LESSON 10 - FUNCTIONS

AGENDA

- Review Variables & Conditionals
- Functions
- Anonymous Functions
- Weather Application

HOUSEKEEPING NOTES

- Last day of class + presentations will be on February 26th
- We will skip the student/instructor choice classes in favor of more lab time for projects
- Homework will continue to be due on Mondays at 5pm before class

LEARNING OBJECTIVES:

AFTER TODAY, YOU SHOULD BE ABLE TO...

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

REVIEW - VARIABLES & CONDITIONALS

WHAT ARE VARIABLES?

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

You can declare, assign, and access variables in Javascript.

- **Declare:** `var myAge;`
- **Assign:** `myAge = 30;`
- **Declare & Assign:** `var myAge = 30;`
- **Access:** `myAge;`

You can also re-assign variables as many times as you want

```
var myName = "Mansoor";

myName = "Sudi";
```

VARIABLE CONVENTIONS

- you should never use a reserved word -- List of Reserved Words
- they should always start with a lower case letter
- if they contain multiple words, subsequent words should start with an upper case letter:

```
var numberOfStudents = 10;
```

DATA TYPES

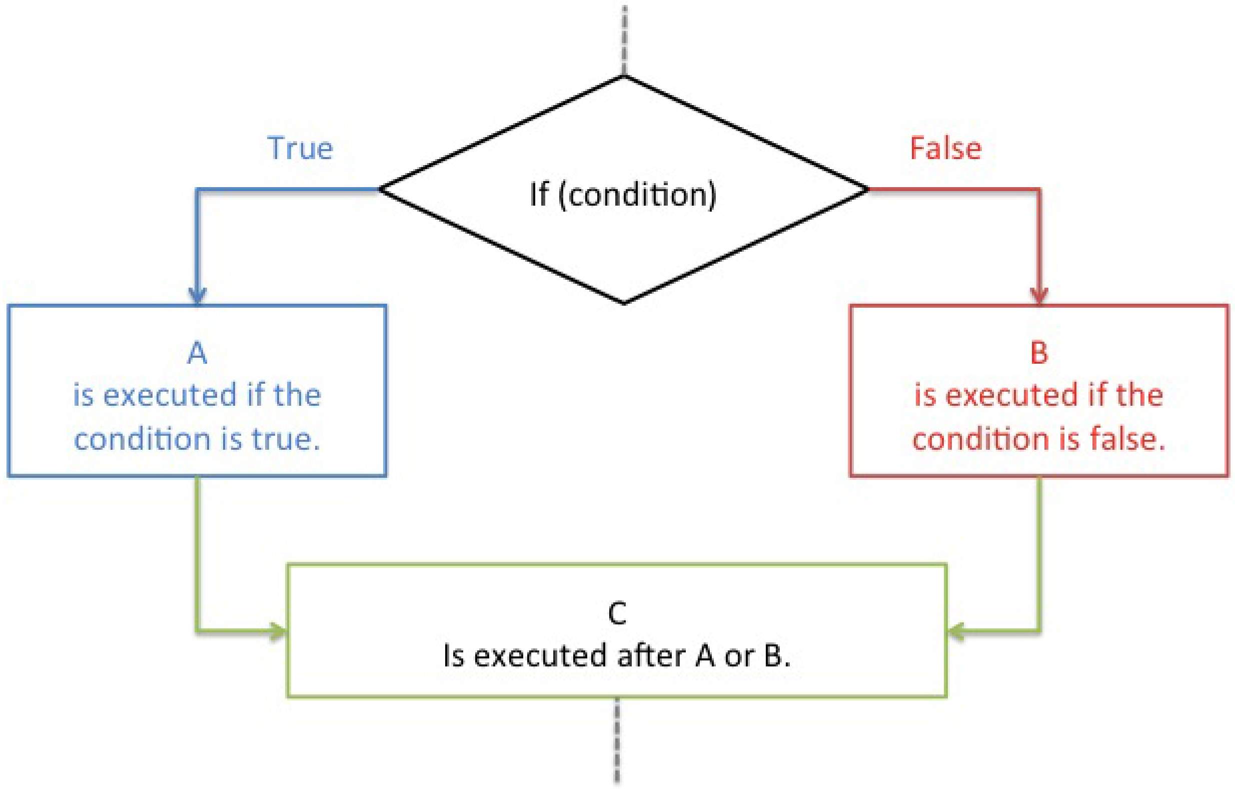
Variables can contain various different **data types**. Below are the 3 most common:

