Week 1 Workshop

JavaScript Fundamentals

Agenda

Activity	Estimated Duration
Welcome & Housekeeping	5 mins
Instructor & Student Introductions	20 mins
Your Bootcamp Overview	20 mins
Week 1 Review	1 hour 15 mins
Break	10 mins
Workshop Introduction & Demo	5 mins
Workshop Assignment	1 hour 30 mins
Code Review and Wrap-up	15 mins

Set up

- Cameras on and mic muted ☺
- Log into Discord and have the class channel open
- Log into Learn and have Week 1 open

Navigating the BBB Platform

- Speak Up If You Have A Question
- Public Chat and Private Chat are available
- We will be **Sharing Screens** often

Your Instructor Melanie McCall

<u>**Based in Atlanta, GA (Eastern Time Zone)**</u>

We're all rooting for you!

<u>My Story</u>

College - Computer Science

2012-2017

5 Years of Experience in The Industry

2016-2022

Work from home mom

2021 - *present*

Teaching with Nucamp

2019 - *present*

<u>Fun Facts:</u>

I failed my first coding class in college - MATLAB

I am a part of a salsa dance company

Student Introductions

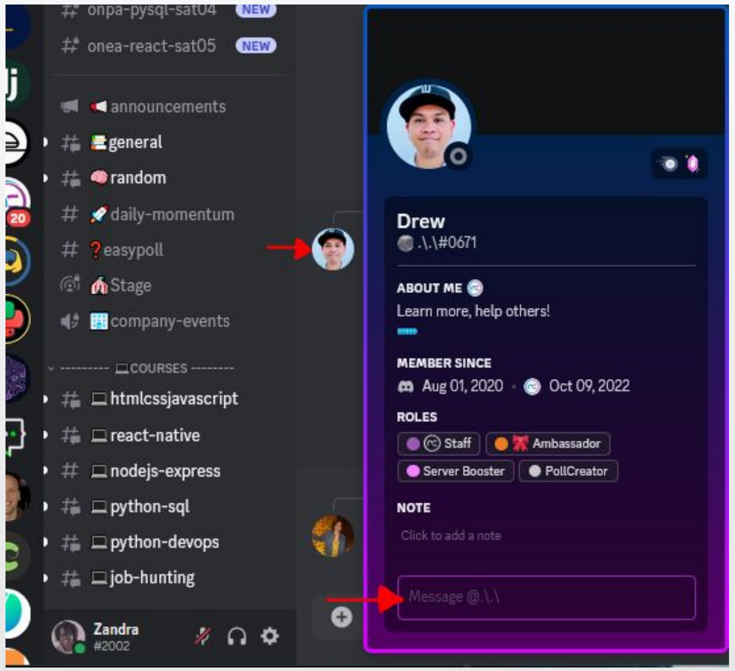
- Please **introduce yourself!**
- Share:
  - Your name (or preferred nickname)
  - Where are you from? Where do you live now?
  - What is your coding experience prior to this bootcamp?
  - What motivates you to take this bootcamp?
  - Name a musical artist you’ve been listening to lately

Using Discord

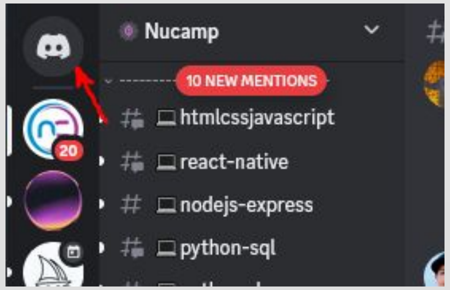
- **Make sure you are in the following Discord channels:**
  - #javascript
  - Our class channel (e.g. #onea-javascript-sat02)

- **Complete the following tasks right now in our class channel:**
  - Post a message in the class channel
  - Follow the #javascript and #debugginator channels
  - Find a friend request and Direct Message from me

This Tool is a vital part of this Bootcamp,
don't isolate yourself!



Send a DM: Click your instructor's image



Your DMs are under the top-left Discord icon

Instructors @ Nucamp

- Most of us are full-time Engineers who choose to teach part-time as well, please be respectful of your instructor’s time
- Utilize your the Discord channels for times when your instructor may be busy at work or sleeping

Asking For Help in Discord Channels

- Post to the class channel or the main **#javascript** channel (not a DM)

- Instead of saying "Can anyone help me with a problem?"
  1. State which exercise you are working on,
  2. **ask a direct question about your problem/issue,**
  3. include what you have already tried,
  4. attach your relevant code file(s), and
  5. if you are seeing error messages, attach a screenshot of the errors

- **There is also the #debuginnator channel for AI assistance**
- *Be respectful of your Instructor's time. They are dedicating a lot of time to help students and they all have full-time jobs. We are not available 24/7*

Your Bootcamp Learning Routine

- **Our lecture is a review** and no longer than 1.5 hours. Please make sure you are completing the exercises and practicing the concepts you’ve learned in your spare time. This is key to being successful in the workshop.
- Your time and effort will determine your success

What to Expect From Me

Workshops every Saturday @ 9AM (Eastern Time)

Weekly Beginner-Friendly Info Sessions

We Won’t Be Doing the Exercises Together During Workshop. They should be done before Saturday!

