JavaScript if

**Summary**: in this tutorial, you will learn how to use the JavaScript if statement to execute a block when a condition is true.

Introduction to the JavaScript if statement

The if statement executes block if a condition is true. The following shows the syntax of the if statement:

if( condition )

statement;

Code language: JavaScript (javascript)

The condition can be a value or an expression. Typically, the condition evaluates to a [Boolean](https://www.javascripttutorial.net/javascript-data-types/#boolean) value, which is true or false.

If the condition evaluates to true, the if statement executes the statement. Otherwise, the if statement passes the control to the next statement after it.

The following flowchart illustrates how the if statement works:

If the condition evaluates to a non-Boolean value, JavaScript implicitly converts its result to a Boolean value by calling the [Boolean()](https://www.javascripttutorial.net/javascript-boolean/) function.

If you have more than one statement to execute, you need to use wrap them in a block using a pair of curly braces as follows:

if (condition) {

*// statements to execute*

}

Code language: JavaScript (javascript)

However, it’s a good practice to always use curly braces with the if statement. By doing this, you make your code easier to maintain and avoid possible mistakes.

JavaScript if statement examples

The following example uses the if statement to check if the age is equal to or greater than 18:

let age = 18;

if (age >= 18) {

console.log('You can sign up');

}

Code language: JavaScript (javascript)

Output:

You can sign up

Code language: JavaScript (javascript)

How it works.

First, declare and initialize the [variable](https://www.javascripttutorial.net/javascript-variables/) age to 18:

let age = 18;

Code language: JavaScript (javascript)

Second, check if the age is greater or equal to 18 using the if statement. Because the expression age >= 18 is true, the code inside the if statement executes that outputs a message to the console:

if (age >= 18) {

console.log('You can sign up');

}

Code language: JavaScript (javascript)

The following example also uses the if statement. However, the age is 16 which causes the condition to evaluate to false. Therefore, you won’t see any message in the output:

let age = 16;

if (age >= 18) {

console.log('You can sign up');

}

Code language: JavaScript (javascript)

Nested if statement

It’s possible to use an if statement inside another if statement. For example:

let age = 16;

let state = 'CA';

if (state == 'CA') {

if (age >= 16) {

console.log('You can drive.');

}

}

Code language: JavaScript (javascript)

Output:

You can drive.

Code language: JavaScript (javascript)

How it works.

First, declare and initialize the age and state variables:

let age = 16;

let state = 'CA';

Code language: JavaScript (javascript)

Second, check if the state is 'CA' using an if statement. If yes, check if the age is greater than 16 using a nested if statement and output a message to the console:

if (state == 'CA') {

if (age == 16) {

console.log('You can drive.');

}

}

Code language: JavaScript (javascript)

In practice, you should avoid using nested if statements as much as possible. For example, you can use the && operator to combine the conditions and use an if statements as follows:

let age = 16;

let state = 'CA';

if (state == 'CA' && age == 16) {

console.log('You can drive.');

}

Code language: JavaScript (javascript)

Summary

* Use the JavaScript if statement to execute a statement if a condition is true.
* Avoid using nested if statement as much as possible.

JavaScript if else

**Summary**: in this tutorial, you will learn how to use the JavaScript if...else statement to execute a block based on a condition.

Introduction to the JavaScript if…else statement

The [if](https://www.javascripttutorial.net/javascript-if/) statement executes a block if a condition is true. When the condition is false, it does nothing. But if you want to execute a statement if the condition is false, you can use an if...else statement.

The following shows the syntax of the if...else statement:

if( condition ) {

*// ...*

} else {

*// ...*

}

Code language: JavaScript (javascript)

In this syntax, the condition is a value or an expression that evaluates to true or false. If the condition is true, the if...else statement executes the block that follows the if branch.

If the condition is false, the if...else statement executes the block that follows the else branch.

Typically, the condition evaluates to a boolean value, which is true or false. However, if it evaluates to a non-boolean value, the if...else statement will convert it to the boolean value.

The following flowchart illustrates how the if...else statement works:

JavaScript if…else statement examples

The following example uses the if...else statement to check if the age is greater than or equal to 18:

let age = 18;

if (age >= 18) {

console.log('You can sign up.');

} else {

console.log('You must be at least 18 to sign up.');

}

Code language: JavaScript (javascript)

In this example, the age is 18. Therefore, the expression age >= 18 is true. Hence, you’ll see the following message in the console:

You can sign up.

