JS

JavaScript Fundamentals

By: Ahmed El-Mahdy

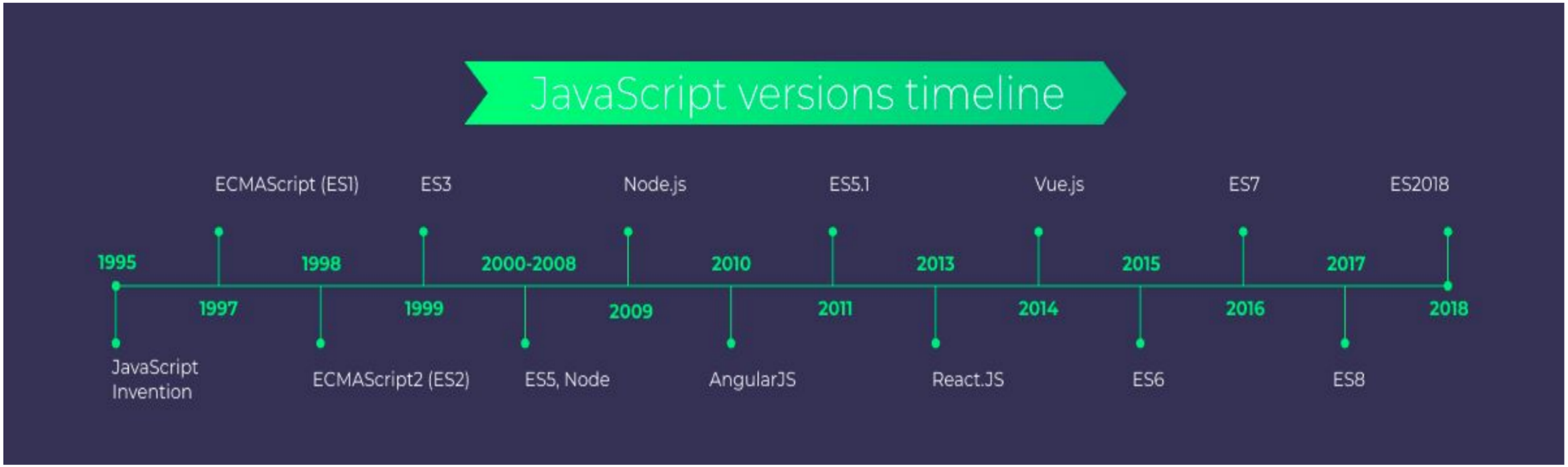
What is JavaScript

- Most popular and widely used client-side scripting language.
- Client-side scripting refers to scripts that run within your web browser.
- Designed to add interactivity and dynamic effects to the web pages by manipulating the content returned from a web server.
- JavaScript was originally developed as LiveScript by Netscape in the mid 1990s.
- It was later renamed to JavaScript in 1995, and became an ECMA standard in 1997.
- Now JavaScript is the standard client-side scripting language for web-based applications, and it is supported by virtually all web browsers available today, such as Google Chrome, Mozilla Firefox, Apple Safari, etc.
- JavaScript is an object-oriented language, and it also has some similarities in syntax to Java programming language.
- JavaScript is not related to Java in any way. JavaScript is officially maintained by ECMA (European Computer Manufacturers Association) as ECMAScript. ECMAScript 6 (or ES6) is the latest major version of the ECMAScript standard

JS Engine.

- V8 – in Chrome and Opera.
- SpiderMonkey – in Firefox.
- There are other codenames like “Trident” and “Chakra” for different versions of IE, “ChakraCore” for Microsoft Edge
- “Nitro” and “SquirrelFish” for Safari, etc

JavaScript history Engine



Another important milestone in JavaScript history was ECMA standardization (also in 1996). This is, actually, the reason why JavaScript versions are called “ECMAScript”.

What You Can Do with JavaScript ?