Slides & Resources Will be Posted in JavaScript Forums section and Discord class channel

Grades and Feedback are Posted by Tuesday (You need at least a 6 to pass)

~1Hr Per Day of Student Assistance (Ask Qs Early in the Week)!

Student Check-ins

Read Weekly Feedback

Ask me ANYTHING through Discord or Nucamp Portal!

The 20-minute rule

If you ask for help too soon:

- You will not learn how to tackle problems or remove obstacles, which is core to coding.
- Remember the process or path that resolved the issue is more important than the solution.

If you ask for help too late:

- You will get frustrated and tired.
- You will miss an opportunity to go deeper on the same topic (time is limited).

10-minute rule during workshops:

- During the week, go by the 20-minute rule.
- During workshops, go by a 10-minute rule – try to solve the issue yourself (or with your classmate, if working together) for 10 minutes before asking your instructor for help.

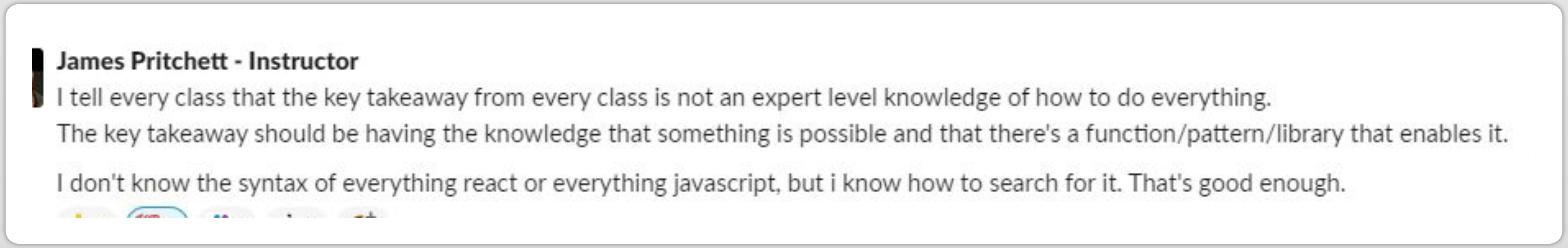
20 minutes, then ask for help!

Next 5 Weeks Overview

- **Week 1:** JavaScript Basics
- **Week 2:** Loops and Arrays
- **Week 3:** Objects and Classes
- **Week 4:** DOM Manipulation and Events
- **Week 5:** Async & APIs

Your Learning Goals

- How to **WRITE** good, clean code
- How to **READ** code and documentation
- How to **TALK ABOUT** your code with others
- How to **DEBUG** your code
- How to **SEARCH** for answers



James Pritchett - Instructor
I tell every class that the key takeaway from every class is not an expert level knowledge of how to do everything.
The key takeaway should be having the knowledge that something is possible and that there's a function/pattern/library that enables it.

I don't know the syntax of everything react or everything javascript, but i know how to search for it. That's good enough.

How to Achieve Your Learning Goals

- **How to WRITE good, clean code** - Use tools to keep your code organized and well-formatted, Always look for ways to improve your code
- **How to READ code and documentation** - Try to translate code into easy-to-understand terms, Take a few extra minutes to read documentation on new things
- **How to TALK ABOUT your code with others** - Work with your team, do not stay on mute for the entire workshop. Work together to solve problems
- **How to DEBUG your code** - Take notes on how a problem was solved so you can learn how to solve common bugs, Use **console.log()** for insight on what's happening inside your functions
- **How to SEARCH for answers** - Start using google and AI tools to ask questions

#1 Tip for Learning (Beginners) - What Do You Think It is? :)

BE PATIENT

- Take Breaks (Don’t Hurt Your Brain)
- ***You are Learning a New Language. It’s Going to Take Some Time to Sink In***

What is Git?

Git helps us save progress of our code, synchronize our code with others, and access earlier versions of our code.

Git is a widely popular software being used by millions of developers every day!

YOU NEED TO KNOW GIT just like you need to know how to code. They go hand in hand

Version control is essential when developing software long term and working in teams

Git Tutorial THIS Wednesday!

~9:30 PM EST, look for email calendar invite and announcements on Discord

Questions about Bootcamp?

Week 1 Review

JavaScript Basics

Week 1 Recap

- What is JavaScript?
- HTML: The script element
- The JavaScript console
- Variables
- Data types and Assignment
- JS Grammar and Syntax
- HTML: The onclick attribute
- Functions
- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Conditional Statements
- Code Review: Magical Mystery Theatre Challenge

What is JavaScript?

How would you describe JavaScript to a friend/stranger? Assume they do not know anything about code

What is ES6?

