Variables & Conditionals

Lesson 9

Learning Objectives

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

Agenda

- Review Intro to jQuery
- Variables
    - Declaring, Assigning, and Accessing
- Data Types
- Conditionals & Control Flow
- Lab Time

Intro to jQuery Review

Today, we will be diving back into the meat of Javascript.

But first, let's briefly review jQuery with a Code Along...

Code Along

Open: **MSiddeeq.com jQuery Exploration**

Directions:

Let's review some of the jQuery methods we learned in our previous class. To do this, we will use the **Chrome Developer Tools Console** in the browser

Timing:

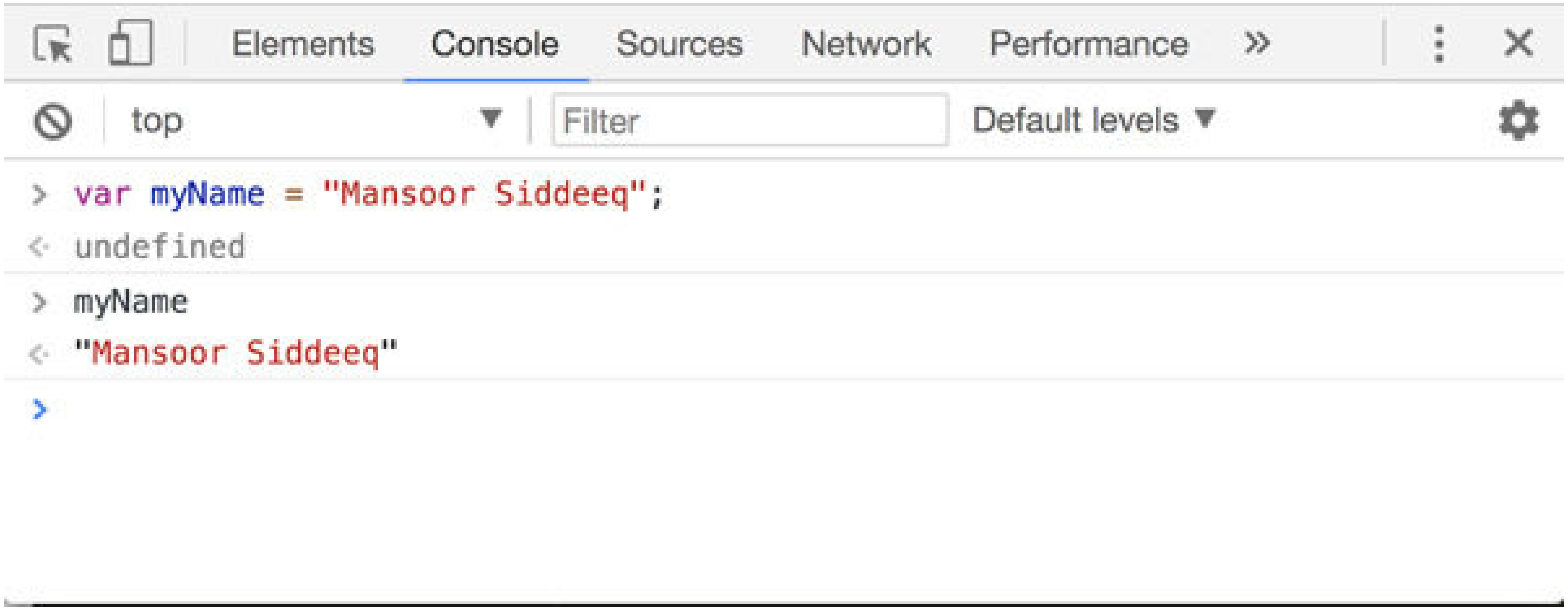
- **10 minutes** - Navigate to msiddeeq.com and open your browser's console.

Tips:

- If you encounter a jQuery method you are unfamiliar with, Google it!
- Try to use the jQuery Documentation whenever possible.

