JavaScript: Logic & Loops

What is logic (as it pertains to programming)?

- The control flow of your program
- Think of logic as a fork in the road. Which direction you'll head is based on the result of a test

First stop: Testing

- Any test returns a boolean true or false
- To test if two strings are equal:

```
'stringone' === 'string two';
›false
```

- Using three equals signs instead of two also checks the type
- If you don't check type, these are both true:
  - (10-5) == 5;
  - (10-5) == '5';
- Will cause bugs down the road!

More testing examples

To test if two strings are NOT equal:

```
'stringone' !== 'string two';
> true
```

To test if one number is greater than another:

5 > 10;

false

<, >, <=, >= are also valid comparison operators

The `if` statement

The `if` statement allows us to run code only if a certain test evaluates to `true`

```
if (5 > 10) {
  console.log("You'll never see this in the console because 5 is not greater than 10");
}

if (5 < 10) {
  console.log('But you'll definitely see this');
}
```

The `else` statement

The `else` statement runs only if the statement in the `if` statement is false

```
if (5 > 10) {
  console.log("You'll never see this because 5 is not greater than 10");
} else {
  console.log("You will see this though, since 5 < 10 evaluates to false");
}
```

Exercises

- Write a program that checks if a variable is less than 10. If it is, alert the user that their variable is less than 10. If it is not, let the user know what the variable was and that it was greater than 10.
- Try running the script and then changing the variable's value to see how this affects the program output
- If you have extra time, write a similar program to check if a string stored in a variable is the same as another string

The `else if` statement

If you want to run another test before getting to `else`, you'll want to use `else if`

```
if (5 > 10) {
  console.log("You'll never see this because 5 is not greater than 10");
} else if (5 === 5) {
  console.log("Yes, 5 really is equal to 5, so this will show up in the console")
} else {
  console.log("We won't get here because our else if evaluates to true");
}
```

Functions

- A `function` is a way to encapsulate code for later use
- It can take arguments, which are used as variables inside the function
- It can return a value, which may be used later on or displayed immediately

Functions

```
// Declare a function called addNumbers that takes
// two arguments: numberOne and otherNumber

function addNumbers(numberOne, otherNumber){
  // return the sum of numberOne, 10, and otherNumber
  return numberOne + 10 + otherNumber;
}

// call your new function, giving it 2 argument values
// numberOne = 4, otherNumber = 14
// log the result to the console
console.log(addNumbers(4, 14));
// Calling a function is also known as "invoking" it
>28
```

Another function example (Named function)

```
// declare a function with the name fullName
// that takes two arguments
function fullName(fname, lname){
  console.log(fname + ' ' + lname);
}
// invoke the function
fullName('Lucille', 'Bluth');
```

Functions: Stored in a variable

```
var myGreatFunction = function() {
  console.log('Do Something');
};
```

// Immediately creates a variable with a function stored inside

```
myGreatFunction(); // Invoke your function
```

Exercises

- Declare a function that takes no arguments but outputs something to the console. Invoke it after it has been declared
- Declare a function that takes your first name as an argument and alerts your name. Invoke it after it has been declared
- Declare a function that takes a "door" as an argument. Depending upon which "door" is provided, tell the user they've received a different "prize" in the console. Invoke it after it has been declared a few times, with each door option.

Identifying data types

- JavaScript has plenty of useful built in functions (and properties)
- Assuming you have a variable with some data stored inside of it, and are unaware of its data type…
- You can ask the variable for its datatype using the typeof( ) function
- Possible types: String, Number, Boolean, Object

```
var yourData = "This is my data.";
typeof(yourData);
>string
```

Accessing Array items

- If you're unsure of an items index number inside of an array

```
var cartoons = ['garfield', 'snoopy', 'popeye'];
var snoopyPosition = cartoons.indexOf('snoopy');
console.log(cartoons[snoopyPosition]);
>"Snoopy"
```

Array length

- How many items in an array?

```
var cartoons = ['garfield', 'snoopy', 'popeye'];
console.log(cartoons.length);
> 3
```

Loops

- A loop is a block of code that gets repeated for a defined amount of iterations (known as a "for loop") or until a certain condition is met (known as a "while loop")
- Typically one variable or condition in the loop changes each time it is run

Loops - `for loop`

```
for(var i = 0; i < 10; i++) {
  console.log(i)
}

>>0
1
2
3
...
9
```

Loops - `for loop`

```
var beers = ['Lagunitas', 'Peak']
for (var i = 0; i < beers.length; i++) {
  console.log(beers[i])
}

>>"Lagunitas"
"Peak"
```

Loops - `while loop`

```
var x = 6;
while(x < 10){
  console.log("On number " + x)
  x++;
}
```

```
>>On number 6
On number 7
On number 8
On number 9
```

Exercise - `for loop`

- Create an array filled with several strings of names
- Use a loop to print out the names, followed by 'is my friend'. Like so:

```
>>Jake is my friend
John is my friend
Samantha is my friend
Billy is my friend
```

Resources

Codecademy

JavaScript - Functions, 'For' Loops in JavaScript, 'While' Loops in JavaScript

TeamTreeHouse

JavaScript Loops, Arrays and Objects - Simply Repetitive Tasks with Loops