Intro

JavaScript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. — you can bet that JavaScript is probably involved.

JavaScript enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else. (Okay, not everything, but it is amazing what you can achieve with a few lines of JavaScript code.)

Node JS

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command-line tools and for server-side scripting. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server-side and client-side scripts.

Coding Platform

https://replit.com/

Running Javascript Code

```
node filename.js
```

Variables:

We use programming languages like JavaScript to store and manipulate information. Variables in JavaScript are used to store different kinds of data. Think of a variable like a bottle. We use bottles to store water, in much the same way we use variables to store various forms of data in JavaScript.

Example:

```
var number = 200
```

Syntax: You can also declare multiple variables in one statement as seen in the syntax below.

Multiple Variables Example:

```
var a = 100, b = 200, c = 300;
```

The data inside variables is not constant. This means the data inside a variable can be changed.

Example:

```
var a = 200
a = 100
```

In the above example the variable called a first contained the value 200 but a = 100 means that a now contains the value 100.

NOTE: You can name variables whatever you want but try to give them good/descriptive names that tell the reader what the variable is used for

Data Types In Javascript:

There are 7 types of data in JavaScript but we will focus on 2 types of data for now.

Strings:

The first type of data is a String. This is used to store a sequence of characters used to represent text.

Example:

```
var name = "Masai School"
```

Any data within quotes " " is a String in JavaScript.

Numbers:

The second type of data we want to know is a Number, which is used to store any kind of numbers. We have already seen this type of data in the variables example. Numbers can store both Whole Numbers/Integers and Decimals.

Example:

```
var num = 100
var dec = 100.001
```

Checking the type of data: Lets say you have some data but you don't know what type it is. You can user the typeof() inbuilt code to find the type of the data.

Example:

```
var name = "Masai School"
console.log(typeof(name))
```

Output:

```
string
```

Mathematical operators in JavaScript

Common Operators

Javascript supports all the commonly used mathematical operators. Namely `+` `-` `*` `/`.

Example:

```
var a = 2
var b = 3
var c = a + b
var d = a * b
var e = a / b
var f = a - b
```

Output:

```
c = 5
d = 6
e = 0.6666666666666666
f = -1
```

`Modulo` or Remainder Operator:

Many programming languages including JavaScript have a modulo operator `%`. This operator returns the remainder when one variable is divided by another.

Example:

```
var a = 10 % 7
```

Output:

```
a = 3
```

This operator is often useful when you want to check if a number is odd or even.

Example:

```
var a = 10 % 2
var b = 11 % 2
```

Output:

```
a = 0
b = 1
```

Try this out for yourself, any even number `%2` returns 0 while any odd number `%2` returns 1.

`Exponentiation` operator:

This operator is represented by `**`. This returns the value of the first operand raised to the power of the second operand. For example 2^4 = 16.

Example:

```
var a = 2 ** 4
var b = 3 ** 2
var c = 10 ** 1.5
```

Output:

```
a = 16
b = 9
c = 31.622776601683793
```

String concatenation

A special property of `Strings` is that they can be combined or concatenated with one another.

Example:

```
var word1 = "Welcome"
var word2 = "Masai"
var word3 = word1 + " to " + word2 + " school!"
console.log(word3)
```

Output:

```
Welcome to Masai school!
```

Strings can also be combined with other types like `numbers`.

Example:

```
var num1 = 1
var num2 = 2
var output = "1 + 2 = " + (num1 + num2)
console.log(output)
```

Output:

```
1 + 2 = 3
```

Note: Notice the circular brackets between `num1 + num2` this tells javascript that we want to add the two numbers mathematically. Without the brackets the output would be `1 + 2 = 12`.

`Booleans`:

The last data type we are going to learn about is a `Boolean`. This data type has only two values `true` and `false`.

Example:

```
var x = true
var y = false
```

Relational operators

These operators allow you to test the relation between 2 values and returns a `boolean`. JavaScript unlike other languages allows you to compare any type with any other type!

Greater than and greater than equal to

The **greater than** operator `>` allows you to check if one value is greater than the other. It returns `true` if the first value is greater than the second and `false` if the second value is greater.

Example:

```
20 > 10
10 > 20
10 > 10
```

Output:

```
true
false
false
```

The **greater than equal to operator** `>=` also checks if the second value could be equal to the first value.

```
10 > 10
10 >= 10
```

Output:

```
false
true
```

Lesser than and lesser than equal to

The **lesser than** operator `>` allows you to check if one value is lesser than the other. It returns `false` if the first value is greater than the second and `true` if the second value is greater.

Example:

```
20 < 10
10 < 20
10 < 10
```

Output:

```
false
true
false
```

The **lesser than equal to operator** `<=` also checks if the second value could be equal to the first value.

```
10 < 10
10 <= 10
```

Output:

```
false
true
```

Comparison Operators

Equality

The **equality** operator `==` lets you test if two values are equal or not. It accepts 2 inputs of any type and outputs `true` if they are equal and `false` if the are not equal.

Example:

```
1 == 1
1 == 2
"Masai" == "Masai"
"Masai" == "masai"
```

Output:

```
true
false
true
false
```

Inequality Operator

The **inequality** operator `!=` performs the opposite function of the equality operator. It accepts 2 inputs of any type and outputs `false` if they are equal and `true` if the are not equal.

Example:

```
1 != 1
1 != 2
"Masai" != "Masai"
"Masai" != "masai"

1 != '1' // false
1 !== '1' // true
```

Similar to `===`, `!==` will check for type as well.

It is recommended to use `===` and `!==` when it comes to comparison operators

Output:

```
false
true
```

```
false
true
```