The following example is the same as above except that the age is 18 instead of 16:

let age = 16;

if (age >= 18) {

console.log('You can sign up.');

} else {

console.log('You must be at least 18 to sign up.');

}

Code language: JavaScript (javascript)

Output:

You must be at least 18 to sign up.

In this example, the age is 16. Therefore, the expression age >= 18 evaluates to false. Hence, the statement in the else branch executes that output the message to the console.

The following example uses a logical operator AND (&&) as the condition in the if block:

let age = 16;

let country = 'USA';

if (age >= 16 && country === 'USA') {

console.log('You can get a driving license.');

} else {

console.log('You are not eligible to get a driving license.');

}

Code language: JavaScript (javascript)

Because the age is 16 and the country is the USA, the following expression returns true.

age >= 16 && country === 'USA'

Code language: JavaScript (javascript)

And you see the following output:

You can get a driving license.

Code language: JavaScript (javascript)

Summary

* Use the JavaScript if...else statement to execute a block if a condition is true and another block otherwise.

# JavaScript if else if

**Summary**: In this tutorial, you will learn how to use the JavaScript if...else...if statement to check multiple conditions and execute the corresponding block if a condition is true.

## Introduction to the JavaScript if else if statement

The [if](https://www.javascripttutorial.net/javascript-if/) an [if…else](https://www.javascripttutorial.net/javascript-if-else/) statements accept a single condition and execute the if or else block accordingly based on the condition.

To check multiple conditions and execute the corresponding block if a condition is true, you use the if...else...if statement like this:

if (condition1) {

*// ...*

} else if (condition2) {

*// ...*

} else if (condition3) {

*//...*

} else {

*//...*

}

Code language: JavaScript (javascript)

In this syntax, the if...else...if statement has three conditions. In theory, you can have as many conditions as you want to, where each else...if branch has a condition.

The if...else...if statement checks the conditions from top to bottom and executes the corresponding block if the condition is true.

The if...else...if statement stops evaluating the remaining conditions as soon as a condition is true. For example, if the condition2 is true, the if...else...if statement won’t evaluate the condition3.

If all the conditions are false, the if...else...if statement executes the block in the else branch.

The following flowchart illustrates how the if...else...if statement works:

## JavaScript if else if examples

Let’s take some examples of using the if...else...if statement.

### **1) A simple JavaScript if…else…if statement example**

The following example uses the if...else...if statement to get the month name from a month number:

let month = 6;

let monthName;

if (month == 1) {

monthName = 'Jan';

} else if (month == 2) {

monthName = 'Feb';

} else if (month == 3) {

monthName = 'Mar';

} else if (month == 4) {

monthName = 'Apr';

} else if (month == 5) {

monthName = 'May';

} else if (month == 6) {

monthName = 'Jun';

} else if (month == 7) {

monthName = 'Jul';

} else if (month == 8) {

monthName = 'Aug';

} else if (month == 9) {

monthName = 'Sep';

} else if (month == 10) {

monthName = 'Oct';

} else if (month == 11) {

monthName = 'Nov';

} else if (month == 12) {

monthName = 'Dec';

} else {

monthName = 'Invalid month';

}

console.log(monthName);

Code language: JavaScript (javascript)

Output:

Jun

In this example, we compare the month with 12 numbers from 1 to 12 and assign the corresponding month name to the monthName variable.

Since the month is 6, the expression month==6 evaluates to true. Therefore, the if...else...if statement assigns the literal string 'Jun' to the monthName variable. Therefore, you see Jun in the console.

If you change the month to a number that is not between 1 and 12, you’ll see the Invalid Month in the console because the else clause will execute.

### **2) Using JavaScript if…else…if statement to calculate the body mass index**

The following example calculates a body mass index (BMI) of a person. It uses the if...else...if statement to determine the weight status based on the BMI:

let weight = 70; *// kg*

let height = 1.72; *// meter*

*// calculate the body mass index (BMI)*

let bmi = weight / (height \* height);

let weightStatus;

if (bmi < 18.5) {

weightStatus = 'Underweight';

} else if (bmi >= 18.5 && bmi <= 24.9) {

weightStatus = 'Healthy Weight';

} else if (bmi >= 25 && bmi <= 29.9) {

weightStatus = 'Overweight';

} else {

weightStatus = 'Obesity';

}

console.log(weightStatus);

Code language: JavaScript (javascript)

Output:

Healthy Weight

How it works.