Console

Console (cont'd)



```
> var myName = "Mansoor Siddeeq";
< undefined
> myName
< "Mansoor Siddeeq"
>
```

jQuery Review

- Navigate to msiddeeq.com link in Chrome and open Console.
- Type the following and hit enter: **$('h1').slideUp()**
- What happened?
- Try to use other jQuery methods. What happens?
- Examples:
  - $('body').css(...)
  - $('p').click(function(){ ... })
  - $('.card').toggleClass('hvr-bob').toggleClass('full-off').toggleClass('full-on')

Variables

What are Variables?

JS Variables

Variables are essentially containers for storing data values in your JS code.

Declaring Variables

The act of creating/naming the variable is called a **declaration**.

```
var myAge;
```

Variable Declaration Conventions

- Cannot have spaces in the name
- **Never** use a reserved word as a variable name
- Should start with a lower case letter (generally)
- If they contain multiple words, subsequent words should start with an upper case letter (this is called **camel-casing**):

```
var numberOfStudents;
```

Assigning Variables

The action of saving a value to a variable is called **assignment**.

```
myAge = 30;
```

Declaring + Assigning

The declaration and assignment of a variable can be done at the same time:

```
var legalAge = 18;
```

Code Along

Open: Intro to Variables Codepen

Pulse Check

Key Objective:

Open the Intro to Variables Codepen in your browser, and follow the directions outlined in the JS comments.

Timing:

- **5 minutes** - As a class, work to declare and assign the variables as specified.

Accessing Variables

By **accessing the variable**, you can get the value from it in order to use it in your code.

```
legalAge;
```

Re-Assigning Variables

You can also **re-assign** a variable as many times as you want.

```
var myName = "Mansoor";

myName = "Sudi";
myName; // What will this output as?
```

Pulse Check

Key Objective:

Open the Intro to Variables Codepen again. This time, let's access the variables we've defined.

Timing:

- **5 minutes** - As a class, access each of the variables and output them to the console.

Bonus:

- Re-assign the variables to have different values.

EcmaScript 6 Improvements

LET and CONST

In ES6 (EcmaScript 6), we have 2 new ways to declare variables: **let** and **const**.

They were created to resolve a scoping issue with the **var** keyword.

Code Along

Open: **starter_code/score_keeper**

Directions:

1. Open the **starter_code/score_keeper** folder
2. Demonstrate to the class what the completed assignment looks like.
3. Together, add the appropriate JS/jQuery to the index.js file to get it to work

Timing:

- **20 minutes** - As a class, build out the JS code to make this lab interactive.

Data Types

Variables & Data Types

You can store all different types of data in your variables.

However, each one comes with their own special way it can be manipulated using JS.

Data Types (Primitive)

- Strings
- Numbers
- Boolean
- more...

Data Types: Boolean

A boolean in JS can represent one of two values: **true** or **false**.

```
true;

false;
```

Data Types: Strings

Strings

Strings are used to store textual information.

The actual value of a string will always be surrounded by quotes:

```
"How is the weather today?" //double quotes

'Today, it is warm.' //single quotes

"What's your name?" //sentence with apostrophe

'Today, we are reading, "The Count of Monte Cristo"' //sentence with quotes
```

Concatenation

You can chain multiple strings together to build complex strings. This is called **concatenation**.

```
"I live in Atlanta, but " + "I am from Indianapolis.";

// This will result in the following:
"I live in Atlanta, but I am from Indianapolis.";
```

Template Literals

You can also use something called **template literals** to create strings in JS.

Wrap your content with back-ticks (` `) to create template literals.

```
`Can I borrow Sarah's copy of "Frankenstein"?`;
```

Template Literal Benefits

Template literals allow for:

- multi-line strings
- expression interpolation
- and more...

