**JAVASCRIPT – Overview**

**Q1. What is JavaScript?**

1. JavaScript is a dynamic programming language that's used for web development, in web applications, for game development, and lots more. It allows you to implement dynamic features on web pages that cannot be done with only HTML and CSS.

**Q2. Client-Side JavaScript?**

1. Client-Side JavaScript (CSJS) -- an extended version of JavaScript that enables the enhancement and manipulation of web pages and client browsers

**Q3. Advantages of JavaScript**

**Speed -** JavaScript tends to be very fast because it is often run immediately within the client's browser. So long as it doesn't require outside resources, JavaScript isn't slowed down by calls to a backend server. Also, major browsers all support JIT (just in time) compilation for JavaScript, meaning that there's no need to compile the code before running it.

**Simplicity -** JavaScript's syntax was inspired by Java's and is relatively easy to learn compared to other popular languages like C++.

**Popularity -** JavaScript is everywhere on the web, and with the advent of Node.js, is increasingly used on the backend. There are countless resources to learn JavaScript. Both StackOverflow and GitHub show an increasing amount of projects that use JavaScript, and the traction it's gained in recent years is only expected to increase.

**Interoperability -** Unlike PHP or other scripting languages, JavaScript can be inserted into any web page. JavaScript can be used in many different kinds of applications because of support in other languages like Pearl and PHP.

**Server Load -** JavaScript is client-side, so it reduces the demand on servers overall, and simple applications may not need a server at all.

**Rich interfaces -** JavaScript can be used to create features like drag and drop and components such as sliders, all of which greatly enhance the user interface and experience of a site.

**Extended Functionality -** Developers can extend the functionality of web pages by writing snippets of JavaScript for third party add-ons like Greasemonkey.

**Q4. Limitations of JavaScript?**

**Client-Side Security -** Since JavaScript code is executed on the client-side, bugs and oversights can sometimes be exploited for malicious purposes. Because of this, some people choose to disable JavaScript entirely.

**Browser Support -** While server-side scripts always produce the same output, different browsers sometimes interpret JavaScript code differently. These days the differences are minimal, and you shouldn't have to worry about it as long as you test your script in all major browsers.

**Q5. JavaScript Development Tools**

1. [**Visual Studio**](https://visualstudio.microsoft.com/)

[**WebStorm**](https://www.jetbrains.com/webstorm/)

[**Sublime Text**](https://www.sublimetext.com/)

**Q.6. Where is JavaScript Today?**

1. The 8th edition, known as ECMAScript 2017, is the current JavaScript version, **<!doctype html>**
2. released in June 2017

**JAVASCRIPT – Syntax**

**Q.1. Your First JavaScript Code**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>My Web Page</title>**

**<script>**

**alert('hello world!');**

**</script>**

**</head>**

**Q.2. Whitespace and Line Breaks**

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

JavaScript ignores whitespaces. For example, the following are same:

var employee = "Amit";

var employee="Amit";

Line breaks in JavaScript:

<!DOCTYPE html>

<html>

<body>

<p>Line-break in an alert box. Click below button.</p>

<button onclick="alert('Hi\nWelcome to Tutorialspoint')">Click</button>

</body>

</html>

**Q.2. Semicolons are Optional**

1. Semicolons are not required for JavaScript programming, nevertheless I advise you to use them. It makes your code more readable and is actually a good practice, and almost all cool programming languages use it. Take a stand and use it, it's up to you now!

**Q.3. Case Sensitivity**

1. JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers Time and TIME will convey different meanings in JavaScript.

**Q.4. Comments in JavaScript**

JavaScript comments can be used to explain JavaScript code, and to make it more readable. JavaScript comments can also be used to prevent execution, when testing alternative code.

## **Single Line Comments**

## Single line comments start with //

Any text between // and the end of the line will be ignored by JavaScript (will not be executed).

let x = 5; // Declare x, give it the value of 5

let y = x + 2; // Declare y, give it the value of x + 2

## **Multi-line Comments**

Multi-line comments start with /\* and end with \*/.

Any text between /\* and \*/ will be ignored by JavaScript.

/\*

The code below will change

the heading with id = "myH"

and the paragraph with id = "myP"

in my web page:

\*/

**JAVASCRIPT – Enabling**

**Q1.JavaScript in Internet Explorer**

1. On the Tools menu, click Internet Options, and then click the Security tab.
2. Click the Internet zone.
3. If you do not have to customize your Internet security settings, click Default Level. Then do step 4

If you have to customize your Internet security settings, follow these steps:

a. Click Custom Level.

b. In the Security Settings – Internet Zone dialog box, click Enable for Active Scripting in the Scripting section.

