



Frontend development course

JavaScript

Author: Daniel Chabr

What is JavaScript?

JavaScript is a cross-platform, object-oriented scripting language. It is a small and lightweight language. Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them.

JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements. Core JavaScript can be extended for a variety of purposes by supplementing it with additional objects; for example:

- Client-side JavaScript extends the core language by supplying objects to control a browser and its Document Object Model (DOM). For example, client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation.
- Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server. For example, server-side extensions allow an application to communicate with a database, provide continuity of information from one invocation to another of the application, or perform file manipulations on a server.

Even though JavaScript has similar syntax to Java, the two languages are to be considered completely separate. The naming resemblance of Java is just a coincidence.

ECMAScript Specification

- JavaScript is standardized at Ecma International — the European association for standardizing information and communication systems (ECMA was formerly an acronym for the European Computer Manufacturers Association) to deliver a standardized, international programming language based on JavaScript
- This standardized version of JavaScript, called ECMAScript, behaves the same way in all applications that support the standard. Companies can use the open standard language to develop their implementation of JavaScript

ECMAScript Specification

3	December 1999	Added regular expressions, better string handling, new control statements, try/catch exception handling, tighter definition of errors, formatting for numeric output and other enhancements	Mike Cowlishaw
4	Abandoned	Fourth Edition was abandoned, due to political differences concerning language complexity. Many features proposed for the Fourth Edition have been completely dropped; some are proposed for ECMAScript Harmony.	
5	December 2009	Adds "strict mode", a subset intended to provide more thorough error checking and avoid error-prone constructs. Clarifies many ambiguities in the 3rd edition specification, and accommodates behaviour of real-world implementations that differed consistently from that specification. Adds some new features, such as getters and setters, library support for JSON, and more complete reflection on object properties. ^[10]	Pratap Lakshman, Allen Wirfs-Brock
5.1	June 2011	This edition 5.1 of the ECMAScript Standard is fully aligned with third edition of the international standard ISO/IEC 16262:2011.	Pratap Lakshman, Allen Wirfs-Brock
6	June 2015 ^[11]	The Sixth Edition, known as ECMAScript 2015, ^[12] adds significant new syntax for writing complex applications, including classes and modules, but defines them semantically in the same terms as ECMAScript 5 strict mode. Other new features include iterators and for/of loops, Python-style generators and generator expressions, arrow functions, binary data, typed arrays, collections (maps, sets and weak maps), promises, number and math enhancements, reflection, and proxies (metaprogramming for virtual objects and wrappers). As the first "ECMAScript Harmony" specification, it is also known as "ES6 Harmony".	Allen Wirfs-Brock

Host Environment

- Unlike most programming languages, the JavaScript language has no concept of input or output. It is designed to run as a scripting language in a host environment, and it is up to the host environment to provide mechanisms for communicating with the outside world
- The most common host environment is the browser, but JavaScript interpreters can also be found in a huge list of other places, including Adobe Acrobat, Adobe Photoshop, SVG images, Yahoo's Widget engine, server-side environments such as Node.js, NoSQL databases like the open source Apache CouchDB, embedded computers, complete desktop environments like GNOME

Web Console

- The Web Console shows you information about the currently loaded Web page, and also includes a command line that you can use to execute JavaScript expressions in the current page.
- To open web console press (Ctrl+Shift+K) in Firefox or (Ctrl+Alt+I) in Chrome and select the Console tab
- For longer snippets of code you can use Scratchpad in Firefox (Shift+F4) or Snippets in Chrome (Sources tab -> Snippets tab on the left)

Syntax

- JavaScript is case-sensitive
- instructions are called statements and are separated by a semicolon (;)
- Spaces, tabs and newline characters are called whitespace

