Dashboard > Tutorials > 10 Days of Javascript > Day 2: Loops

Table Of Contents

for
while
do-while
for-in
for-of

Day 2: Loops

Problem Submissions Leaderboard Discussions Editorial Tutorial

Loops

JavaScript Loops

Loops are a quick and easy way to repeatedly perform a series of instructions, and they are typically run a finite number of times. JavaScript has the following types of loops:

- *for*
- *while*
- *do-while*
- *for-in*
- *for-of*

for

The *for* statement creates a loop that consists of three optional expressions, enclosed in parentheses and separated by semicolons, followed by one or more statements that will be executed in the loop.

Basic Syntax

```
for (initialization; condition; finalExpression) {
    statement(s);
}
```

Components

- $initialization$: An expression or variable declaration that is typically used to initialize a counter variable.
- $condition$: This is the *termination condition*, which is an expression that's evaluated before each pass through the loop. If this expression evaluates to *true*, then $statement$ is executed. If the expression evaluates to *false*, execution jumps to the first line of code after the end of the loop. If this statement is omitted, then $condition$ always evaluates to *true*.
- $finalExpression$: An expression to be evaluated at the end of each loop iteration. This occurs before the next evaluation of $condition$.
- $statement$: The statement (or statements) that is executed each time $condition$ evaluates to *true*.

It's important to note that:

- The $initialization$, $condition$, and $finalExpression$ in the head of the *for* loop

Go to Top

are *optional*, but are generally always used.

- The head of a for loop typically looks like `for (var i = 0; i < maxValue; i++)`, where $maxValue$ is the maximum value you wish to iterate until.

- EXAMPLE

Print all the integers in the range from 1 to some number given as input.

```
process.stdin.on('data', function (data) {
    main(+(data));
});
/**** Ignore above this line. ****/

function main(input) {
    for (var i = 1; i <= input; i++) {
        process.stdout.write(i + " ");
    }
```

Input

```
10
```

Run

Output

Initialize

In this example, we omit the $initialization$ expression and instead initialize the variable used in $condition$ and $finalExpression$ before our loop:

```
process.stdin.on('data', function (data) {
    main(+(data));
});
/**** Ignore above this line. ****/

function main(input) {
    var i = 1;

    for (; i <= input; i++) {
        process.stdout.write(i + " ");
    }
```

Input

```
10
```

Run

Output

Condition

In this example, we omit the $condition$ expression and instead add an *if* statement inside the loop that terminates the loop once a the condition `i > input` is satisfied:

```
process.stdin.on('data', function (data) {
    main(+(data));
});
/**** Ignore above this line. ****/

function main(input) {

    for (var i = 1;; i++) {
        if (i > input) {
```

Table Of Contents

for

while

do-while

for-in

for-of

Go to Top

```
            break;
        }

        process.stdout.write(i + " ");
    }
```

Input

10

Run

Output

Infinite Loop

If we omit all three blocks, our loop will run infinitely or until such a time as we call `break;` from inside the loop. In this example, we do just that:

```
process.stdin.on('data', function (data) {
    main(+(data));
});
/**** Ignore above this line. ****/

function main(input) {
    var i = 1;

    for (;;) {
        if (i > input) {
            break;
        }

        process.stdout.write(i + " ");
        i++;
    }
```

Input

10

Run

Output

while

The *while* statement creates a loop that executes its internal statement(s) as long as the specified ***condition*** evaluates to *true*. The condition is evaluated before executing the statement.

Basic Syntax

```
while (condition) {
    statement(s);
}
```

- ***condition***: This is the *termination condition*, which is an expression that's evaluated before each pass through the loop. If this expression evaluates to *true*, then ***statement*** is executed; if it evaluates to *false*, execution jumps to the first line of code after the end of the loop.
- ***statement***: The statement (or statements) that is executed each time ***condition*** evaluates to *true*.

Table Of Contents

for

while

do-while

for-in

for-of

Go to Top

- EXAMPLE

Print all the integers from 1 to 10.

```
process.stdin.on('data', function (data) {
    main(+(data));
});
/**** Ignore above this line. ****/

function main(input) {
    var i = 1;

    while (i <= input) {
        process.stdout.write(i + " ");

        i++;
    }
```

Input

```
10
```

Run

Output

do-while

The *do-while* statement creates a loop that executes its internal statement(s) until the specified $condition$ evaluates to false. The condition is evaluated after executing the internal statement(s), so the contents of the loop always execute *at least* once.

Basic Syntax

```
do {
    statement(s);
} while (condition);
```

- $condition$: This is the *termination condition*, and it's evaluated *after* each pass through the loop (meaning the loop will always run at least once). Once the statement(s) inside the loop is executed, $condition$ is evaluated. If this expression evaluates to *true*, then $statement$ is executed again; if it evaluates to *false*, execution jumps to the first line of code after the end of the loop.
- $statement$: The statement (or statements) that is executed each time $condition$ evaluates to *true*.

