

DEVOPS

brings you closer to software!

00 00 00

---

From “Hello, World” and beyond

# JavaScript cheat sheet: Tips & tricks

## What is JavaScript?

Briefly, it is a powerful language used for programming on the web. While HTML and CSS take care for defining the content skeleton of the web page and specifying its layout respectively, JavaScript is responsible for programming its behavior. It is a high-level interpreted runtime language that supports imperative, object-oriented and functional programming styles. It was created in 1995 by Brendan Eich. Soon, in 1997 it was standardized in the ECMAScript language specification.

## Hello World example

I will begin with the well-established tradition of acquaintance with a new programming language, namely how to write Hello World. You can display this message in the following ways:

- By using an **alert box** that pop-ups in the browser window

```
alert("Hello world!");
```

- By using the **browser console**

```
console.log("Hello world!");
```

- By writing into the HTML output using **document.write()**.

```
document.write("Hello world!");
```

- By writing into an HTML element, using **innerHTML**.

```
document.getElementById("para").innerHTML = "Hello world!";
```

[SEE MORE: Mastering blockchain: Meet Lisk, a blockchain platform for JavaScript developers](#)

You can also write these statements in two places:

- Between `<script>` and `</script>` tags, inserted directly in HTML. This script can be placed in the `<body>`, in the `<head>` section of an HTML page, or in both.
- External JavaScript file, which is useful, when you have to use it in a lot of different pages. You should simply specify the name of the script file. All JavaScript files have .js extension. `<script src="script.js"></script>`

## Common mistakes

In this section, I will talk about the most common mistakes people usually make and how to avoid them.

### Forgetting about JavaScript block-level scope

One of the most noticeable quirks of JavaScript is that it does not create a new scope for each code block. Consider the following code:

```
1 | for ( var i = 0 ; i < 10 ; i ++ ) {  
2 |   // some code  
3 | }  
4 | console . log ( i ); // What will be logged here?
```

Most of the experienced developers will say that the output would be `null`, `undefined` or an `error`. All of them are wrong, although this is true in many other languages. The actual output for the `i` variable is `10`. This is so because the scope of the variable is not restricted to the `for` loop scope and its value remains as it was during the last loop iteration. You can solve this problem by using the `let` keyword instead of `var`. It will overcome the upper problem and the code will act as it is in many languages. You can find more information about `let` [here](#).

### Undefined is not null

JavaScript uses both `undefined` and `null`, but they are used in different context. `Null` is default value for objects, while `undefined` is for variables and properties. For an object to be `null`, it must be defined, otherwise it will be `undefined`. The following code depicts this case.

```
1 | if ( obj !== null && typeof obj !== "undefined" ) // throws an error, obj  
2 | if ( typeof myObj !== "undefined" && myObj !== null ) //
```

### Equality

Unlike some other languages, JavaScript has an additional operator for comparison: `===`. The basic difference between `==` and `===` is called type coercion (More information [here](#)). Shortly, it means that when the operands of an operator are different types, one of them will be converted according to ECMA-262 specification.

When you use `==`, values may be considered equal, even if they are different types, since the operator will force coercion of one or both operators into a single type before performing a comparison. For example:

```

1 console . log ( 5 == '5' ); // Output: true. Both 5 and '5' are coerced to
2 console . log ( null == undefined ); // Output: true. Both null and 'unde
3 console . log ( '' == 0 ); // Output: true. Both '' and 0 are coerced to
4 console . log ( '0' == false ); // Output: true. '0' is coerced to false

```

However, if you use the non-converting comparison operator `===`, no conversion occurs. In other words, it only compares the values when they are of the same type. If the operands are of different types the answer is always false.

```

1 console . log ( 7 === 7 ); // Output: true
2 console . log ( 4 === '4' ); // Output: false.

```

## “This” keyword

“This” keyword is one of the strangest and confusing things for both new and skilled JavaScript developers. In JavaScript ‘this’ is the current execution context of a function. It is a reference to the object which calls the function.

Consider the following example:

```

1 window . name = "Super Window" ;
2
3 var mark = { age : 21 , name : "Mark" };
4 var sayHi = function () {
5   return "Hi, I am " + this . name ;
6 };
7
8 mark . sayHi = sayHi ;
9 sayHi (); // Hi, I am Super Window
10 mark . sayHi (); // Hi, I am Mark

```

When we call `sayHi()` the execution context is the global object. In this case it is window. It will be important and useful to remember that the value of `this` used in a function is the object that “owns” the function. The value of `this` used in an object is the object itself. This is not a variable. It is a keyword. You cannot change the value of `this`. (More information [here](#).)