The JavaScript console: Your BFF

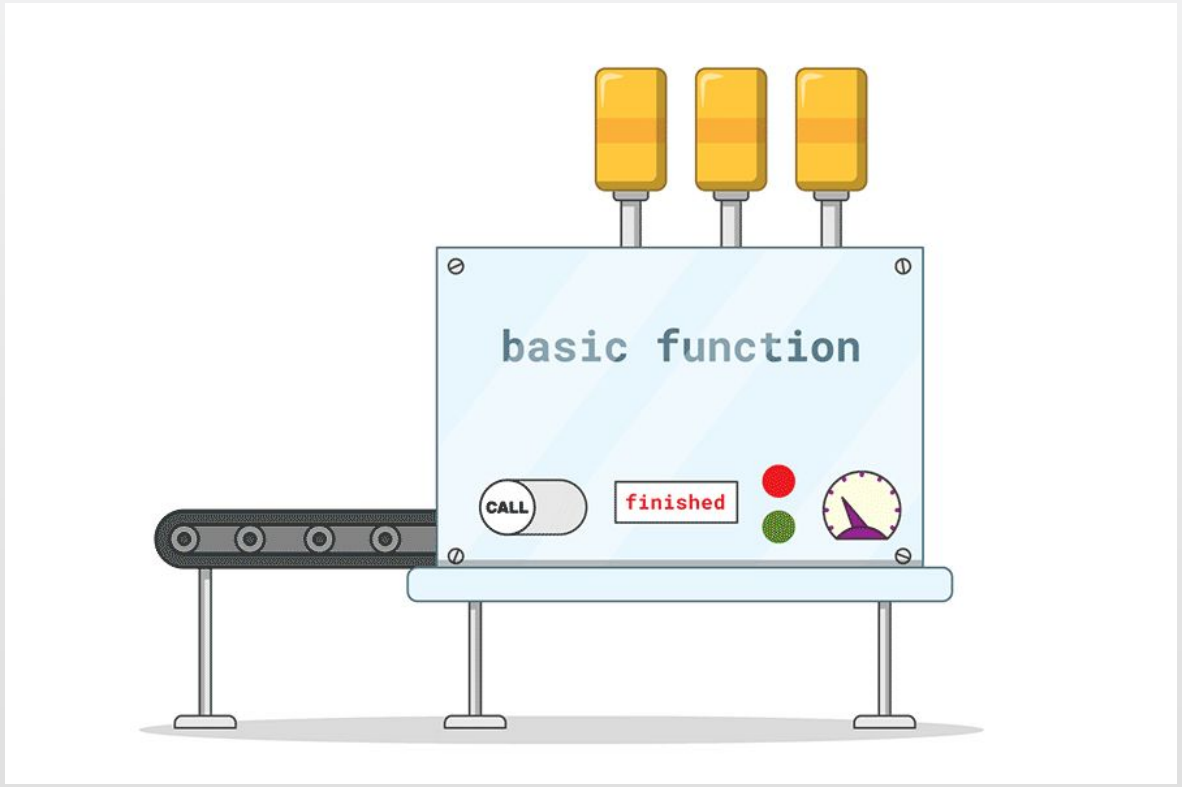
Three primary uses for the console:

1. View error/warning messages
2. Log your own messages using **console.log('…');**
3. Test out small pieces of JavaScript and have their values immediately evaluated and echoed back to you

Where can you find the developer‘s Console?

If your code isn’t working, check your console for errors!

Functions



- A function is a block of code that does something.

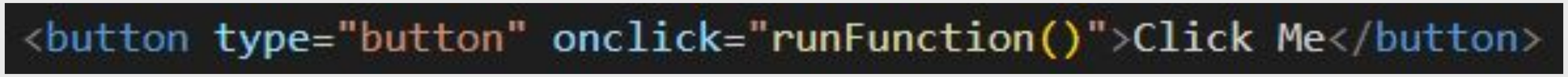
```
function sayHello() {
  alert("Hello World!");
}
```

- Functions can take inputs (called parameters) and returns an output
- You call a function with its name followed by ()

```
<button onClick="sayHello()">Click me!</button>
```

HTML: The onclick attribute

- Add **onclick** attribute to HTML element such as button to run JS function when element is clicked:



```
<button type="button" onclick="runFunction()">Click Me</button>
```

- This attribute connects your static HTML elements to JavaScript functionality
- There are multiple ways to trigger JavaScript from an HTML page; this is just one way

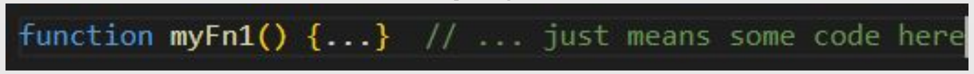
Functions – Parameters and Arguments

- Function definitions must include a **parameter list**
  - Variable names for values that will be passed in when function is called
- Function calls pass an **argument list** for the parameter values

Pass arguments **TO** functions when calling it

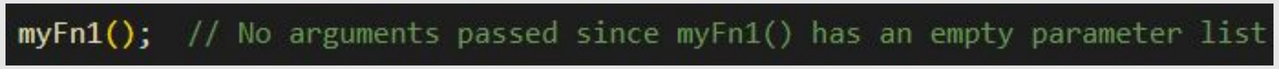
The arguments passed then become parameters (variables) to be used inside a function

- **Parameter list** can be empty:



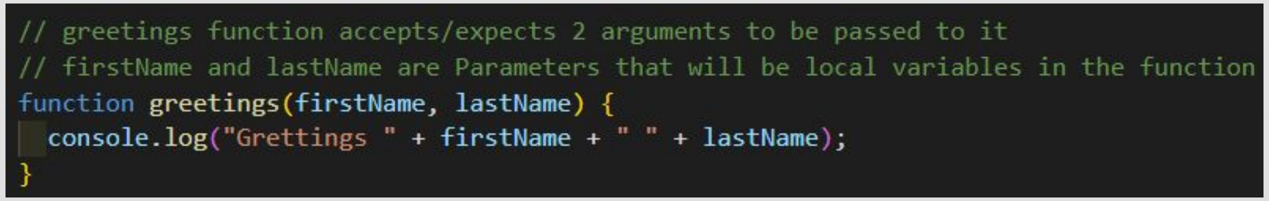
```
function myFn1() {...}  // ... just means some code here
```