Comments

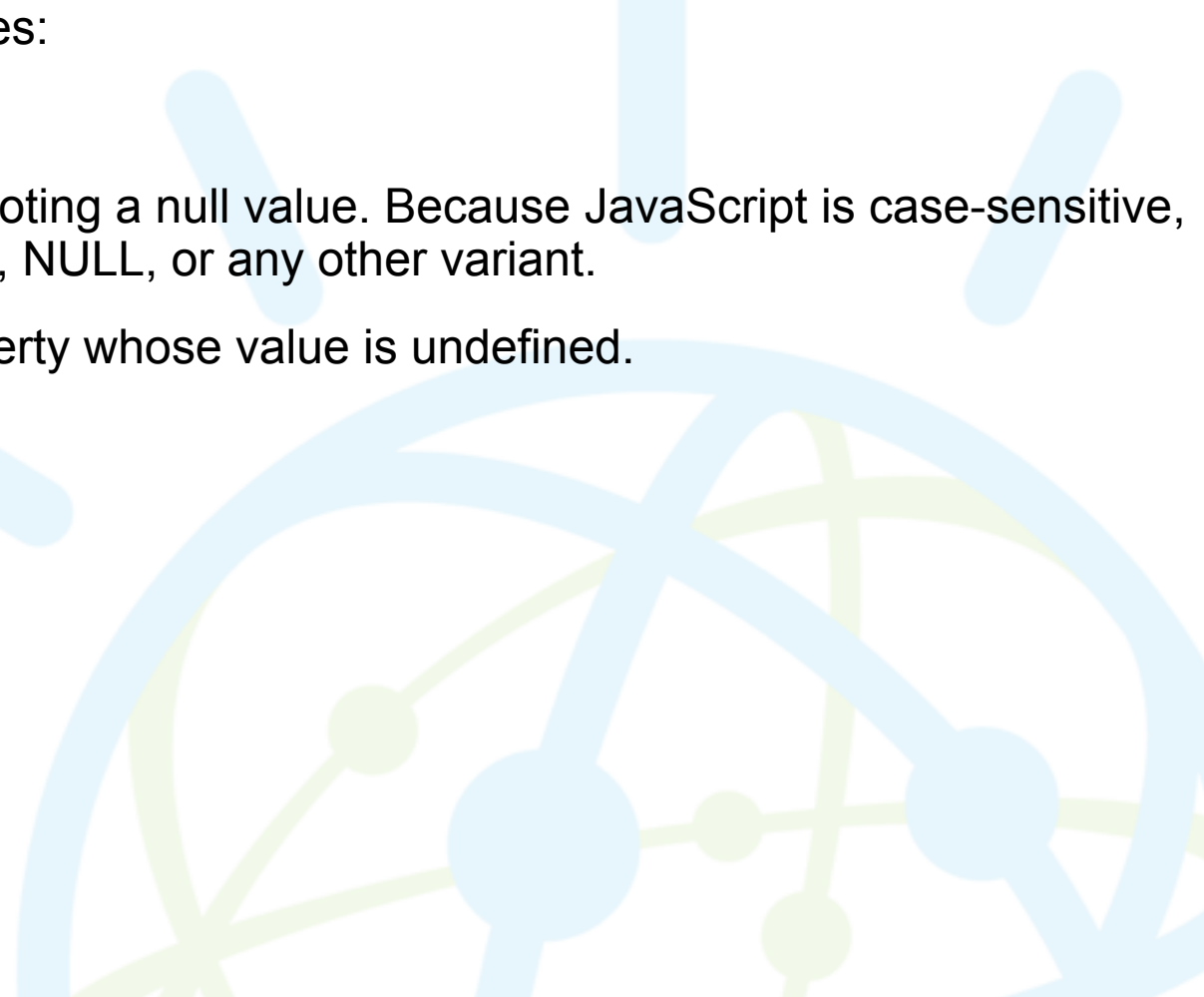
The syntax of comments is the same as in C++ and in many other languages

```
1 // a one line comment
2
3 /* this is a longer,
4    multi-line comment
5    */
6
7 /* You can't, however, /* nest comments */ SyntaxError */
```


Declarations

- **var** - Declares a variable, optionally initializing it to a value.
- ECMA6 - **let** - Declares a block scope local variable, optionally initializing it to a value.
- ECMA6 - **const** - Declares a read-only named constant.
- for example, `var x = 42`. This syntax can be used to declare both **local and global** variables.
- by simply assigning it a value. For example, `x = 42`. This **always declares a global variable**. It generates a strict JavaScript warning. You **shouldn't use this variant**.
- A variable declared using the `var` or `let` statement with no initial value specified has the **value undefined**.

