Fundusze Europejskie
Wiedza Edukacja Rozwój

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Społeczny

Website programming

Mechanisms for creating scripts using JavaScript

Mgr inż. Bartłomiej Salak

Contents

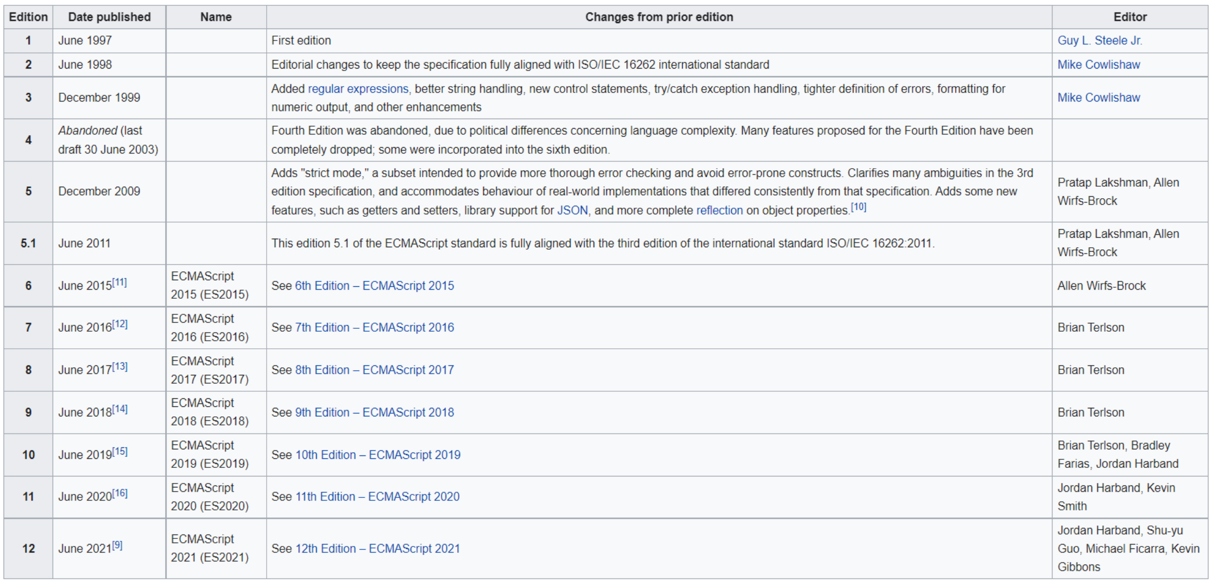
JavaScript programming language

JavaScript is one of the 3 languages all programmers need to learn to build web applications:

1. HTML (Hypertext Markup Language) to define the content of websites;
2. CSS (Cascading Style Sheets), to define the layout of the web pages;
3. JavaScript for programming website behavior.

JavaScript - is a fully fledged scripting language in which we can use the entire repertoire of classic language constructs. It enables support for dynamic content creation on a website, multimedia control, image animation and many other functionalities.

JavaScript was invented by Brendan Eich in 1995 and became an ECMA standard in 1997. ECMA-262 is the official name of the standard and ECMAScript is the official name of the language.



Edition	Date published	Name	Changes from prior edition	Editor
1	June 1997		First edition	Guy L. Steele Jr.
2	June 1998		Editorial changes to keep the specification fully aligned with ISO/IEC 16262 international standard	Mike Cowlishaw
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* (last draft 30 June 2003)		Fourth Edition was abandoned, due to political differences concerning language complexity. Many features proposed for the Fourth Edition have been completely dropped; some were incorporated into the sixth edition.	
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 the third edition of the international standard ISO/IEC 16262:2011.	Pratap Lakshman, Allen Wirfs-Brock
6	June 2015[11]	ECMAScript 2015 (ES2015)	See 6th Edition – ECMAScript 2015	Allen Wirfs-Brock
7	June 2016[12]	ECMAScript 2016 (ES2016)	See 7th Edition – ECMAScript 2016	Brian Terlson
8	June 2017[13]	ECMAScript 2017 (ES2017)	See 8th Edition – ECMAScript 2017	Brian Terlson
9	June 2018[14]	ECMAScript 2018 (ES2018)	See 9th Edition – ECMAScript 2018	Brian Terlson
10	June 2019[15]	ECMAScript 2019 (ES2019)	See 10th Edition – ECMAScript 2019	Brian Terlson, Bradley Farias, Jordan Harband
11	June 2020[16]	ECMAScript 2020 (ES2020)	See 11th Edition – ECMAScript 2020	Jordan Harband, Kevin Smith
12	June 2021[9]	ECMAScript 2021 (ES2021)	See 12th Edition – ECMAScript 2021	Jordan Harband, Shu-yu Guo, Michael Ficarra, Kevin Gibbons