- If so, function is called with empty **argument list**:



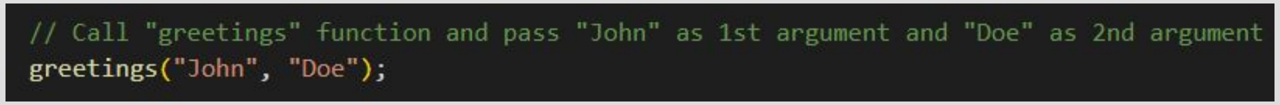
```
myFn1();  // No arguments passed since myFn1() has an empty parameter list
```

- Otherwise, call with arguments that correspond to parameters:



```
// greetings function accepts/expects 2 arguments to be passed to it
// firstName and lastName are Parameters that will be local variables in the function
function greetings(firstName, lastName) {
  console.log("Grettings " + firstName + " " + lastName);
}
```

No need to declare these variables using let or const! The parameter list does it for you



```
// Call "greetings" function and pass "John" as 1st argument and "Doe" as 2nd argument
greetings("John", "Doe");
```

Comparison Operators

- Equality Operators
  - **Strict equality** (aka triple equals/identity): `===`
  - Loose equality (aka double equals/equality): `==`
  - **Strict inequality** (aka non-identity): `!==`
  - Loose inequality (aka inequality): `!=`

Discuss: What's the difference between the strict and loose versions of the equality operators, and which are best practice to use?

== does NOT evaluate the data type (i.e 1 == "1" will return true)

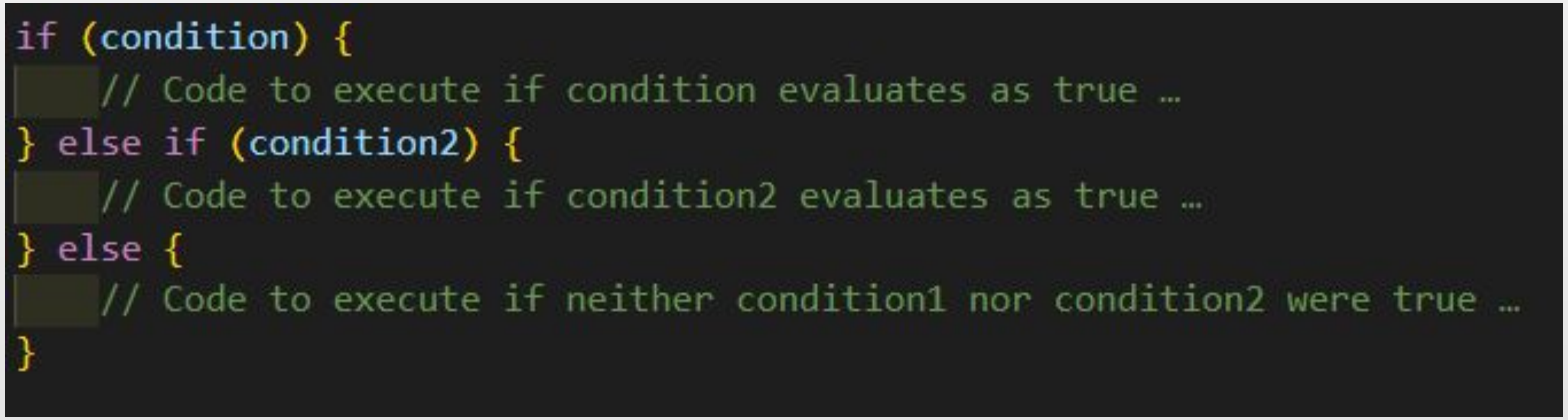
=== is a strict equality where the data types must match (i.e 1 === "1" will return false)

=== Strict is best practice

- Relational Operators
  - `>` `>=` `<` `<=`
  - Greater than, greater than or equal to, less than, less than or equal to
  - Works as you would expect with numbers
  - Works in lexicographical order with strings; 'a' is lower/less than 'z'

If … Else If … Else

- Conditional statement, allows forks in your code



```
if (condition) {
    // Code to execute if condition evaluates as true …
} else if (condition2) {
    // Code to execute if condition2 evaluates as true …
} else {
    // Code to execute if neither condition1 nor condition2 were true …
}
```