- Modify the content of a web page by adding or removing elements.
- Change the style and position of the elements on a web page.
- Monitor events like mouse click, hover, etc. and react to it.
- Perform and control transitions and animations.
- Create alert pop-ups to display info or warning messages to the user.
- Perform operations based on user inputs and display the results.
- Validate user inputs before submitting it to the server.
- The list does not end here, there are many other interesting things that you can do with JavaScript.

What You Can Do with JavaScript ?

Examples:

- Add new HTML to the page, change the existing content, modify styles.
- React to user actions, run on mouse clicks, pointer movements, key presses.
- Send requests over the network to remote servers, download and upload files (so-called *AJAX* and *COMET* technologies).
- Get and set cookies, ask questions to the visitor, show messages.
- Remember the data on the client-side (“local storage”).
- Increases interactivity with the users. (eg: hides/shows or enables/disable fields)
- Richer interfaces (e.g: drag and drop or smoother scrolling).
- Immediate feedback to users.

What You Can Not Do with JavaScript ?

JavaScript's abilities in the browser are limited for the sake of the user's safety. The aim is to prevent an evil webpage from accessing private information or harming the user's data.

- Limited connectivity to I/O devices (eg: network or files)
- Does not support multi-threading

What You Can Not Do with JavaScript ?

- May not read/write arbitrary files on the hard disk, copy them or execute programs. It has no direct access to OS system functions.

There are ways to interact with camera/microphone and other devices, but they require a user's explicit permission.

- Different tabs/windows generally do not know about each other.

- Its ability to receive data from other sites/domains is crippled. Though possible, it requires explicit agreement (expressed in HTTP headers) from the remote side..

Languages over JavaScript

The syntax of JavaScript does not suit everyone's needs. Different people want different features.
That's to be expected, because projects and requirements are different for everyone.
So recently a plethora of new languages appeared, which are transpired (converted) to JavaScript before they run in the browser.

Examples of such languages:

- CoffeeScript is a "syntactic sugar" for JavaScript. It introduces shorter syntax, allowing us to write clearer and more precise code. Usually, Ruby devs like it.
- TypeScript is concentrated on adding "strict data typing" to simplify the development and support of complex systems. It is developed by Microsoft.
- Flow also adds data typing, but in a different way. Developed by Facebook.
- Dart is a standalone language that has its own engine that runs in non-browser environments (like mobile apps), but also can be transpiled to JavaScript. Developed by Google

Adding JavaScript to Your Web Pages

Three ways to add JavaScript to a web page:

- Embedding the JavaScript code between a pair of <script> and </script> tag.
- Creating an external JavaScript file with the .js extension and then load it within the page through the src attribute of the <script> tag.
- Placing the JavaScript code directly inside an HTML tag using the special tag attributes such as onclick, onmouseover, onkeypress, onload, etc.

Adding JavaScript to Your Web Pages

Embedding the JavaScript Code

* Embed the JavaScript code directly within your web pages by placing it between the <script> and </script> tags.
* The <script> tag indicates the browser that the contained statements are to be interpreted as executable script and not HTML.

Calling an External JavaScript File

place your JavaScript code into a separate file with a .js extension, and then call that file in your document through the src attribute of the <script> tag, like this:

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

This is useful if you want the same scripts available to multiple documents.
It saves you from repeating the same task over and over again, and makes your website much easier to maintain.

Adding JavaScript to Your Web Pages

Placing the JavaScript Code Inline

* insert it directly inside the HTML tag using the special tag attributes such as onclick, onmouseover, onkeypress, onload, etc.

* Avoid placing large amount of JavaScript code inline as it clutters up your HTML with JavaScript and makes JavaScript code difficult to maintain.

Tip: You should always keep the content and structure of your web page (i.e. HTML) separate out from presentation (CSS), and behavior (JavaScript).

Adding JavaScript to Your Web Pages

Positioning of Script inside HTML Document
The <script> element can be placed in the <head>, or <body> section of an HTML document. But ideally, scripts should be placed at the end of the body section, just before the closing </body> tag, it will make your web pages load faster, since it prevents obstruction of initial page rendering.