Pulse Check

Strings or Nah?

```
"Obama"

'Atlanta, GA'

"2018"

true

`Welcome to the Jungle!`

750
```

Data Types: Numbers

Numbers

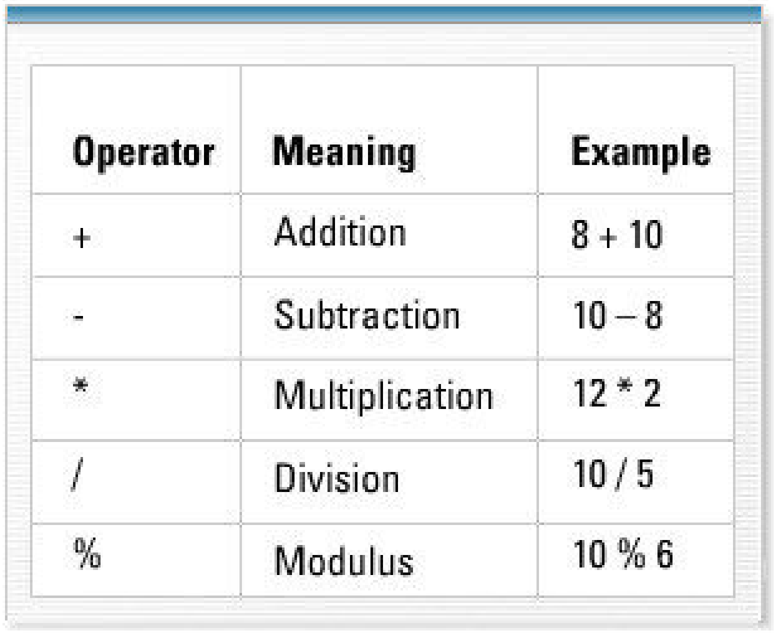
Numbers in Javascript represent numerical data.

They can be positive or negative, and either **integers** (whole numbers) or **floats** (decimals).

```
42;

3.14159265359;
```

Mathematical Operators



Operator	Meaning	Example
+	Addition	8 + 10
-	Subtraction	10 – 8
*	Multiplication	12 * 2
/	Division	10 / 5
%	Modulus	10 % 6

Modulus Operator

The **modulus operator** (%) returns the division remainder.

```
10 % 3;
// Equals 1
```

Numbers Quiz

Pulse Check

```
5209 + 13

125.34 * 55

141 / 23

566 % 34

78 - 2234

750 % 5
```

Break time!

Let's take 5-10 minutes to decompress...

Conditionals

What is Control Flow?

Control Flow

The **control flow** is the order in which the computer executes statements in a script.

Typically, this is from **left to right, top to bottom**.

```
var myName = 'Mansoor';
console.log(myName);

myName = 'Sudi';
console.log(myName);
```

Control Flow (cont'd)

