

Need response times for mission critical applications within 30 minutes? Learn more →

We're
hiring

Blog

Docs

Get
Support

Contact
Sales



Tutorials

Questions

Learning Paths

For Businesses

Product Docs

Social Impact

C

CONTENTS

While Loop

Infinite Loops

Do...While Loop

Conclusion

RELATED

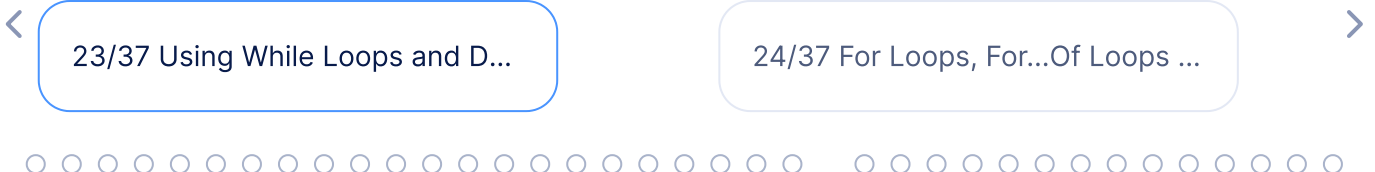
Codelgniter: Getting Started With a Simple Example

[View](#)

How To Install Express, a Node.js Framework, and Set Up Socket.io on a VPS

[View](#)

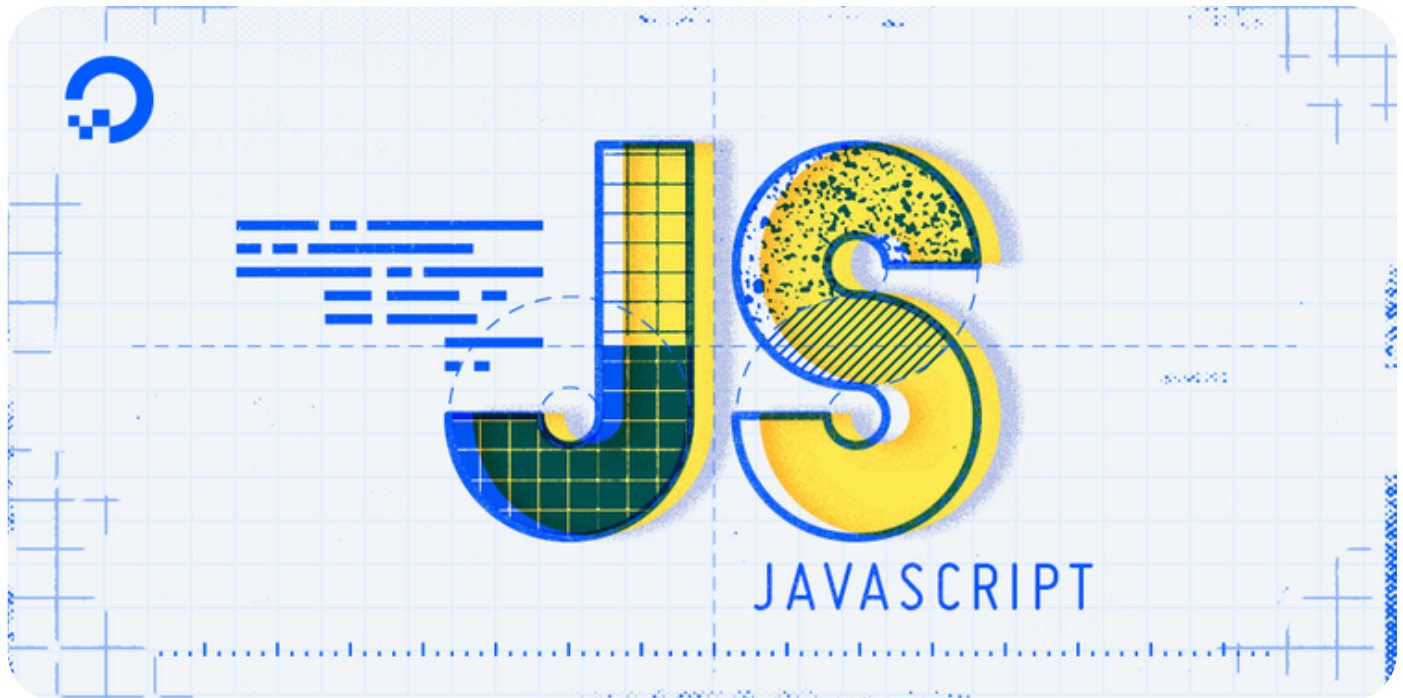
Tutorial Series: How To Code in JavaScript



