Conditionals & Functions

But, first...

Learning Objectives

- **Practice** controlling the flow of a program using conditionals
- **Identify** the need for functions and **practice** using them
- **Identify** and use both comparison and logical operators effectively
- **Understand** the purpose of parameters and return values

Agenda

- Surveys
- Conditionals
  - Comparison Operators
  - Type Coercion
  - Logical Operators
- Functions
  - Parameters
  - Return values

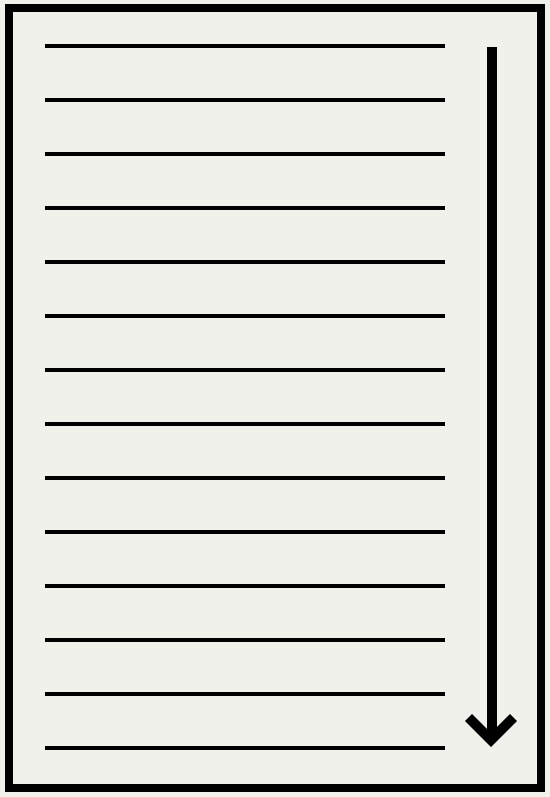
Review

Review

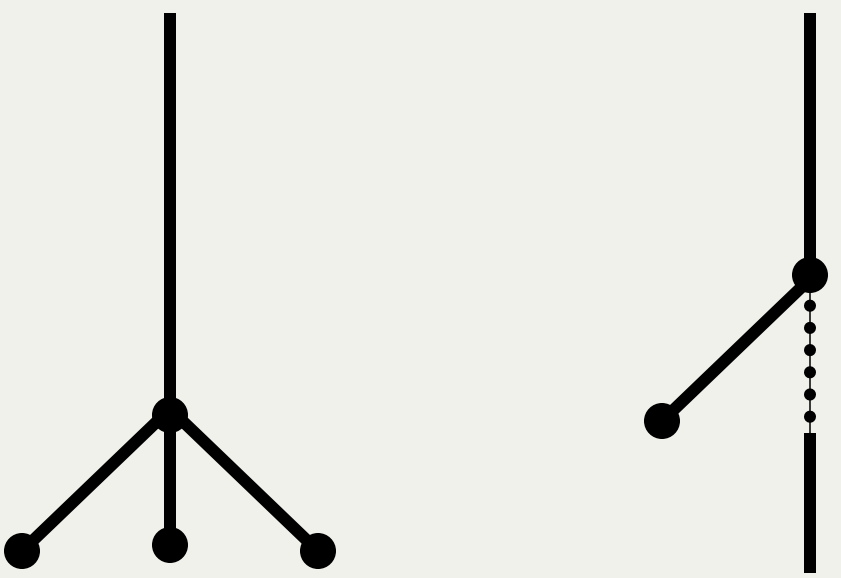
- What is JavaScript and what can it do?
- How do we include it in a page?
- What is pseudo-code?
- What is a data type?
- What are some data types that are available in JS?
- What are properties?
- What are methods?
- What are variables and why would you use them?
- How should we name variables?

Conditionals

Execution of a program



With Conditionals...



What are they?

- Conditional statements execute or skip parts of a program based on the value of an expression
- These are the decision points of your code, or the "branches"
- They rely quite heavily on boolean(ish) values

The if statement

- This is the fundamental control statement that allows JavaScript to make decisions

```
if ( CONDITION ) {
    // Things to execute go here!
}

if ( true ) {
    console.log( "This runs" );
}

if ( 42 === 41 ) {
    console.log( "This does not run" );
}
```

The if statement

```
if ( expression ) {
    // statement(s)
} else {
    // statement(s)
}

if ( true ) {
    console.log( "This will run" );
} else {
    console.log( "This won't" );
}

if ( !false ) {
    console.log( "The opposite is true" );
}
```

Comparison Operators

Operator	Meaning	Example
==	Equality	4 == "4"
===	Strict Equality	4 === 4
!=	Inequality	1 != "5"
!==	Strict Inequality	8 !== 2
<	Less	4 < 5
<=	Less than or equal	3 <= 5
>	Greater than	2 > 1
>=	Greater than or equal	42 >= 21