- **else if** and **else** are optional, you do not need them

  You can have either or both, following an **if** block

  You can have multiple **else if** blocks

  You can have only one **else** block at the very end

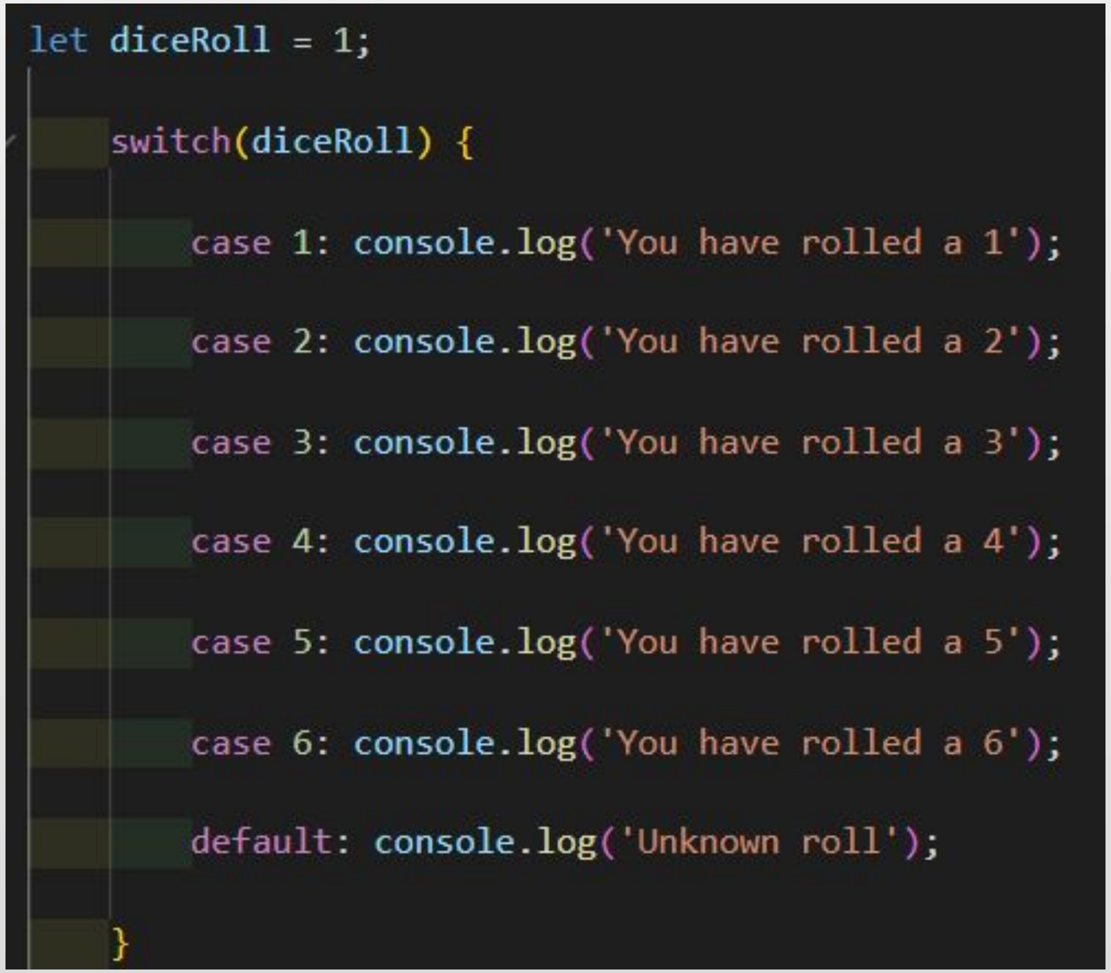
```
*Syntax*

    if (condition) {
           //block of code
   }
```

```
*Syntax*

    if (condition) {
           //block of code if condition is true
   } else {
           //block of code if none of the condition(s) is true.
   }
```

```
*Syntax*

    if (condition) {
           //block of code if condition is true
   } else if (condition_2) {
           //block of code if condition is false and condition_2 is true.
   } else if (condition_3) {
           //block of code if condition is false and condition_2 is false and condition_3
is true.
   } else {
           //block of code if none of the condition(s) is true.
   }
```

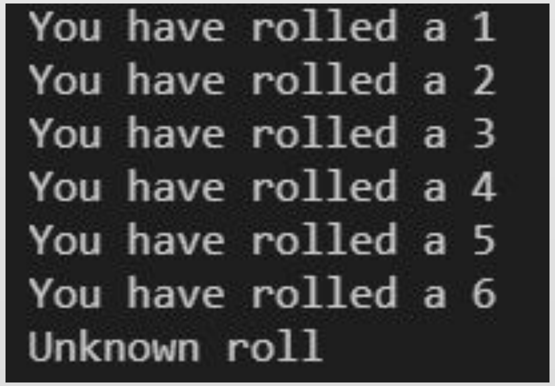
Switch

- What is wrong with this code? What would happen if you ran



```
let diceRoll = 1;

switch(diceRoll) {

    case 1: console.log('You have rolled a 1');

    case 2: console.log('You have rolled a 2');

    case 3: console.log('You have rolled a 3');

    case 4: console.log('You have rolled a 4');

    case 5: console.log('You have rolled a 5');

    case 6: console.log('You have rolled a 6');

    default: console.log('Unknown roll');

}
```

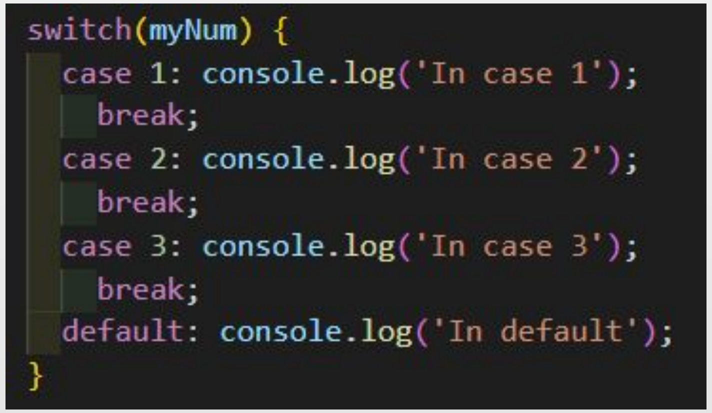
There is no **break;** for each case, which will result in running the matched case + all cases below it