Data Types

- Six data types that are primitives:
 - Boolean. true and false.
 - null. A special keyword denoting a null value. Because JavaScript is case-sensitive, null is not the same as Null, NULL, or any other variant.
 - undefined. A top-level property whose value is undefined.
 - Number. 42 or 3.14159.
 - String. "Howdy"
 - and Object
 - Function
 - Array
 - Date
 - RegExp
- 

Data Type Conversions

```
var answer = 42;  
answer = "Thanks for all the fish...";
```

Numbers to strings

```
x = "The answer is " + 42 // "The answer is 42"  
"37" + 7 // "377"
```

Strings to numbers

```
parseInt("54") // 54  
parseFloat("1.1") // 1.1  
"1.1" + "1.1" // "1.11.1"  
(+"1.1") + (+ "1.1") // 2.2  
"37" - 7 // 30
```

Quiz: what does `3 + 4 + "5"`; evaluate to?

Strings

```
"hello".length; // 5
```

```
"hello".charAt(0); // "h"
```

```
"hello, world".replace("hello", "goodbye"); // "goodbye, world"
```

```
"hello".toUpperCase(); // "HELLO"
```



Arrays

```
var coffees = ["French Roast", "Colombian", "Kona"];
```

- we can also mix data types

```
var myList = ['home', 3, 'school'];
```

- elements are then accessed by index, which starts at zero

```
coffees[1]          // "Colombian"
```

A deprecated way for creating objects is `var coffees = new Array();`

Objects

An object literal is a list of zero or more pairs of property names and associated values of an object

```
var car = {  
  brand: "Chrysler",  
  owner: "James",  
  color: "black",  
  id: 452846  
};  
car.brand // "Chrysler"  
car.type = "pickup";  
car.type  // "pickup"  
car["owner"] // "James"
```

A deprecated way for creating objects is `var car = new Object();`

Conditional Statements

if...else statement

Use the if statement to execute a statement if a logical condition is true. Use the optional else clause to execute a statement if the condition is false.

```
if (condition_1) {  
    statement_1;  
} else if (condition_2) {  
    statement_2;  
} else if (condition_n) {  
    statement_n;  
} else {  
    statement_last;  
}
```

switch statement

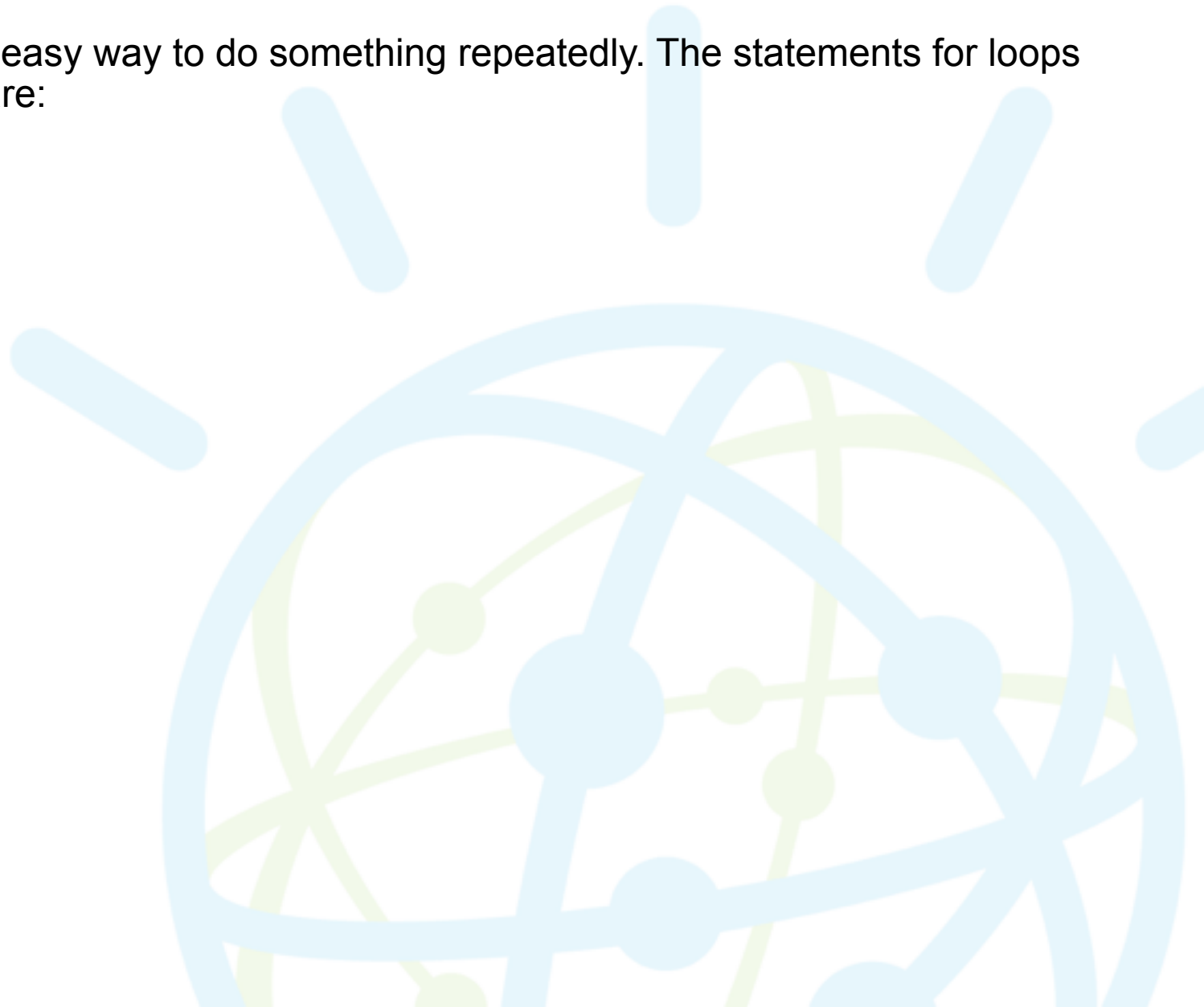
A switch statement allows a program to evaluate an expression and attempt to match the expression's value to a case label. If a match is found, the program executes the associated statement.

```
switch (expression) {  
    case label_1:  
        statements_1  
        [break;]  
    case label_2:  
        statements_2  
        [break;]  
    ...  
    default:  
        statements_def  
        [break;]  
}
```

