JavaScript Basics For Absolute Beginners

Mohammed Abdulhady

May 1, 2017

Contents

Chapter 1

Introduction

1.1 History of JavaScript

JavaScript was created in 1995 by Brendan Eich who worked at Netscape. JavaScript was originally named Mocha by Marc Andreessen, the founder of Netscape. The name was changed to LiveScript and then to JavaScript. It was some kind of a marketing move because of the popularity of Java at that time.

1.2 What is JavaScript?

JavaScript is an object-oriented programming language that is used to make web pages interactive and creating web applications. JavaScript is an interpreted language, it needs an interpreter to translate the codes and executes the program. The interpreter, which is the browser in our case, executes instructions directly line by line without previously compiling it into machine language.

Many people confused between Java (developed in the 1990s at Sun Microsystems) and JavaScript. They are completely different. Only the names are similar.

1.3 Why JavaScript

JavaScript is a very simple, powerful and popular programming language. It has become an essential technology along with HTML and CSS.
You will need to learn JavaScript if you want to get into web development, if you are planning to be a front-end or you can even use it for back-end development. As o this moment, JavaScript has extended to desktop apps development, mobile apps development and games development. It has become a very important and useful skill to master.

Chapter 2

Numbers

2.1 What Are Numbers?

Numbers are the values that you can use mathematical operations on.
There is not any special syntax for numbers in JavaScript. You can just write them straight into JavaScript.

Example

```
123456;
```

2.2 Decimals and fractions

Numbers in JavaScript can be written with, or without, decimals. Unlike any other programming language, JavaScript does not distinguish between numbers. It has only one type of numbers. So, you can use whole number and decimals together without worrying or having to convert from one to another.

Example

```
1 + 2.345;
```

Output

```
3.345
```

2.3 Negative Numbers

If you want to make any number negative, you can do that by placing the `-` operator in front of it.

Example

```
-100;
```

Output

```
-100
```

Also, you can get a negative number by subtracting a big number from a smaller one.

Example

```
3 - 100;
```

Output

```
-97
```

2.4 Arithmetic Operators

Operators are those symbols that you use between two or more values to perform different operations such as addition, subtraction and more.
In this book, we will focus on the ones that we think you will see most often.

2.4.1 Addition

the `+` operator is used to add two or more numbers.

Example

```
3 + 2 + 1;
```

Output

```
6
```

2.4.2 Subtraction

the `-` operator is used to subtract numbers from each others.

Example

```
3 - 2 - 1;
```

Output

```
0
```

2.4.3 Multiplication

the $\boxed{*}$ operator is used to multiply two or more numbers. notice that we use $\boxed{*}$ instead of $\boxed{\times}$ symbol that is commonly used in math.

Example

```
3 * 2;
```

Output

```
6
```

2.4.4 Division

the $\boxed{/}$ operator is used to divide numbers by numbers. notice that we use $\boxed{/}$ instead of $\boxed{\div}$ symbol that is commonly used in math.

Example

```
6 / 2;
```

Output

```
3
```

2.4.5 Modulus

the $\boxed{\%}$ operator is used to get the division remainder of two numbers.

Example

```
5 % 2;
```

Output

```
1
```

2.4.6 Increment

the $\boxed{++}$ operator is used to increment the number by 1.

Example

```
5++;
5+1;
```

Output

```
6
6
```

2.4.7 Decrement

the `--` operator is used to decrement the number by 1.

Example

```
5--;
5-1;
```

Output

```
4
4
```

2.5 Ordering and Grouping

JavaScript expressions follow the **order of operations** , so even if the `+` and `-` operators come first in the following example, the `*` and `/` operators will be performed first between the second and third number.

Example

```
5 + 10 * 2;
5 - 10 / 2;
```

Output

```
25
0
```

However, you can have more control over the order of operations using the grouping operator `()`.
Using these operators to group other values and operations will force JavaScript to perform the grouped operations first, no matter what the ordering is.

Example

```
(5 + 10) * 2;
(5 - 10) / 2;
```

Output

```
30
-2.5
```

Chapter 3

Strings

3.1 What Are Strings ?

Strings are values made up with series of characters. They can be made of letters, symbols, numbers or anything.

A JavaScript String must be contained inside a pair of double quotation `" "` or ever single quotation `' '`.

Example

```
'This is our string.';
"This is our string.";
```

3.2 Enclosing quotation marks

Let us say you want to use the quotation marks inside your string. You can not do this normally like any other character, because quotation marks are one of the **special characters** family.
To solve that problem, you have to easy options.
First option, you will need to use opposite quotation marks inside and outside. This means for each string that include single quotes needs to use double quotes and vice versa.

Example

```
"I've eaten an apple.";
'I said "I have eaten an apple"';
```

Second option, you can use a backslash \ right before each quote inside the string. This makes the JavaScript know that you want to use a **special characters**.