Each <script> tag blocks the page rendering process until it has fully downloaded and executed the JavaScript code, so placing them in the head section (i.e. <head> element) of the document without any valid reason will significantly impact your website performance.

Tip: You can place any number of <script> element in a single document.
However, they are processed in the order in which they appear in the document, from top to bottom.

Adding JavaScript to Your Web Pages

Difference Between Client-side and Server side Scripting

*Client-side scripting languages such as JavaScript, VBScript, etc. are interpreted and executed by the web browser.

*Server-side scripting languages such as PHP, ASP, Java, Python, Ruby, etc. runs on the web server and the output sent back to the web browser in HTML format.

*Client-side scripting has many advantages over traditional server-side scripting approach.

Example: use JavaScript to check if the user has entered invalid data in form fields and show notifications for input errors accordingly in real-time before submitting the form to the web server for final data validation and further processing in order to prevent unnecessary network bandwidth usages and the exploitation of server system resources.

*Response from a server-side script is slower as compared to a client side script, because server-side scripts are processed on the remote computer not on the user's local computer.

Syntax

- Javascript code can be used in an HTML page in 2 different ways:

1. Inline script:

```
<script>
      // javascript code goes here
</script>
```

2. Include a js file:

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

Note: As a rule, only the simplest scripts are put into HTML. More complex ones reside in separate files.

- Whitespaces (e.g: spaces ,tabs or newlines are ignored).
- Statements can either end with semicolon or newline (semicolon is recommended).
- Javascript is case-sensitive language (i.e. fullName is different from FullName).
- Javascript is executed in the same order the code appears on the page -> (event loop).
- Comments are made using "//" or "/* */".
- By convention, all javascript variables and function names follow camelCasing (i.e firstName, getSalary()).

Variable

What is Variable?

- Variables are fundamental to all programming languages.
- Variables are used to store data, like string of text, numbers, etc.
- The data or value stored in the variables can be set, updated, and retrieved whenever needed.
- variables are symbolic names for values.

You can create a variable with the var keyword, whereas the assignment operator (=) is used to assign value to a variable, like this: var varName = value;

Tip: Always give meaningful names to your variables. Additionally, for naming the variables that contain multiple words, camelCase is commonly used. In this convention all world after the first should have uppercase first letters, e.g. myLongVariableName.

In JavaScript, variables can also be declared without having any initial values assigned to them. This is useful for variables which are supposed to hold values like user inputs.

Note: In JavaScript, if a variable has been declared, but has not been assigned a value explicitly, is automatically assigned the value undefined.

Variable

Declaring Multiple Variables at Once

can declare multiple variables and set their initial values in a single statement. Each variable are separated by commas.

The let and const Keywords ES6

- ES6 introduces two new keywords let and const for declaring variables.
- The const keyword works exactly the same as let, except that variables declared using const keyword cannot be reassigned later in the code.

- Var declare function-scoped variables, **BUT** both let and const keywords declare variables, scoped at block-level ({}).
- Block scoping means that a new scope is created between a pair of curly brackets {}.

Note: The let and const keywords are not supported in older browsers like IE10. IE11 support them partially.

Variable

Naming Conventions for JavaScript Variables

Rules for naming a JavaScript variable:

- must start with a letter, underscore (_), or dollar sign ($).
- cannot start with a number.
- can only contain alpha-numeric characters (A-z, 0-9) and underscores.
- cannot contain spaces.
- cannot be a JavaScript keyword or a JavaScript reserved word.

Note: Variable names in JavaScript are case sensitive, it means $myvar and $myVar are two different variables. So be careful while defining variable names.

Understanding the JavaScript Syntax

- The syntax of JavaScript is the set of rules that define a correctly structured JavaScript program.
- A JavaScript consists of JavaScript statements that are placed within the <script></script> HTML tags in a web page, or within the external JavaScript file having .js extension.

Variable

Case Sensitivity in JavaScript
This means that variables, language keywords, function names, and other identifiers must always be typed with a consistent capitalization of letters.

