



Javascript ES6

JavaScript Keywords

JavaScript statements often start with a **keyword** to identify the JavaScript action to be performed.

Our [Reserved Words Reference](#) lists all JavaScript keywords.

Here is a list of some of the keywords you will learn about in this tutorial:

Keyword	Description
var	Declares a variable
let	Declares a block variable
const	Declares a block constant
if	Marks a block of statements to be executed on a condition
switch	Marks a block of statements to be executed in different cases
for	Marks a block of statements to be executed in a loop
function	Declares a function
return	Exits a function
try	Implements error handling to a block of statements

JavaScript keywords are reserved words. Reserved words cannot be used as names for variables.

Must be Assigned

JavaScript `const` variables must be assigned a value when they are declared:

Correct

```
const PI = 3.14159265359;
```

Incorrect

```
const PI;  
PI = 3.14159265359;
```

When to use JavaScript const?

Always declare a variable with `const` when you know that the value should not be changed.

Use `const` when you declare:

- A new Array
- A new Object
- A new Function
- A new RegExp

JavaScript Arithmetic Operators

Arithmetic Operators are used to perform arithmetic on numbers:

Arithmetic Operators Example

```
let a = 3;  
let x = (100 + 50) * a;
```

Try it Yourself »

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement



JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator



JavaScript has 8 Datatypes

1. String
2. Number
3. BigInt
4. Boolean
5. Undefined
6. Null
7. Symbol
8. Object

The Object Datatype

The object data type can contain:

1. An object
2. An array
3. A date

Examples

```
// Numbers:
let length = 16;
let weight = 7.5;

// Strings:
let color = "Yellow";
let lastName = "Johnson";

// Booleans
let x = true;
let y = false;

// Object:
const person = {firstName:"John", lastName:"Doe"};

// Array object:
const cars = ["Saab", "Volvo", "BMW"];

// Date object:
const date = new Date("2022-03-25");
```

JavaScript Objects


[< Previous](#)

[Next >](#)

Real Life Objects, Properties, and Methods

In real life, a car is an **object**.

A car has **properties** like weight and color, and **methods** like start and stop:

Object	Properties	Methods
	<div>car.name = Fiat</div> <div>car.model = 500</div> <div>car.weight = 850kg</div> <div>car.color = white</div>	<div>car.start()</div> <div>car.drive()</div> <div>car.brake()</div> <div>car.stop()</div>

All cars have the same **properties**, but the property **values** differ from car to car.

All cars have the same **methods**, but the methods are performed **at different times**.



Object Definition

You define (and create) a JavaScript object with an object literal:

Example

```
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```


[Try it Yourself »](#)

Spaces and line breaks are not important. An object definition can span multiple lines:

Example

```
const person = {  
  firstName: "John",  
  lastName: "Doe",  
  age: 50,  
  eyeColor: "blue"  
};
```

[Try it Yourself »](#)



Object Methods

Objects can also have **methods**.

Methods are **actions** that can be performed on objects.

Methods are stored in properties as **function definitions**.

Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue
fullName	function() {return this.firstName + " " + this.lastName;}

A method is a function stored as a property.

Example

```
const person = {  
  firstName: "John",  
  lastName : "Doe",  
  id       : 5566,  
  fullName : function() {  
    return this.firstName + " " + this.lastName;  
  }  
};
```

Here is a list of some common HTML events:

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

The list is much longer: [W3Schools JavaScript Reference HTML DOM Events](#).

JavaScript Event Handlers

Event handlers can be used to handle and verify user input, user actions, and browser actions:

- Things that should be done every time a page loads
- Things that should be done when the page is closed
- Action that should be performed when a user clicks a button
- Content that should be verified when a user inputs data
- And more ...

Many different methods can be used to let JavaScript work with events:

- HTML event attributes can execute JavaScript code directly
- HTML event attributes can call JavaScript functions
- You can assign your own event handler functions to HTML elements
- You can prevent events from being sent or being handled
- And more ...

JavaScript String Methods

[< Previous](#)[Next >](#)

Basic String Methods

Javascript strings are primitive and immutable: All string methods produces a new string without altering the original string.

[String.length](#)[String.charAt\(\)](#)[String.charCodeAt\(\)](#)[String.at\(\)](#)[String\[_\]](#)[String.slice\(\)](#)[String.substring\(\)](#)[String.substr\(\)](#)[String.toUpperCase\(\)](#)[String.toLowerCase\(\)](#)[String.concat\(\)](#)[String.trim\(\)](#)[String.trimStart\(\)](#)[String.trimEnd\(\)](#)[String.padStart\(\)](#)[String.padEnd\(\)](#)[String.repeat\(\)](#)[String.replace\(\)](#)[String.replaceAll\(\)](#)[String.split\(\)](#)

See Also:

[String Search Methods](#)[String Templates](#)

Template Strings allow variables in strings:

Example

```
let firstName = "John";  
let lastName = "Doe";  
  
let text = `Welcome ${firstName}, ${lastName}!`;
```

Try it Yourself »

Automatic replacing of variables with real values is called **string interpolation**.

Expression Substitution

Template Strings allow expressions in strings:

Example

```
let price = 10;  
let VAT = 0.25;  
  
let total = `Total: ${((price * (1 + VAT)).toFixed(2))}`;
```

The else if Statement

Use the `else if` statement to specify a new condition if the first condition is false.

Syntax

```
if (condition1) {  
    // block of code to be executed if condition1 is true  
} else if (condition2) {  
    // block of code to be executed if the condition1 is false and condition2 is true  
} else {  
    // block of code to be executed if the condition1 is false and condition2 is false  
}
```

Example

If time is less than 10:00, create a "Good morning" greeting, if not, but time is less than 20:00, create a "Good day" greeting, otherwise a "Good evening":

```
if (time < 10) {  
    greeting = "Good morning";  
} else if (time < 20) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}
```

The result of greeting will be:

Good day




The JavaScript Switch Statement

Use the `switch` statement to select one of many code blocks to be executed.

Syntax

```
switch(expression) {  
  case x:  
    // code block  
    break;  
  case y:  
    // code block  
    break;  
  default:  
    // code block  
}
```

This is how it works:

- The switch expression is evaluated once.
 - The value of the expression is compared with the values of each case.
 - If there is a match, the associated block of code is executed.
 - If there is no match, the default code block is executed.
- 



Different Kinds of Loops

JavaScript supports different kinds of loops:

- `for` - loops through a block of code a number of times
 - `for/in` - loops through the properties of an object
 - `for/of` - loops through the values of an iterable object
 - `while` - loops through a block of code while a specified condition is true
 - `do/while` - also loops through a block of code while a specified condition is true
- 