**JavaScript**

Is a programming language that is used to make web pages interactive in a way that it gives functions and most often used in survey, quizzes or polls in HTML pages. Scripts can be embedded in or included from HTML pages.

**Control flow**

[Block](http://devdocs.io/javascript/statements/block)

A block statement is used to group zero or more statements. The block is delimited by a pair of curly brackets.

[break](http://devdocs.io/javascript/statements/break)

Terminates the current loop, switch, or label statement and transfers program control to the statement following the terminated statement.

[continue](http://devdocs.io/javascript/statements/continue)

Terminates execution of the statements in the current iteration of the current or labeled loop, and continues execution of the loop with the next iteration.

[Empty](http://devdocs.io/javascript/statements/empty)

An empty statement is used to provide no statement, although the JavaScript syntax would expect one.

[if...else](http://devdocs.io/javascript/statements/if...else)

Executes a statement if a specified condition is true. If the condition is false, another statement can be executed.

[switch](http://devdocs.io/javascript/statements/switch)

Evaluates an expression, matching the expression's value to a case clause, and executes statements associated with that case.

[throw](http://devdocs.io/javascript/statements/throw)

Throws a user-defined exception.

[try...catch](http://devdocs.io/javascript/statements/try...catch)

Marks a block of statements to try, and specifies a response, should an exception be thrown.

**Declarations**

[var](http://devdocs.io/javascript/statements/var)

Declares a variable, optionally initializing it to a value.

[let](http://devdocs.io/javascript/statements/let)

Declares a block scope local variable, optionally initializing it to a value.

[const](http://devdocs.io/javascript/statements/const)

Declares a read-only named constant.

**Functions and classes**

[function](http://devdocs.io/javascript/statements/function)

Declares a function with the specified parameters.

[function\*](http://devdocs.io/javascript/statements/function*)

Generators functions enable writing [iterators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/The_Iterator_protocol) more easily.

[async function](http://devdocs.io/javascript/statements/async_function)

Declares an async function with the specified parameters.

[return](http://devdocs.io/javascript/statements/return)

Specifies the value to be returned by a function.

[class](http://devdocs.io/javascript/statements/class)

Declares a class.

**Iterations**

[do...while](http://devdocs.io/javascript/statements/do...while)

Creates a loop that executes a specified statement until the test condition evaluates to false. The condition is evaluated after executing the statement, resulting in the specified statement executing at least once.

[for](http://devdocs.io/javascript/statements/for)

Creates a loop that consists of three optional expressions, enclosed in parentheses and separated by semicolons, followed by a statement executed in the loop.

[for each...in](http://devdocs.io/javascript/statements/for_each...in)

Iterates a specified variable over all values of object's properties. For each distinct property, a specified statement is executed.

[for...in](http://devdocs.io/javascript/statements/for...in)

Iterates over the enumerable properties of an object, in arbitrary order. For each distinct property, statements can be executed.

[for...of](http://devdocs.io/javascript/statements/for...of)

Iterates over iterable objects (including [arrays](http://devdocs.io/javascript/global_objects/array), array-like objects, [iterators and generators](https://developer.mozilla.org/en-US/docs/JavaScript/Guide/Iterators_and_Generators)), invoking a custom iteration hook with statements to be executed for the value of each distinct property.

[while](http://devdocs.io/javascript/statements/while)

Creates a loop that executes a specified statement as long as the test condition evaluates to true. The condition is evaluated before executing the statement.

**Others**

[debugger](http://devdocs.io/javascript/statements/debugger)

Invokes any available debugging functionality. If no debugging functionality is available, this statement has no effect.

[export](http://devdocs.io/javascript/statements/export)

Used to export functions to make them available for imports in external modules, another scripts.

[import](http://devdocs.io/javascript/statements/import)

Used to import functions exported from an external module, another script.

[label](http://devdocs.io/javascript/statements/label)

Provides a statement with an identifier that you can refer to using a break or continue statement.

[with](http://devdocs.io/javascript/statements/with)

Extends the scope chain for a statement.

**Expressions and operators**

This chapter documents all the [JavaScript expressions and operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators).

For an alphabetical listing see the sidebar on the left.

**Primary expressions**

Basic keywords and general expressions in JavaScript.

[this](http://devdocs.io/javascript/operators/this)

The this keyword refers to the function's execution context.

[function](http://devdocs.io/javascript/operators/function)

The function keyword defines a function expression.

[class](http://devdocs.io/javascript/operators/class)

The class keyword defines a class expression.