Example: the variable myVar must be typed myVar not MyVar or myvar.
Similarly, the method name getElementById() must be typed with the exact case not as getElementByID().

Comments

- A comment is a line of text that is completely ignored by the JavaScript interpreter.
- Comments are usually added with the purpose of providing extra information pertaining to source code.
- Not only help you understand your code when you look after a period of time but also others who are working with you on the same project.
- JavaScript support single-line as well as multi-line comments.
- Single-line comments begin with a double forward slash (//), followed by the comment text.

JS

Data Types

Data Types in JavaScript
Data types basically specify what kind of data can be stored and manipulated within a program.

Primitive Data Types (or primary)
- String
- Number
- Boolean

Composite Data Types (or reference)
- Object
- Array
- Function

Special data types
- Undefined
- Null

Primitive data types can hold only one value at a time, whereas composite data types can hold collections of values and more complex entities. Let's discuss each one of them in detail.

Data Types

Primitive Data Types (or primary)

The String Data Type

- used to represent textual data (i.e. sequences of characters).
- created using single or double quotes surrounding one or more characters.

The Number Data Type

- used to represent positive or negative numbers with or without decimal place, or numbers written using exponential notation e.g. 1.5e-4 (equivalent to 1.5x10-4).
- Also, it includes some special values which are: Infinity, -Infinity and NaN.
- Infinity represents the mathematical Infinity ∞, which is greater than any number.
- Infinity is the result of dividing a nonzero number by 0.
- NaN represents a special Not-a-Number value. It is a result of an invalid or an undefined mathematical operation, like taking the square root of -1 or dividing 0 by 0, etc.

The Boolean Data Type

- can hold only two values: true or false.
- used to store values like yes (true) or no (false), on (true) or off (false), etc.
- Boolean values also come as a result of comparisons in a program.

Data Types

Composite Data Types (or reference)

The Object Data Type

- The object is a complex data type that allows you to store collections of data.
- An object contains properties, defined as a key-value pair.
- A property key (name) is always a string, but the value can be any data type, like strings, numbers, booleans, or complex data types like arrays, function and other objects.

The Array Data Type

- Type of object used for storing multiple values in single variable.
- Each value (also called an element) in an array has a numeric position, known as its index, and it may contain data of any data type numbers, strings, booleans, functions, objects, and even other arrays.
- The array index starts from 0, so that the first array element is arr[0] not arr[1].
- The simplest way to create an array is by specifying the array elements as a comma-separated list enclosed by square brackets.

Data Types

Composite Data Types (or reference)

The Function Data

- An object that executes a block of code.
- Since functions are objects, so it is possible to assign them to variables.
- can be used at any place any other value can be used.
- can be stored in variables, objects, and arrays.
- can be passed as arguments to other functions,
- can be returned from functions.

Data Types

Special data types

The Undefined Data Type

- Can only have one value-the special value undefined.
- If a variable has been declared, but has not been assigned a value, has the value undefined.

The Null Data Type

- can have only one value the null value.
- null value means that there is no value.
- It is not equivalent to an empty string ("") or 0, it is simply nothing.
- A variable can be explicitly emptied of its current contents by assigning it the null value.

Data Types

The typeof OperatorThe Undefined Data Type

- Can be used to find out what type of data a variable or operand contains.
- It can be used with or without parentheses (typeof(x) or typeof x).
- It is useful in the situations when you need to process the values of different types differently.
- Be careful, it may produce unexpected result in some cases.

Note: if we test the null value using the typeof operator, it returned "object" instead of "null".
This is a long-standing bug in JavaScript, but since lots of codes on the web written around this behaviour, and thus fixing it would create a lot more problem, so idea of fixing this issue was rejected by the committee that design and maintains JavaScript.

JS

JavaScript Operators

What are Operators in JavaScript

- Symbols or Keywords that tell the JavaScript engine to perform some sort of actions.

Example: the addition (+) symbol is an operator that tells JavaScript engine to add two variables or values, while the equal-to (==), greater-than (>) or less-than (<) symbols are the operators that tells JavaScript engine to compare two variables or values, and so on.