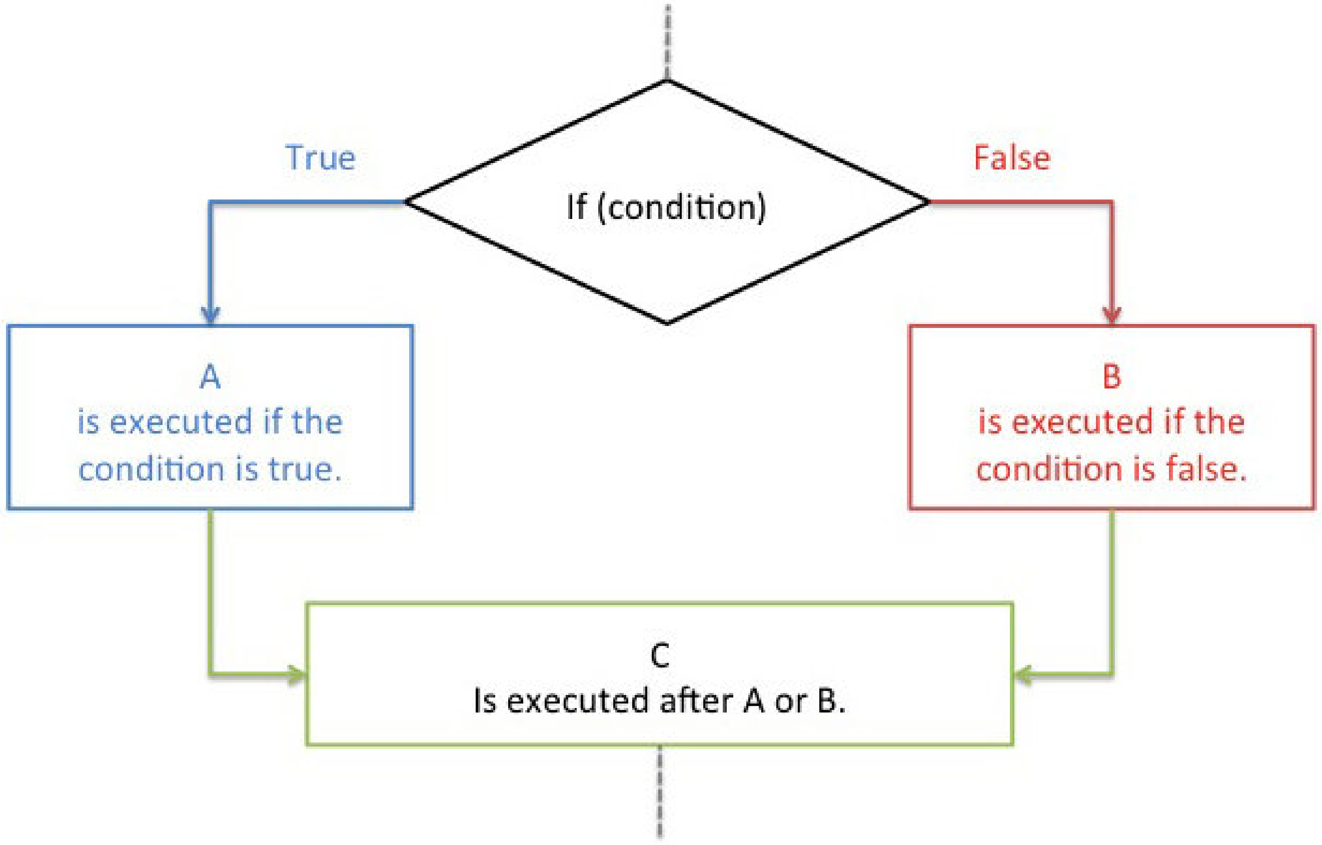
However, the computer can come across a structure that changes this **control flow** -- such as functions, loops, or conditionals.

Conditional Statements

Often, when you write code, you will want to perform different actions for different decisions.

You can use **conditional statements** in your code for this.

Conditional Statement Diagram


True
If (condition)
False
A
is executed if the condition is true.
B
is executed if the condition is false.
C
Is executed after A or B.

If/Else Statements

An **if/else statement** simply checks to see if something is either TRUE or FALSE...

Then does something based on the outcome.

"If" Statement Example

If you are older than 17, then log "You are an adult":

```
if (age > 17){
        console.log("You are an adult");
}

// Otherwise, continue running script...
```

"If/Else" Statement Example

If you are older than 17, then log "You are an adult."; else, log "You are a child.":

```
if (age > 17){
    console.log("You are an adult.");
}else{
    console.log("You are a child.");
}

// Continue running script...
```