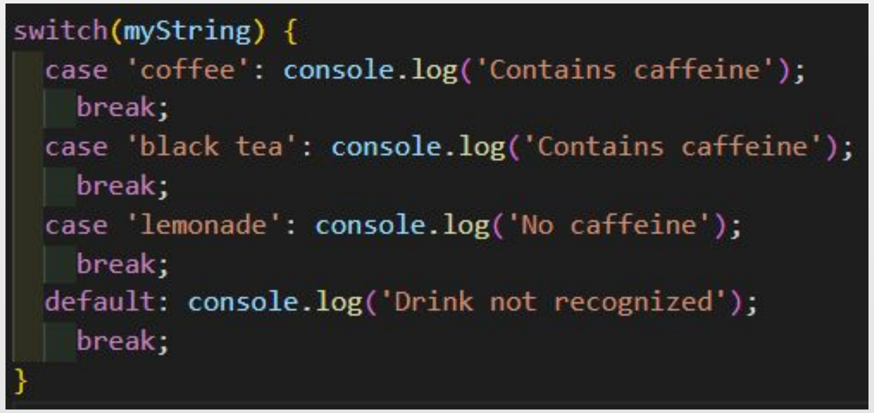
```
You have rolled a 1
You have rolled a 2
You have rolled a 3
You have rolled a 4
You have rolled a 5
You have rolled a 6
Unknown roll
```

Switch

- Conditional statement – evaluates an expression depending on its value, then executes one of multiple **case** clauses and an optional **default** clause:



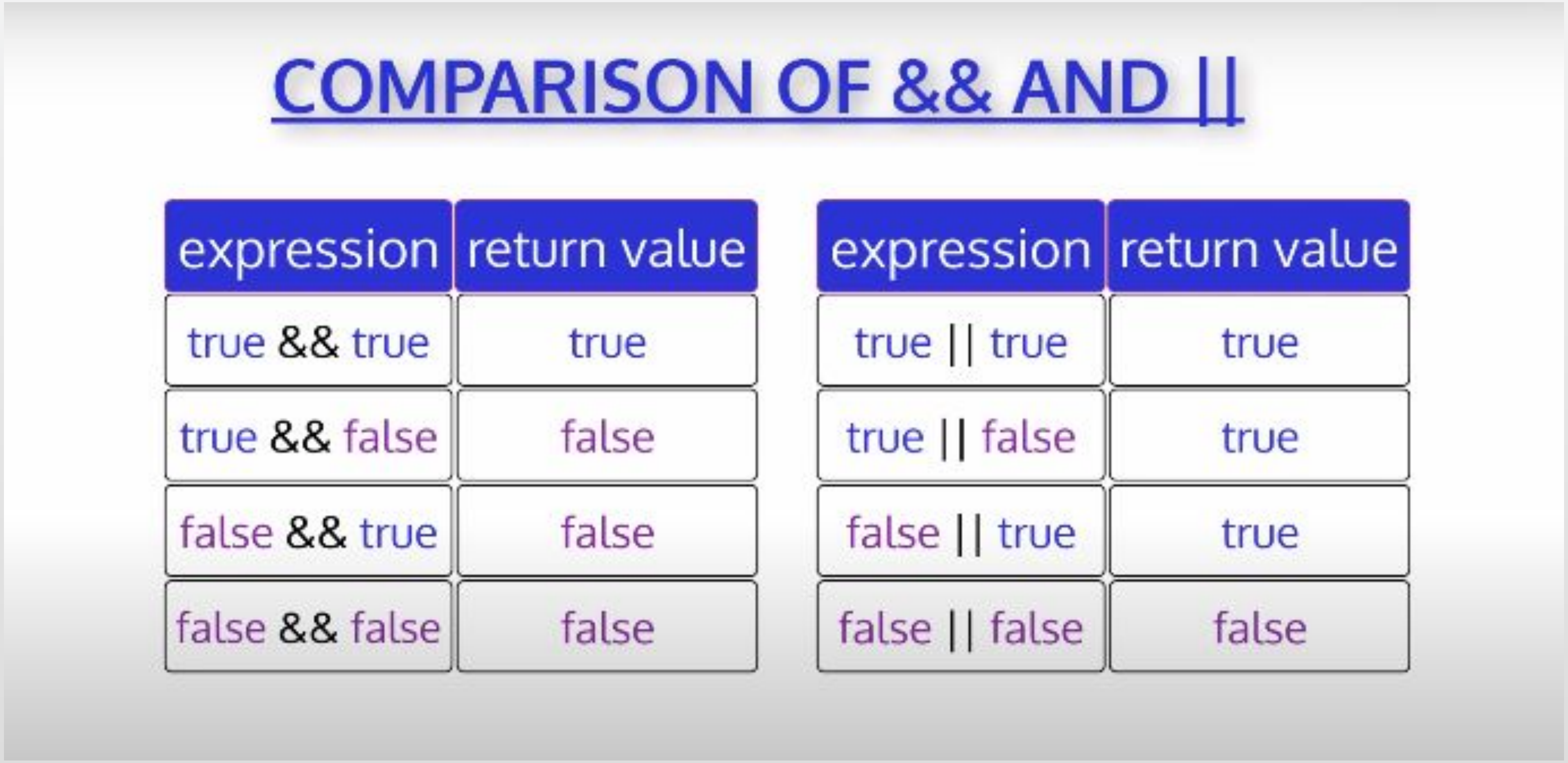
```
switch(myNum) {
  case 1: console.log('In case 1');
    break;
  case 2: console.log('In case 2');
    break;
  case 3: console.log('In case 3');
    break;
  default: console.log('In default');
}
```



```
switch(myString) {
  case 'coffee': console.log('Contains caffeine');
    break;
  case 'black tea': console.log('Contains caffeine');
    break;
  case 'lemonade': console.log('No caffeine');
    break;
  default: console.log('Drink not recognized');
    break;
}
```

