



Faça login em Medium com o Google



Rogério Soares (Merovingio)

rgrsoares@yahoo.com.br

Continuar como Rogério

Pug Templates: A Developer's Faithful Companion



Jawara Gordon · Follow

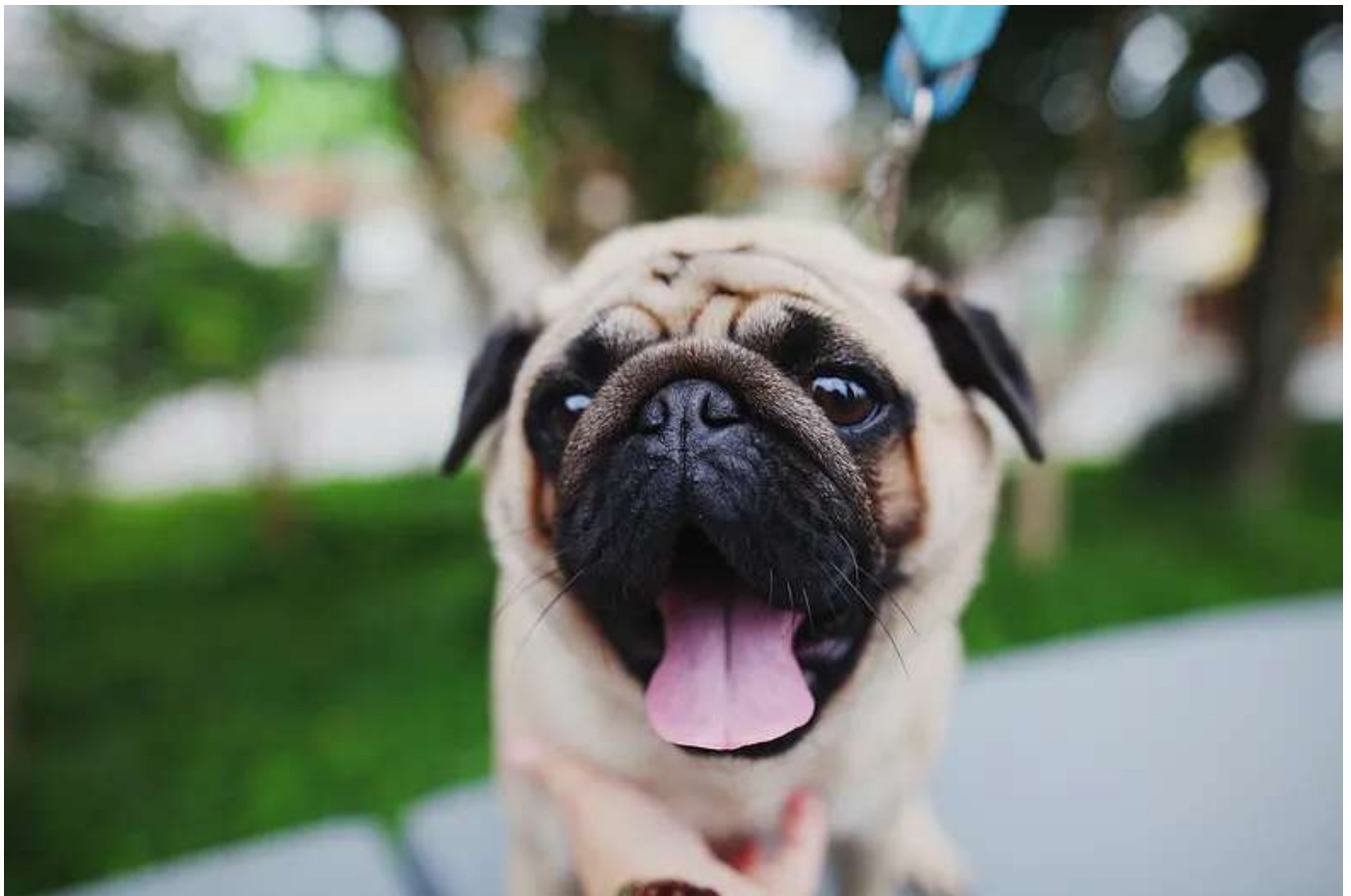
8 min read · Mar 12



Listen



Share

Photo by [Silvana Carlos](#) on [Unsplash](#)

Are you tired of working like a dog when it comes to creating HTML for your projects?