Equality vs. Strict Equality

That is, a comparison between **==** and **===**

Always use threequals! It makes sure that the types are the same too

Remember that one equals sign is assignment

Logical Operators

Operator	Meaning	Example
&&	AND	1 === 1 && 2 === 2
||	OR	true || false
!	NOT	!false

The if statement

```
if ( 5 > 4 ) {
    console.log( "Yes, it is!" );
}

var myNumber = 42;
if ( myNumber === 42 ) {
    console.log( "Equal" );
}

if ( 3 >= 2 && 7 === 7 ) {
    console.log( "Yep" );
}

if ( false || true ) {
    console.log( "Yep" );
}
```

The if statement

```
var age = 42;

if ( age >= 18 ) {
    console.log( "You can vote" );
} else {
    console.log( "You can't" );
}
```

If && Else If && Else

```
if ( someCondition ) {

} else if ( someOtherCondition ) {

} else {

}

if ( 4 === 3 ) {
    console.log( "First statement" );
} else if ( 42 !== 42 ) {
    console.log( "Second statement" );
} else {
    console.log( "Third statement" );
}
```

If && Else If && Else

```
var age = 42;

if (age >= 35) {
    console.log('You can vote AND hold any place in government!');
} else if (age >= 25) {
    console.log('You can vote AND run for the Senate!');
} else if (age >= 18) {
    console.log('You can vote!');
} else {
    console.log('You have no voice in government!');
}
```

Exercise!

If Statements, Comparison and Logical Operators

Functions

What are they?

- A reusable section of code that has a purpose and a name
- The bread and butter of JS
- They can associate names with subprograms

What are they?

Creating new words is normally bad practice, though fun.
It is essential in programming!

We give a name to a part of our program, and in doing so, we make it flexible, reusable and more readable

How do they work?

- We **define** a function
- We **call** (or **execute**) it when we want it to run

What can they do?

They can perform any code!

- Calculations
- Animations
- Change CSS
- Change, add, or delete elements on the page
- Speak to the server
- Anything!

How do we define them?

```
function sayHello () {
    console.log( "Hello" );
}
// A Function Declaration

var sayHi = function () {
    console.log( "Hi" );
};
// A Function Expression
```

How do we call them?

```
function sayHello () {
    console.log( "Hello!" );
}

sayHello(); // The callsite
```

How do we call them?

```
var sayHello = function () {
    console.log( "Hello!" );
}

sayHello(); // The callsite
```

Parameters

They aren't dynamic... yet!

This brings us to parameters. **Parameters** (or **arguments**) allow us to give a function a little bit of extra information or detail

Parameters

```
function sayHello ( name ) {
    var greeting = "Hello " + name;
    console.log( greeting );
}

sayHello( "Groucho" );

sayHello( "Harpo" );

sayHello(); // ???
```

Parameters

```
function multiply (x, y) {
    console.log( x * y );
}

multiply( 5, 4 );

multiply( 10, -2 );

multiply( 100, 0.12 );
```

Pseudocode Examples

```
moveToLeft function
    RECEIVE an element to animate
    STORE the current left position as currentLeft
    STORE the new left position, as desiredLeft, by adding 100px to currentLeft
    UPDATE the left position of the provided element to be desiredLeft

changeTheme function
    RECEIVE a themeChoice ("light" or "dark")
    IF themeChoice === "light"
        CHANGE the body background to "white"
        CHANGE the text color to "black"
    ELSE
        CHANGE the body background to "black"
        CHANGE the text color to "white"
```

Return Values

Sometimes your function calculates something and you want the result!

Return values allow us to do that

Return Values

```
function squareNumber ( x ) {
    var square = x * x;
    return square;
};

var squareOfFour = squareNumber( 4 );
```

Return Values

```
function squareNumber ( x ) {
    var square = x * x;
    return square;
};

var squareOfFour = squareNumber( 4 );

var squareOfTwelve = squareNumber( 12 );

squareNumber(4) + squareNumber(12);

squareOfFour + squareOfTwelve;
```

Return Values

- A **return** value means that a function has a result
- It is always the last line that executes

```
function sayHello () {
  return "No.";
  console.log( "Hi!" );
};

sayHello();
```

Passing in Variables

```
var addTwoNumbers = function (x, y) {
  return x + y;
};

var firstNumber = 10;

addTwoNumbers( firstNumber, 4 );
addTwoNumbers( firstNumber, 6 );
```

Exercise!

JavaScript Functions

Homework

- Finish off the Conditional Exercises
- Finish off the Function Exercises
- Prepare for Milestone 1

Next Lesson

- More JavaScript
  - Loops (for, while)
  - Collections (Arrays, Objects)

Q & A

Feedback Time

Lesson 10: **Conditionals and Functions**

https://ga.co/fewd32syd

Thanks!