CONTENTS

RELATED

CodeIgniter: 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

21/37 How To Write Conditional ...

22/37 How To Use the Switch S...

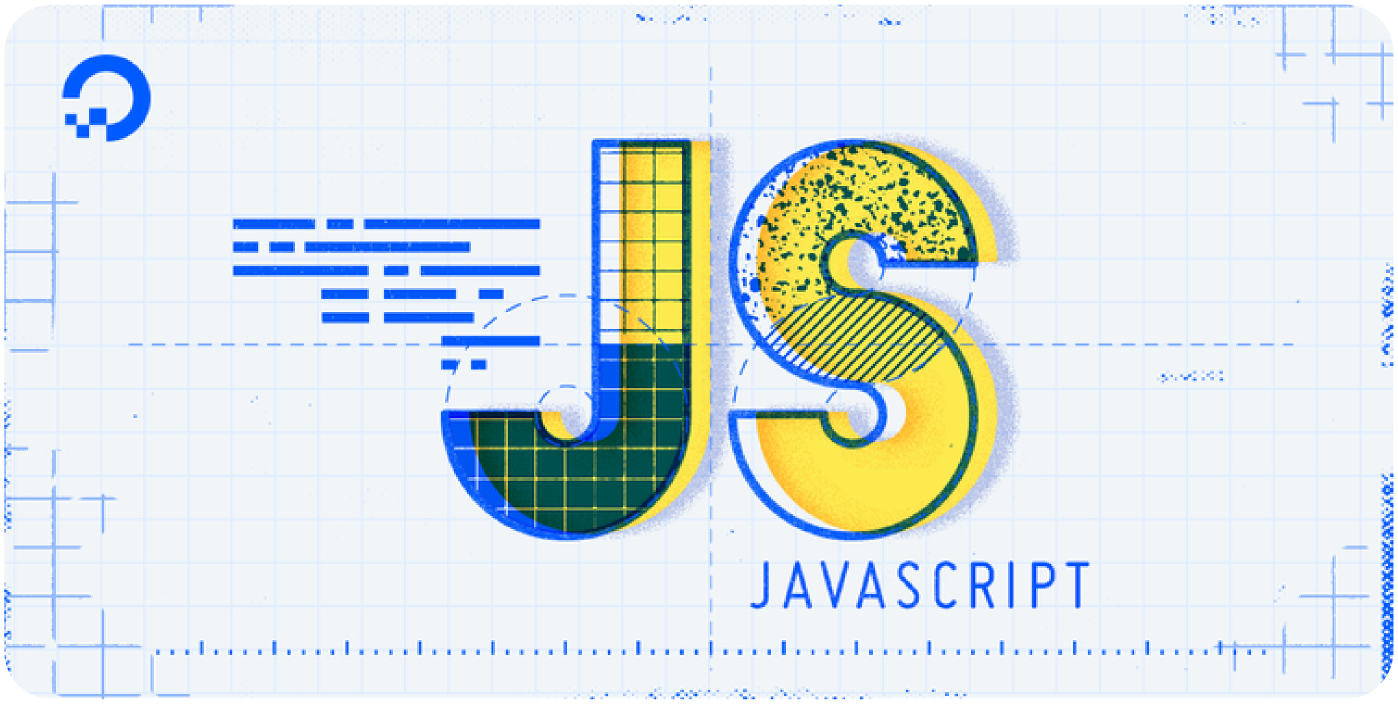
// Tutorial //

How To Write Conditional Statements in JavaScript

Published on August 29, 2017 · Updated on August 25, 2021

JavaScript Development

By Tania Rascia



# Introduction

In programming, there will be many occasions in which you will want different blocks of code to run depending on user input or other factors.

As an example, you might want a form to submit if each field is filled out properly, but you might want to prevent that form from submitting if some required fields are missing. In order to achieve tasks like these we have **conditional statements**, which are an integral part of all programming languages.