[function\*](http://devdocs.io/javascript/operators/function*)

The function\* keyword defines a generator function expression.

[yield](http://devdocs.io/javascript/operators/yield)

Pause and resume a generator function.

[yield\*](http://devdocs.io/javascript/operators/yield*)

Delegate to another generator function or iterable object.

[async function\*](http://devdocs.io/javascript/operators/async_function)

The async function defines an async function expression.

[await](http://devdocs.io/javascript/operators/await)

Pause and resume an async function and wait for the promise's resolution/rejection.

[[]](http://devdocs.io/javascript/global_objects/array)

Array initializer/literal syntax.

[{}](http://devdocs.io/javascript/operators/object_initializer)

Object initializer/literal syntax.

[/ab+c/i](http://devdocs.io/javascript/global_objects/regexp)

Regular expression literal syntax.

[( )](http://devdocs.io/javascript/operators/grouping)

Grouping operator.

**Left-hand-side expressions**

Left values are the destination of an assignment.

[Property accessors](http://devdocs.io/javascript/operators/property_accessors)

Member operators provide access to a property or method of an object  
(object.property and object["property"]).

[new](http://devdocs.io/javascript/operators/new)

The new operator creates an instance of a constructor.

[new.target](http://devdocs.io/javascript/operators/new.target)

In constructors, new.target refers to the constructor that was invoked by [new](http://devdocs.io/javascript/operators/new).

[super](http://devdocs.io/javascript/operators/super)

The super keyword calls the parent constructor.

[...obj](http://devdocs.io/javascript/operators/spread_operator)

The spread operator allows an expression to be expanded in places where multiple arguments (for function calls) or multiple elements (for array literals) are expected.

**Increment and decrement**

Postfix/prefix increment and postfix/prefix decrement operators.

[A++](http://devdocs.io/javascript/operators/arithmetic_operators#Increment)

Postfix increment operator.

[A--](http://devdocs.io/javascript/operators/arithmetic_operators#Decrement)

Postfix decrement operator.

[++A](http://devdocs.io/javascript/operators/arithmetic_operators#Increment)

Prefix increment operator.

[--A](http://devdocs.io/javascript/operators/arithmetic_operators#Decrement)

Prefix decrement operator.

**Unary operators**

A unary operation is operation with only one operand.

[delete](http://devdocs.io/javascript/operators/delete)

The delete operator deletes a property from an object.

[void](http://devdocs.io/javascript/operators/void)

The void operator discards an expression's return value.

[typeof](http://devdocs.io/javascript/operators/typeof)

The typeof operator determines the type of a given object.

[+](http://devdocs.io/javascript/operators/arithmetic_operators#Unary_plus)

The unary plus operator converts its operand to Number type.

[-](http://devdocs.io/javascript/operators/arithmetic_operators#Unary_negation)

The unary negation operator converts its operand to Number type and then negates it.

[~](http://devdocs.io/javascript/operators/bitwise_operators#Bitwise_NOT)

Bitwise NOT operator.

[!](http://devdocs.io/javascript/operators/logical_operators#Logical_NOT)

Logical NOT operator.

**Relational operators**

A comparison operator compares its operands and returns a Boolean value based on whether the comparison is true.

[in](http://devdocs.io/javascript/operators/in)

The in operator determines whether an object has a given property.

[instanceof](http://devdocs.io/javascript/operators/instanceof)

The instanceof operator determines whether an object is an instance of another object.

[<](http://devdocs.io/javascript/operators/comparison_operators#Less_than_operator)

Less than operator.

[>](http://devdocs.io/javascript/operators/comparison_operators#Greater_than_operator)

Greater than operator.

[<=](http://devdocs.io/javascript/operators/comparison_operators#Less_than_or_equal_operator)

Less than or equal operator.

[>=](http://devdocs.io/javascript/operators/comparison_operators#Greater_than_or_equal_operator)

Greater than or equal operator.

Note: => is not an operator, but the notation for [Arrow functions](http://devdocs.io/javascript/functions/arrow_functions).

**Equality operators**

The result of evaluating an equality operator is always of type Boolean based on whether the comparison is true.

[==](http://devdocs.io/javascript/operators/comparison_operators#Equality)

Equality operator.

[!=](http://devdocs.io/javascript/operators/comparison_operators#Inequality)

Inequality operator.

[===](http://devdocs.io/javascript/operators/comparison_operators#Identity)

Identity operator.

[!==](http://devdocs.io/javascript/operators/comparison_operators#Nonidentity)

Nonidentity operator.