1. Click the Back button to return to the previous page, and then click the Refresh button to run scripts.

**Q2.JavaScript in Firefox**

1. Click on the **Tools** menu

2. Choose **Options...** from the menu

3. Click the **Content** tab in the **Options** pop up

4. Make sure that **Enable JavaScript** is checked

5. Click **OK** to finish the process

**Q2.JavaScript in Chrome**

1. To enable JavaScript in your browser. Please follow the instructions below:
2. Click the Chrome menu icon on the browser toolbar.
3. Select Settings.
4. On the "Settings" page, click the Show advanced settings... link.
5. In the "Privacy" section, click Content settings...
6. Select Allow all sites to run JavaScript (recommended) in the "JavaScript" section.
7. Click Done.

**Q3.JavaScript in Opera**

1. On the Tools menu, click Preferences.
2. On the Advanced tab, click Content.
3. Click to select the Enable JavaScript check box, and then click OK.
4. Click the Back button to return to the previous page, and then click the Reload button to run scripts.

**Q4.Warning for Non-JavaScript Browsers**

To let users know about non-JavaScript web browsers, use the <noscript> tag. The HTML <noscript> tag is used to handle the browsers, which do recognize <script> tag but do not support scripting. This tag is used to display alternate text message

Here’s an example,

<!DOCTYPE html>

<html>

<head>

<title>HTML noscript Tag</title>

</head>

<body>

<script>

<!--

document.write("Hello JavaScript!")

-->

</script>

<noscript>

Your browser does not support JavaScript!

</noscript>

</body>

</html>

**JAVASCRIPT – Placement**

**Q1.JavaScript in <head>...</head> Section**

1. In this example, a JavaScript function is placed in the <head> section of an HTML page. The function is invoked (called) when a button is clicked:

<!DOCTYPE html>

<html>

<head>

<script>

function myFunction() {

document.getElementById("demo").innerHTML = "Paragraph changed.";

}

</script>

</head>

<body>

<h2>Demo JavaScript in Head</h2>

<p id="demo">A Paragraph</p>

<button type="button" onclick="myFunction()">Try it</button>

</body>

</html>

**Q2.JavaScript in <body>...</body> Section**

In this example, a JavaScript function is placed in the <body> section of an HTML page.

The function is invoked (called) when a button is clicked:

<!DOCTYPE html>

<html>

<body>

<h2>Demo JavaScript in Body</h2>

<p id="demo">A Paragraph</p>

<button type="button" onclick="myFunction()">Try it</button>

<script>

function myFunction() {

document.getElementById("demo").innerHTML = "Paragraph changed.";

}

</script>

</body>

</html>

**Q3.JavaScript in <body> and <head> Sections**

<html>

<head>

<script type="text/javascript">

function book(){

alert ("JavaScript is Awesome");

}

</script>

</head>

<body>

<script type="text/javascript">

document.write("Leave it!! No need to develop.");

</script>

<input type="button" onClick="book()" value="DEVELOP">

</body>

</html>

**Q3.JavaScript in External File**

1. Scripts can also be placed in external files:

### **External file: myScript.js**

function myFunction() {

document.getElementById("demo").innerHTML = "Paragraph changed.";

}

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension **.js**.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