Let me introduce you to my new friend Pug.

Pug is a popular templating engine that works with your favorite JavaScript libraries and frameworks.

Formerly known as Jade, this engine allows developers to write clean, concise, and reusable HTML.

Pug uses indentation and minimal markup to create templates that are easier to read than traditional HTML. It also includes features like mixins, and filters which allow developers to reuse code, keeping things nice and organized.

Whether you're new to Pug or have some experience with it, this article will walk you through the basics of Pug syntax and show you how to get started with using Pug in a Node and Express.js environment.



Photo by [Jantine Doornbos](#) on [Unsplash](#)

A Dog-Eat-Dog World

There are many templating engines to choose from such as: [EJS](#), [Handlebars](#), [Mustache](#), and [Nunjucks](#).

These engines all have their own strengths and weaknesses depending on your needs. However, they all share a common goal of simplifying the process of creating

web applications using JavaScript.

I decided to use Pug for a MERN stack project I'm working on due to its popularity with the Node community.

Pug was created in 2011 by TJ Holowaychuk, a prominent Node.js developer, and was originally called Jade. It was renamed to Pug in 2016 after a naming conflict with the already existing Jade software. Pug is now maintained by a team of developers and has become one of the most popular templating engines for Node.js and Express.

Pug's popularity is based on these core concepts:

- **Simplifies code:** Pug's syntax simplifies the structure of your code, making it easier to read and maintain. The reduced amount of code also means faster load times and performance.
- **Creates reusable code:** Features like mixins, filters, and includes enable developers to reuse code and keep their templates organized. This can save time and effort by eliminating the need to duplicate code or make changes to multiple files.
- **Supports dynamic content:** Pug makes it easy to include dynamic content in your templates, like variables and loops, which can be rendered based on data passed to each template. This allows for more efficient and flexible development.



Photo by [Sigmund](#) on [Unsplash](#)

Off The Leash

Now that we've learned what Pug is, it's time to put it to use with a coding example.

We're going to install Pug, create an express app, then use the Pug template to create a simple HTML template.

Let's get started!

First, make sure you have Node and npm (node package manager) installed on your system. You can find out by typing **node -v** into your terminal window.

```
$ node -v
```

You can do the same for npm:

```
$ npm -v
```

If you don't have them both installed, follow this step-by-step guide to get up to speed: <https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>

Next we need to install Express, a “fast, unopinionated, and minimalist” Node.js framework.

You can check to see if you have Express installed by typing **express — version** in the terminal:

```
$ express --version
```

If you receive an error, follow this simple guide to get it installed:

<https://expressjs.com/en/starter/installing.html>

Installing Pug

Now that we have those core requirements in place, we can install pug and all of its dependencies.

First, create a new directory inside of your text editor (if you have not already) and then navigate into that directory:

```
$ mkdir pug  
$ cd pug
```

Next install Pug by typing **npm install pug** in the terminal:

```
$ npm install pug
```

You'll see a loading bar progress across the screen and details about the installation when it's completed.

```
● → pug npm install pug  
  
added 44 packages, and audited 45 packages in 5s  
  
8 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities
```

The easiest way to get started with an Express app is to use the Express application generator. Type the following command in your terminal **express --view=pug myapp** which will set Pug as the default view engine.

```
$ express --view=pug myapp
```

You can change “myapp” to whatever name you would like to use.

The Express app generator will automatically create this list of files and folders:

```
create : myapp  
create : myapp/package.json  
create : myapp/app.js  
create : myapp/public  
create : myapp/public/javascripts  
create : myapp/public/images  
create : myapp/routes  
create : myapp/routes/index.js  
create : myapp/routes/users.js  
create : myapp/public/stylesheets  
create : myapp/public/stylesheets/style.css  
create : myapp/views  
create : myapp/views/index.pug  
create : myapp/views/layout.pug  
create : myapp/views/error.pug  
create : myapp/bin  
create : myapp/bin/www
```

Next, navigate into the newly created project folder

```
$ cd myapp
```

Then install dependencies:

```
$ npm install
```

You can run your app on macOS and Linux with this command:

```
$ DEBUG=myapp:* npm start
```

In a windows Command Prompt use this command:

```
> set DEBUG=myapp:* & npm start
```

On Windows PowerShell, use this command:

```
PS> $env:DEBUG='myapp:*'; npm start
```

Then load <http://localhost:3000/> in your browser to access the app.