- **String** (text data)
- **Int/Float** (number data)
- **Boolean** (true or false)

PULSE CHECK - WHAT AM I?

- "Hello world!"
- "23 + 5"
- true
- 30 * 2
- "false"

CONTROL FLOW



IF/ELSE STATEMENT

This simply checks to see if something is either **TRUE** or **FALSE**

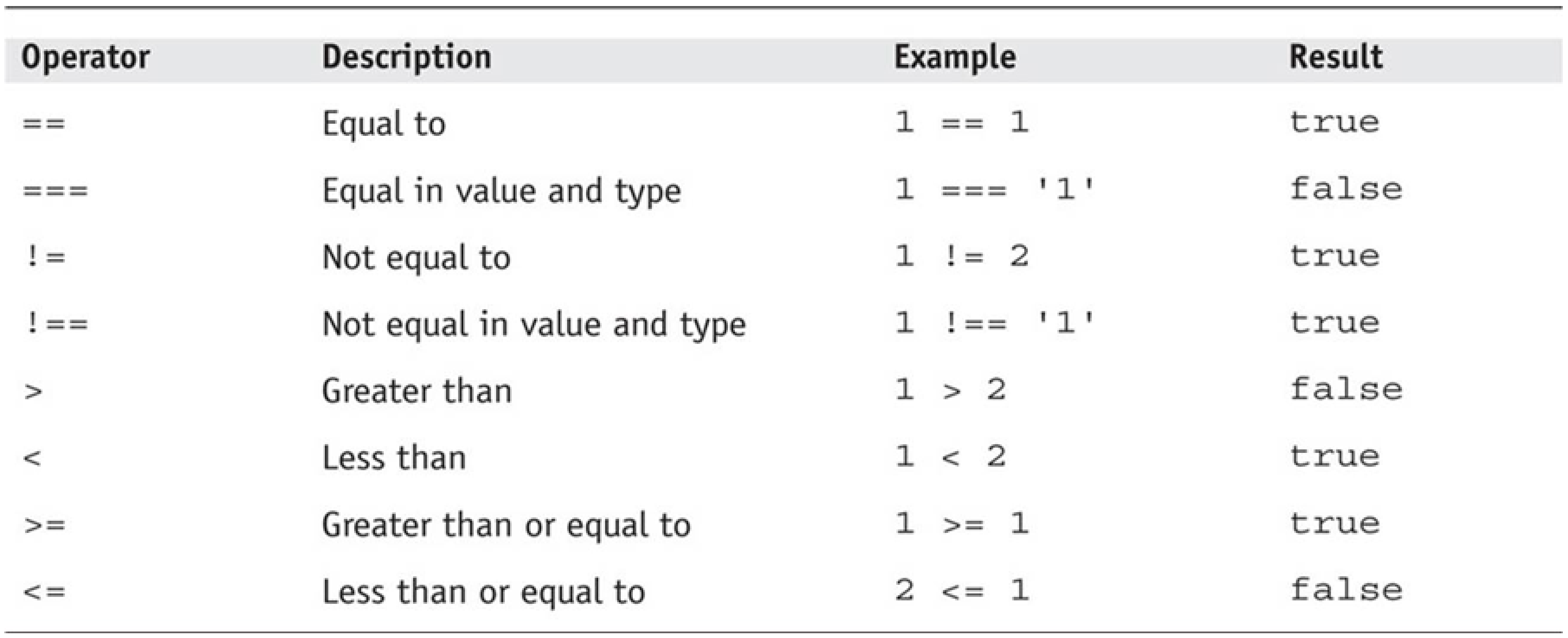
Then does something based on the outcome.