Fig 1. ECMA Script version history

Link to the Picture: https://en.wikipedia.org/wiki/ECMAScript

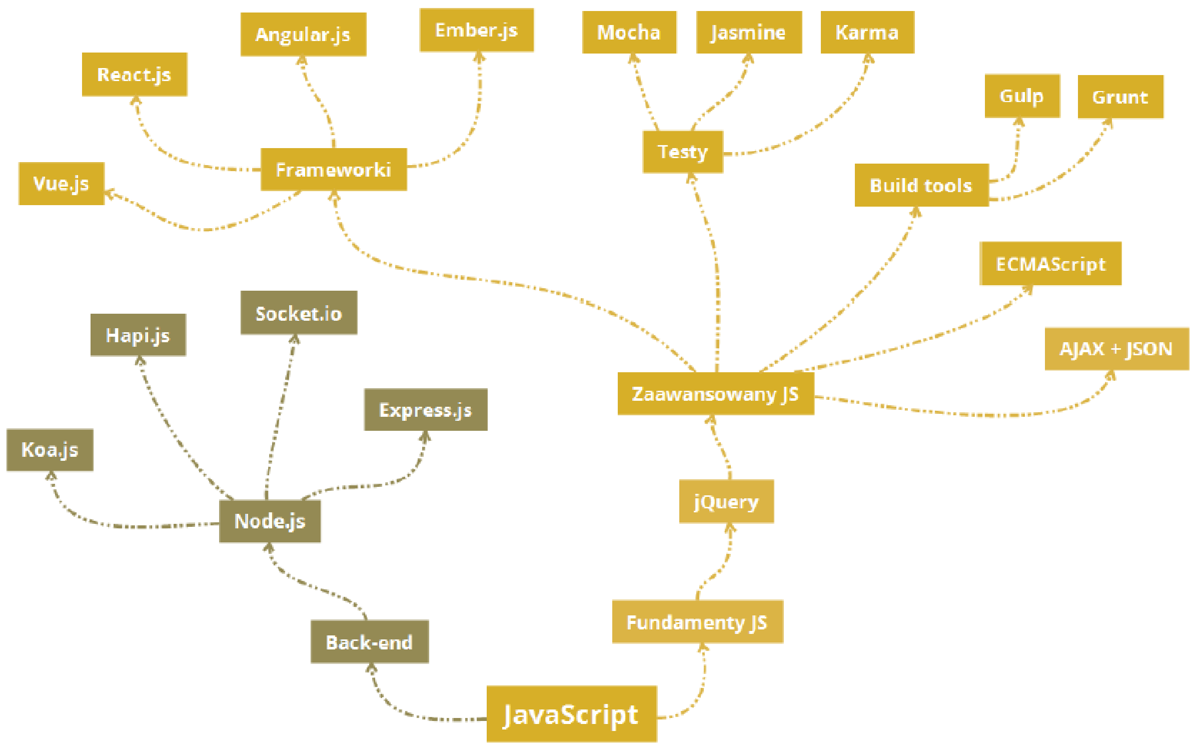


Fig 2. JS tool tree

At the very beginning of our adventure with a web developer, we most often use JS to improve the interface of the website, enriching it with additional functionalities not available in HTML or CSS. JS scripts will allow us to create effective sliders, animated photo galleries, pop-up panels with navigation, interactive menus, clocks, animations, etc. However, with time we find that it was only an introduction to the true power of this language - in some applications JS now even plays the role of back-end (Node.js technology).

