SYNTAX — CALLING A FUNCTION

- To run the code in a function, we 'call' the function by using the function name followed by parenthesis.

```
myFunction();
```

Function name

## SYNTAX — DECLARING A FUNCTION (WITH PARAMETERS)

Parameters

```
function myFunction(param1, param2) {
    return param1 * param2;
}
```

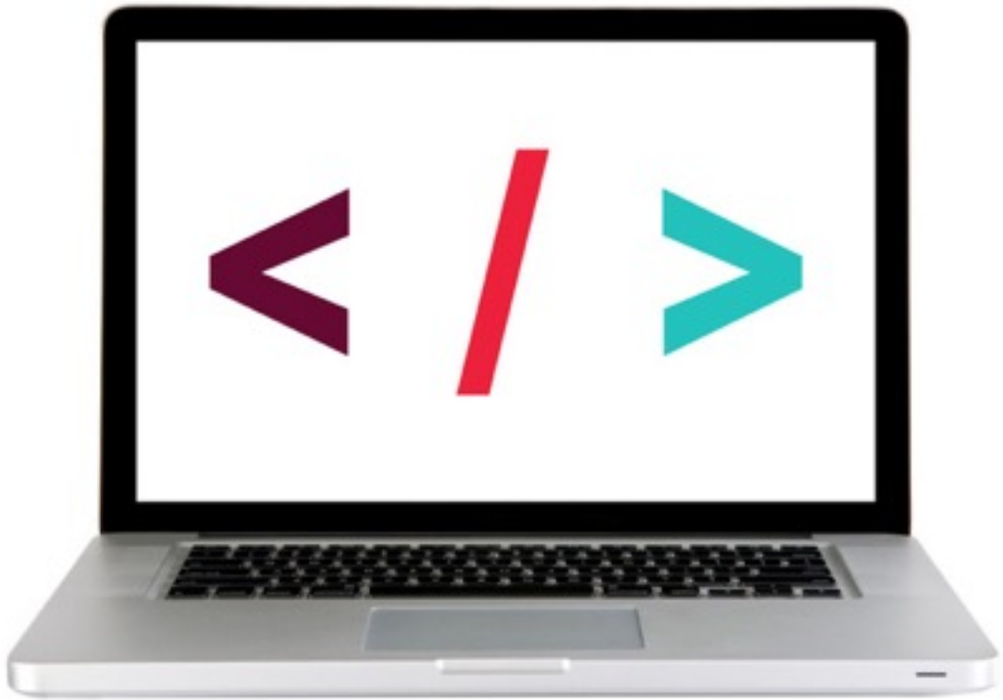
We can use these parameters like variables from within our function

## SYNTAX — CALLING A FUNCTION (WITH ARGUMENTS)

Arguments

myFunction(350, 140)