As much as I write about the most common mistakes, they will never end up. I recommend you to look at the references about common mistakes [here](#) and [here](#) for further information.

**SEE MORE: Bad maintenance means your favorite JavaScript libraries might be vulnerable**

## Regular expressions

Can you imagine if you had to search for data in a text and to have something to describe what you are searching for? As you already know, regular expressions will do a great job for your task. They are patterns (sequence of characters) used to match character combinations in strings. This sequence forms a *search pattern* which can be used for operations like *text search* and *replace*.

You can create regular expressions in two ways:

1. Using a literal, which consists of a pattern enclosed between slashes and a modifier.

```
var pattern = /dreamix/ i ; // The template is / pattern / modifiers;
```

2. Calling the constructor function of the RegExp object, as follows:

```
var regExpr = new RegExp ( 'dreamix' );
```

In the first example `/dreamix/i` is a regular expression, `dreamix` is a pattern which is used in the search and `i` is a modifier (modifies the search to be case-insensitive).

## Working with regular expressions

You must have noticed the “modifier” word in the previous bullet. Modifiers are used to perform different type of searching. (For example: `i` – case-insensitive matching; `g` – all matches without stopping after the first match; `m` – multiline matching.)

Also, there are special characters that have special value when they are used in regular expression. You can check their full list for better understanding.

There are two sets of methods to deal with regular expressions. The first set is composed by so called String methods – `match()`, `replace()`, `search()`, and `split()`. The second one consists of `test()` and `exec()` and are used with `RegExp`. Examples:

```
1  var str = "The best company is Dreamix";
2
3  var n = str . search ( /dreamix/ i ); // Output: 20. Returns the index c
4
5  var repl = str . replace ( /company/ i , "company in the world" ); // Ou
6
7  var res = str . match ( /company/ g ); // Output: ["company"]. Returns a
8
9  var res = str . split ( /\s/ ); // Output: ["The", "best", "company", "i
10
11 var res = /^The/ . test ( str ) // Output: true. Tests for a match in a
12
13 var res = /best/ . exec ( str ); // Output: ["best", index: 4, input: "I
```

If you are interested in regular expressions you can check the following pages at the Mozilla Development network on [RegExo](#) and [Regular Expressions](#) for more details.

## JSON

Do you know that the data which is exchanged between a client (web-browser) and a server can only be text? Did you remember that there is a flexible format which can convert any JavaScript object into text and send it to server, and vice versa. The solution here is JSON (JavaScript Object Notation). It is a file format for transmitting data which is built on two structures:

- A collection of *key-value* pairs. This collection can be considered as object, dictionary, hash table or associative array in most programming languages.
- A *list* of values. In various languages this is called array, vector or list.

Example:

```
1  {
2  "companyName" : "Dreamix",
3  "address" :{ "city" : "Sofia",
4  "country" : "Bulgaria"
5  },
6  "website" : "http://dreamix.eu",
7  "keywords" :[ "Java EE", "AngularJS", "Oracle ADF", "Oracle SOA And BPM"
8  }
```

As you see from the example above all of the keys and string values are put in *double quotes*. The values must be only `String`, `Number`, `Object`, `Array`, `true`, `false` or `null`.

[SEE MORE: JavaScript as a smart contract language](#)

## JavaScript object – JSON and JSON – JavaScript object

```
1 | var company = { "name" : "Dreamix" , "city" : "Sofia" };
2 | var companyJSON = JSON . stringify ( company ); // Output: {"name":"John"}
```

Note that `JSON.stringify(obj)` function is used for converting a JavaScript object into a string following JSON notation.

```
1 | var companyJSON = '{ "name":"Dreamix", "city":"Sofia" }';
2 | var obj = JSON . parse ( companyJSON );
3 | console . log ( obj . name ); // Output: Dreamix
```

In this case `JSON.parse(json)` is used for converting a JSON string to a valid JavaScript object. The main benefit of JSON is that it is a *platform and language independent*. It is easy for machines to generate to parse it and for people to read or edit it. It is more useful than XML, because JSON can be parsed by a simple JS function, while you will need a XML parser for XML. What is more, JSON is shorter, does not use tags and it is quicker to read and understand.