JavaScript capabilities

- JavaScript can place dynamically changing text on the page;

- JavaScript handles events - for example, you can write code that responds to clicking the mouse on a given element;

- JavaScript can change the content of HTML elements on the page;

- JavaScript can be used for data validation, i.e. checking its correctness before sending it to the server;

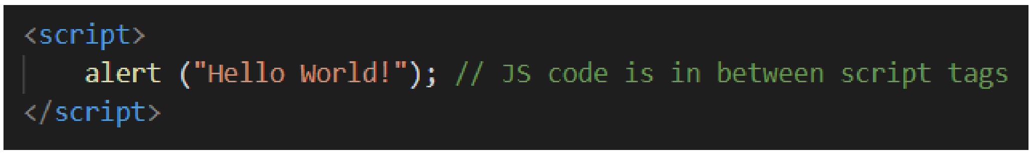
- JavaScript can recognize your browser type, which can be used to load the appropriate page (depending on your browser);

- JavaScript has access to cookies.

Embedding a JS script in an HTML dokument

There are two basic ways to embed ("hook") a JS script into the HTML of your website:

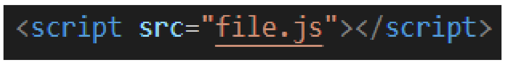
1. Placing the source code of the script directly between the script tags:



```
<script>
    alert ("Hello World!"); // JS code is in between script tags
</script>
```

The disadvantage of this solution is the weaker separation of the website content layer (HTML tags) from its front-end functionality (written in JavaScript).

2. Placing the source code of the script in an external file with the * .js extension, and then including it in an HTML dokument



```
<script src="file.js"></script>
```

The advantage of this solution is a good separation of HTML tags and JS code - after all, the entire script "sits" in an external file.

Variables. Declaring variables in JS

A variable is simply a container (drawer, box) for data - we can store in it, for example, a number, a string (in programming we often call a string: string), a logical value (true / false) or data of another type. Of course, the value stored in such a variable actually "sits" in the computer's RAM - and sometimes we do need to remember a value in the script.

Variable - is a program element that allows you to store data of different types. In JavaScript, unlike other programming languages, you do not need to specify the type of a variable when declaring it. Moreover, the type of the variable may be modified during the script execution - e.g. a string variable can be assigned a numerical value.

The variable consists of three elements:
- variable name (identifier);

- address (memory location);

- value.

The general rules for constructing variable names (unique identifiers) are:

- names can contain letters, numbers, underscores and dollar signs;

- names must start with a letter;

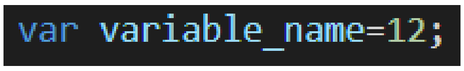
- names can start with $ and _ (not recommended in JS);

- names are case sensitive (y and Y are different variables);

- reserved words (like JavaScript keywords) cannot be used as names.

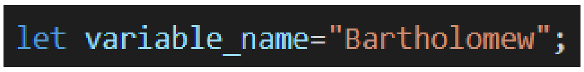
Variable declarations
In JavaScript as of ES6, there are three types of declarations.

var - declares a variable with functional (global) scope, optionally initializing it with the given value.



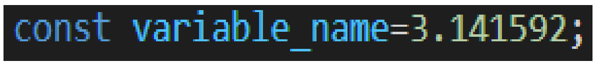
```
var variable_name=12;
```

let - declares a block-scope (local) variable limited to the block it is contained in, optionally initializing it with the given value.



```
let variable_name="Bartholomew";
```

const - declares a block-scoped variable, which is a read-only constant, the value of which cannot be changed while the entire script is running.



```
const variable_name=3.141592;
```

Capturing an HTML element in JavaScript

The selected HTML element (which has a specific identifier) can be "caught" for editing in JavaScript using the document.getElementById() method. The handle allows easy access to many attributes, but also accordingly. In the recreation below, an edit field with the identifier "field" was caught, and then the value of the attribute was read (attribute of ten stored texts lasting in the text field):

HTML

```
<input type="text" id="pole">
```

JavaScript

```
var napis = document.getElementById("pole").value;
```

When you use the above line in a variable napis there will be the text taken from the selected edit field. Importantly - the handle itself is not yet a value! It was only after we used the handle to get to the attribute that it enabled us to write a value.

Displaying the variable value on the screen

document.write() - instruction built into the javascript language, responsible for displaying the text on the page. Allows the use of all HTML tags to edit the text on the page. The biggest disadvantage of this simple instruction is the destruction of the existing content of the site - the entire document will be overwritten and filled with the new content specified in brackets (only the value of the variable will remain on the screen).