EXAMPLE IF/ELSE STATEMENT

```
if(condition is true) {
    // Do thing A
}else if{
    // Do thing B
}else{
    // Do something else
}
```

COMPARISON OPERATORS

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



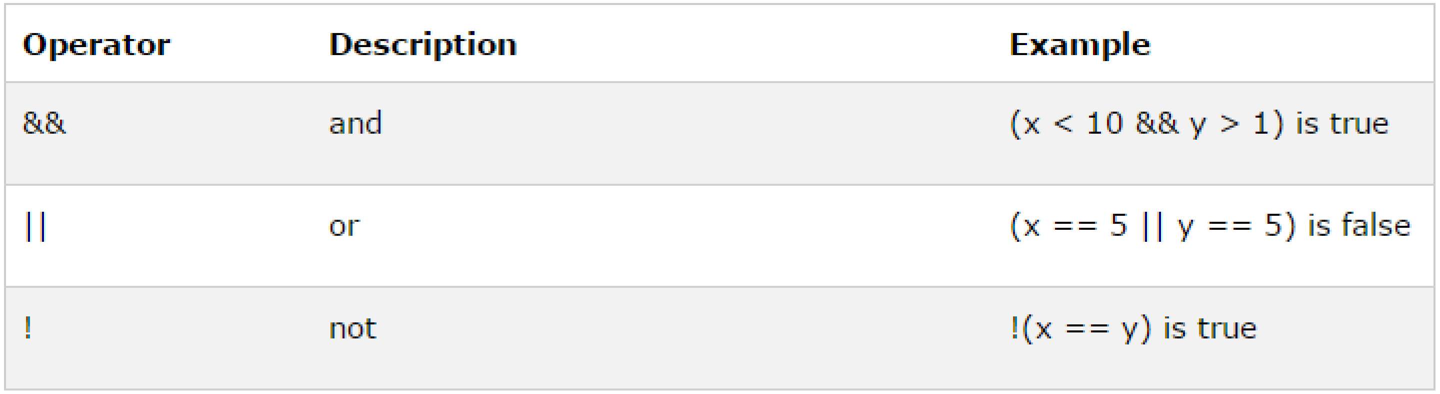
Operator	Description	Example	Result
`==`	Equal to	`1 == 1`	`true`
`===`	Equal in value and type	`1 === '1'`	`false`
`!=`	Not equal to	`1 != 2`	`true`
`!==`	Not equal in value and type	`1 !== '1'`	`true`
`>`	Greater than	`1 > 2`	`false`
`<`	Less than	`1 < 2`	`true`
`>=`	Greater than or equal to	`1 >=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

EXAMPLE COMPARISON OPERATORS:

```
if (age >= 25){
    // You can rent a car!
}
```

LOGICAL OPERATORS

We can also check multiple conditions in a single conditional statement using **logical operators**