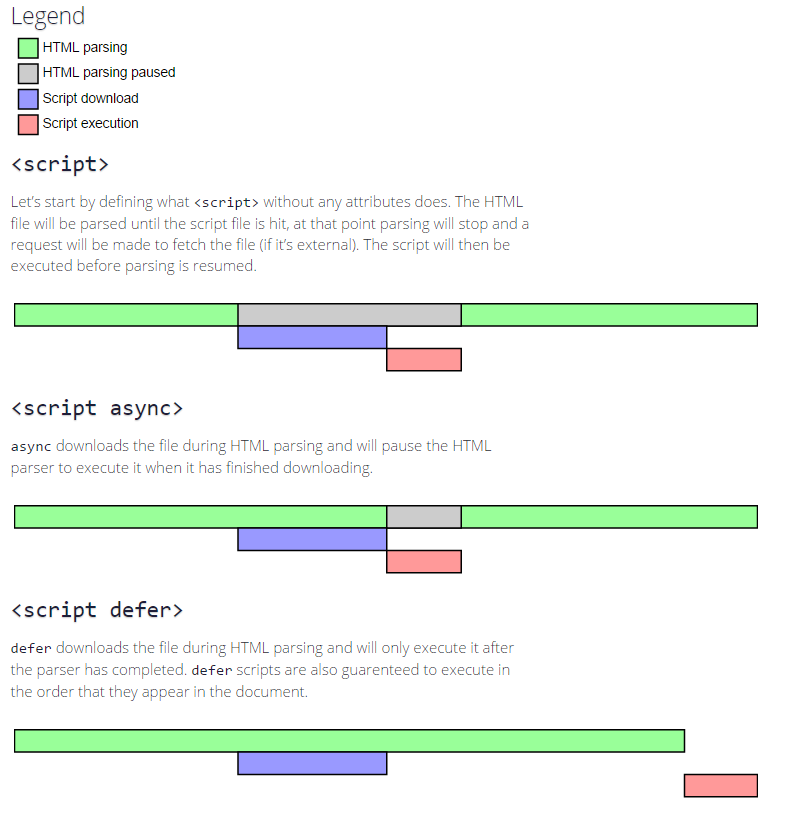
<script src="myScript.js"></script>

**Q4.Defer in script**

1. The defer is a Boolean value, used to indicate that script is executed after the document has been parsed. It works only with external scripts (i.e., works only when we are specifying the src attribute in <script> tag). It declares that the script will not create any content. So, the browser can continue the parsing of the rest of the page. The <script> with the defer attribute does not block the page.

**Q5.Async in script**

1. The async attribute is a boolean attribute. When present, it specifies that the script will be executed asynchronously as soon as it is available.



**JAVASCRIPT – Variables**

**Q1.JavaScript Data Types**

1. JavaScript variables can hold different data types: numbers, strings, objects and more:

let length = 16; // Number

let lastName = "Johnson"; // String

let x = {firstName:"John", lastName:"Doe"}; // Object

**Q2.JavaScript Variables**

1. **4 Ways to Declare a JavaScript Variable:**

* Using var
* Using let
* Using const
* Using nothing

**Q3.JavaScript Variable Scope**

1. Scope determines the accessibility (visibility) of variables.

JavaScript has 3 types of scope:

* Block scope
* Function scope
* Global scope

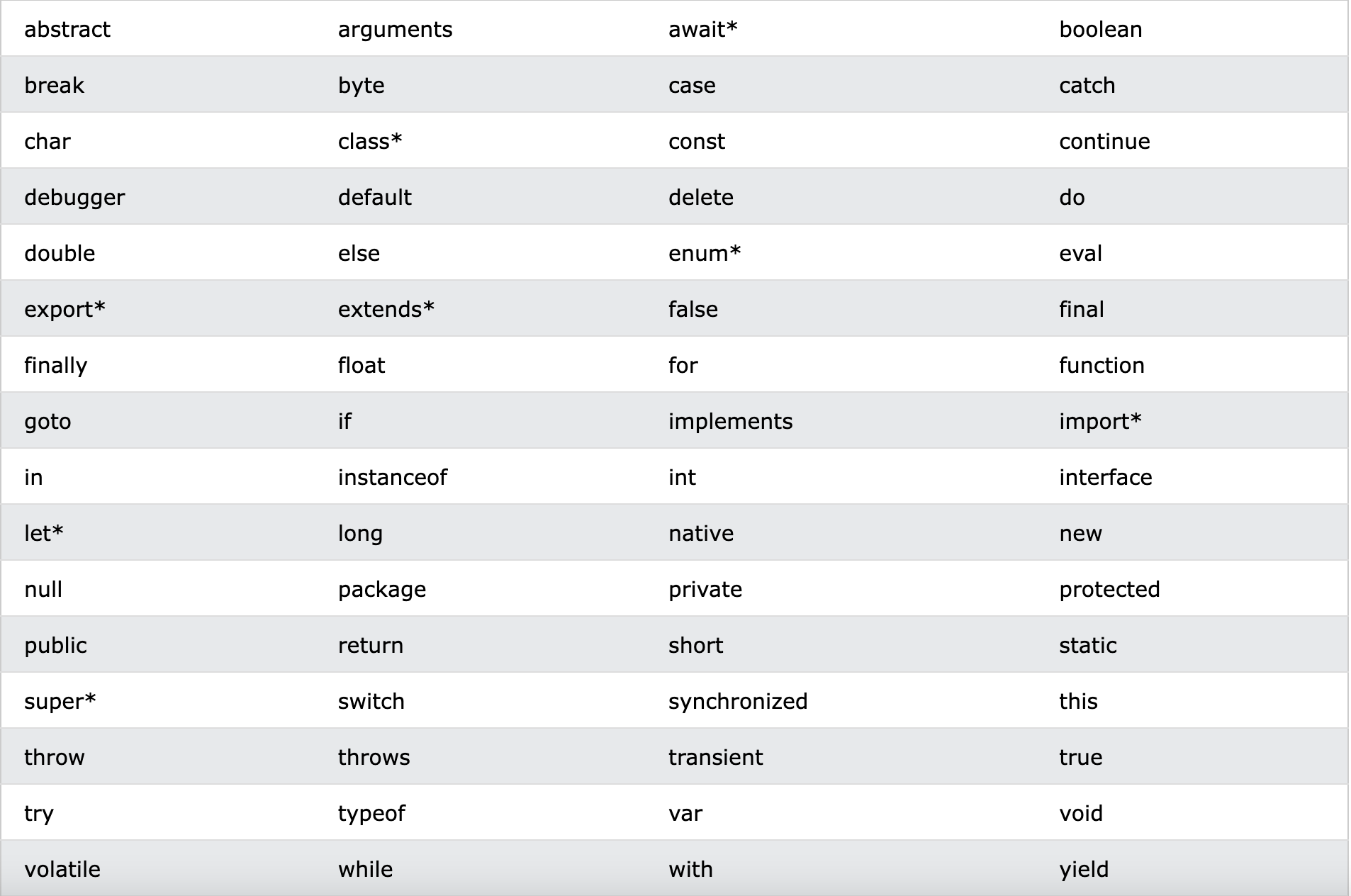
**Q4.JavaScript Variable Names**

A variable name should accurately identify your variable. When you create good variable names, your JavaScript code becomes easier to understand and easier to work with. Properly naming variables is really important! Here are rules JavaScript has for naming variables:

* Variable names cannot contain spaces.
* Variable names must begin with a letter, an underscore (\_) or a dollar sign ($).
* Variable names can only contain letters, numbers, underscores, or dollar signs.
* Variable names are case-sensitive.

**Q5.JavaScript Reserved Words**

1. In JavaScript you cannot use these reserved words as variables, labels, or function names:



**JAVASCRIPT – Variables**

**Q1.What is an Operator?**

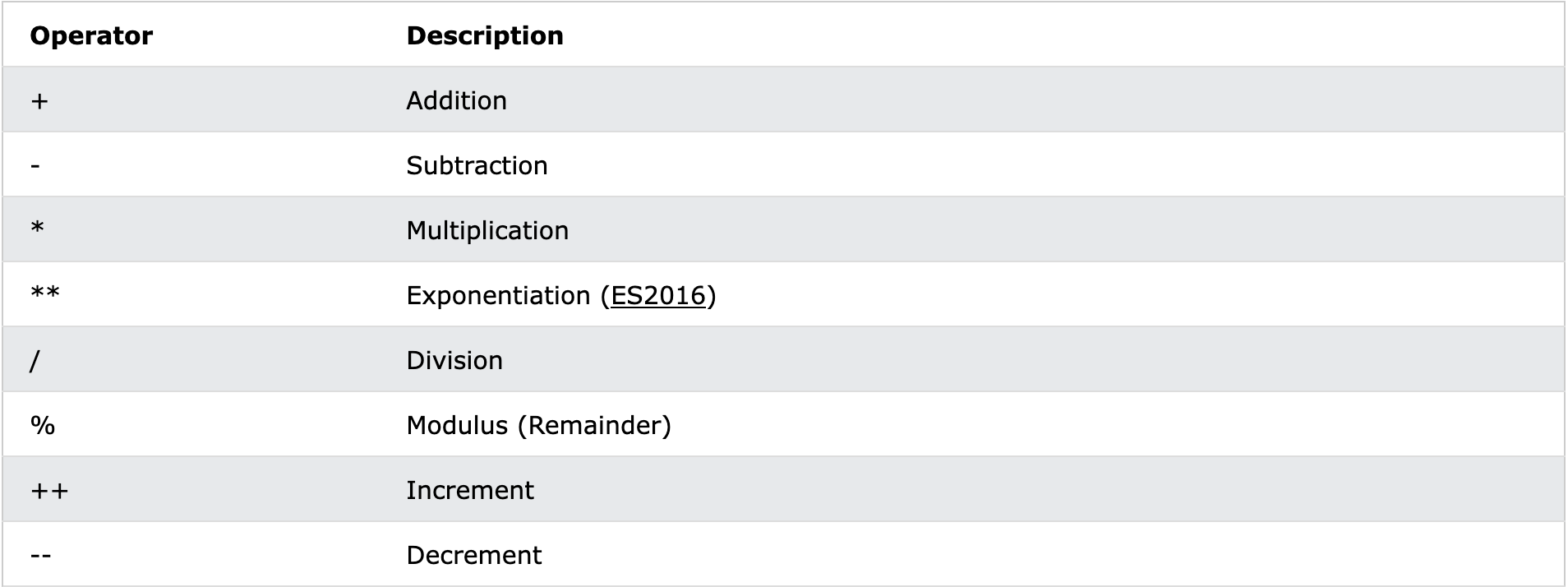
1. In JavaScript, an operator is a special symbol used to perform operations on operands (values and variables). For example,

2 + 3; // 5

Here + is an operator that performs addition, and 2 and 3 are operands.