**Bitwise shift operators**

Operations to shift all bits of the operand.

[<<](http://devdocs.io/javascript/operators/bitwise_operators#Left_shift)

Bitwise left shift operator.

[>>](http://devdocs.io/javascript/operators/bitwise_operators#Right_shift)

Bitwise right shift operator.

[>>>](http://devdocs.io/javascript/operators/bitwise_operators#Unsigned_right_shift)

Bitwise unsigned right shift operator.

**Binary bitwise operators**

Bitwise operators treat their operands as a set of 32 bits (zeros and ones) and return standard JavaScript numerical values.

[&](http://devdocs.io/javascript/operators/bitwise_operators#Bitwise_AND)

Bitwise AND.

[|](http://devdocs.io/javascript/operators/bitwise_operators#Bitwise_OR)

Bitwise OR.

[^](http://devdocs.io/javascript/operators/bitwise_operators#Bitwise_XOR)

Bitwise XOR.

**Binary logical operators**

Logical operators are typically used with boolean (logical) values, and when they are, they return a boolean value.

[&&](http://devdocs.io/javascript/operators/logical_operators#Logical_AND)

Logical AND.

[||](http://devdocs.io/javascript/operators/logical_operators#Logical_OR)

Logical OR.

**Conditional (ternary) operator**

[(condition ? ifTrue : ifFalse)](http://devdocs.io/javascript/operators/conditional_operator)

The conditional operator returns one of two values based on the logical value of the condition.

**Assignment operators**

An assignment operator assigns a value to its left operand based on the value of its right operand.

[=](http://devdocs.io/javascript/operators/assignment_operators#Assignment)

Assignment operator.

[\*=](http://devdocs.io/javascript/operators/assignment_operators#Multiplication_assignment)

Multiplication assignment.

[/=](http://devdocs.io/javascript/operators/assignment_operators#Division_assignment)

Division assignment.

[%=](http://devdocs.io/javascript/operators/assignment_operators#Remainder_assignment)

Remainder assignment.

[+=](http://devdocs.io/javascript/operators/assignment_operators#Addition_assignment)

Addition assignment.

[-=](http://devdocs.io/javascript/operators/assignment_operators#Subtraction_assignment)

Subtraction assignment

[<<=](http://devdocs.io/javascript/operators/assignment_operators#Left_shift_assignment)

Left shift assignment.

[>>=](http://devdocs.io/javascript/operators/assignment_operators#Right_shift_assignment)

Right shift assignment.

[>>>=](http://devdocs.io/javascript/operators/assignment_operators#Unsigned_right_shift_assignment)

Unsigned right shift assignment.

[&=](http://devdocs.io/javascript/operators/assignment_operators#Bitwise_AND_assignment)

Bitwise AND assignment.

[^=](http://devdocs.io/javascript/operators/assignment_operators#Bitwise_XOR_assignment)

Bitwise XOR assignment.

[|=](http://devdocs.io/javascript/operators/assignment_operators#Bitwise_OR_assignment)

Bitwise OR assignment.

[[a, b] = [1, 2]](http://devdocs.io/javascript/operators/destructuring_assignment)  
[{a, b} = {a:1, b:2}](http://devdocs.io/javascript/operators/destructuring_assignment)

Destructuring assignment allows you to assign the properties of an array or object to variables using syntax that looks similar to array or object literals.

**Comma operator**

[,](http://devdocs.io/javascript/operators/comma_operator)

The comma operator allows multiple expressions to be evaluated in a single statement and returns the result of the last expression.

**Non-standard features**

[Legacy generator function](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Legacy_generator_function)

The function keyword can be used to define a legacy generator function inside an expression. To make the function a legacy generator, the function body should contains at least one [yield](http://devdocs.io/javascript/operators/yield) expression.

[Expression closures](http://devdocs.io/javascript/operators/expression_closures)

The expression closure syntax is a shorthand for writing simple function.

[[for (x of y) x]](http://devdocs.io/javascript/operators/array_comprehensions)

Array comprehensions.

[(for (x of y) y)](http://devdocs.io/javascript/operators/generator_comprehensions)

Generator comprehensions.

Script Example  
<!DOCTYPE html>

<html>

<head>

<title>Sample Code</title>

</head>

<body>

<button id="button">Hello</button>

<script>

document.getElementById('button').onclick = function() {

alert('Sample Code!');

var myTextNode = document.createTextNode('This is an example.');

document.body.appendChild(myTextNode);

};

</script>

</body>

</html>