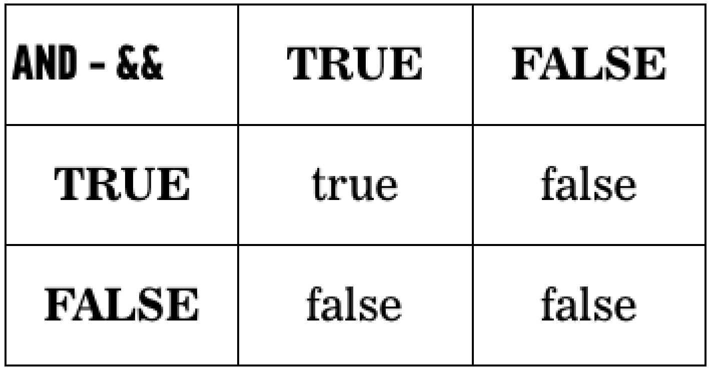


Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true

EXAMPLE LOGICAL OPERATORS

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

THE LOGICAL AND (&&) TRUTH TABLE

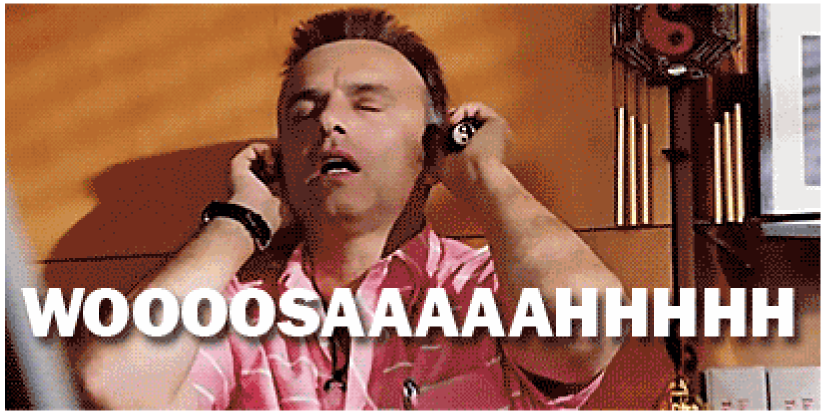


AND - &&	TRUE	FALSE
TRUE	true	false
FALSE	false	false

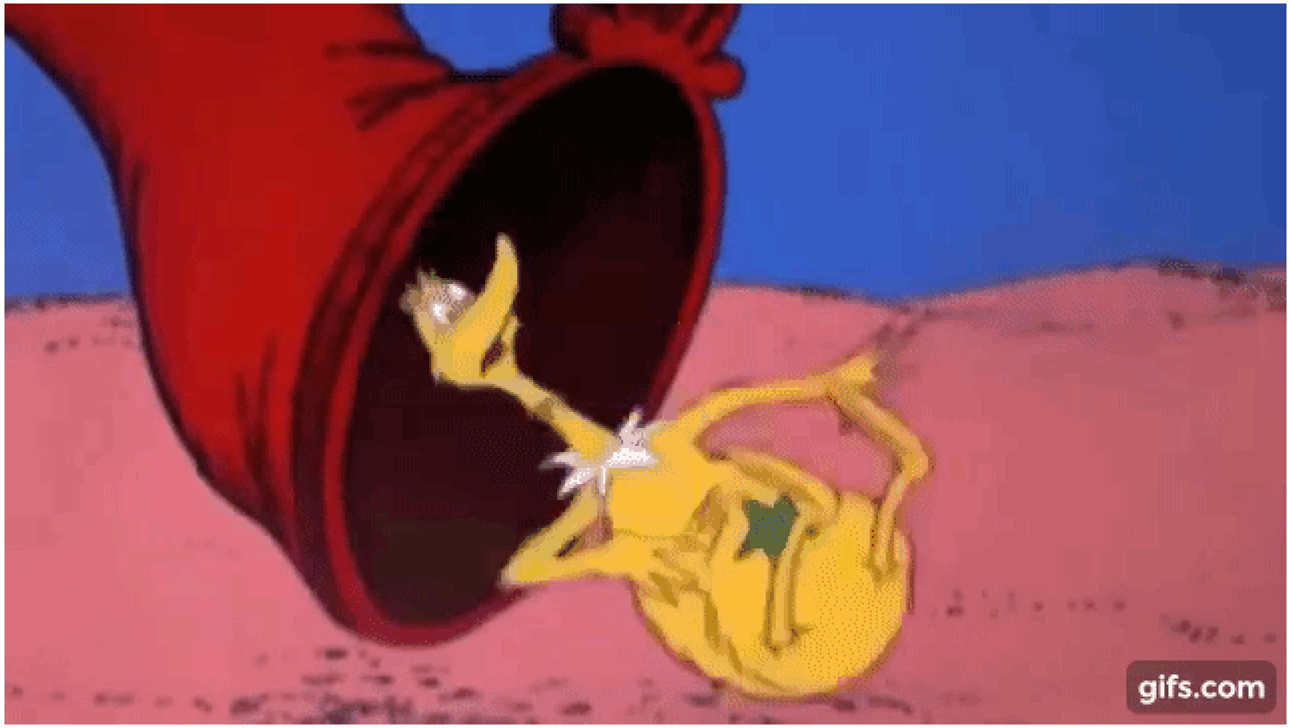
THE LOGICAL OR (||) TRUTH TABLE

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

Let's breathe for a second...