* First, declare two variables that hold the weight in kilogram and height in meter. In a realworld application, you’ll get these values from a web form.
* Second, calculate the body mass index by dividing the weight by the square of the height.
* Third, determine the weight status based on the BMI using the if...else..if statement.
* Finally, output the weight status to the console.

## Summary

* Use the JavaScript if...else...if statement to check multiple conditions and execute the corresponding block if a condition is true.

# JavaScript Ternary Operator



**Summary**: in this tutorial, you will learn how to use the JavaScript ternary operator to make your code more concise.

## Introduction to JavaScript ternary operator

When you want to execute a block if a condition evaluates to true, you often use an [if…else](https://www.javascripttutorial.net/javascript-if-else/) statement. For example:

let age = 18;

let message;

if (age >= 16) {

message = 'You can drive.';

} else {

message = 'You cannot drive.';

}

console.log(message);

Code language: JavaScript (javascript)

In this example, we show a message that a person can drive if the age is greater than or equal to 16. Alternatively, you can use a ternary operator instead of the [if-else](https://www.javascripttutorial.net/javascript-if-else/) statement like this:

let age = 18;

let message;

age >= 16 ? (message = 'You can drive.') : (message = 'You cannot drive.');

console.log(message);

Code language: JavaScript (javascript)

Or you can use the ternary operator in an expression as follows:

let age = 18;

let message;

message = age >= 16 ? 'You can drive.' : 'You cannot drive.';

console.log(message);

Code language: JavaScript (javascript)

Here’s the syntax of the ternary operator:

condition ? expressionIfTrue : expressionIfFalse;

Code language: JavaScript (javascript)

In this syntax, the condition is an expression that evaluates to a Boolean value, either true or false.

If the condition is true, the first expression (expresionIfTrue) executes. If it is false, the second expression (expressionIfFalse) executes.

The following shows the syntax of the ternary operator used in an expression:

let variableName = condition ? expressionIfTrue : expressionIfFalse;

Code language: JavaScript (javascript)

In this syntax, if the condition is true, the variableName will take the result of the first expression (expressionIfTrue) or expressionIfFalse otherwise.

## JavaScript ternary operator examples

Let’s take some examples of using the ternary operator.

### **1) Using the JavaScript ternary operator to perform multiple statements**

The following example uses the ternary operator to perform multiple operations, where each operation is separated by a comma. For example:

let authenticated = true;

let nextURL = authenticated

? (alert('You will redirect to admin area'), '/admin')

: (alert('Access denied'), '/403');

*// redirect to nextURL here*

console.log(nextURL); *// '/admin'*

Code language: JavaScript (javascript)

In this example, the returned value of the ternary operator is the last value in the comma-separated list.

### **2) Simplifying ternary operator example**

See the following example:

let locked = 1;

let canChange = locked != 1 ? true : false;

Code language: JavaScript (javascript)

If the locked is 1, then the canChange variable is set to false, otherwise, it is set to true. In this case, you can simplify it by using a Boolean expression as follows:

let locked = 1;

let canChange = locked != 1;

Code language: JavaScript (javascript)

### **3) Using multiple JavaScript ternary operators example**

The following example shows how to use two ternary operators in the same expression:

let speed = 90;

let message = speed >= 120 ? 'Too Fast' : speed >= 80 ? 'Fast' : 'OK';

console.log(message);

Code language: JavaScript (javascript)

Output:

Fast

Code language: JavaScript (javascript)

It’s a good practice to use the ternary operator when it makes the code easier to read. If the logic contains many if...else statements, you should avoid using the ternary operators.

## Summary

* Use the JavaScript ternary operator (?:)to make the code more concise.

# JavaScript switch case

**Summary**: in this tutorial, you will learn how to use the JavaScript switch statement to execute a block based on multiple conditions.

## Introduction to the JavaScript switch case statement

The switch statement evaluates an expression, compares its result with case values, and executes the statement associated with the matching case value.

The following illustrates the syntax of the switch statement:

switch (expression) {

case value1:

statement1;

break;

case value2:

statement2;

break;

case value3:

statement3;

break;

default:

statement;

}

Code language: JavaScript (javascript)

How it works.

* First, evaluate the expression inside the parentheses after the switch keyword.
* Second, compare the result of the expression with the value1, value2, … in the case branches from top to bottom. The switch statement uses the strict comparison (===).
* Third, execute the statement in the case branch where the result of the expression equals the value that follows the case keyword. The [break](https://www.javascripttutorial.net/javascript-break/) statement exits the switch statement. If you skip the break statement, the code execution falls through the original case branch into the next one. If the result of the expression does not strictly equal to any value, the switch statement will execute the statement in the default branch.