- Once the program enters a **case**, it will execute all following statements until it reaches the end of the switch block, or a **break**, *even the statements for other cases.*
- Always use a **break** unless you know what you're doing, and you want that behavior.
- **default** clause is like the "else" in an if statement, will run if nothing else matches, best practice is to always use it

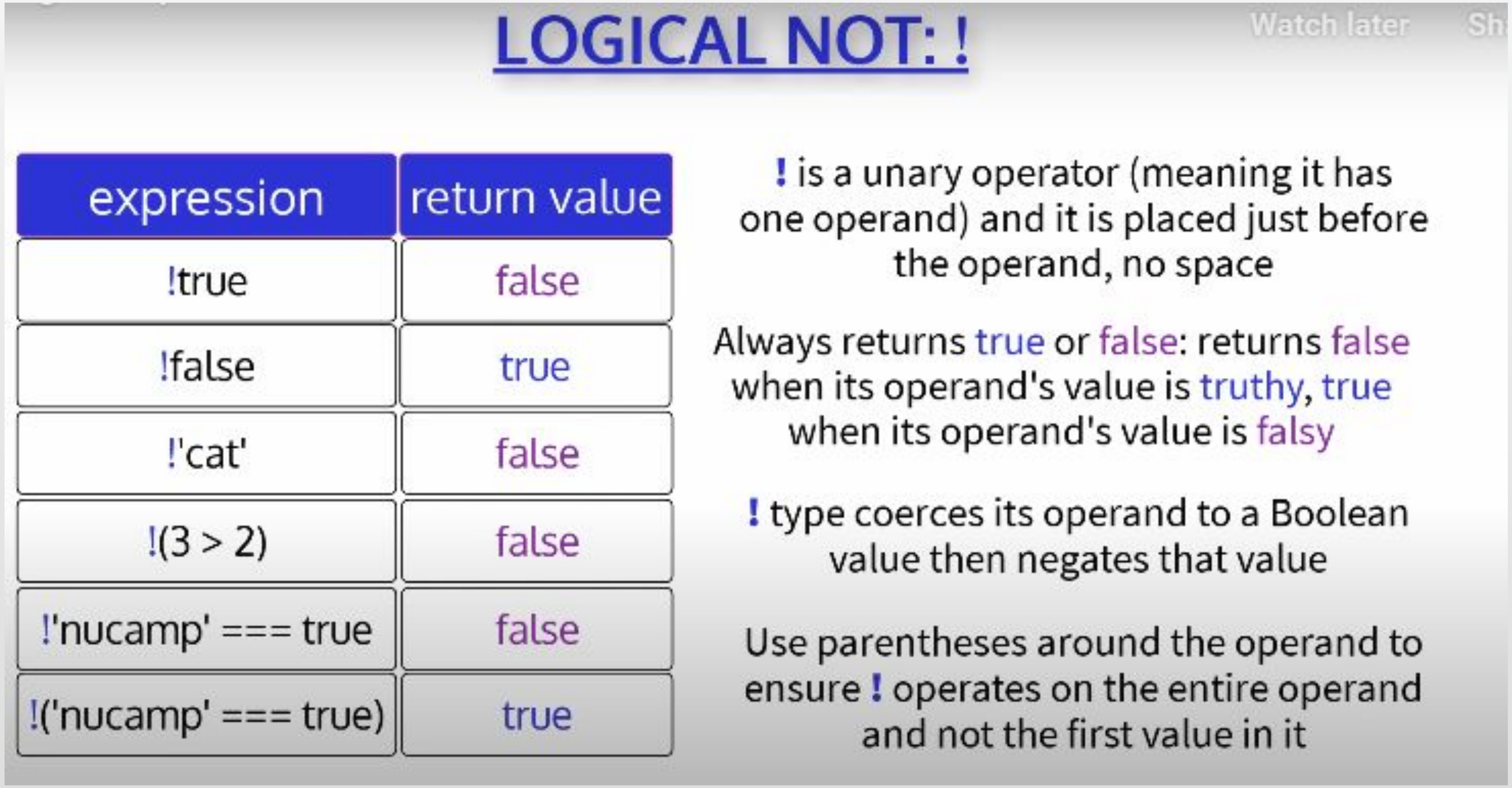
Logical Operators



COMPARISON OF && AND ||

expression	return value
true && true	true
true && false	false
false && true	false
false && false	false

expression	return value
true || true	true
true || false	true
false || true	true
false || false	false



LOGICAL NOT: !

expression	return value
!true	false
!false	true
!'cat'	false
!(3 > 2)	false
!'nucamp' === true	false
!('nucamp' === true)	true

! is a unary operator (meaning it has one operand) and it is placed just before the operand, no space

Always returns true or false: returns false when its operand's value is truthy, true when its operand's value is falsy

! type coerces its operand to a Boolean value then negates that value

Use parentheses around the operand to ensure **!** operates on the entire operand and not the first value in it

Truthy vs falsy

Discuss:

- What do the terms **truthy** and **falsy** mean?