General write:

object.method (method arguments);

Example:

```
<script type="text/javascript">
        document.write("First text");
</script>
```

Dialog boxes: alert, prompt, confirm

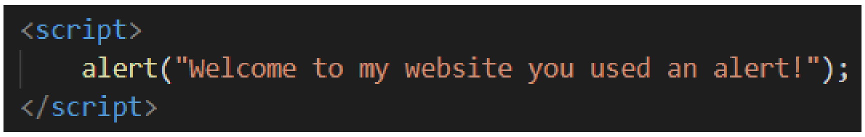
JavaScript enables the use of three types of dialog boxes from the window object. It is now possible to use these methods without having to call them from the window object.

Three types of dialog boxes:

- window.alert ("message");
- window.confirm ("message");
- window.prompt ("message", "default value").

Information windows - the task is to convey specific information. I have no influence on the further operation of the script. Its structure is extremely simple. It displays the text specified as the argument to the alert method and has one OK button to close the window. After clicking on the button, no value is returned. This window only informs the user about various events.

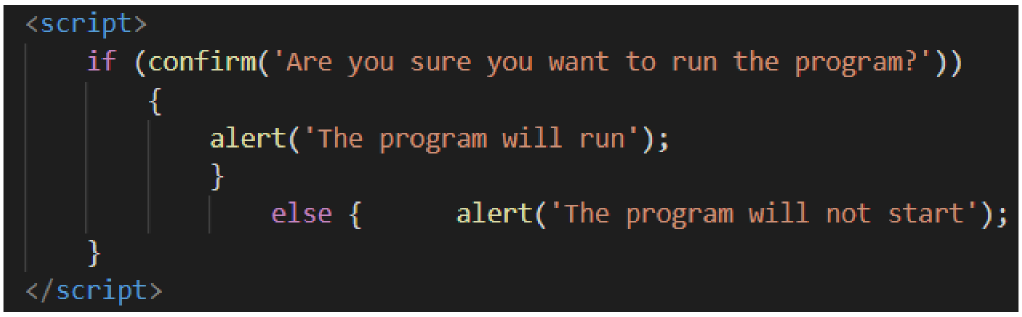
Example:



```
<script>
    alert("Welcome to my website you used an alert!");
</script>
```

Decision windows - it is responsible for displaying the content of the message constituting an argument of the confirm method. It provides two buttons: OK and Cancel, which, when pressed, return a boolean value of true for the "OK" button or false for the "Cancel" button. The window is therefore only used to confirm certain content by the user.

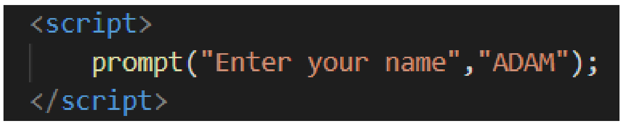
Example:



```
<script>
    if (confirm('Are you sure you want to run the program?'))
        {
            alert('The program will run');
            }
                else {      alert('The program will not start');
    }
</script>
```

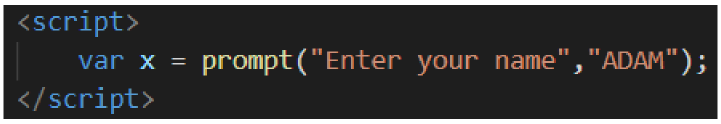
Text windows - displays the content of the message as arguments prompt method and enabling the user to enter data. Default text may appear in the field when calling up the text window.

Example:



```
<script>
    prompt("Enter your name","ADAM");
</script>
```

Secend example:

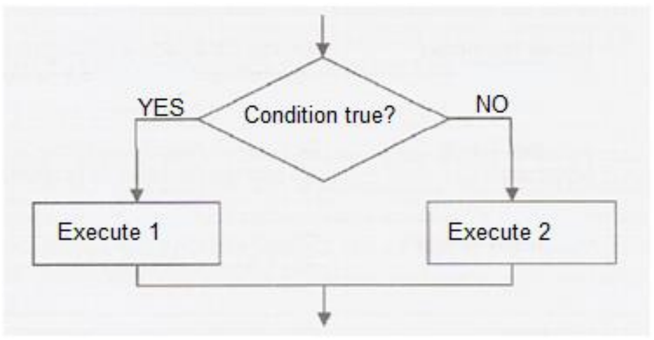


```
<script>
    var x = prompt("Enter your name","ADAM");
</script>
```

Conditional statement if

The keyword representing the conditional statement is IF. It decides which of the script fragments will be executed based on the specified condition, entered between the brackets.