That the switch statement will stop comparing the expression‘s result with the remaining case values as long as it finds a match.

The switch statement is like the [if…else…if](https://www.javascripttutorial.net/javascript-if-else-if/) statement. But it has more readable syntax.

The following flowchart illustrates the switch statement:

In practice, you often use a switch statement to replace a complex if...else...if statement to make the code more readable.

Technically, the switch statement is equivalent to the following  if...else...if statement:

if (expression === value1) {

statement1;

} else if (expression === value2) {

statement2;

} else if (expression === value3) {

statement3;

} else {

statement;

}

Code language: JavaScript (javascript)

## JavaScript switch case examples

Let’s take some examples of using the JavaScript switch statement.

### **1) Using JavaScript switch statement to get the day of the week**

The following example uses the switch statement to get the day of the week based on a day number:

let day = 3;

let dayName;

switch (day) {

case 1:

dayName = 'Sunday';

break;

case 2:

dayName = 'Monday';

break;

case 3:

dayName = 'Tuesday';

break;

case 4:

dayName = 'Wednesday';

break;

case 5:

dayName = 'Thursday';

break;

case 6:

dayName = 'Friday';

break;

case 7:

dayName = 'Saturday';

break;

default:

dayName = 'Invalid day';

}

console.log(dayName); *// Tuesday*

Code language: JavaScript (javascript)

Output:

Tuesday

How it works.

First, declare the day variable that holds the day number and the day name variable (dayName).

Second, get the day of the week based on the day number using the switch statement. If the day is 1, the day of the week is Sunday. If the day is 2, the day of the week is Monday, and so on.

Third, output the day of the week to the console.

### **2) Using the JavaScript switch statement to get the day count based of a month**

The following example uses the switch statement to get the day count of a month:

let year = 2016;

let month = 2;

let dayCount;

switch (month) {

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

dayCount = 31;

break;

case 4:

case 6:

case 9:

case 11:

dayCount = 30;

break;

case 2:

*// leap year*

if ((year % 4 == 0 && !(year % 100 == 0)) || year % 400 == 0) {

dayCount = 29;

} else {

dayCount = 28;

}

break;

default:

dayCount = -1; *// invalid month*

}

console.log(dayCount); *// 29*

Code language: JavaScript (javascript)

In this example, we have four cases:

* If the month is 1, 3,5, 7, 8, 10, or 12, the number of days in a month is 31.
* If the month is 4, 6, 9, or 11, the number of days in that month is 30.
* If the month is 2, and the year is not the leap year, the number of days is 28. If the year is the leap year, the number of days is 29.
* If the month is not in the valid range (1-12), the default branch executes and sets the dayCount variable to -1, which indicates the invalid month.

## Summary

* The switch statement evaluates an expression, compare its result with case values, and execute the statement associated with the matching case.
* Use the switch statement to rather than a complex if...else...if statement to make the code more redable.
* The switch statement uses the strict comparison (===) to compare the expression with the case values.

JavaScript while Loop

**Summary**: in this tutorial, you will learn how to use the JavaScript while statement to create a loop that executes a block as long as a condition is true .

Introduction to the JavaScript while loop statement

The JavaScript while statement creates a loop that executes a block as long as a condition evaluates to true.

The following illustrates the syntax of the while statement:

while (expression) {

*// statement*

}

Code language: JavaScript (javascript)

The while statement evaluates the expression before each iteration of the loop.

If the expression evaluates to true, the while statement executes the statement. Otherwise, the while loop exits.

Because the while loop evaluates the expression before each iteration, it is known as a pretest loop.

If the expression evaluates to false before the loop enters, the while loop will never execute.

The following flowchart illustrates the while loop statement:

Note that if you want to execute the statement a least once and check the condition after each iteration, you should use the [do…while](https://www.javascripttutorial.net/javascript-do-while/) statement.

JavaScript while loop example

The following example uses the while statement to output the odd numbers between 1 and 10 to the console:

let count = 1;

while (count < 10) {

console.log(count);

count +=2;

}

Code language: JavaScript (javascript)

Output:

1

3

5

7

9

How the script works

* First, declare and initialize the count variable to 1.
* Second, execute the statement inside the loop if the count variable is less than 10. In each iteration, ouput the count to the console and increase the count by 2.
* Third, after 5 iterations, the count is 11. Therefore, the condition count < 10 is false, the loop exits.