Loops and iterating

Loops offer a quick and easy way to do something repeatedly. The statements for loops provided in JavaScript are:

- for statement
- do...while statement
- while statement
- label statement
- break statement
- continue statement
- for...in statement
- for...of statement



for and while loops

for statement

```
var text = "Ahoj";  
for (var i = 0; i <  
text.length; i++) {  
  
    console.log( text.charAt(i)  
    + " ");  
}
```

1. first `var i = 0;` is evaluated
2. condition `i < text.length` is checked, if it evaluates as false execution skips to end of for block
3. statements inside are executed
4. the increment expression `i++` is evaluated and execution goes back to step 2.

while statement

A while statement executes its statements as long as a specified condition evaluates to true.

```
n = 0;  
x = 0;  
while (n < 3) {  
    n++;  
    x += n;  
}
```

Functions

A function definition (also called a function declaration, or function statement) consists of the function keyword, followed by:

- The name of the function.
- A list of arguments to the function, enclosed in parentheses and separated by commas.
- The JavaScript statements that define the function, enclosed in curly brackets, { }.

```
function square(number) {  
    return number * number;  
}
```

```
var x = square(4) // x gets the value 16
```

```
var square = function(number) { return number * number };  
var y = square(6) // x gets the value 36
```

Exercise

You can insert the below code into web console of any website and it will return array of all text elements in the variable `elements`. Create a function that reverses character order in a string and use this function to change text on a website. Element text can be accessed and changed by its property `nodeValue`.

```
var elements = [];  
var recurse = function (element) {  
    if (element.childNodes.length > 0)  
        for (var i = 0; i < element.childNodes.length; i++)  
            recurse(element.childNodes[i]);  
  
    if (element.nodeType == Node.TEXT_NODE &&  
        element.nodeValue.trim() != '')  
        elements.push(element);  
};  
var html = document.getElementsByTagName('html')[0];  
recurse(html);
```

Exercise Solution

```
function reverse (text) {  
    var reverseText = '';  
    for (var i = text.length-1; i >= 0; i--) {  
        reverseText += text[i];  
    }  
    return reverseText;  
}  
  
for (var i = 0; i < elements.length; i++) {  
    if (elements[i].nodeValue.length < 6) {  
        elements[i].nodeValue = reverse(elements[i].nodeValue);  
    }  
}
```

Resources

<https://developer.mozilla.org/>

<http://caniuse.com/>

<https://google.github.io/styleguide/javascriptguide.xml#Strings>