The following sections describe the different operators used in JavaScript.

JavaScript Arithmetic Operators

used to perform common arithmetical operations, such as addition, subtraction, multiplication etc. Here's a complete list of JavaScript's arithmetic operators:

Operator	Description	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y.
*	Multiplication	$x * $y	Product of $x and $y.
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y

JS

JavaScript Operators

JavaScript Assignment Operators

used to assign values to variables.

Operator	Description	Example	Is The Same As
=	Assign	x = y	x = y
+=	Add and assign	x += $	x = x + y
-=	Subtract and assign	x -= y	x = x - y
*=	Multiply and assign	x *= y	x = x * y
/=	Divide and assign quotient	x /= y	x = x / y
%=	Divide and assign modulus	x %= y	x = x % y

JavaScript String Operators

Operator	Description	Example	Result
+	Concatenation	str1 + str2	Concatenation of str1 and str2
+=	Concatenation assignment	str1 += str2	Appends the str2 to the str1

JS

JavaScript Operators

JavaScript Incrementing and Decrementing Operators

used to increment/decrement a variable's value.

Operator	Name	Effect
++x	Pre-increment	Increments x by one, then returns x
x++	Post-increment	Returns x, then increments x by one
--x	Pre-decrement	Decrements x by one, then returns x
x--	Post-decrement	Returns x, then decrements x by one

JavaScript Logical Operators

used to combine conditional statements.

Operator	Name	Example	Result
&&	And	x && y	True if both x and y are true
||	Or	x || y	True if either x or y is true
!	Not	!x	True if x is not true

JS

JavaScript Operators

JavaScript Comparison Operators

used to compare two values in a Boolean fashion.

Operator	Name	Example	Result
==	Equal	x == y	True if x is equal to y
===	Identical	x === y	True if x is equal to y, and they are of the same type
!=	Not equal	x != y	True if x is not equal to y
!==	Not identical	x !== y	True if x is not equal to y, or they are not of the same
<	Less than	x < y	True if x is less than y
>	Greater than	x > y	True if x is greater than y
>=	Greater than or equal to	x >= y	True if x is greater than or equal to y
<=	Less than or equal to	x <= y	True if x is less than or equal to y

Input And Output

Creating Alert Dialog Boxes For Output

- The most simple dialog box. It enables you to display a short message to the user.
- Includes OK button, and the user has to click this OK button to continue.
- You can create alert dialog boxes with the alert() method.

Creating Prompt Dialog Box For Input

- Used to prompt the user to enter information.
- Includes a text input field, an OK and a Cancel button.
- You can create prompt dialog boxes with the prompt() method.
- The prompt() method returns the text entered in the input field when the user clicks the OK
- button, and null if user clicks the Cancel button.
- If the user clicks OK button without entering any text, an empty string is returned.
- For this reason, its result is usually assigned to a variable when it is used.
- The value returned by the prompt() method is always a string.
  (This means if the user enters 10 in the input field, the string "10" is returned instead of the number 10).
- if you want to use the returned value as a number you must covert it or cast to Number, like this: var age = Number(prompt("What's your age?"));

Tip: To display line breaks inside the dialog boxes, use newline character or line feed (\n); a backslash followed

Input And Output

Confirm

shows a message and waits for the user to press “OK” or “Cancel”. It returns true for OK and false for Cancel/Esc.

Two limitations shared by all the methods above:

1. The exact location of the modal window is determined by the browser. Usually, it’s in the center.
2. The exact look of the window also depends on the browser. We can’t modify it

JS

Use strict

- The directive looks like a string: "use strict" or 'use strict’.
- When it is located at the top of a script, the whole script works the “modern” way.
- "use strict" switches the engine to the “modern” mode, changing the behaviour of some built-in features.
- Strict mode is enabled by placing "use strict" at the top of a script or function. Several
- language features, like “classes” and “modules”, enable strict mode automatically.
- Strict mode is supported by all modern browsers.
- starting scripts with "use strict” is recommended.