Falsy – Any value that is 0, an empty string (""), undefined, null, or NaN will be evaluated as false

Truthy – Any other value that is NOT falsy will be evaluated as true

- How are they different from **true** and **false**?

Logical Operators: *Dealing with non-boolean values (Intermediate)*

- Logical And **&&** -- Returns first falsy value or last truthy value
- Logical Or **||** -- Returns first truthy value or last falsy value

This is called "short circuiting"

- Discuss:
  - What is returned from evaluating **(true && (3 >= 5))**?
  - What is returned from evaluating **(false || (5 - 10))**?

false

-5
-5 is the first truthy value and is returned

Tips for writing your own functions

There are multiple ways to approach something in Javascript

Use console.log() to track your progress

Complete your code in the way that makes sense to you

More advanced concepts like short circuiting can be used later when you're more experienced

Take a look at others approaches to code challenges for insight

Break Time! See You At 12:15 EST!

Coffee Break Timer

Challenge Code Review:

Magical Mystery Theatre

Workshop Assignment

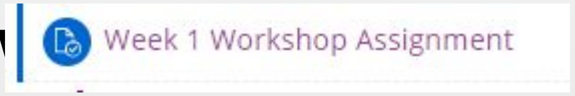
- It's time to start the workshop assignment!
- Today’s challenge:
  - ✓ Create a simple ticketing portal for the “Wild Waves Aquatic Park”
  - ✓ Ask the user which attraction they want to visit
  - ✓ Ask the user for their height in inches to ensure they are tall enough to ride
  - ✓ Alert the user whether they can access that ride or not
  - ✓ Bonus: grant the rider an all-access pass if they meet the criteria
- Work together in your groups to figure out how to code each step
- Use the “Magical Mystery Theatre” as an example if you get stuck
- Follow the workshop instructions very closely – accuracy matters!
- Try to complete the assignment by the end of the workshop

Assignment Submission

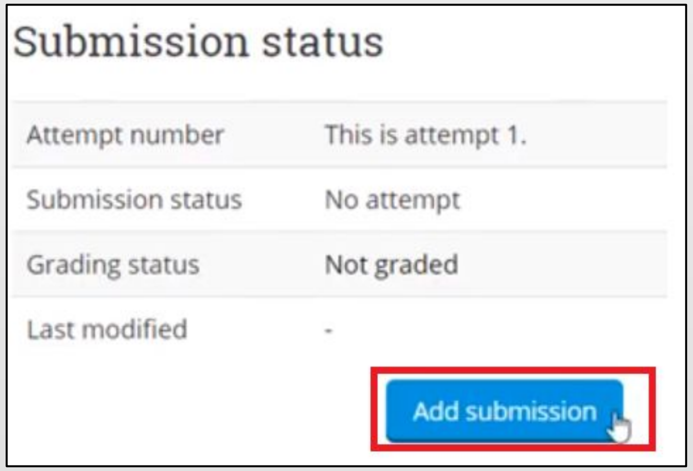
- Submit your **html** page at the bottom of the Week 1 assignment page in the learning portal.
- Submit your Week 1 feedback about your instructor and the course
- You may start on Week 2 as soon as you submit your assignment and your feedback
- Example on the next slide □

Submitting Your Assignment

- Go to https://learn.nucamp.co
  - Click "**Workshop Assignment: Students'**

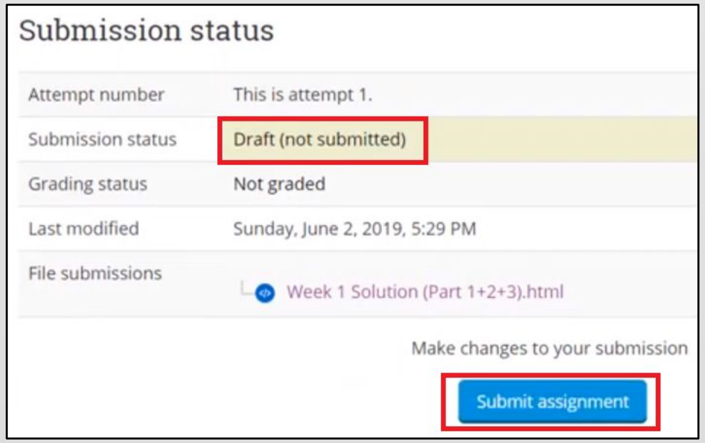

Week 1 Workshop Assignment

  - Upload your work by clicking "**Add Submission**", select the file, and then click "save"


Submission status

Attempt number	This is attempt 1.
Submission status	No attempt
Grading status	Not graded
Last modified	-

- Note that your work is in Draft status
  - Click "**Submit assignment**" to submit it


Submission status

Attempt number	This is attempt 1.
Submission status	Draft (not submitted)
Grading status	Not graded
Last modified	Sunday, June 2, 2019, 5:29 PM
File submissions	Week 1 Solution (Part 1+2+3).html

Make changes to your submission

nucamp

Happy learning!