Summary

* Use a while loop statement to create a loop that executes a block as long as a condition evaluates to true.

# JavaScript do…while Loop

**Summary**: in this tutorial, you will learn how to use the JavaScript do...while statement to create a loop that executes a block until a condition is false.

## Introduction to the JavaScript do…while statement

The do...while loop statement creates a loop that executes a block until a condition evaluates to false. The following statement illustrates the syntax of the do...while loop:

do {

statement;

} while(expression);

Code language: JavaScript (javascript)

Unlike the [while](https://www.javascripttutorial.net/javascript-while-loop/) loop, the do-while loop always executes the statement at least once before evaluating the expression.

Because the do...while loop evaluates expression after each iteration, it’s often referred to as a post-test loop.

Inside the loop body, you need to make changes to some [variables](https://www.javascripttutorial.net/javascript-variables/) to ensure that the expression is false after some iterations. Otherwise, you’ll have an indefinite loop.

Note that starting with ES6+, the trailing semicolon (;) after the while(expression) is optional. So you can use the following syntax:

do {

statement;

} while(expression)

Code language: JavaScript (javascript)

The following flowchart illustrates the do-while loop statement:

In practice, you often use the do...while statement when you want to execute the loop body at least once before checking the condition.

## JavaScript do while statement examples

Let’s take some examples of using the do...while statement.

### **1) Simple JavaScript do while statement example**

The following example uses the do...while statement to output five numbers from 0 to 4 to the console:

let count = 0;

do {

console.log(count);

count++;

} while (count < 5)

Code language: JavaScript (javascript)

Output:

0

1

2

3

4

In this example:

* First, declare and initialize the count variable to zero.
* Second, show the count and increase its value by one in each iteration until its value is greater or equal to 5.

### **2) Using the JavaScript do while statement to make a simple number guessing game**

The following example uses the do...while statement to generate a number guessing game.

The script generates a random integer between 1 and 10. And you have to make a number of guesses until your number matches the random number.

*// generate a secret number between 1 and 10*

const MIN = 1;

const MAX = 10;

let secretNumber = Math.floor(Math.random() \* (MAX - MIN + 1)) + MIN;

let guesses = 0; *// for storing the number of guesses*

let hint = ''; *// for storing hint*

let number = 0;

do {

*// get input from user*

let input = prompt(`Please enter a number between ${MIN} and ${MAX}` + hint);

*// get the integer*

number = parseInt(input);

*// increase the number of guesses*

guesses++;

*// check input number with the secret number provide hint if needed*

if (number > secretNumber) {

hint = ', and less than ' + number;

} else if (number < secretNumber) {

hint = ', and greater than ' + number;

} else if (number == secretNumber) {

alert(`Bravo! you're correct after ${guesses} guess(es).`);

}

} while (number != secretNumber);

Code language: JavaScript (javascript)

How it works.

First, declare the MIN and MAX constants and initialize their values to 1 and 10:

const MIN = 1;

const MAX = 10;

Code language: JavaScript (javascript)

Second, use Math.random() function to generate a random floating-point number with the value in the range of 0 and 1 (inclusive of zero but not one).

To generate a random number between MIN and MAX (exclusive), you use the following expression:

Math.random() \* (MAX - MIN + 1)

Code language: JavaScript (javascript)

However, the result is a floating-point number. Therefore, you need to use the Math.floor() function to convert it to an integer:

Math.floor(Math.random() \* (MAX - MIN + 1))

Code language: JavaScript (javascript)

To generate a random number between min and max, you use the following expression:

let secretNumber = Math.floor(Math.random() \* (MAX - MIN + 1)) + MIN;

Code language: JavaScript (javascript)

Third, define three variables for storing the number of guesses, hints, and user’s input number:

let guesses = 0; *// for storing the number of guesses*

let hint = ''; *// for storing hint*

let number = 0;

Code language: JavaScript (javascript)

Fourth, use the input() function to get the input from the user:

let input = prompt(`Please enter a number between ${MIN} and ${MAX}` + hint);

Code language: JavaScript (javascript)

Note that the input() function only works on web browsers. If you run the code in another environment such as node.js, please check the corresponding function.