Example

```
'I\'ve eaten an apple.';
"I said \"I have eaten an apple\"";
```

3.3 Methods and properties

Strings in JavaScript have their own predefined list of operations you can perform to string. They are call "methods and properties".
Here are some of the most helpful and commonly used :-

3.3.1 Length

The string length is a property that returns the number of characters that the string has.

Example

```
"What is my length?".length;
```

Output

```
18
```

3.3.2 toLowerCase

The String toLowerCase() Method returns a copy of the string but with all capital letters converted to small letters.

Example

```
"I aM A StRiNg !!".toLowerCase();
```

Output

```
"i am a string !!"
```

3.3.3 toUpperCase

The String toUpperCase() Method returns a copy of the string but with all small letters converted to capital letters.

Example

```
"I aM A StRiNg !!".toUpperCase();
```

Output

```
"I AM A STRING !!"
```

3.3.4 Trim

The trim() method cuts off any whitespaces at the begin and end of the string.

Example

```
"     Trim me, please  ".trim();
```

Output

```
"Trim me, please"
```

3.3.5 Replace

The String replace() method replaces a specified value with another value in a string and returns the result.

Example

```
"I love red oranges".replace("oranges", "apples");
```

Output

```
"I love red apples"
```

Chapter 4

Booleans

Booleans are the values that can only be one of two things: either `true` or `false`.
It's really useful to store booleans in **variables** to keep track of their value and change them as you like.

Example

```
var flag = true;

flag = false;

flag;
```

Output

```
false
```

Booleans are also essential for **conditional** work. We will discuss that later.

Chapter 5

Variables

Variables are named values that we use to store any type of JavaScript Value. The following example is how to declare a variable:-W

```
Example

var myNumber = 100;
```

And this is what is happening in the example above:-

- var is a keyword you use to declare a variable.
- myNumber is the name of the variable.
- = is an operator used for assigning the value to the variable.
- 100 is the value you want to store at the variable.

5.1 Using Variables

After declaring your variable, you can reference it by its name anywhere after the declaration in your code for further use.

```
Example

var myNumber = 100;

myNumber + 50;

Output

150
```

You can even use variable while declaring another one.

Example

```
var x = 100;

var y = x + 50;

y;
```

Output

```
150
```

5.2 Reassigning Variables

What if you needed to change a value of a variable you have declared before ?! is it possible ?!
Well, of course it is !!

Example

```
var color = "red";

color = "blue";

color;
```

Output

```
"blue"
```

5.3 Naming Variables

Naming variable is really easy and flexible as long as you follow these simple rules:-

- You can only start them with a letter, underscore, or a dollar sign.
- After the first letter, you can use numbers as you want.
- You can not use any JavaScript **reserved** keyword.

Keeping that in your mind, these are some valid variable names.

Example

```
var camelCase = "first word lowercase, then everything goes uppercase.";

var book2write = "My JavaScript mini-book for absolute beginners.";

var I_FEEL_EXCITED = false;

var $_$ = "Money eyes";

var _1000000$_ = "I wish that was my salary";
```

And these are some invalid variable name. Can you try to tell what is wrong with each ?

Example

```
var total% = 100;

var 2books2write = false;

var function = "myFunction";

var money+money = "a lot of money";
```

In JavaScript, Variable names are case-sensitive, so myVar , MyVar , MYVAR , myvar are all different variables. However, it is a good practice to avoid naming variables so similarly. Things could really get mixed up.

Chapter 6

Functions

Variables are named blocks of code that you can call by its name and reuse anywhere where you like.
This is how to declare a function:-

```
Example

function addNumbers (x, y) {
   return x + y;
}
```

A lot is going on in the example above. So, let us have a look at each part separately.

- $\boxed{\text{function}}$ is a keyword for declaring a function.

- $\boxed{\text{addNumbers}}$ is the name of the function, which is customizable just like **reserved** names.

- $\boxed{\text{(x, y)}}$ are parameters that you pass to the function to perform different operation on. **return** is a keyword that exits the function and share only 1 value outside.

  In our case the **return** will share sum of **x** and **y**.

6.1 Using Functions

Once you define your function, you can call it by referencing its name with parentheses () right after it. Notice that the following function does nt have any parameters.

Example

```
function sayHello() {
  return "Hello !!";
}

sayHello();
```

Output

```
"Hello !!"
```

If the function has any parameters, you will need to pass the values that will be represented by the parameters names inside the function.

Example

```
function sayHello(name) {
  return "Hello, " + name;
}
sayHello("Mohamed");
```

Output

```
"Hello, Mohamed"
```

Chapter 7

Conditionals

Conditionals can control the behavior of your code. It determine whether or not a block of code runs according to a specified condition.

7.1 If

Conditionals statement is the most common type of conditionals. It the code inside it only runs if the condition inside its parentheses is **true**.
This is how an If statement should look like:-

Example

```
if (1 < 10) {
   var output = "Code Block";
}
output;
```

Output

```
"Code Block"
```

Pretty sure you understand it all. But, let us discuss every part just in case you missed something.