# LET'S TAKE A CLOSER LOOK - PARAMETERS/ARGUMENTS



See example in Codepen

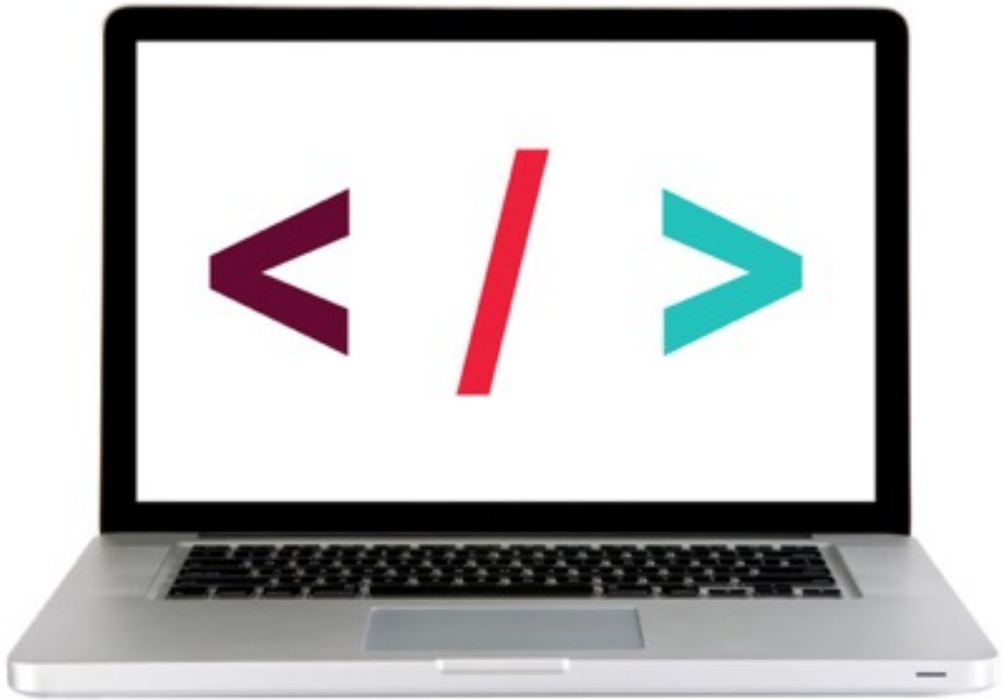
## RETURNING VALUES FROM A FUNCTION

- To return a value from a function, we use the **return** keyword
- From within a function, the **return** keyword 'hands' a value back to the code that called the function
- We can then do something with that value, or store it in a variable for use later in the script

```
function greeting(name) {
  var sayHello = "Hello " + name;
  return sayHello;
}

var sayHi = greeting("Sarah");
```

# LET'S TAKE A CLOSER LOOK — RETURN VALUES



See example in Codepen

# ANONYMOUS FUNCTIONS

# ANONYMOUS FUNCTIONS

```
var myVariable = function() {
    // Do something
}
```

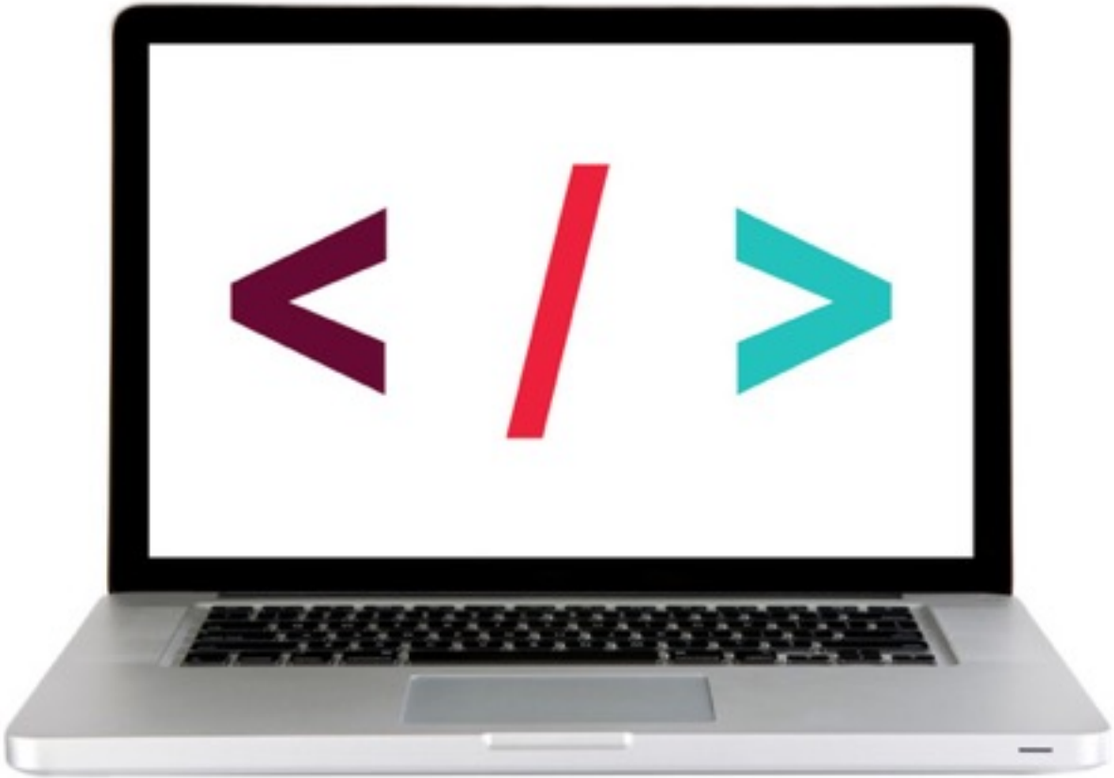
## WHAT IS AN ANONYMOUS FUNCTION?

- An anonymous function is a function without a name
- It can be stored in a variable and called the same way you would call a named function

## WHAT'S THE MAIN DIFFERENCE?

- You cannot call this function before the interpreter has discovered it.