The input() function returns a string, therefore, you need to use the parseInt() function to convert it to an integer:

number = parseInt(input);

Code language: JavaScript (javascript)

Fifth, increase the number of guesses in each iteration:

guesses++;

Sixth, check the input number with the secret number (random) number and give hint. If the numbers are matched, show the message using the [alert()](https://www.javascripttutorial.net/javascript-bom/javascript-alert/) function:

if (number > secretNumber) {

hint = ', and less than ' + number;

} else if (number < secretNumber) {

hint = ', and greater than ' + number;

} else if (number == secretNumber) {

alert(`Bravo! you're correct after ${guesses} guess(es).`);

}

Code language: JavaScript (javascript)

Seventh, perform the next iteration until the number matches the secret number.

while (number != secretNumber);

Code language: JavaScript (javascript)

## Summary

* Use the do…while statement to create a loop that executes a code block until a condition is false.

# JavaScript for Loop

**Summary**: in this tutorial, you will learn how to use the JavaScript for loop statement to create a loop with various options.

## Introduction to the JavaScript for loop statement

The for loop statement creates a loop with three optional expressions. The following illustrates the syntax of the for loop statement:

for (initializer; condition; iterator) {

*// statements*

}

Code language: JavaScript (javascript)

### **1) iterator**

The for statement executes the initializer only once the loop starts. Typically, you declare and initialize a local loop variable in the initializer.

### **2) condition**

The condition is a boolean expression that determines whether the for should execute the next iteration.

The for statement evaluates the condition before each iteration. If the condition is true (or is not present), it executes the next iteration. Otherwise, it’ll end the loop.

### **3) iterator**

The for statement executes the iterator after each iteration.

The following flowchart illustrates the for loop:

In the for loop, the three expressions are optional. The following shows the for loop without any expressions:

for ( ; ; ) {

*// statements*

}

Code language: JavaScript (javascript)

## JavaScript for loop examples

Let’s take some examples of using the for loop statement.

### **1) A simple JavaScript for loop example**

The following example uses the for loop statement to show numbers from 1 to 4 to console:

for (let i = 1; i < 5; i++) {

console.log(i);

}

Code language: JavaScript (javascript)

Output:

1

2

3

4

How it works.

* First, declare a variable  counter and initialize it to 1.
* Second, display the value of counter in the console if counter is less than 5.
* Third, increase the value of counter by one in each iteration of the loop.

### **2) Using the JavaScript for loop without the initializer example**

The following example uses a for loop that has no initializer expression:

let j = 1;

for (; j < 10; j += 2) {

console.log(j);

}

Code language: JavaScript (javascript)

Output:

1

3

5

7

9

### **3) Using the JavaScript for loop without the condition example**

Similar to the initializer expression, the condition expression is optional. If you omit the condition expression, you need to use a [break](https://www.javascripttutorial.net/javascript-break/) statement to terminate the loop.

for (let j = 1; ; j += 2) {

console.log(j);

if (j > 10) {

break;

}

}

Code language: JavaScript (javascript)

Output:

1

3

5

7

9

11

### **3) Using the JavaScript for loop statement without any expression example**

All three expressions of the for loop statements are optional. Therefore, you can omit all of them. For example:

let j = 1;

for (;;) {

if (j > 10) {

break;

}

console.log(j);

j += 2;

}

Code language: JavaScript (javascript)

Output:

1

3

5

7

9

### **4) Using the JavaScript for loop without the loop body example**

JavaScript allows the for statement to have an empty statement. In this case, you place a semicolon (;) immediately after the for statement.

For example, the following uses a for loop to calculate the sum of 10 numbers from 1 to 10:

let sum = 0;

for (let i = 0; i <= 9; i++, sum += i);

console.log(sum);

Code language: JavaScript (javascript)

Output:

55

## Summary

* Use the JavaScript for statement to create a loop that execute a block based using various options.

# JavaScript break

**Summary**: in this tutorial, you’ll learn how to use the JavaScript break statement to terminate a loop prematurely.

## The label statement

In JavaScript, you can label a statement for later use. Here’s the syntax of the label statement:

label: statement;

Code language: HTTP (http)

In this syntax, the label can be any valid identifier. For example, the following shows how to label a for loop using the outer label:

outer: for (let i = 0; i < 5; i++) {

console.log(i);

}

Code language: JavaScript (javascript)

Once defining a label, you can reference it in the break or [continue](https://www.javascripttutorial.net/javascript-continue/) statement.