"If/Else if" Statement Example

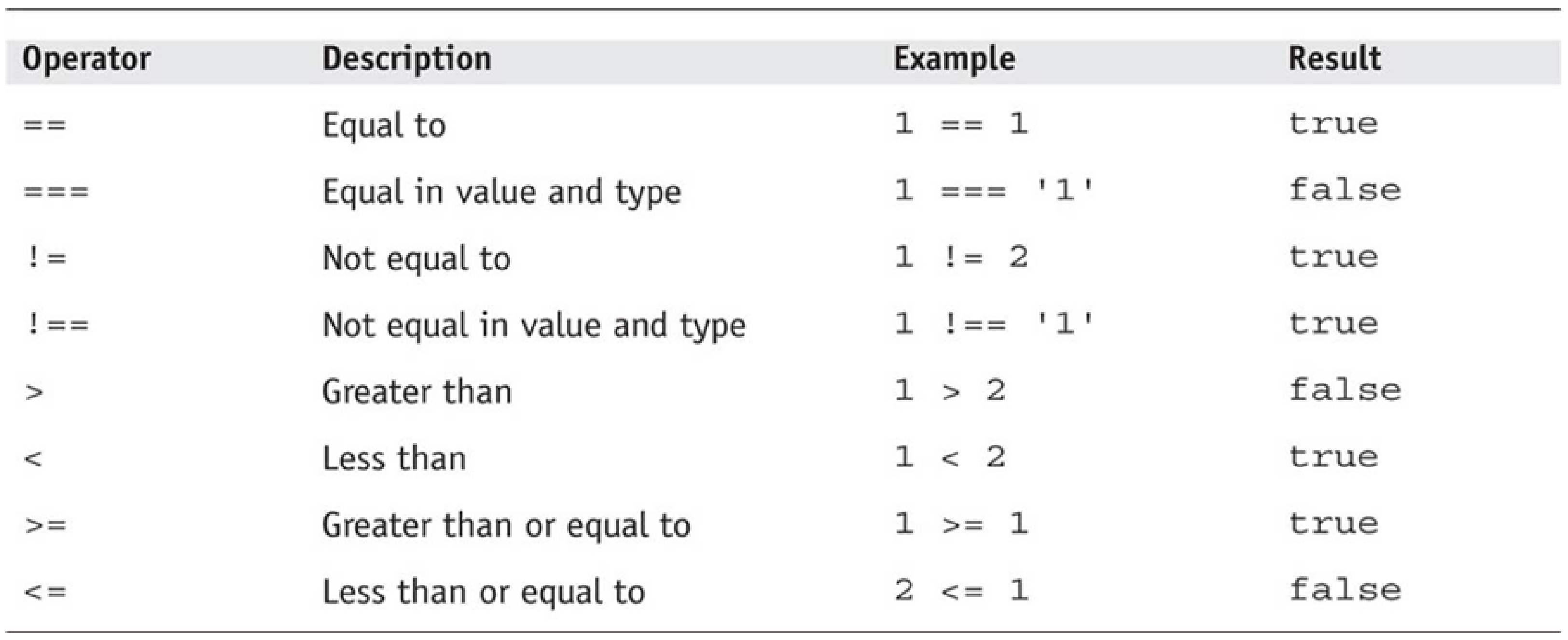
You can even chain multiple if/else statements for more conditions...

```
if (age > 17){
    console.log("You are an adult.");
}else if(age > 12){
    console.log("You are a teenager.");
}else{
    console.log("You are a child.");
}

// Continue running script...
```

Comparison Operators

To check if something is true or not, we need **comparison operators** to compare the criteria.



Operator	Description	Example	Result
`==`	Equal to	`1 == 1`	`true`
`===`	Equal in value and type	`1 === '1'`	`false`
`!=`	Not equal to	`1 != 2`	`true`
`!==`	Not equal in value and type	`1 !== '1'`	`true`
`>`	Greater than	`1 > 2`	`false`
`<`	Less than	`1 < 2`	`true`
`>=`	Greater than or equal to	`1 >= 1`	`true`
`<=`	Less than or equal to	`2 <= 1`	`false`

Code Along

Open: **starter_code/compare_that**

Directions:

1. Open the **starter_code/compare_that** folder
2. Demonstrate to the class what the completed assignment looks like.
3. Together, add the appropriate JS/jQuery to the index.js file to get it to work

Timing:

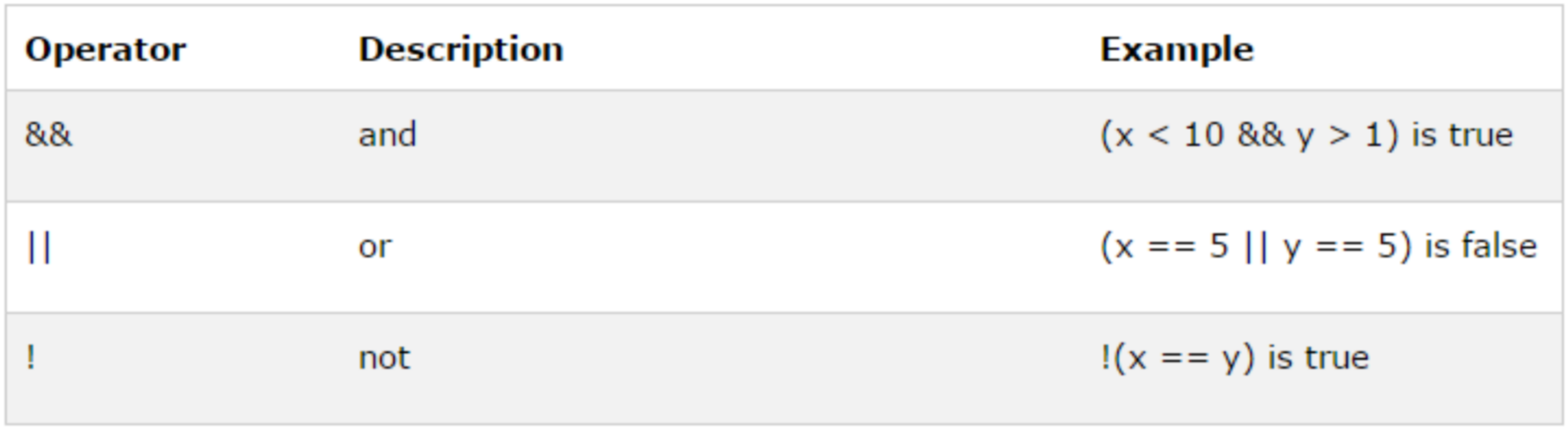
- **15 minutes** - As a class, build out the JS code to make this lab interactive.

Logical Operators

Logical operators allow you to check multiple criteria in a single conditional statement.

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

Logical Operators (cont'd)

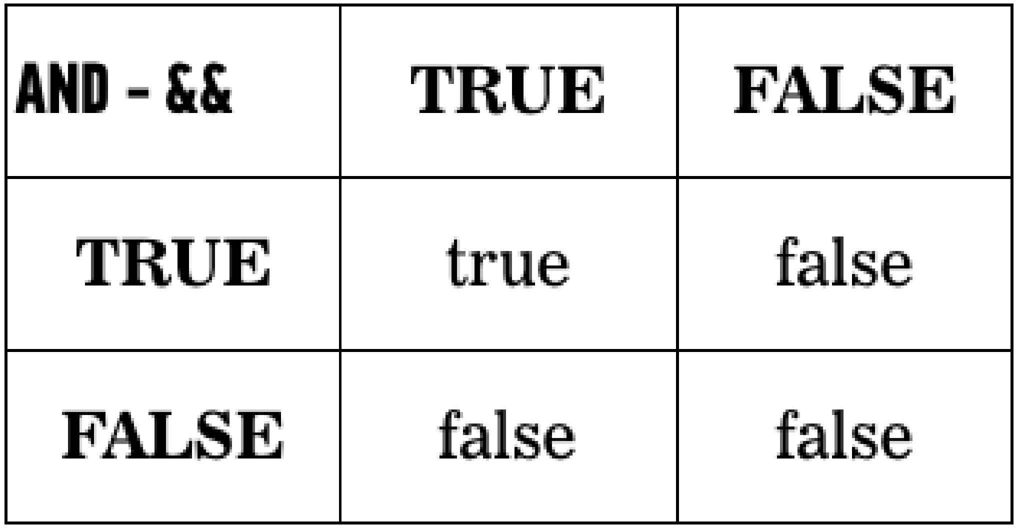


Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true

`x = 6 and y =3`

Logical AND (&&) Operator

This checks to see if **ALL** of the conditions are true before running the code inside of the conditional statement.



