Introduction to JavaScript

Class 2

Welcome Ladies

Let's Look back at it!

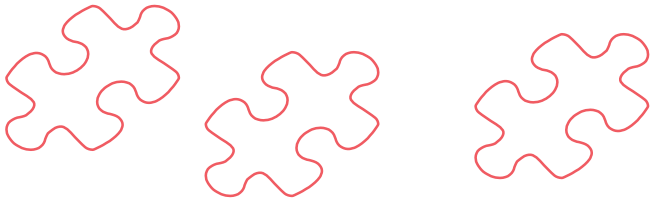
In the code spot the

- Comment
- Variables
- Operators

```
var billPreTip = 10;
var tipPercent = 0.15; // Can be changed

var billTip = billPreTip * tipPercent;
var receipt = 'Meal: ' + billPreTip + ' Tip: ' + billTip +
' Total: ' + (billPreTip + billTip);

console.log(receipt);
```

2

Functions



Functions are separable, reusable pieces of code.

Declaring Functions

- To declare (create) a function, you can give it a name, then include all the code for the function inside curly brackets { }
- Functions can have multiple lines.

```
function parrotFacts() {
  console.log('Some parrot species can live for over 80 years');
  console.log('Kakapos are a critically endangered flightless parrot');
}
```

Using Functions

- To invoke (use) a function, you type its name, followed by parenthesis `()`
- We'll talk about what can go inside those parenthesis in a minute! For now, leave them empty.

```
parrotFacts();
```

What's going on here?

- A function is a group of code you can reuse many times. Whenever you invoke a function by using it's name, you tell the browser to run the code inside the function.
- You must declare a function before you can use it.

Let's Develop It

- Write a function that outputs a sentence. Then invoke that function later in your code.

Arguments

- Functions can accept input values, called **`arguments`**.

```
function callKitten(kittenName){
  console.log('Come here, ' + kittenName + '!');
}

callKitten('Fluffy'); // outputs 'Come here, Fluffy!'

function addNumbers(a, b){
  console.log(a + b);
}

addNumbers(5, 7);  // outputs 12
addNumbers(9, 12); // outputs 21
```

Arguments

- You can also pass **variables** into functions. These variables do not need to have the same name as the function arguments.

```
function addOne(num){
  var newNumber = num + 1;
  console.log('You now have ' + newNumber);
}

// Declare variables
var numberOfKittens = 5;
var numberOfPuppies = 4;

// Use them in functions
addOne(numberOfKittens);
addOne(numberOfPuppies);
```

Let's Develop It

- Write a simple program to combine a first name and a last name inside a function. Then update the function to accept a first and last name as arguments.

Returning Values

- You can have a function give you back a value, to use later.
- `return` will immediately end a function.

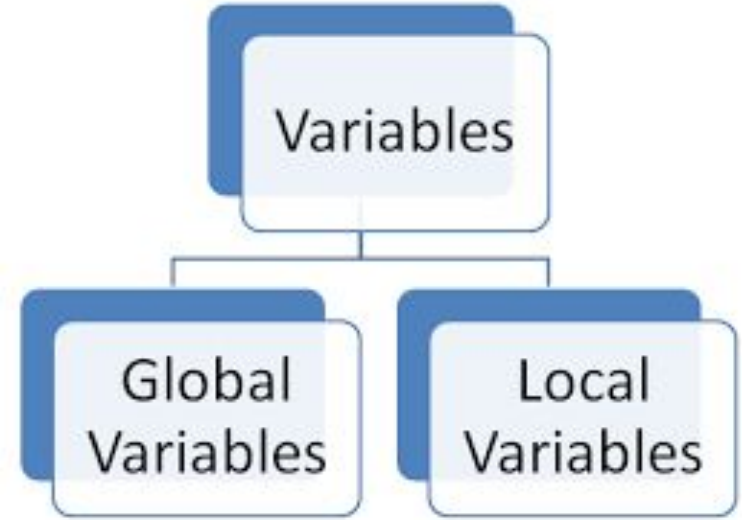
```
function square(num) {
  return num * num;
}

console.log(square(4));        // outputs '16'

var squareOfFive = square(5); // squareOfFive equals '25'
```

Let's Develop It

- Add a return statement to your 'name' function. Use that function to set the value of a variable.