Using While Loops and Do...While Loops in JavaScript

Published on September 27, 2017 · Updated on August 26, 2021

JavaScript Development

By [Tania Rascia](#)

Introduction

Automation is the technique of making a system operate automatically; in programming, we use **loops** to automate repetitious tasks. Loops are one of the most useful features of programming languages, and in this article we will learn about the `while` and `do...while` loops in JavaScript.

The `while` and `do...while` statements in JavaScript are similar to [conditional statements](#), which are blocks of code that will execute if a specified condition results in `true`. Unlike an `if` statement, which only evaluates once, a loop will run multiple times until the condition no longer evaluates to `true`.

Another common type of loop you will encounter is the [for statement](#), which executes a set number of times. `while` and `do...while` loops are conditionally based, and therefore it is not necessary to know beforehand how many times the loop will run.



While Loop

In JavaScript, a `while` statement is a loop that executes as long as the specified condition evaluates to `true`.

The syntax is very similar to an `if` statement, as seen below.

```
while (condition) {  
    // execute code as long as condition is true  
}
```

[Copy](#)

The `while` statement is the most basic loop to construct in JavaScript.

As an example, let's say we have an aquarium that has a population limit. For each iteration of the loop, we will add one fish. Once the aquarium has 10 fish, the population limit will be reached, and the program will cease to add more fish.

aquarium.js

```
// Set population limit of aquarium to 10  
const popLimit = 10;  
  
// Start off with 0 fish  
let fish = 0;  
  
// Initiate while loop to run until fish reaches population limit  
while (fish < popLimit) {  
    // add one fish for each iteration  
    fish++;  
    console.log("There's room for " + (popLimit - fish) + " more fish.");  
}
```

[Copy](#)

Once we run the above program, we'll receive the following output, showing the iteration of the program through the `while` loop until the conditions are no longer evaluated as `true`.

Output

```
There's room for 9 more fish.  
There's room for 8 more fish.  
There's room for 7 more fish.  
There's room for 6 more fish.  
There's room for 5 more fish.  
There's room for 4 more fish.  
There's room for 3 more fish.  
There's room for 2 more fish.  
There's room for 1 more fish.  
There's room for 0 more fish.
```

In our example, we set our `while` loop to run as long as the number of fish was less than the population limit of the aquarium. For each iteration, one fish is added to the aquarium until all 10 spots are filled. At that point, the loop stops running.

Infinite Loops

An **infinite loop**, as the name suggests, is a loop that will keep running forever. If you accidentally make an infinite loop, it could crash your browser or computer. It is important to be aware of infinite loops so you can avoid them.

A common infinite loop occurs when the condition of the `while` statement is set to `true`. Below is an example of code that will run forever. It is not necessary to test any infinite loops.

infiniteLoop.js

Copy

```
// Initiate an infinite loop
while (true) {
    // execute code forever
}
```

An infinite loop will run forever, but the program can be terminated with the `break` keyword.

In the below example, we will add an `if` statement to the `while` loop, and when that condition is met, we will terminate the loop with `break`.

polarBears.js

Copy

```
// Set a condition to true
const iceCapsAreMelting = true;
let polarBears = 5;

// Initiate infinite loop
while (iceCapsAreMelting) {
    console.log(`There are ${polarBears} polar bears.`);
    polarBears--;
    // Terminate infinite loop when following condition is true
    if (polarBears === 0) {
        console.log("There are no polar bears left.");
        break;
    }
}
```



When we run the code above, the output will be as follows.