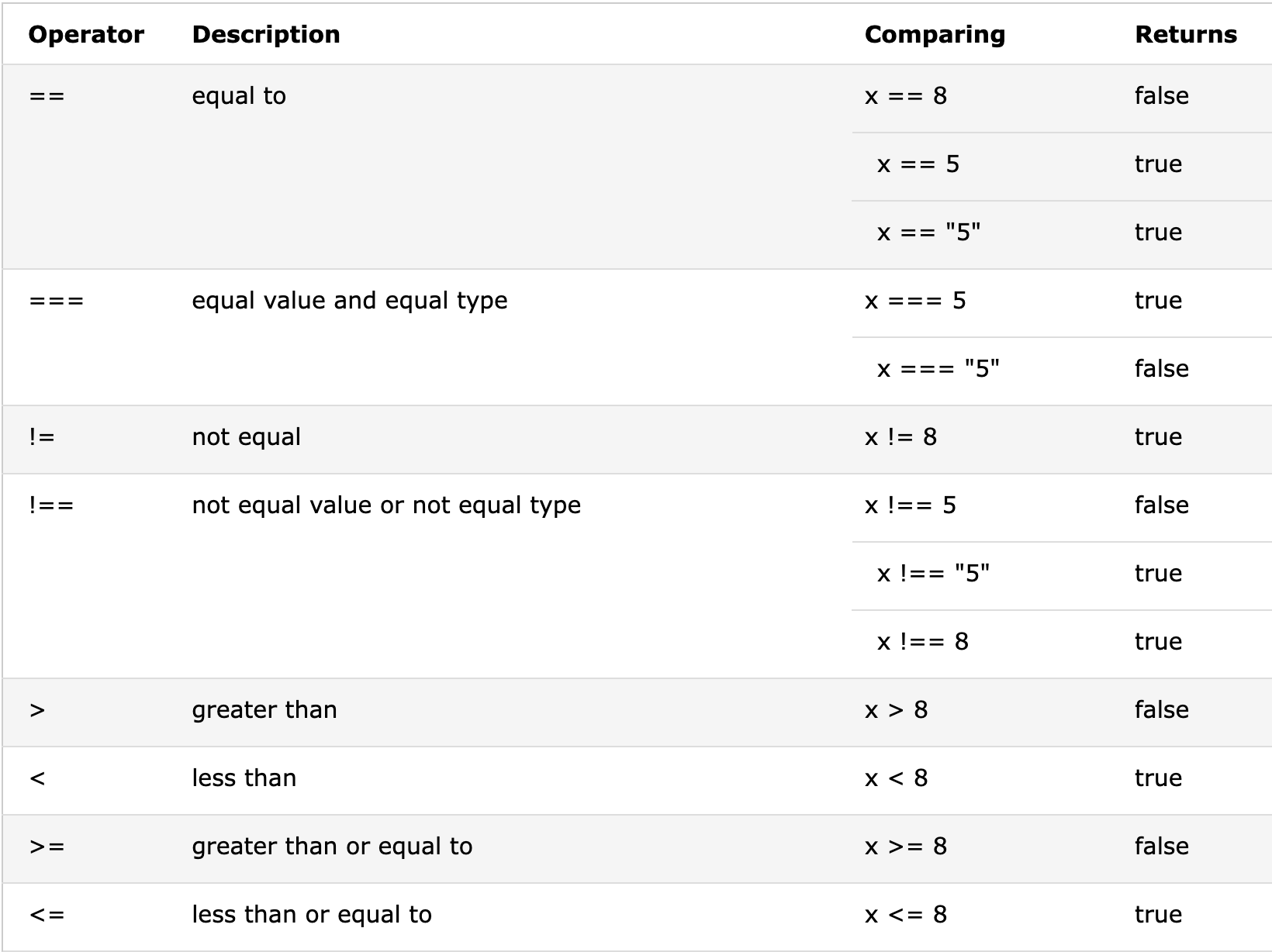
**Q2.Arithmetic Operators**

1. Arithmetic operators perform arithmetic on numbers (literals or variables).



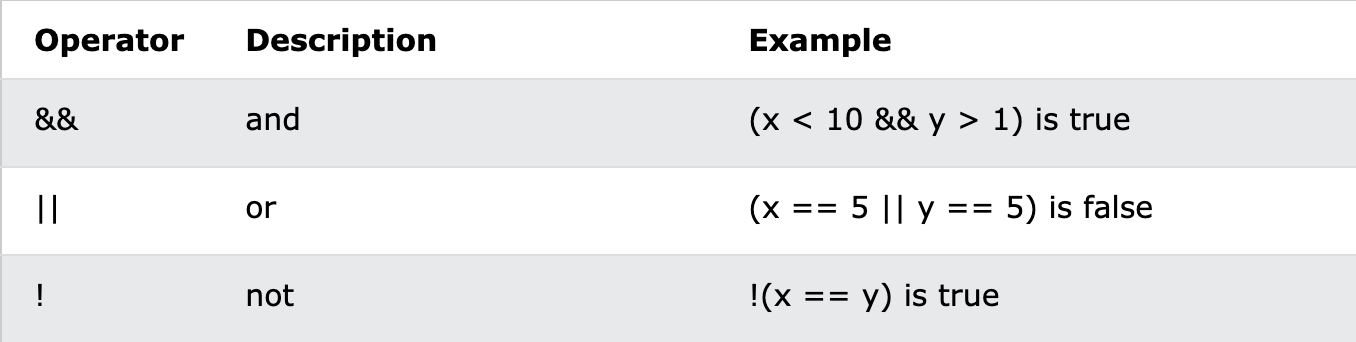
**Q3.Comparison Operators**

1. Comparison operators are used in logical statements to determine equality or difference between variables or values.Given that x = 5, the table below explains the comparison operators:

****

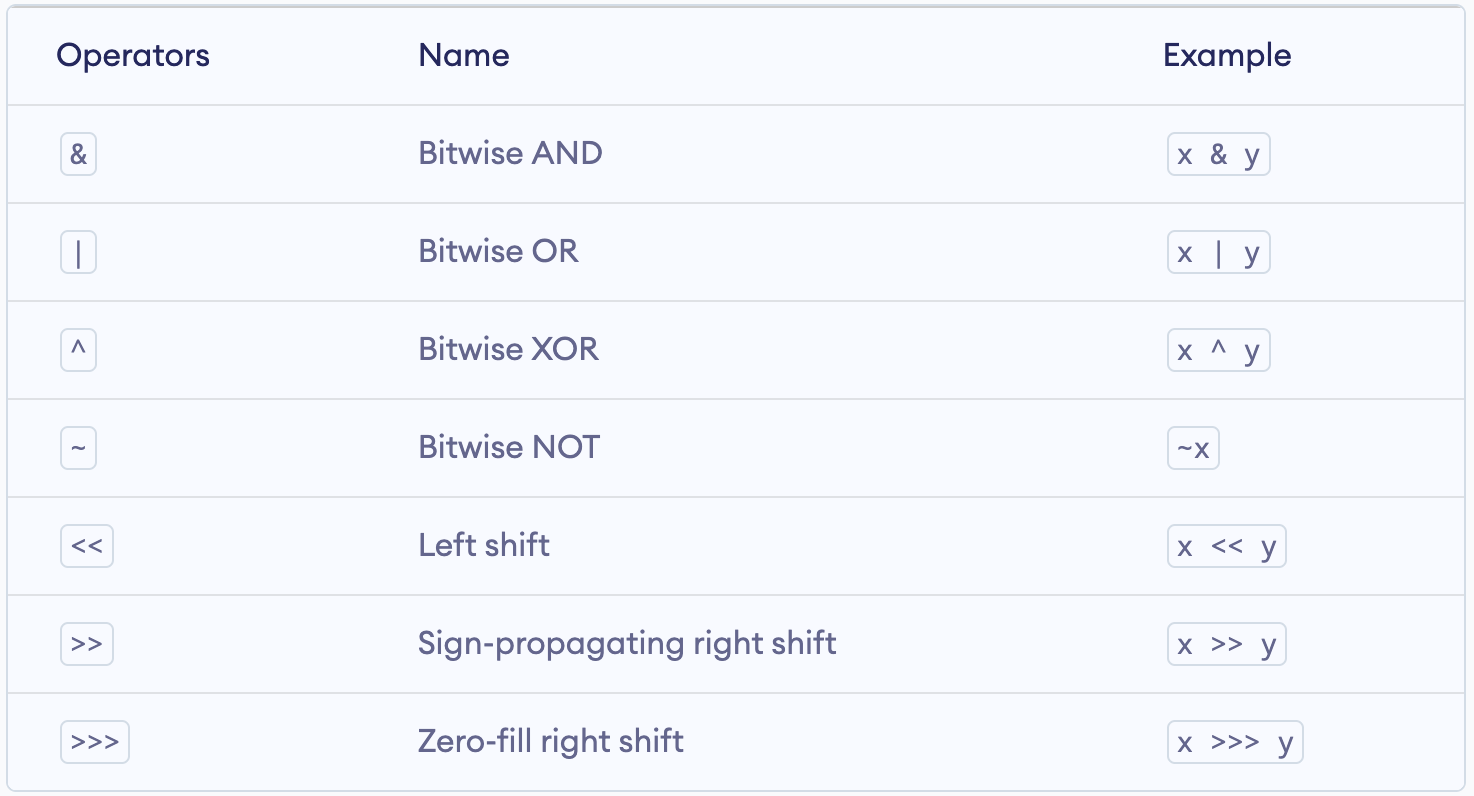
**Q3.Logical Operators**

1. Logical operators are used to determine the logic between variables or values. Given that x = 6 and y = 3, the table below explains the logical operators:



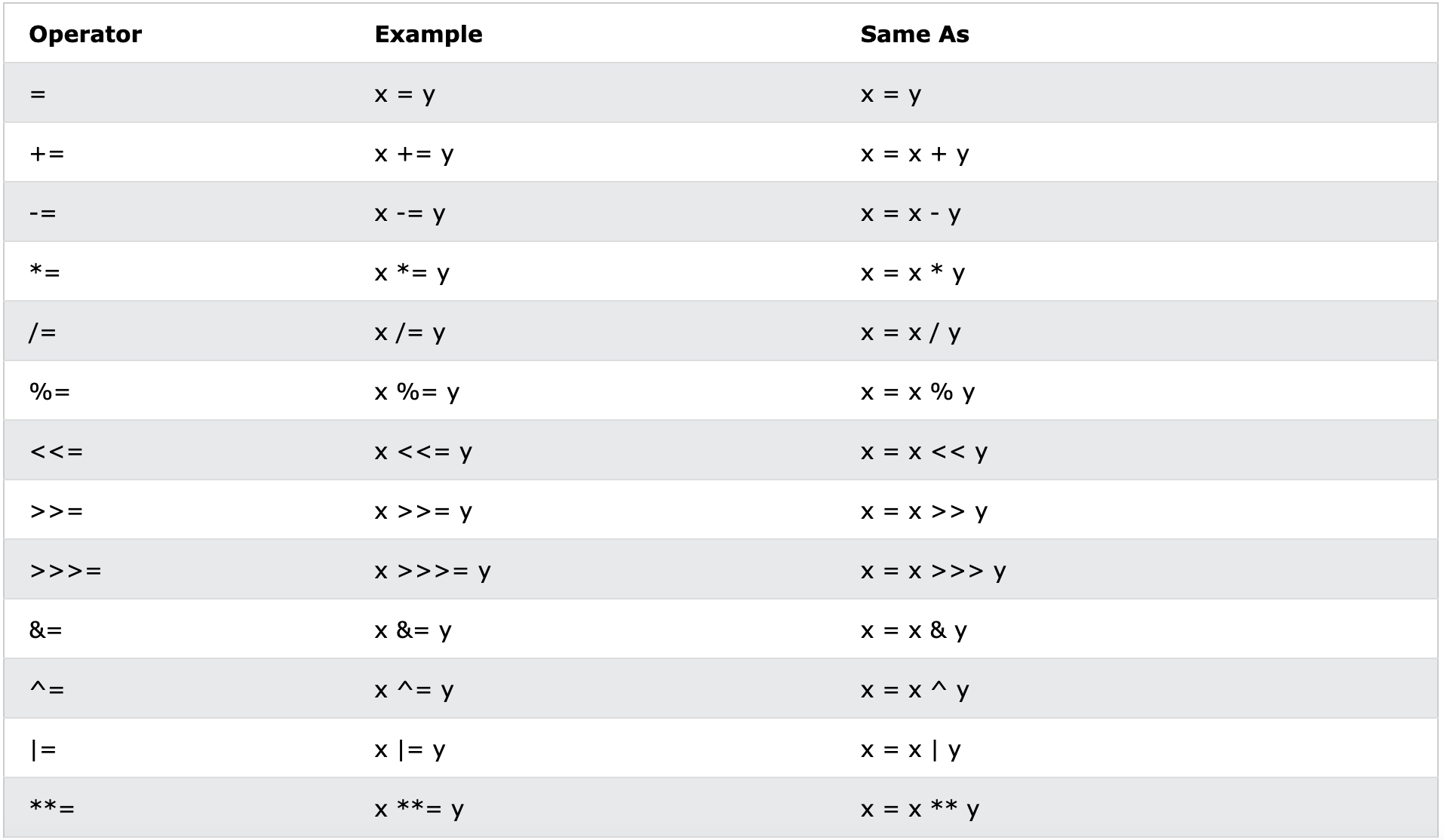
**Q4.Bitwise Operator**

1. Bitwise operators treat its operands as a set of 32-bit binary digits (zeros and ones) and perform actions. However, the result is shown as a decimal value.