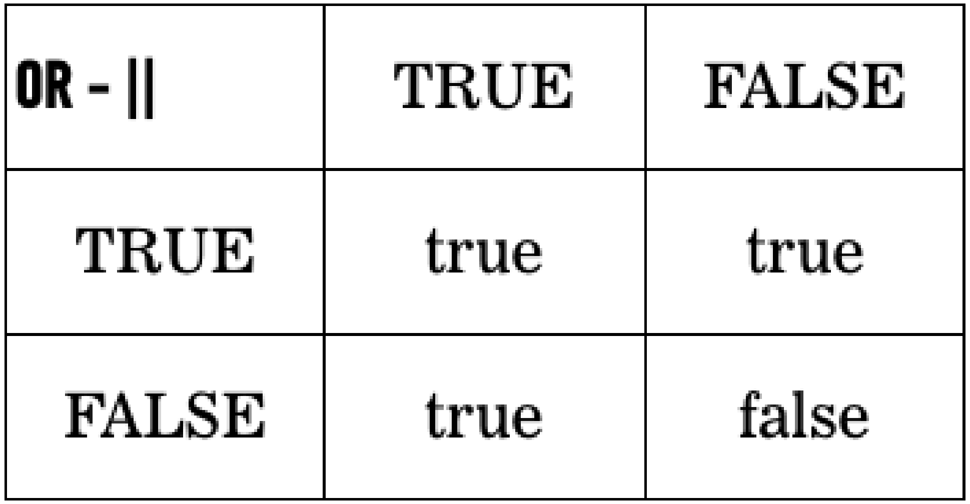
AND - &&	TRUE	FALSE
TRUE	true	false
FALSE	false	false

Logical AND (&&) Example

```
if (username == "email" && password == "secret"){
        // Allow access to Facebook Account
}
```

Logical OR (| |) Operator

This checks to see if **AT LEAST ONE** of the conditions are true before running the code inside of the conditional statement.



OR - ||	TRUE	FALSE
TRUE	true	true
FALSE	true	false

Logical OR (||) Example

```
if (day == "Saturday" || day == "Sunday"){
        // It is the weekend
}
```

Additional Resources

- Data Types (W3Schools)
- JS Operators (W3Schools)

Lab Time

Open: **starter_code/blackout**

Directions:

1. Open the **starter_code/blackout** folder
2. Show how it should work
3. Build an application that turns the "light" on or off when you click the button

Timing:

- **10 minutes** - As a class, preview write out the pseudocode in the js file
- **20 minutes** - build out the JS code to make this lab interactive.
- **10 minutes** - everyone take 1 minute to share what they have with the class (if we have time)

Exit Tickets

Take 5-10 minutes to give us some

(Link is in Slack Room)

Learning Objectives Review

- We defined variables and identify best cases to use them.
- We differentiated between strings, integers and floats.
- We applied conditionals to change the programs control flow.

Week 4 HW Due Today

- **Due today by 11:59pm ET**
- Once you upload your HW to your repo, please PM the TA with a link to the Assignment folder.
- This is how we will know whether an assignment has been completed or not
- Once graded, the TA will reply with feedback

Final Project Milestone 1

- **Due tonight at 11:59pm ET**
- Wireframe/proposal should be uploaded to your forked repo and placed inside of the **Final_Project/Milestone 1** folder
- Does NOT have to be high quality (can be hand drawn/written, etc.)
- **REMEMBER:** These milestones are for YOUR benefit!
- Be prepared to discuss in subsequent labs

Next Class...

Lesson 10 - JS Lab