Output

```
There are 5 polar bears.
There are 4 polar bears.
There are 3 polar bears.
There are 2 polar bears.
There are 1 polar bears.
There are no polar bears left.
```

Note that this is not necessarily a practical method of creating and terminating a loop, but `break` is a useful keyword to be aware of.

Do...While Loop

We already learned about the `while` loop, which executes a block of code for as long as a specified condition is true. Building on that is the `do...while` statement, which is very similar to `while` with the major difference being that a `do...while` loop will always execute once, even if the condition is never true.

Below we will demonstrate the syntax of the `do...while` loop.

```
do {
    // execute code
} while (condition);
```

Copy

As you can see, the `do` portion of the loop comes first, and is followed by `while (condition)`. The code block will run, then the condition will be tested as it is in a normal `while` loop.

To test this, we can set a variable to `0`, increment it inside the `do` statement, and set our condition to `false`.

falseCondition.js

```
// Set variable to 0
let x = 0;

do {
    // Increment variable by 1
    x++;
} while (false);
```

Copy

```
    console.log(x);  
  } while (false);
```

Output

1

Our output came out to `1`, meaning that the code block iterated through the loop once (from `0`) before it was stopped by an unsuccessful `while` condition.

While keeping in mind that the loop will iterate at least once, the `do...while` loop can be used for the same purposes as a `while` loop.

Conclusion

In this tutorial, we learned about the `while` loop, the `do...while` loop, and infinite loops in JavaScript.

Automation of repetitive tasks is an extremely important part of programming, and these loops can help make your programs more efficient and concise.

To learn more, read about the [while](#) and [do...while](#) loops on the Mozilla Developer Network.

Thanks for learning with the DigitalOcean Community. Check out our offerings for compute, storage, networking, and managed databases.

[Learn more about us →](#)

Next in series: [For Loops, For...Of Loops and For...In Loops in JavaScript →](#)

Want to learn more? Join the DigitalOcean Community!



Join our DigitalOcean community of over a million developers for free! Get help and share knowledge in our Questions & Answers section, find tutorials and tools that will

help you grow as a developer and scale your project or business, and subscribe to topics of interest.

Sign up now →

Tutorial Series: How To Code in JavaScript

JavaScript is a high-level, object-based, dynamic scripting language popular as a tool for making webpages interactive.

Subscribe

JavaScript Development

Browse Series: 37 articles

[1/37 How To Use the JavaScript Developer Console](#)

[2/37 How To Add JavaScript to HTML](#)

[3/37 How To Write Your First JavaScript Program](#)

Expand to view all

About the authors



[Tania Rascia](#) Author



[Lisa Tagliaferri](#) Editor

Still looking for an answer?

Ask a question

Search for more help

Was this helpful?

Yes

No



Comments

Leave a comment

B *I* U H₁ H₂ H₃ “”



Leave a comment...

This textbox defaults to using **Markdown** to format your answer.

You can type **!ref** in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

Sign In or Sign Up to Comment





This work is licensed under a Creative Commons Attribution-NonCommercial- ShareAlike 4.0 International License.

Try DigitalOcean for free

Click below to sign up and get **\$200 of credit** to try our products over 60 days!

Sign up →

Popular Topics

Ubuntu

Linux Basics

JavaScript

Python


MySQL



Docker


Kubernetes

[All tutorials →](#)

[Free Managed Hosting →](#)

 Congratulations on unlocking the whale ambience easter egg! Click the whale button in the bottom left of your screen to toggle some ambient whale noises while you read.

  Thank you to the [Glacier Bay National Park & Preserve](#) and [Merrick079](#) for the sounds behind this easter egg.

 Interested in whales, protecting them, and their connection to helping prevent climate change? We recommend checking out the [Whale and Dolphin Conservation](#).