If done correctly you'll see the following text: **Express Welcome to Express.**



Express

Welcome to Express

Its Bark is Worse Than its Bite

Don't get intimidated by the large number of files, folders, and code that suddenly appeared in your project directory.

Express gives you this file structure for getting projects started quickly:

```
.
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.pug
    ├── index.pug
    └── layout.pug
app.js
package.json
```


We're only going to make changes to three different files in this tutorial — `style.css`, `index.js`, and `index.pug`.

Pro Tip: Before we get started let's add a tool called **nodemon** that will watch for our files to make changes, then automatically restart our server for us (so we don't have to do it manually).

Type **npm install nodemon** into your terminal:

```
$ npm install nodemon
```

After it's installed, open your `package.json` file in your text editor and you'll see that it was added as a dependency.

Add this line to your "scripts" underneath "start":

"dev": "nodemon index.js":

Make sure you add a comma after the previous line!

```
{
  "name": "myapp",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www",
    "dev": "nodemon index.js"
  },
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "morgan": "~1.9.1",
    "nodemon": "^2.0.21",
    "pug": "2.0.0-beta11"
  }
}
```

When you're done updating the package.json file, open the terminal window where your app is currently running and type **control + c** to stop the local server.

Type **npm run dev** in the terminal to start **nodemon**.

```
$ npm run dev
```

Let's start making changes to our HTML template with index.js located in the routes folder.

Open index.js in your text editor and you'll see the following code:

```
const express = require('express');
const router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

module.exports = router;
```

You can change the title value on line 6 from 'Express' to anything you'd like. I changed mine to 'Getting Started:'

Refresh your web browser and you'll see the change.

Next, open index.pug in your text editor which is located in the views folder. You'll see the following Pug syntax:

```
extends layout

block content
  h1= title
  p Welcome to #{title}
```

This allows you to add (extend) the boilerplate HTML located in the layout.pug file to any other template you choose.

You'll also see **block content** which is a section block that contains HTML elements for that section. The `h1= title` is dynamically displaying the name you just assigned in the previous step along with a `<p>` tag with shortened syntax with the remaining text showing in the browser.

Pug uses indentation to help with readability and to keep code organized. It's crucial that you follow this syntax throughout your templates for them to work correctly.

Delete all of the code from this file and replace it with the following:

```
extends layout

block content
  div.center-container
    h1= title
    p
      span Welcome to my
      img.emoji(src="https://jawaragordon.com/img/pug-icon.png")
    p demo app!
```

Refresh your browser and you'll see these changes:

Open in app ↗

Sign up

Sign in



Search



Getting Started:



Welcome to my
demo app!

Finally, let's add some styling by opening up the style.css in your text editor by navigating to public > stylesheets > style.css.

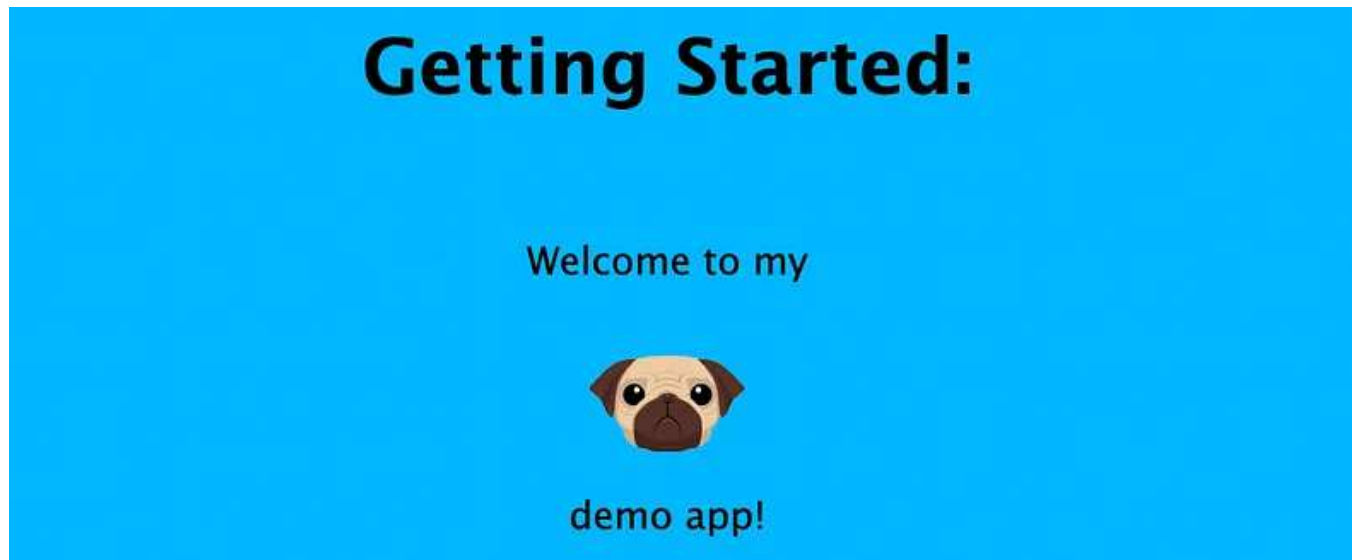
You can add CSS classes to your block content as I did with the div and the img tags from the previous step.

