Data Types

**Variable**

**Explanation**

**Example**

String

A sequence of text known as a string. To signify that the value is a string, you must enclose it in quote marks.

let myVariable = 'Bob';

Number

A number. Numbers don't have quotes around them.

let myVariable = 10; but if you but in quotes it can be a string.

Boolean

A True/False value. The words true and false are special keywords in JS, and don't need quotes.

let myVariable = true, if but into quotes it is again a string

Array

A structure that allows you to store multiple values in one single reference.

1st item in an array is always 0.

let myVariable = [1,'Bob','Steve',10];  
Refer to each member of the array like this: Arrays are always [ ]  
myVariable[0], myVariable[1], etc.

Object

Basically, anything. Everything in JavaScript is an object and can be stored in a variable. Keep this in mind as you learn.

let myVariable = document.querySelector('h1');  
All of the above examples too.

**JavaScript VARIABLES**

• In a programming language, variables are used to store data values.

• JavaScript uses the var keyword to declare variables.

• An equal sign is used to assign values to variables.

In this example, x is defined as a variable. Then, x is assigned (given) the value 6:

var x;  
x = 6;

**JavaScript OPERATORS**

JavaScript uses arithmetic operators ( + - \* / ) to compute values

(5 + 6) \* 10

JavaScript uses an **assignment operator** ( = ) to **assign** values to variables:

var x, y;  
x = 5;  
y = 6;

**TYPES OF OPERATORS**

**Operator Description**

+ Addition

- Subtraction

\* Multiplication

\*\* Exponentiation (ES2016)

/ Division

% Modulus (Division Remainder)

++ Increment

-- Decrement

\' ' Single quote This is to put in a **‘** into an element rick/’s

\" " Double quote This is to put in a **“** into an element

\\ \ Backslash This is to put in a **\** into an element

**JavaScript ASSIGNMENT OPERATORS**

**Operator Example Same As**

= x = y x = y

+= x += y x = x + y

-= x -= y x = x - y

\*= x \*= y x = x \* y

/= x /= y x = x / y

%= x %= y x = x % y

\*\*= x \*\*= y x = x \*\* y

**JavaScript COMPARISON OPERATORS**

**Operator Description**

== equal to Loosely equal to

=== equal value and equal type

!= not equal Loosely not equal

!== not equal value or not equal type to strictly not equal to

> greater than

< less than

>= greater than or equal to

<= less than or equal to

? ternary operator conditional operator that assigns a value to a variable based on some condition SEE BELOW

variablename = (condition) ? value1 (if true) :value2 (if false)

if (condition) {

value1

} else {

Value2

}

Example  
var voteable = (age < 18) ? "Too young":" Old enough"; **MUST HAVE A COLON**

If the variable age is a value below 18, the value of the variable voteable will be "Too young", otherwise the value of voteable will be "Old enough"

JavaScript Logical Operators

**COMPLEX DATA=typeof note: the Of is not capitalized**

function

object

The typeof operator returns "object" for objects, arrays, and null.

The typeof operator does not return "object" for functions.

**Example**

typeof {name:'John', age:34} // Returns "object"

typeof [1,2,3,4] // Returns "object" (not "array", see note below)

typeof null // Returns "object"

typeof function myFunc(){} // Returns "function"

The typeof operator returns "object" for arrays because in JavaScript arrays are objects.

**JavaScript EXPRESSIONS**

An expression is a combination of values, variables, and operators, which computes to a value.

The computation is called an evaluation.

For example, 5 \* 10 evaluates to 50:  
5 \* 10  
Expressions can also contain variable values:

x \* 10

The values can be of various types, such as numbers and strings.

For example, "John" + " " + "Doe", evaluates to "John Doe":

"John" + " " + "Doe"

**JavaScript KEYWORDS**

JavaScript keywords are used to identify actions to be performed.

The var keyword tells the browser to create variables:

var x, y;  
x = 5 + 6;  
y = x \* 10;

**JavaScript IDENTIFIERS**

Identifiers are names.

In JavaScript, identifiers are used to name variables (and keywords, and functions, and labels).

The rules for legal names are much the same in most programming languages.

In JavaScript, the first character must be a letter, or an underscore (\_), or a dollar sign ($).

Subsequent characters may be letters, digits, underscores, or dollar signs.

**JavaScript FOR LOOPS**

for ([initialExpression]; [condition]; [incrementExpression])

statement

While statement

let i = 0;

do {

i += 1;

console.log(i);

} while (i < 5);

**JavaScript METHODS**

DEFINITION: Is an object within a function. Whenever ever you are using a function it must be user.speak   
IE

**JavaScript FUNCTION SYNTAX**

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().  
Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).  
The parentheses may include parameter names separated by commas:  
(parameter1, parameter2, ...)  
The code to be executed, by the function, is placed inside curly brackets: {}  
Example:  
function name(parameter1, parameter2, parameter3) {  
  // code to be executed Sample return parameter1 \* parameter2 \* parameter3; NOTE: closed semicolon  
}  
Function parameters are listed inside the parentheses () in the function definition.

Function arguments are the values received by the function when it is invoked.

Inside the function, the arguments (the parameters) behave as local variable

Function myFunction (parameters= arguments) {

What should the function do within the arguments

Why use Functions, you and run the exact same code to get different values.

Call ( ) takes arguments one by one separately

Function simpleFunction (x,y) {

Return x \* y;

}

Console.log(simpleFunction =x,y);

newArray = {22, 2];

simpleObject = simpleFunction.apply (simpleObject, newArray);

Apply ( ) makes it into an array

REALLY IMPORTANT WITH ANY JAVASCRIPT, YOU MUST DEFINE EXACTLY WHAT IS NEEDED, THAN FIGURE OUT HOW TO ACHIEVE IT WITH DIFFERENT EXPRESSIONS.

Here is how to think about Javascript.

• 1st think about what you are trying to do.

• Draw up a plan

• Write code

• Reflecting back on how we got there

In this case

• Unhide h1

• 1sec delay

• Unhide p

• H1 change text to “Hello”

• P change text to “World”

What you are doing is trying to tell the elements what to do.

All code in any code-base should look like a single person typed it, no matter how many people contributed.

Output codes

**Code**

**Output**

\’

Single Quote

For comments single line

\”

Double Quote

For multiple line needs closed out if continuing

\\

Backslash

= Notes will not run code.

\n

Newline

\r

Carriage Return

\t

Tab

\b

Backspace

\f

Form Feed

**Example** var myStr = “FirstLine\n ‘New line’\t ‘tab’\\SecondLine\nThirdLine”

10 % 2 =5 which = 0 because there is nothing remaining these would be ‘even’