Variable Scope

- The scope of a variable determines where it's value is accessible throughout your program.



Global Scope

- A variable declared outside of a function has a **`global scope`** and can be accessed anywhere, even inside of functions.

```
var awesomeGroup = 'Girl Develop It'; // Global scope

function whatIsAwesome() {
  console.log(awesomeGroup + ' is pretty awesome.'); // Will work
}

whatIsAwesome();
```

Local Scope

- A variable declared inside a function has a **local scope** and can only be accessed within that function.

```
function whatIsAwesome() {
  var awesomeGroup = 'Girl Develop It';                    //
Local scope
  console.log(awesomeGroup + ' is pretty awesome.'); //
Will work
}

whatIsAwesome();

console.log(awesomeGroup + ' is pretty awesome.'); // Won't
work
```

Boolean Variables

- Boolean variables represent the logical values `true` and `false`

```
var catsAreBest = true;
var dogsRule = false;
```

Boolean Variables

- Some values are considered **`falsy`** and will evaluate to **`false`** in a Boolean context.
- Boolean variables represent the logical values **`true`** and **`false`**

```
// the following variables will evaluate as false
var myName = '';
var numOfKids = 0;
var isMarried;      // remember a variable with no value is
undefined
```

`null` and **`NaN`** will also evaluate as **`false`**.

Everything else evaluates as **`true`**.

Control Flow

The if statement

- Use `if` statements to decide which lines of code to execute, based on a condition.

```
if (condition) {
  // statements to execute
}
```

```
var age = 30;
if (age > 18) {
  console.log('You are an adult');
}
```

Comparison Operators

Example	Name	Result
`a == b`	Equal	TRUE if `a` is equal to `b` (can be different types)
`a===b`	Identical	TRUE if `a` is equal to `b`, and the same type.
`a!=b`	Not equal	TRUE if `a` is not equal to `b` (can be different types).
`a!==b`	Not identical	TRUE if `a` is not equal to `b`, or they are not the same type.
`a < b`	Less than	TRUE if `a` is strictly less than `b`.
`a > b`	Greater than	TRUE if `a` is strictly greater than `b`.
`a <= b`	Less than or equal to	TRUE if `a` is less than or equal to `b`.
`a >= b`	Greater than or equal to	TRUE if `a`is greater than or equal to `b`.

WATCH OUT!

- Don't mix up `=` and `==` and `===`

Let's Develop It

- Make a variable called "temperature". Write some code that tells you to put on a coat if it is below 50 degrees.

The if/else statement

- Use `else` to provide an alternate set of instructions.

```
var age = 30;

if (age >= 16) {
  console.log('Yay, you can drive!');
} else {
  console.log('Sorry, you have ' + (16 - age) +
  ' years until you can drive.');
}
```

The if/else statement

- If you have multiple conditions, you can use `else if`.

```
var age = 30;

if (age >= 35) {
  console.log('You can vote AND run for President!');
} else if (age >= 30) {
  console.log('You can vote AND run for the Senate!');
} else if (age >= 18) {
  console.log('You can vote!');
} else {
  console.log('You can\'t vote, but you can write your
representatives.');
}
```

Let's Develop It

- Modify your "wear a coat" code for these conditions:
  1. If it's less than 50 degrees, wear a coat.
  2. If it's less than 30 degrees, wear a coat and a hat.
  3. If it's less than 0 degrees, stay inside.
  4. Otherwise, wear whatever you want.

Comparison Operators

Example	Name	Result
`a && b`	And	TRUE if both a and b are TRUE.
`a || b`	Addition	TRUE if either a or b is TRUE.
`!a`	Subtraction	Difference of `a` and `b`.

Using logical operators

- You can use these operators to combine conditions.

```
var age = 30;
var yearsAsCitizen = 30;

if (age >=30 && yearsAsCitizen > 9) {
  console.log('You can run for the Senate!');
} else {
  console.log('You are not eligible to run for the
Senate');
}
```

Let's Develop It

- Add a logical operator to your "what to wear" program.

Resources

- JavaScript Guide, from the Mozilla Developers Network.
- Code Academy, with interactive JavaScript lessons to help you review.

YOU DID IT!

Any questions?