- if is the keyword to start the conditional statement.
- $(1 < 10)$ is the condition, which is true in this case.
- The curly braces {} are the containers of the code block.
- Because our condition is true, the variable output will be assigned to the value "Code Block"

7.2 else

However, you can extend your `if` statement with an `else` statement, which runs only if the condition was **false**:-

Example

```
if (false) {
   var output = "If Block";
} else {
   var output = "Else Block";
}
output;
```

Output

```
"Else Block"
```

You can also use multiple `else if` statements, if you have more conditions to use.

Example

```
if (1 > 5) {
   var output = "If Block";
} else if (10 > 5) {
   var output = "First Else If Block";
} else if (100 > 5) {
   var output = "Second Else If Block";
} else {
   var output = "Else Block";
}
output;
```

Output

```
"First Else If Block"
```

Note that only the first `else if` statement that passes the condition runs, regardless of the following statements.

Chapter 8

Arrays

Arrays are containers that can hold multiple values inside. These values call **elements**.

Example

```
var food = ["meat", "rice"];
food;
```

Output

```
["meat", "rice"]
```

In JavaScript, array elements do not have to be the same value type. They can even be another array.

Example

```
var items = ["book", 100, false, [5, "pen"], 1.5];
items;
```

Output

```
["book", 100, false, [5, "pen"], 1.5]
```

8.1 Accessing Elements

To access elements inside an array, you need to use the array name followed by square brackets and the number of the element inside the brackets.
Keep in mind that JavaScript numbering start with 0 not 1. So, the first element is number 0.

Example

```
var colors = ["black", "white"];
colors[0];
```

Output

```
"black"
```

This can also work for setting values to to elements, not just getting them.

Example

```
var colors = ["black", "white", "gren"];
colors[2] = "green";
colors;
```

Output

```
["black", "white", "green"]
```

8.2 Methods And Properties

Arrays also have their own predefined list of operations you can perform to string. They are call "methods and properties".
Here are some of the most helpful and commonly used :-

8.2.1 length

The length property is used to get the number of elements inside an array.

Example

```
[1, 2, 3].length;
```

Output

```
3
```

8.2.2 concat()

concat() is a method that returns a new array that combines the values of the other two array.

Example

```
["red"].concat(["green", "blue"]);
```

Output

```
["red", "green", "blue"]
```

8.2.3 push()

push() is a method that appends an element to the end of an array and returns the new length.

Example

```
["red", "green", "blue"].push("yellow");
```

Output

```
4
```

8.2.4 pop()

pop() is a method that removes the last element from the array and returns that element.

Example

```
["red", "green", "blue", "yellow"].pop();
```

Output

```
"yellow"
```

8.2.5 reverse()

reverse() is a method that returns the array in an opposite order.

Example

```
["red", "green", "blue", "yellow"].reverse();
```

Output

```
["yellow", "blue", "green", "red"]
```

Chapter 9

Objects

Objects are containers that can contain multiple values of multiple types using **keys** to name these values.

And this is how Object look like in JavaScript:-

```
Example

var student = {
   name:  "John Cena",
   age:  21,
   grade:  3
};
```

9.1 Getting Values

To get a value using keys, you have two options:-
You can use a dot `.` notation.

```
Example

var student = {
   name:  "John Cena",
   age:  21,
   grade:  3
};
student.name;

Output
"John Cena"
```

Or, you can use the square brackets and the double quotations `[""]` with the key inside them.

Example

```
var student = {
   name:  "John Cena",
   age:  21,
   grade:  3
};
student["name"];
```

Output

```
"John Cena"
```

9.2 Setting Values

To set values or overwrite them using keys. you can use the two options we discussed before.

Example

```
var student = {
   name:  "John Cena",
   age:  21,
   grade:  3
};
student["name"] = "Mohammed";
student.age = 23;
```

Output

```
{
   "name":  "Mohammed",
   "age":  23,
   "grade":  3
};
```

Chapter 10

Outro

As for now, you can do some very simple app but you still can not really build these awesome applications you had in mind, YET. However, you should be familiar with JavaScript whenever you see it.

We have not really got to the real JavaScript. That was just a small talk about the basics of JavaScript and pretty much every other high level programming languages. There is much more going on.

If you like JavaScript by far (and I am sure you do now) you can start digging through the Internet to get deeply into the real JavaScript. There are many website that will really help you start. I personally recommend you **The W3School** or **The MDN Website**. They are covering almost every this about JavaScript and many other web-related programming languages.

Bibliography

[1] JavaScript Website.
`https://www.javascript.com/learn/javascript`

[2] W3School.
`https://www.w3schools.com/js`

[3] Mozilla Developer Network.
`https://developer.mozilla.org/en-US/docs/Web/JavaScript`