- EXAMPLE

Print all the integers in the range from 1 to some number given as input.

```
process.stdin.on('data', function (data) {
    main(+(data));
});
/**** Ignore above this line. ****/

function main(input) {
    var i = 1;

    do {
        process.stdout.write(i + " ");

        i++;
    } while (i <= input);
}
```

Table Of Contents

for

while

do-while

for-in

for-of

Go to Top

Input

10

Run

Output

for-in

This loop iterates (in an arbitrary order) over the *name* of each enumerable property in an object, allowing statements to be executed for each distinct property.

Basic Syntax

```
for (var variable in object) {
    // insert code that uses variable here
}
```

- $variable$: A variable that refers to a different property *name* during each iteration of the loop. You can declare this with `var` or `let`.
- $object$: The object whose enumerable properties are being iterated through.

- EXAMPLE

In the code below, we create an object (referenced by the $actress$ variable) and iterate over its enumerable properties:

```
var actress = {
    firstName: "Julia",
    lastName: "Roberts",
    dateOfBirth: "October 28, 1967",
    nationality: "American",
    firstMovie: "Satisfaction"
};

for (var property in actress) {
    console.log("actress." + property + " = " + actress[property]);
```

Output

Run

The code above produces the following output:

```
actress.firstName = Julia
actress.lastName = Roberts
actress.dateOfBirth = October 28, 1967
actress.nationality = American
actress.firstMovie = Satisfaction
```

In this code, we create a *Monster* object named $monster$, then print the object followed by its individual properties.

Input Format

The first line contains a string, $name$, denoting the type of monster.
The second line contains a string, $home$, denoting the location where the monster lives.
The third line contains a string, $description$, describing the monster.

```
'use strict';
```

Table Of Contents

for

while

do-while

for-in

for-of

Go to Top

```
process.stdin.on('data', function (data) {
    main(String(data).trim().split(new RegExp("[\n]+")));
});
/**** Ignore above this line. ****/

class Monster {
    constructor(name, home, description) {
        this.name = name;
        this.home = home;
        this.description = description;
    }
}

function main(input) {
    var monster = new Monster(input[0], input[1], input[2]);

    // Print array
    console.log(monster);

    // Print each of its elements on a new line
    for (let property in monster) {
        console.log(property + ": " + monster[property]);
    }
```

Input

```
Minotaur
Labyrinth
```

Run

Output

The code above produces the following output for the given input:

```
Monster {
  name: 'Minotaur',
  home: 'Labyrinth',
  description: 'Bull head, man body.' }
name: Minotaur
home: Labyrinth
description: Bull head, man body.
```

Table Of Contents

for

while

do-while

for-in

for-of

for-of

This loop iterates over iterable objects such as an *Array*, *Map*, *Set*, *String*, *TypedArray*, *arguments object*, etc. It essentially iterates over the *value* of each distinct property in the structure, such as each letter in a word or each element in an array.

Basic Syntax

```
for (let variable of iterable) {
    statement(s);
}
```

- ***variable***: A variable that refers to a different property *value* during each iteration of the loop. You can declare this with `var` or `let`.
- ***object***: The object whose enumerable properties are being iterated through.

- EXAMPLE

The code below splits the input into an array and prints it. It then iterates over each element of the array and prints it on a new line.

Input Format
Space and/or newline-separated words.

Go to Top

```
'use strict';
process.stdin.on('data', function (data) {
    main(String(data).trim());
});
/**** Ignore above this line. ****/

function main(input) {
    // Split the words read as input into an array of words
    var array = input.split(new RegExp("[ \n]+"));

    // Print array
    console.log(array);

    // Print each of its elements on a new line
    for (let value of array) {
        console.log(value);
    }
```

Input

```
hi bye
hello goodbye
```

Run

Output

The code above produces the following output:

```
[ 'hi', 'bye', 'hello', 'goodbye' ]
hi
bye
hello
goodbye
```

In this code, we iterate over the set of *Key-Value* pairs in a *Map*, first printing each *Key-Value* pair and then printing each individual *Key* and its paired *Value*.

```
'use strict';

let actress = new Map([
    ["firstName", "Julia"],
    ["lastName", "Roberts"],
    ["dateOfBirth", "October 28, 1967"],
    ["nationality", "American"],
    ["firstMovie", "Satisfaction"]
]);

// Print each Key-Value pair in the map
for (let info of actress) {
    console.log(info);
}

// Print each Key and Value as "Key: Value"
console.log();
for (let info of actress) {
    console.log(info[0] + ": " + info[1]);
```

Output

Run

The code above produces the following output:

```
[ 'firstName', 'Julia' ]
[ 'lastName', 'Roberts' ]
[ 'dateOfBirth', 'October 28, 1967' ]
[ 'nationality', 'American' ]
[ 'firstMovie', 'Satisfaction' ]
```

Table Of Contents

for

while

do-while

for-in

for-of

Go to Top

```
firstName: Julia
lastName: Roberts
dateOfBirth: October 28, 1967
nationality: American
firstMovie: Satisfaction
```

Table Of Contents

for

while

do-while

for-in

for-of

Go to Top