# Javascript Fundamentals

## Statements

Statements make up the Javascript program, consisting of <i>*expressions*</i> and <i>*operations*</i> that perform a certain task, like the assignment of a value to a variable or execution of a function. Each new line is inexplicitly considered a new Javascript statement, but it is better to end statements explicitly using semicolons. Statements can be grouped into a <i>*block*</i> when wrapped in braces, which are generally used to with functions and conditionals.

<pre>

var greeting = "Hello world";

function sayHi() {

console.log(`${greeting}`);

} // output: Hello world

</pre>

## Variables

**var**

- global

- property of object (window)

- local variable if inside function

- breaks through blocks (variable can be called even outside of a block, except function)

- default value: undefined

**let**

- global

- not property of object

**const**

- global

- not property of object

- value cannot be changed after declaration

- usually written in uppercase letters

## Datatypes

**Primary Datatypes**

- number

- special: infinity, -infinity, nan

- calculating mathematical operations is safe, because dividing by zero still produced an output rather than halting the operation

- integer

- floating point

- string

- boolean

**Composite Datatypes**

- object

- array

**Special Datatypes**

- null

- undefined (also considered a *NaN*, or not a number)

\* value of undefined and null are equal

## Coercion

Javascript is a <i>*loosely typed*</i> language, meaning that variables do not have a predetermined type when declared. This allows almost any value to be assigned to any variable, and variables, if already declared, can be reassigned to values of another type.

<pre>

var num = 8;

num = "number";

num; // output: number

</pre>

<b>**Coercion**</b> is the conversion of data types before executing the attached operation.

<pre>

var num = 1;

var str = " more time";

var x = num+str; //the num variable is turned into a string, then concatenated with str

x; // output: 1 more time

</pre>

## Functions

Also referred to as “global methods”, it executes the operations or arguments inside the function.

<pre>

function alert() {

let warning = "trepasser!";

return warning;

}

</pre>

Functions can also become objects (that is, assigned to a variable) or parameters to other functions.

<pre>

var today = function (dd) {

return "Today is " + dd;

}

var greeting = function (day, fn) {

return tn(day);

}

greeting("tuesday", today); // output: today is tuesday

</pre>

## Operators

+ (add or concatenation)

++ (increment)

- (subtract)

\* (multiply)

/ (divide)

= (assignment)

!, !== (not equal)

== (equal: compare)

=== (strict equal: compare type and value)

..

# Resources/References

Mardan, A. (2013, Sept 25). JS Fundamentals: An Essential Overview of JS. Retrieved from: https://webapplog.com/js-fundamentals/.

(2017, Jan 18). JS Fundamentals. Retrieve from: https://docs.microsoft.com/en-us/scripting/javascript/javascript-fundamentals

https://javascript.info

https://developer.mozilla.org/en-US/docs/Learn/Getting\_started\_with\_the\_web/JavaScript\_basics