**Q5.Assignment Operators**

1. Assignment operators assign values to JavaScript variables.

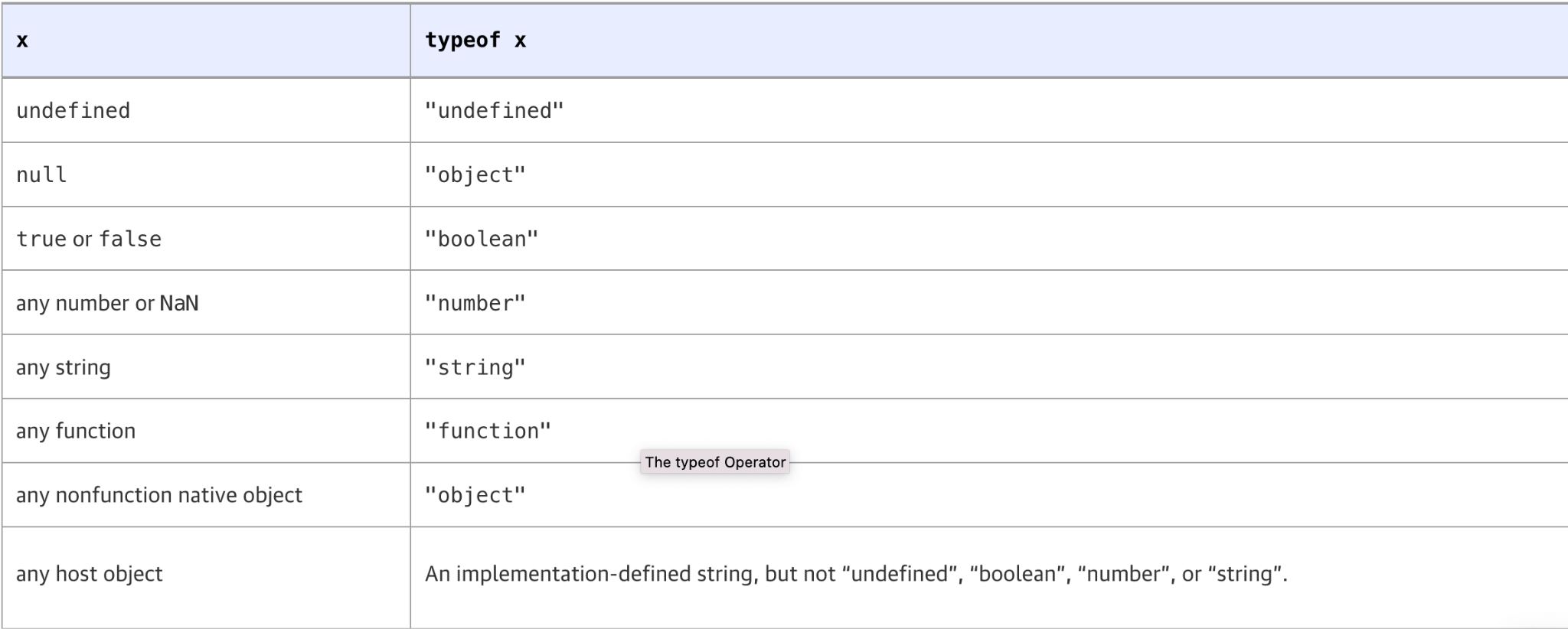


**Q5.Miscellaneous Operators**

## **The Conditional Operator (?:)**

x > 0 ? x : -x

## **The typeof Operator**



## **The delete Operator**

delete is an unary operator that attempts to delete the object property or array element specified as its operand. Like the assignment, increment, and decrement operators, delete is typically used for its property deletion side effect, and not for the value it returns. Some examples:

var o = { x: 1, y: 2}; // Start with an object

delete o.x; // Delete one of its properties

"x" in o // => false: the property does not exist anymore

var a = [1,2,3]; // Start with an array

delete a[2]; // Delete the last element of the array

2 in a // => false: array element 2 doesn't exist anymore

a.length // => 3: note that array length doesn't change, though