[Reset easter egg to be discovered again](#) / [Permanently dismiss and hide easter egg](#)



Get our biweekly newsletter

Sign up for Infrastructure as a Newsletter.

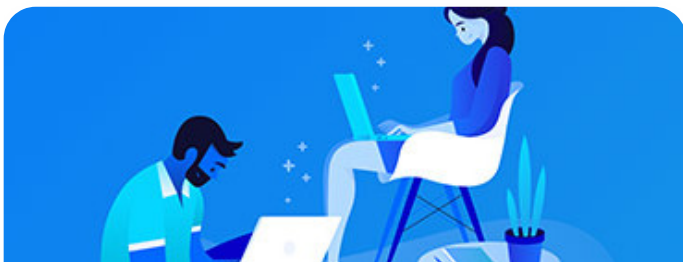
[Sign up](#) →



Hollie's Hub for Good

Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

[Learn more](#) →



Become a contributor

You get paid; we donate to tech nonprofits.

[Learn more](#) →

Featured on Community

[Kubernetes Course](#) [Learn Python 3](#) [Machine Learning in Python](#)
[Getting started with Go](#) [Intro to Kubernetes](#)

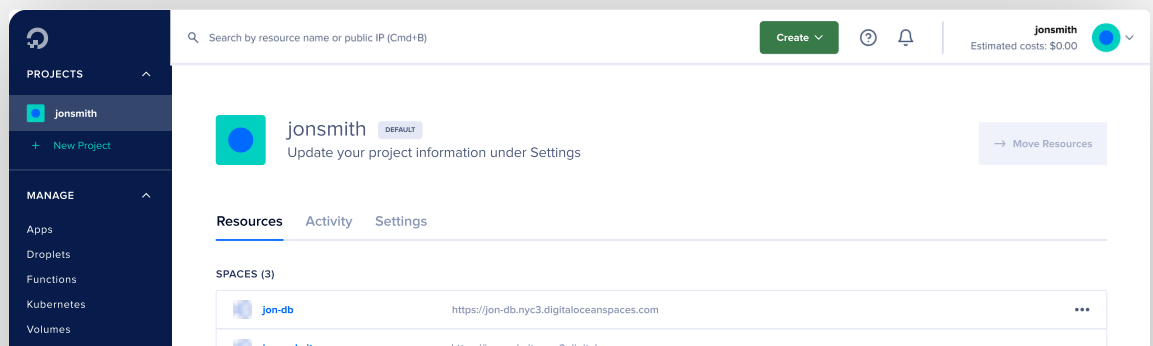
DigitalOcean Products

[Cloudways](#) [Virtual Machines](#) [Managed Databases](#) [Managed Kubernetes](#)
[Block Storage](#) [Object Storage](#) [Marketplace](#) [VPC](#) [Load Balancers](#)

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow – whether you're running one virtual machine or ten thousand.

Learn more →



Company

About

Leadership



Products

Products

Overview

Community

Tutorials

Q&A

Solutions

Website Hosting

VPS Hosting

Contact

Support

Sales

Blog	Droplets	CSS-Tricks	Web & Mobile Apps	Report Abuse
Careers	Kubernetes	Write for DOnations	Game Development	System Status
Customers	App Platform	Currents Research	Streaming	Share your ideas
Partners	Functions	Hatch Startup Program	VPN	
Channel Partners	Cloudways	deploy by DigitalOcean	SaaS Platforms	
Referral Program	Managed Databases	Shop Swag	Cloud Hosting for Blockchain	
Affiliate Program	Spaces	Research Program	Startup Resources	
Press	Marketplace	Open Source		
Legal	Load Balancers	Code of Conduct		
Security	Block Storage	Newsletter Signup		
Investor Relations	Tools & Integrations	Meetups		
DO Impact	API			
	Pricing			
	Documentation			
	Release Notes			
	Uptime			

© 2023 DigitalOcean, LLC. All rights reserved.