## Introduction to JavaScript break statement

The break statement prematurely terminates a loop such as [for](https://www.javascripttutorial.net/javascript-for-loop/), [do...while](https://www.javascripttutorial.net/javascript-do-while/), and [while](https://www.javascripttutorial.net/javascript-while-loop/) loop, a [switch](https://www.javascripttutorial.net/javascript-switch-case/), or a label statement. Here’s the syntax of the break statement:

break [label];

Code language: JavaScript (javascript)

In this syntax, the label is optional if you use the break statement in a loop or switch. However, if you use the break statement with a label statement, you need to specify it.

This tutorial focuses on how to use the break statement to terminate the loop prematurely.

## Using JavaScript break statement in a for loop

The following for loop statement outputs five numbers from 0 to 4:

for (let i = 0; i < 5; i++) {

console.log(i);

}

Code language: JavaScript (javascript)

Output:

0

1

2

3

4

To terminate the for loop prematurely, you can use a break statement. For example, the following illustrates how to use a break statement inside a for loop:

for (let i = 0; i < 5; i++) {

console.log(i);

if (i == 2) {

break;

}

}

Code language: JavaScript (javascript)

Output:

0

1

2

In this example, we use an [if](https://www.javascripttutorial.net/javascript-if/) statement inside the loop. If the current value of i is 2, the if statement executes the break statement that terminates the loop.

This flowchart illustrates how the break statement works in a for loop:

### **Using the break statement to terminate a nested loop**

A nested loop has one loop inside another. For example, the following uses a nested for loop to output a pair of numbers from 1 to 3:

for (let i = 1; i <= 3; i++) {

for (let j = 1; j <= 3; j++) {

console.log(i, j);

}

}

Code language: JavaScript (javascript)

Output:

1 1

1 2

1 3

2 1

2 2

2 3

3 1

3 2

3 3

If you use a break statement inside an inner loop, it only terminates the enclosing loop. For example:

for (let i = 1; i <= 3; i++) {

for (let j = 1; j <= 3; j++) {

if (i + j == 4) {

break;

}

console.log(i, j);

}

}

Code language: JavaScript (javascript)

Output:

1 1

1 2

2 1

In this example, if the sum of i and j is 4, the break statement terminates the inner loop. To terminate the nested loop, you use a label statement. For example:

outer: for (let i = 1; i <= 3; i++) {

for (let j = 1; j <= 3; j++) {

if (i + j == 4) {

break outer;

}

console.log(i, j);

}

}

Code language: JavaScript (javascript)

Output:

1 1

1 2

In this example, we label the outer loop with the label outer. Inside the inner loop, we specify the outer label in the break statement. The break statement to terminate the nested loop if the sum of i and j is 4.

## Using JavaScript break statement in a while loop

The following output five numbers from 1 to 5 to the console using a while loop:

let i = 0;

while (i < 5) {

i++;

console.log(i);

}

Code language: JavaScript (javascript)

Output:

1

2

3

4

5

Like a for loop, the break statement terminates a while loop prematurely. For example:

let i = 0;

while (i < 5) {

i++;

console.log(i);

if (i == 3) {

break;

}

}

Code language: JavaScript (javascript)

Output:

1

2

3

In this example, when the current value of i is 3, the break statement terminates the loop. Therefore, you see only three numbers in the output.

The following flowchart illustrates how the break statement works in a while loop:

## Using JavaScript break statement in a do…while loop

The following example uses a do...while statement to output five numbers from 0 to 5 to the console:

let i = 0;

do {

i++;

console.log(i);

} while (i < 5);

Code language: JavaScript (javascript)

Output:

1

2

3

4

5

Like a while loop, you can use a break statement to terminate a do...while loop. For example:

let i = 0;

do {

i++;

console.log(i);

if (i == 3) {

break;

}

} while (i < 5);

Code language: JavaScript (javascript)

Output:

1

2

3

The following flowchart shows how the break statement works in a do while loop:

## Summary

* Use the break statement to terminate a loop including for, while, and do...while prematurely.
* When used in a nested loop, the break statement terminates the enclosing loop. To terminate the nested loop, you use a label statement.

JavaScript continue

**Summary**: in this tutorial, you will learn how to use the JavaScript continue statement to skip the current iteration of a loop.

Introduction to the JavaScript continue statement

The continue statement terminates the execution of the statement in the current iteration of a loop such as a [for](https://www.javascripttutorial.net/javascript-for-loop/), [while](https://www.javascripttutorial.net/javascript-while-loop/), and [do…while](https://www.javascripttutorial.net/javascript-do-while/) loop and immediately continues to the next iteration.