Conditional statements execute a specific action based on the results of an outcome of `true` or `false`.

A few [illegible]ples of JavaScript conditional statements you might see include:

- Check the location of a user and display the correct language based on country

- Send a form on submit, or display warnings next to missing required fields
- Open a dropdown on a click event, or close a dropdown if it is already open
- Display an alcohol purveyor's website if the user is over the legal drinking age
- Display the booking form for a hotel but not if the hotel is booked

Conditional statements are part of the logic, decision making, or flow control of a computer program. You can compare a conditional statement to a "Choose Your Own Adventure" book, or a flowchart.

In this tutorial, we will go over conditional statements, including the `if`, `else`, and `else if` keywords. We will also cover the ternary operator.

# If Statement

The most fundamental of the conditional statements is the `if` statement. An `if` statement will evaluate whether a statement is true or false, and only run if the statement returns `true`. The code block will be ignored in the case of a `false` result, and the program will skip to the next section.

An `if` statement is written with the `if` keyword, followed by a condition in parentheses, with the code to be executed in between curly brackets. In short, it can be written as `if () {}`.

Here is a longer examination of the basic `if` statement.

Copy

```
if (condition) {
	// code that will execute if condition is true
}
```

The contents of an `if` statement are indented, and the curly brackets containing the block of code to run do not end in a semicolon, just like a function block.

As an example, let's consider a shopping app. Say, for the functionality of this app, a user who has deposited a certain amount of funds into their account would then like to buy an item from the store.

shop.js

Copy

```
// Set balance and price of item
co[illegible]alance = 500;
cons[illegible]ns = 40;

// Check if there are enough funds to purchase item
```

```
if (jeans <= balance) {
  console.log("You have enough money to purchase the item!");
}
```

```
Output
You have enough money to purchase the item!
```

We have an account balance of `500`, and want to buy a pair of jeans for `40`. Using the less than or equal to operator, we can check if the price of jeans is less than or equal to the amount of funds we have. Since `jeans <= balance` evaluates to `true`, the condition will pass and the block of code will run.

In a new example, we will create a new shop item that costs more than the available balance.

shop.js

```
// Set balance and price of item
const balance = 500;
const phone = 600;

// Check if there is enough funds to purchase item
if (phone <= balance) {
        console.log("You have enough money to purchase the item!");
}
```

Copy

This example will have no output, since `phone <= balance` evaluates to `false`. The code block will simply be ignored, and the program will proceed to the next line.

# Else Statement

With `if` statements, we only execute code when a statement evaluates to `true`, but often we will want something else to happen if the condition fails.

For example, we might want to display a message telling the user which fields were filled out correctly if a form did not submit properly. In this case, we would utilize the `else` statement, which is the code that will execute if the original condition does not succeed.

The `else` statement is written after the `if` statement, and it has no condition in pare[illegible]ses. Here is the syntax for a basic `if...else` statement.

```
if (condition) {
        // code that will execute if condition is true
} else {
        // code that will execute if condition is false
}
```
Copy

Using the same example as above, we can add a message to display if the funds in the account are too low.

shop.js

```
// Set balance and price of item
const balance = 500;
const phone = 600;

// Check if there is enough funds to purchase item
if (phone <= balance) {
        console.log("You have enough money to purchase the item!");
} else {
        console.log("You do not have enough money in your account to purchase thi
}
```
Copy

```
Output
You do not have enough money in your account to purchase this item.
```

Since the `if` condition did not succeed, the code moves on to what's in the `else` statement.

This can be very useful for showing warnings, or letting the user know what actions to take to move forward. Usually an action will be required on both success and failure, so `if...else` is more common than a solo `if` statement.

# Else if Statement

With `if` and `else`, we can run blocks of code depending on whether a condition is `true` or `false`. However, sometimes we might have multiple possible conditions and outputs, and need more than simply two options. One way to do this is with the `else if` statement, which can evaluate more than two possible outcomes.