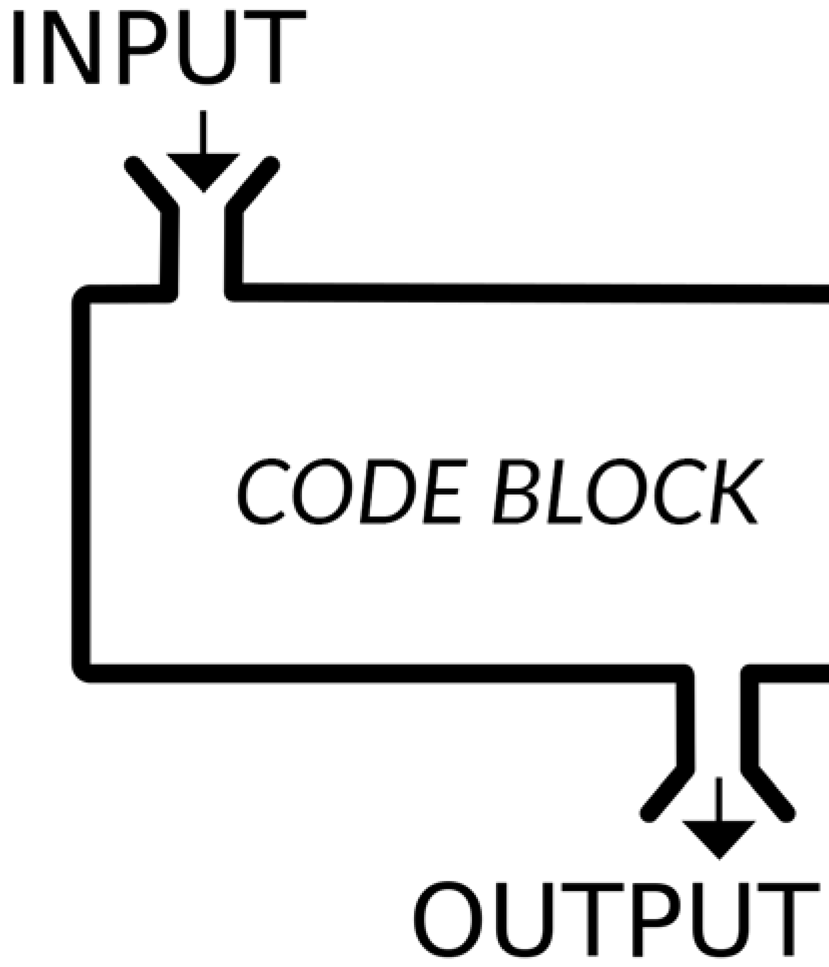


INTRO TO FUNCTIONS



WHAT IS A FUNCTION?

A **function** is a reusable block of code that performs an action or returns a value.


INPUT
CODE BLOCK
OUTPUT

FUNCTIONS SYNTAX

The name of your function

`function` Keyword.
Like the “var” keyword.

```
function functionName(param1, param2) {
  //Body of function
}
```

DECLARING FUNCTIONS (NAMED FUNCTIONS)

```
function addNumbers() {
    ...
}
```

INVOKING FUNCTIONS

```
addNumbers();
```

```
function helloWorld() {
  console.log("Hello Functions");
}

helloWorld(); //Prints “Hello Functions to the
console.
```

The brackets execute the function.
Try calling the function without
them to see what happens.

```
calculator;
```

```
calculator(5,6);
```

FUNCTION ARGUMENTS & PARAMETERS

Parameters are temporary variable names within functions. The **argument** can be thought of as the value that is assigned to that temporary variable.