Delete everything in this file and replace it with the following code:

```
body {  
  background-color: #00B7FF;  
  padding: 50px;  
  font: 1rem "Lucida Grande", Helvetica, Arial, sans-serif;  
}  
.center-container{  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
  justify-content: center;  
}  
  
.emoji{
```

```
width: 72px;  
}
```

We now have a nice blue background, center-aligned text, and a cute Pug-sized fur buddy. Great job!



Here's a list of additional Pug features you can experiment with:

- **Variables:** Variables are used to store and manipulate data in Pug templates. In Pug, variables are defined using the equals sign (=) followed by the variable name.
- **Conditionals:** Conditionals are used to control the flow of execution in Pug templates. In Pug, conditionals are defined using the if, else if, and else keywords.
- **Loops:** Loops are used to iterate over data in Pug templates. In Pug, loops are defined using the each keyword followed by the data to be iterated over.
- **Mixins:** Mixins are used to create reusable blocks of code in Pug templates. In Pug, mixins are defined using the mixin keyword followed by the mixin name and arguments.
- **Filters:** Pug filters allow you to include preprocessor languages in your Pug templates, such as Markdown, Sass, or CoffeeScript. Filters are denoted by a colon followed by the name of the filter and can be applied to a block of text or a file.



Photo by [Dana Tylikova](#) on [Unsplash](#)

Summary

We've only just begun to scratch the surface of Pug. Its syntax leads to cleaner more maintainable code that's easier to read, while its set of rich features makes developing complex web applications a walk in the park.

This article provided a basic example of how to set up Pug with an Express application. With practice and experimentation, you can begin to use Pug's more advanced tools to help you create faster, cleaner, and more maintainable HTML. A skill that leaves you feeling like you're teaching old dogs new tricks.

Sources:

<https://pugjs.org/api/getting-started.html>

<https://www.sitepoint.com/pug-tips-tricks-best-practices/>

https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Displaying_data/Pug_templates

<https://pugjs.org/language/history.html>

https://www.tutorialspoint.com/pugjs/pugjs_advantages.htm

<https://www.sitepoint.com/pug-tips-tricks-best-practices/>

<https://www.geeksforgeeks.org/what-are-the-differences-between-npm-and-npx/>

Resources:

Getting Started

Pug is available via npm: The general rendering process of Pug is simple. `pug.compile()` will compile the Pug source...

pugjs.org



<https://pugjs.org/api/express.html>

Installing

Assuming you've already installed Node.js, create a directory to hold your application, and make that your working...