Here [illegible] basic example of a block of code that contains an `if` statement, multiple `else if` statem[illegible], and an `else` statement in case none of the conditions evaluated to `true`.

```
if (condition a) {
        // code that will execute if condition a is true
} else if (condition b) {
        // code that will execute if condition b is true
} else if (condition c) {
        // code that will execute if condition c is true
} else {
        // code that will execute if all above conditions are false
}
```
Copy

JavaScript will attempt to run all the statements in order, and if none of them are successful, it will default to the `else` block.

You can have as many `else if` statements as necessary. In the case of many `else if` statements, the `switch` statement might be preferred for readability.

As an example of multiple `else if` statements, we can create a grading app that will output a letter grade based on a score out of 100.

The requirements of this app are as follows:

- Grade of 90 and above is an A
- Grade of 80 to 89 is a B
- Grade of 70 to 79 is a C
- Grade of 60 to 69 is a D
- Grade of 59 or below is an F

Below we will create a simple set of `if`, `else`, and `else if` statements, and test them against a given grade.

grades.js

```
// Set the current grade of the student
let grade = 87;

// Check if grade is an A, B, C, D, or F
if (grade >= 90) {
  console.log("A");
} else if (grade >= 80) {
  console.log("B");
} else if (grade >= 70) {
  console.log("C");
} else if (grade >= 60) {
  console.log("D");
} else {
```
Copy

```
    console.log("F");
  }
```

```
Output
B
```

In our example, we first check for the highest score, which will be greater than or equal to `90`. After that, the `else if` statements will check for greater than `80`, `70`, and `60` until it reaches the default `else` of a failing grade.

Although our `grade` value of `87` is technically also true for `C`, `D` and `F`, the statements will stop at the first one that is successful. Therefore, we get an output of `B`, which is the first match.

# Ternary Operator

The **ternary operator**, also known as the conditional operator, is used as shorthand for an `if...else` statement.

A ternary operator is written with the syntax of a question mark (`?`) followed by a colon (`:`), as demonstrated below.

```
(condition) ? expression on true : expression on false
```

Copy

In the above statement, the condition is written first, followed by a `?`. The first expression will execute on `true`, and the second expression will execute on `false`. It is very similar to an `if...else` statement, with more compact syntax.

In this example, we will create a program that checks if a user is `21` or older. If they are, it will print `"You may enter"` to the console. If they are not, it will print `"You may not enter."` to the console.

age.js

```
// Set age of user
let age = 20;

// Place result of ternary operation in a variable
co[illegible]oldEnough = (age >= 21) ? "You may enter." : "You may not enter.";
```

Copy

```
// Print output
oldEnough;
```

```
Output
'You may not enter.'
```

Since the `age` of the user was less than `21`, the fail message was output to the console. The `if...else` equivalent to this would be `"You may enter."` in the `if` statement, and `"You may not enter."` in the `else` statement.

# Conclusion

Conditional statements provide us with flow control to determine the output of our programs. They are one of the foundational building blocks of programming, and can be found in virtually all programming languages.

In this article, we learned about how to use the `if`, `else`, and `else if` keywords, and covered nesting of statements, and use of the ternary operator.

Next in series: How To Use the Switch Statement in JavaScript →

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

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

1 Comments

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

stef5cbfae72eedfb901ba976f • September 4, 2017

In the example with the grades.js is maybe a better example to use the switch instead of the if () else if construction. And a nice addition to the article could be explaining the triple = with the type checking. “0” === 0 will fail but “0” == 0 will work. https://codepen.io/stefferd/pen/OjqaOM

Reply

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

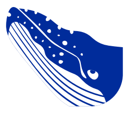
MySQL

Docker

Kubernetes

All tutorials →

Free Managed Hosting →

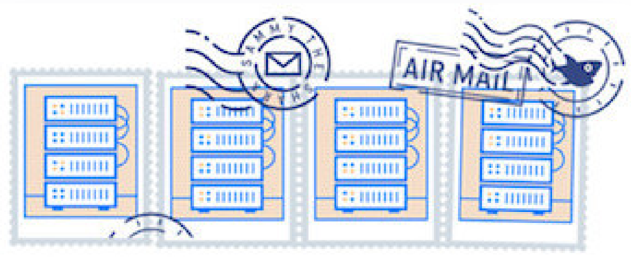


Congratulations on unlocking the whale ambience easter egg! Click the whale button in the bottom left of your screen to toggle some ambient whale noises while you read.