PARAMETERS SYNTAX:

Parameters let you pass data into the function

```
function functionName(param1, param2) {
  //Body of function
}
```

The functions executed code goes between the { } brackets. Much like an “if” statement.

PARAMETERS:

```
function fullName(firstName, lastName) {
    console.log("My name is " + firstName + " " + lastName);
}
```

ARGUMENTS SYNTAX:

```
function addAndPrint(num1, num2) {
  var sum = num1 + num2;
  console.log(sum);
}
```

```
addAndPrint(1, 2); // Result is 3

addAndPrint(8, 2); // Result is 10
```

ARGUMENTS:

```
fullName("Mansoor", "Siddeeq");
```

RETURN FUNCTIONS

Often, when calling a function, you also want to return a value from that function.

To do this, you would need to use a **return statement** in the function.

This tells the program to "return" something back to whatever called it -- whether that was a variable, another function, etc.

```
function calculateSum(num1,num2) {
    return num1 + num2;
}

var sum = calculateSum(1,2);

sum; // accessing this variable will output 3
```

```javascript
function calculateAge(currentYear,birthYear) {
    return currentYear - birthYear;
}

function myBio() {
    console.log("My name is Mansoor, and I am " + calculateAge
}

myBio(); // What will be written to the console?
```

IMPORTANT

When JavaScript reaches a **return statement**, the function will stop executing.

Meaning, if you have code after a return statement, it will not be read.

```
function myName() {
    return "Mansoor";
    return "Sudi";
}

myName(); // This will output "Mansoor" and immediately exit t
```

WE DO - CASH REGISTER

5 MINUTE BREAK

ANONYMOUS FUNCTIONS (FUNCTION EXPRESSIONS)

Earlier, we learned about how declaring functions creates a **named function**

Alternatively, you could assign the function to a variable as an expression, like so:

```
var helloWorld = function() {
    return "Hello World!";
};
```

Any function without a name (aka not **declared** as outlined in our previous slide) is considered an **anonymous function**

DECLARED FUNCTION

```
function area(width,height) {
    return width * height;
}

var size = area(2,4);

console.log(size);
```

FUNCTION EXPRESSION

```
var area = function (width,height) {
    return width * height;
};

var size = area(2,4);

console.log(size);
```

So… what is the point?

TL;DR

HOISTING

This involves how the JS interpreter looks for variable/function **declarations** before going through each piece of the script

DECLARED EXPRESSION

```
console.log(size); // will output 8

function area(width,height) {
    return width * height;
};

var size = area(2,4);
```

FUNCTION EXPRESSION

```
console.log(size); // will result in an error

var area = function (width,height) {
    return width * height;
};

var size = area(2,4);
```

Javascript Functions Additional detail

Now, let's look at the Cash Register example, and convert it so that it uses anonymous functions instead...

CODE ALONG

ANNONYMOUS CASH REGISTER

LAB

YOU DO - TEMP CONVERTER

- Get temperature in Celsius
- Convert temperature to Fahrenheit
- Check temperature to determine what color to change the background
    - if temperature is less than/equal to 65
        - set background to Blue
    - if temperature is greater than 65, but less that 85
        - set background to Yellow
    - if temperature is greater than/equal 85, but less than 95
        - set background to Orange
    - if temperature is greater than 95
        - set background to Red

Now you work on the code to build this. Don't forget your HTML Templating first.

EXIT TICKETS

Let's spend 5-10 minutes to fill out today's Exit Survey

LEARNING OBJECTIVES REVIEW

- We Described arguments as they relate to functions.
- We Predicted values returned by a given function.
- We Differentiated control flow between anonymous and named functions.

WEEK 5 HOMEWORK + FINAL PROJECTS

Homework: Citi Pix

Milestone 3: First draft of JS due on Friday 2/2