Here’s the syntax of the continue statement:

continue [label];

Code language: JavaScript (javascript)

In this syntax, the label is optional. It is a valid identifier associated with the label of a statement. Read the [break](https://www.javascripttutorial.net/javascript-break/) statement tutorial for more information on the label statement.

Typically, you use the continue with an [if](https://www.javascripttutorial.net/javascript-if/) statement like this:

*// inside a loop*

if(condition){

continue;

}

Code language: JavaScript (javascript)

In this syntax, the if statement specifies a condition to execute the continue statement inside a loop.

Using the continue statement in a for loop

When using the continue statement in a for loop, it doesn’t terminate the loop entirely. Instead, it jumps to the iterator expression.

The following flowchart illustrates how the continue statement works in a for loop:

JavaScript continue with for loop

The following example uses a continue in a for loop to display the odd number in the console:

for (let i = 0; i < 10; i++) {

if (i % 2 === 0) {

continue;

}

console.log(i);

}

Code language: JavaScript (javascript)

Output:

1

3

5

7

9

In this example, the for loop iterates over the numbers from 0 to 9.

The i%2 returns the remainder of the division of the current value of i by 2.

If the remainder is zero, the if statement executes the continue statement that skips the current iteration of the loop and jumps to the iterator expression i++. Otherwise, it outputs the value of i to the console.

Using the continue statement in a while loop

When using the continue statement in a while loop, it doesn’t terminate the execution of the loop entirely. Instead, it jumps back to the condition.

The following flowchart show hows the continue statement works in a while loop statement:

The following example uses the continue statement in a while loop to display the odd numbers from 1 to 10:

let i = 0;

while (i < 10) {

i++;

if (i % 2 === 0) {

continue;

}

console.log(i);

}

Code language: JavaScript (javascript)

Output:

1

3

5

7

9

Using the continue statement with a label example

The continue statement can include an optional label like this:

continue label;

Code language: JavaScript (javascript)

The following nested loop displays pairs of numbers from 1 to 2:

for (let i = 1; i < 3; i++) {

for (let j = 1; j < 3; j++) {

console.log(i, j);

}

}

Code language: JavaScript (javascript)

Output:

1 1

1 2

2 1

2 2

The following shows how to use the continue statement with a label:

outer: for (let i = 1; i < 4; i++) {

for (let j = 1; j < 4; j++) {

if (i + j == 3) continue outer;

console.log(i, j);

}

}

Code language: JavaScript (javascript)

Output:

1 1

3 1

3 2

3 3

Summary

* Use the JavaScript continue statement to skip the current iteration of a loop and continue the next one.

JavaScript Comma Operator

**Summary**: in this tutorial, you’ll learn about the JavaScript comma operator and its usage.

Introduction to the JavaScript comma operator

JavaScript uses a comma (,) to represent the comma operator. A comma operator takes two expressions, evaluates them from left to right, and returns the value of the right expression.

Here’s the syntax of the comma operator:

leftExpression, rightExpression

For example:

let result = (10, 10 + 20);

console.log(result);

Code language: JavaScript (javascript)

Output:

30

In this example, the 10, 10+20 returns the value of the right expression, which is 10+20. Therefore, the result value is 30.

See the following example:

let x = 10;

let y = (x++, x + 1);

console.log(x, y);

Code language: JavaScript (javascript)

Output:

11 12

In this example, we increase the value of x by one (x++), add one to x (x+1) and assign x to y. Therefore, x is 11, and y is 12 after the statement.

However, to make the code more explicit, you can use two statements rather than one statement with a comma operator like this:

let x = 10;

x++;

let y = x + 1;

console.log(x, y);

Code language: JavaScript (javascript)

This code is more explicit.

In practice, you might want to use the comma operator inside a [for](https://www.javascripttutorial.net/javascript-for-loop/) loop to update multiple variables each time through the loop.

The following example uses the comma operator in a for loop to display an array of nine elements as a matrix of 3 rows and three columns:

let board = [1, 2, 3, 4, 5, 6, 7, 8, 9];

let s = '';

for (let i = 0, j = 1; i < board.length; i++, j++) {

s += board[i] + ' ';

if (j % 3 == 0) {

console.log(s);

s = '';

}

}

Code language: JavaScript (javascript)

Output:

1 2 3

4 5 6

7 8 9

Summary

* A comma operator takes two expressions and evaluates them from left to right, and returns the value of the right expression.
* Use the comma operator (,) inside a for loop to update multiple variables once.
* Use two statements rather than the comma operator elsewhere to make the code more explicit and easier to understand.