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

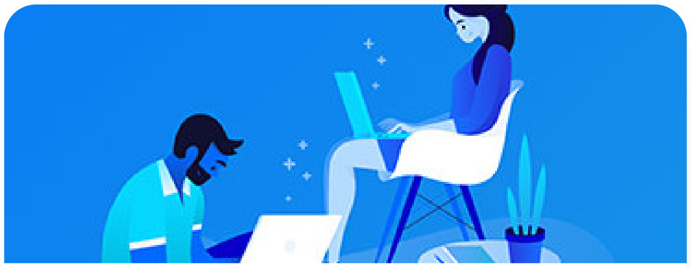
Sign up for Infrastructure as a Newsletter.

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

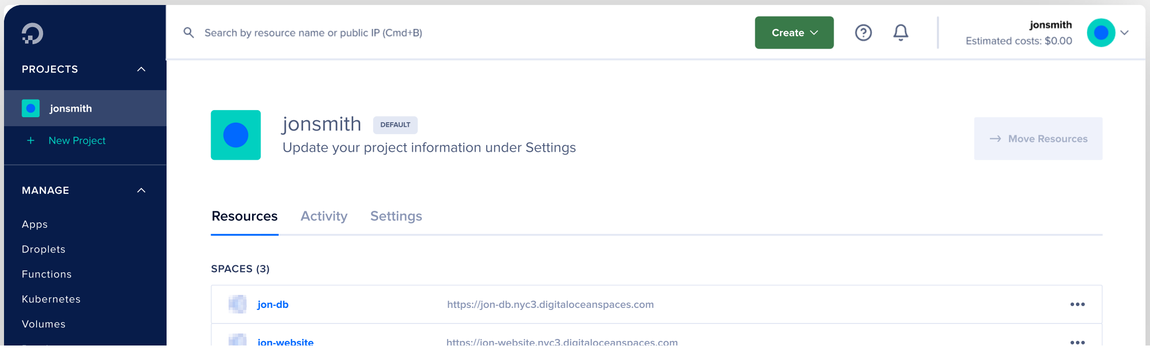
DigitalOcean Products

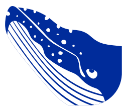
Cloudways Virtual Machines Managed Databases Managed Kubernetes Block Storage Object Storage Marketplace VPC Load Balancers

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
- Blog
- Careers
- Customers
- Partners
- Channel Partners
- Referral Program
- Affiliate Program
- Press
- Legal
- Security
- Investor Relations
- DO Impact

Products

- Products Overview
- Droplets
- Kubernetes
- App Platform
- Functions
- Cloudways
- Managed Databases
- Spaces
- Marketplace
- Load Balancers
- Block Storage
- Tools & Integrations
- API
- Pricing
- Documentation
- Release Notes
- Uptime

Community

- Tutorials
- Q&A
- CSS-Tricks
- Write for DOnations
- Currents Research
- Hatch Startup Program
- deploy by DigitalOcean
- Shop Swag
- Research Program
- Open Source
- Code of Conduct
- Newsletter Signup
- Meetups

Solutions

- Website Hosting
- VPS Hosting
- Web & Mobile Apps
- Game Development
- Streaming
- VPN
- SaaS Platforms
- Cloud Hosting for Blockchain
- Startup Resources

Contact

- Support
- Sales
- Report Abuse
- System Status
- Share your ideas