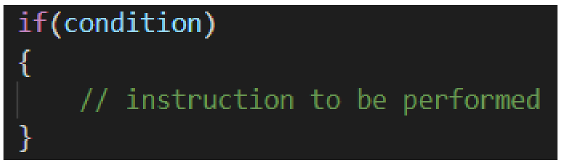
Conditional statement block diagram



The conditional statement is just a separation of the program's operation. Depending on whether the condition in the statement is true or false, the appropriate statements are executed.

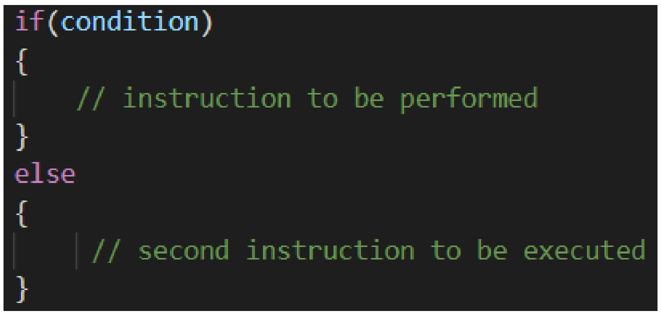
Simplified writing of a conditional statement allows the task to be performed only when the condition is true.

Example:



```
if(condition)
{
    // instruction to be performed
}
```

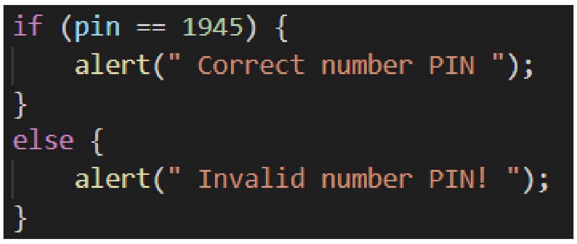
Secend example:



```
if(condition)
{
    // instruction to be performed
}
else
{
     // second instruction to be executed
}
```

The ELSE clause is optional, which means it does not necessarily have to appear - it depends on us and the problem under consideration.

Consider the example of an ATM. To be able to make a transaction, the user must enter the correct PIN number. We can check the correctness with a conditional instruction - let's assume that the correct PIN number is 1945:



```
if (pin == 1945) {
    alert(" Correct number PIN ");
}
else {
    alert(" Invalid number PIN! ");
}
```

Note the comparison operator "==", which is a double equal sign (unlike the value assignment operator, which is single: "=". Conditions can be complex, and hyphens are &&, ||. We also use negation, which we write with the exclamation point operator:!

CONDITION 1	CONDITION 2	OR (||)	AND (&&)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

CONDITION	NOT (!)
0	1
1	0

In the most extensive form of an IF control statement, there can be many conditions and many else if elements (condition), and you can also nest conditional statements inside curly braces.

Exercises

Exercise 1

Write a script that uses document.write () to display the first name in the first line, the last name in the second line, the floating point value in the third line, and the zip code in the fourth line with the city name. For fractional numbers, be sure to use the period sign.

Exercise 2

Using the available materials, write a script that will use all the dialog boxes you have learned.

Exercise 3

Write a script with which you can get the name, surname and age. Then you will display the obtained information on the website.

Exercise 4

Check whether the value of the variable X is greater or less than 20. Display the message on the screen as a dialog box.

Exercise 5

Write a script in which you will reserve a place in the restaurant for the selected number of people. Use the dialog box for this command. If the number of people is in the range from 1 to 2, the message "table for two" is to be displayed, in the range from 3 to 5, the message "table for 5", in the range from 6 to 8, the message "table for 8 people" is to be displayed, range from 9 to 12, the message "table for 12 people", and if the value is lower than 0 or greater than 12, the message "there are no such tables".