# LET'S TAKE A CLOSER LOOK



View example in Codepen

# SCOPE

# VARIABLE SCOPE

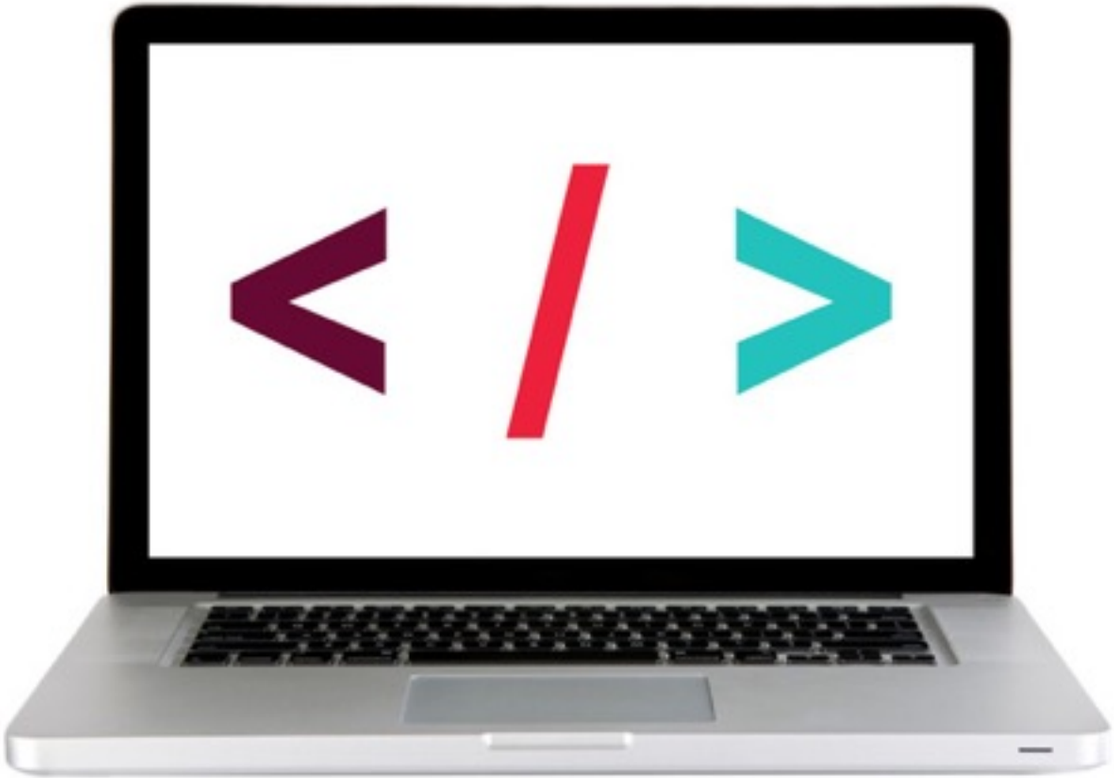
## LOCAL VARIABLES

- A **local** variable is a variable that is declared *inside* a function.
- It can only be used in that function, and cannot be accessed outside of that function

## GLOBAL VARIABLES

- A **global** variable is a variable that is declared *outside* of a function.
- It can be used anywhere in the script.

# LET'S TAKE A CLOSER LOOK



View example in Codepen

# LAB TIME!

# LAB

# ACTIVITY — ROCK PAPER SCISSORS

EXERCISE

## KEY OBJECTIVE

- Practice JS skills by coding a rock paper scissors game

## TYPE OF EXERCISE

- Pair programming

## TIMING

| | |
|---|---|
| *8 min* | Try out the site with a group of 3-4 and write out pseudo code as comments in your main.js |

## TIMING

| | |
|---|---|
| *Until 8:45* | 1. Link to your JS file from your index.html<br>2. Write your Javascript |

***To demo functionality: [Click here]*

# LEARNING OBJECTIVES

- Describe arguments as they relate to functions.
- Predict values returned by a given function.
- Differentiate between named and anonymous functions

# HOMEWORK

# HOMEWORK

## ASSIGNMENT:

Finish Wednesday's lab — Due February 14th at 11:30pm

## FINAL PROJECT:

Final Project Part 1 — Due February 14th at 11:30pm

## OPTIONAL BUT HIGHLY ENCOURAGED READING:

From the textbook - Javascript & jQuery by Jon Duckett

- Read pages 310 - 365 (jQuery - continued from last week's reading)

# EXIT TICKETS