expressjs.com



<https://pugjs.org/api/express.html>

<https://pugjs.org/language/tags.html>

HTML

Nodejs

Expressjs

Pug

Template

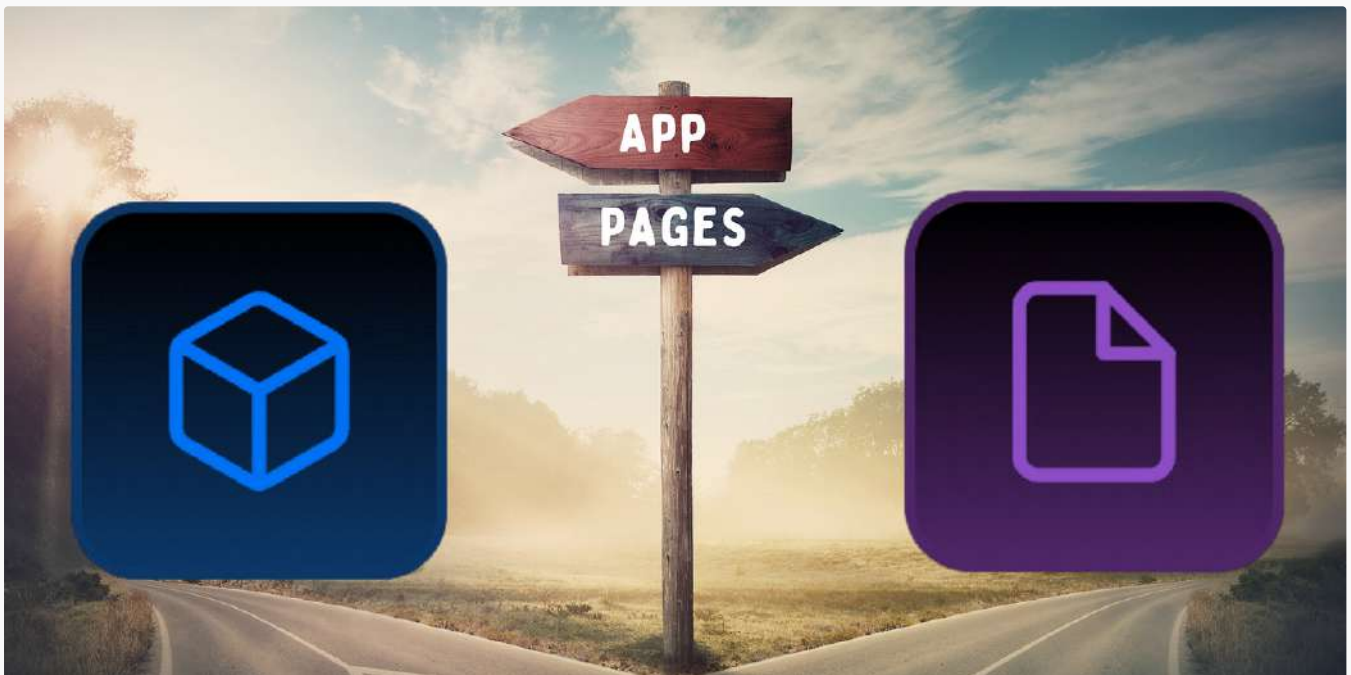
[Follow](#)

Written by Jawara Gordon

41 Followers

Jawara (jah-WAH-rah): Full-Stack Web Developer | Audio Engineer JavaScript, React, HTML, CSS/SASS, Ruby on Rails | Ableton Live

More from Jawara Gordon



Jawara Gordon

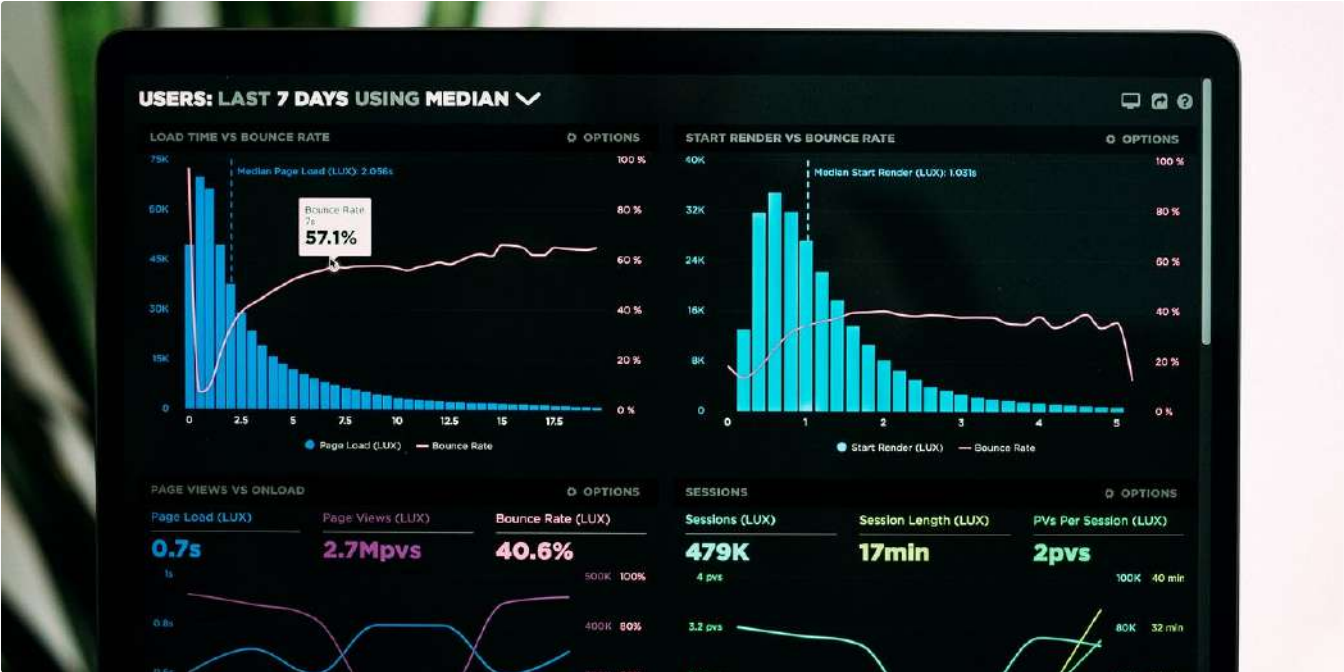
Choose Your Own Adventure: NEXT.js App vs. Pages Router