JS

Use strict

```
"use strict";

// this code works the modern way
...
```

```
alert("some code");
// "use strict" below is ignored--it must be at the top

"use strict";

// strict mode is not activated
```

```
(function() {
  'use strict';

  // ...your code...
})()
```

JS

JavaScript Conditional Statements

- JavaScript also allows you to write code that perform different actions based on the results of a logical or comparative test conditions at run time.
- you can create test conditions in the form of expressions that evaluates to either true or false and based on these results you can perform certain actions.

conditional statements in JavaScript that you can use to make decisions**:**

- The **if** statement
- The **if...else** statement
- The **if...else if....else** statement
- The **switch...case** statement

The if **Statement**
used to execute a block of code only if the specified condition evaluates to true.
This is the simplest JavaScript's conditional statements and can be written like:

```
if(condition) {
    // Code to be executed
}
```

JS

JavaScript Conditional Statements

The if...else **Statement**

- You can enhance the decision-making capabilities of your JavaScript program by providing an alternative through adding an *else* statement to the *if* statement.
- The *if...else* statement allows you to execute one block of code if the specified condition is evaluates to true and another block of code if it is evaluates to false. It can be written, like this:

```
if(condition) {
    // Code to be executed if condition is true
} else {
    // Code to be executed if condition is false
}
```

The if...else if...else **Statement**

used to combine multiple if...else statements

```
if(condition1) {
    // Code to be executed if condition1 is true
} else if(condition2) {
    // Code to be executed if the condition1 is false and condition2
is true
} else {
    // Code to be executed if both condition1 and condition2 are
false
}
```

JS

JavaScript Conditional Statements

The Ternary Operator

- provides a shorthand way of writing the *if...else* statements.
- represented by the question mark (?) symbol and it takes three operands: a condition to check, a result for true, and a result for false. Its basic syntax is:

```
var result = (condition) ? value1 : value2
```

- If the condition is evaluated to true the value1 will be returned, otherwise value2 will be returned.

Tip: Code written using the ternary operator can be difficult to read sometimes. However, it provides a great way to write compact if-else statements

JS

JavaScript Conditional Statements

Using the Switch...Case Statement

- It is an alternative to the *if...else if...else* statement, which does almost the same thing.
- It tests a variable or expression against a series of values until it finds a match, and then executes the block of code corresponding to that match. It's syntax is:

```
switch(x){
    case value1:
        // Code to be executed if x === value1
        break;
    case value2:
        // Code to be executed if x === value2
        break;
    ...
    default:
        // Code to be executed if x is different from all values
}
```

Note: In a switch...case statement, the value of the expression or variable is compared against the case value using the strict equality operator (===).
That means if x = "0", it doesn't match case 0:, because their data types are not equal.

JS

JavaScript Conditional Statements

Using the Switch...Case Statement

- The switch...case statement differs from the *if...else* statement in one important way. The switch statement executes line by line (i.e. statement by statement) and once JavaScript finds a case clause that evaluates to true.

- It's not only executes the code corresponding to that case clause, but also executes all the subsequent case clauses till the end of the switch block automatically.

- To prevent this you must include a break statement after each case.

- The break statement tells the JavaScript interpreter to break out of the *switch...case* statement block once it executes the code associated with the first true case.

- The break statement is however not required for the case or default clause, when it appears at last in a switch statement. Although, it a good programming practice to terminate the last case, or default clause in a switch statement with a break.

- It prevents a possible programming error later if another case statement is added to the switch statement.

- The default clause is optional, which specify the actions to be performed if no case matches the switch expression.

JS

JavaScript Conditional Statements

Using the Switch...Case Statement

- The default clause is optional, which specify the actions to be performed if no case matches the switch expression.

- The default clause does not have to be the last clause to appear in a switch statement.

Multiple Cases Sharing Same Action

Each case value must be unique within a switch statement. However, different cases don't need to have a unique action. Several cases can share the same action.