## Interesting JavaScript plugins and libraries

In general, the strength of a programming language comes from all of its plugins and libraries which are developed for it. In JavaScript, there are great amounts and most of them are useful and offer lightweight and problem-solving solutions that make the web development process much easier and faster for every JS developer.

**Leaflet.js** is an open-source JavaScript library for mobile-friendly interactive maps. It uses OpenLayers and the Google Maps API and it has become one of the most popular JavaScript mapping libraries and is used by major web sites such as FourSquare, Pinterest and Flickr. (More Information and examples [here](#).)

**Chart.js** is a powerful library, which is responsible for rendering different types of charts in the browser. It is perfect for small projects – when you need clean and elegant javascript charts. It includes 6 core chart types (line, bar, radar, polar, pie and doughnut), separated in modules. This means that you can load only the module you need. The entire chart is responsive. (More Information [here](#) and on [GitHub](#).)

[SEE MORE: How to convert Java apps to JavaScript with CheerpJ](#)

**Flexdatalist** is a jQuery plugin for creating autocomplete input fields with support for

**Tabulator** is an easy to use interactive table generation plugin for jQuery. It allows you to create interactive tables in seconds from any HTML Table, Javascript Array, AJAX data source or JSON formatted data. It comes with a complete set of CSS classes to make a perfect styling. (More information and examples [here](#).)

**Marginotes** takes your jQuery selection and adds notes to the margin with the text provided in HTML attributes. If you don't want to use jQuery, there's also a version of marginotes without it. (Examples and source code [here](#).)

**Philter** is a JS plugin giving you the power to control CSS filters with HTML attributes. It is available as either a jQuery plugin or as vanilla JavaScript. (For examples visit its official site [here](#).)

**D3.js** is a library for producing dynamic, interactive data visualizations in web browsers. It helps you bring data to life using HTML, SVG, and CSS. The latest version changed the monolithic conception and it is separated to multiple modules that can be included in case you need one or more of them. (More information and examples [here](#) and [here](#).)

**SEE MORE: [How well do you know your JavaScript trivia?](#)**

As you can guess this is not the full list of all JS libraries and plugins. It is just a quick overview of some of them and the things they can do. More libraries are available in the references section.

## Conclusion

While we have to come to the end of the JavaScript cheat sheet, this definitely does not mean that I have told everything about the language characteristics, libraries or mistakes. There are so many to cover and we only had time for a quick overview. I hope you learned something new which will be helpful for your everyday work.

## Further references

[Top 50 JavaScript Plugins & Libraries for 2017](#)

[Top 10 Trending Javascript & jQuery Plugins in 2017](#)

[50 Amazing jQuery Plugins That You Should Start Using Right Now](#)

[Best JS Plugins & Libraries of 2015](#)

[17 brilliant jQuery plugins](#)

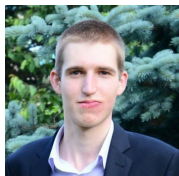
[The 50 Most Useful jQuery Plugins for Frontend Development](#)

**Be the first to share this article with your network!**



---

### Author



 Kostadin Hamanov

Kostadin Hamanov is Java and Oracle Developer at [Dreamix](#), a bespoke software development company. He studies Software Engineering at Sofia University and he describes himself as a highly motivated young person who pursues his dream for professional success and personal development.

---

---

### Recommended For You




# Add some candy to your code: JSweet transpiles from Java to JavaScript

## Leave a Reply

2 Comments on "JavaScript cheat sheet: Tips & tricks"

Join the discussion

 Subscribe ▾

 Guest

Khushbu Chawhan



Usually I never comment on blogs but your article is so convincing that I never stop myself to say something about it. You’re doing a great job ,Keep it up.

Reply

 11 months ago

 Guest

Deepak



great article. very useful to understand the concepts.-Scientechn Easy

Reply

 7 months ago

[Java](#)[DevOps](#)[Machine Learning](#)[Serverless](#)[Blockchain](#)[JavaScript](#)[NetBeans](#)[Careers](#)[Tutorials](#)[Contact](#)[Newsletter](#)[Authors](#)[Found a bug?](#)[Advertise](#)[Privacy Policy](#)[Terms of Use](#)[Imprint](#)[Twitter](#)[Facebook](#)[RSS](#)[JAXenter.de](#)[JAX London](#)[JAX Germany](#)[DevOpsCon](#)[International PHP  
Conference](#)[Webinare](#)[S&S Media](#)

**Software & Support Media Group | [Contact](#) | [Masterclass Terms & Conditions](#)**