7 min read · Oct 1



20





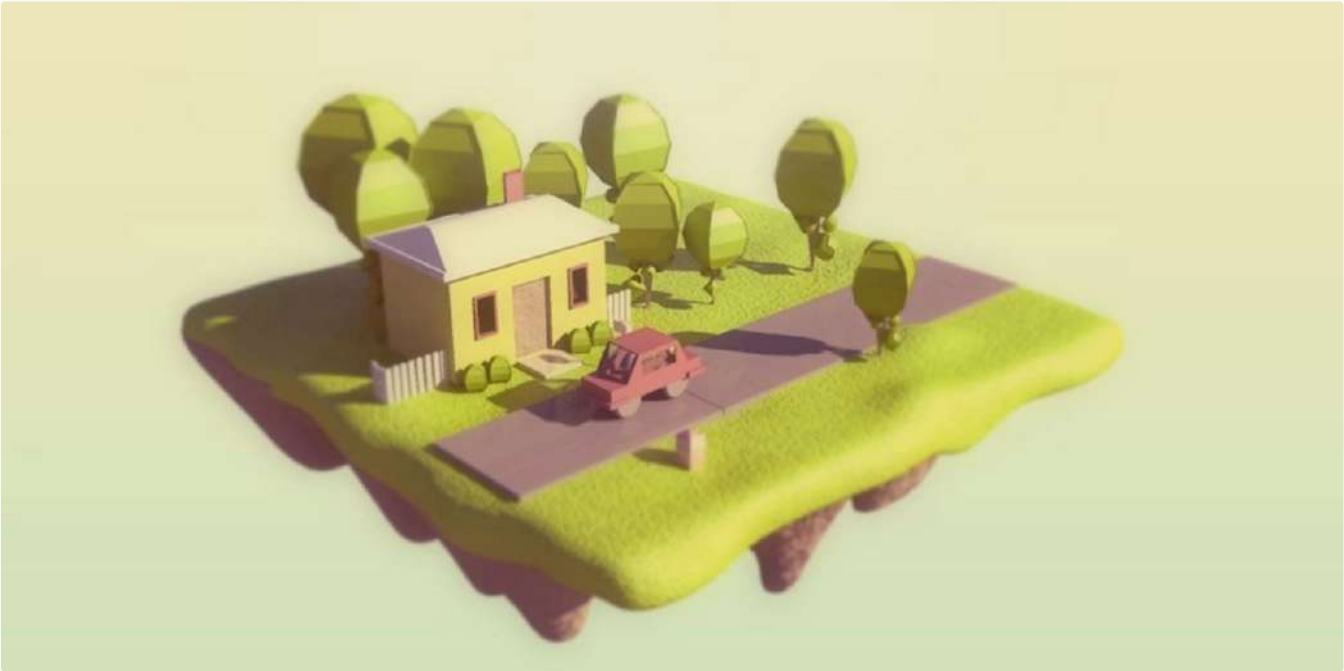
 Jawara Gordon


Getting Started with Data Visualization and Chart.js

What is Chart.js?

7 min read · Feb 11

 51 



 Jawara Gordon

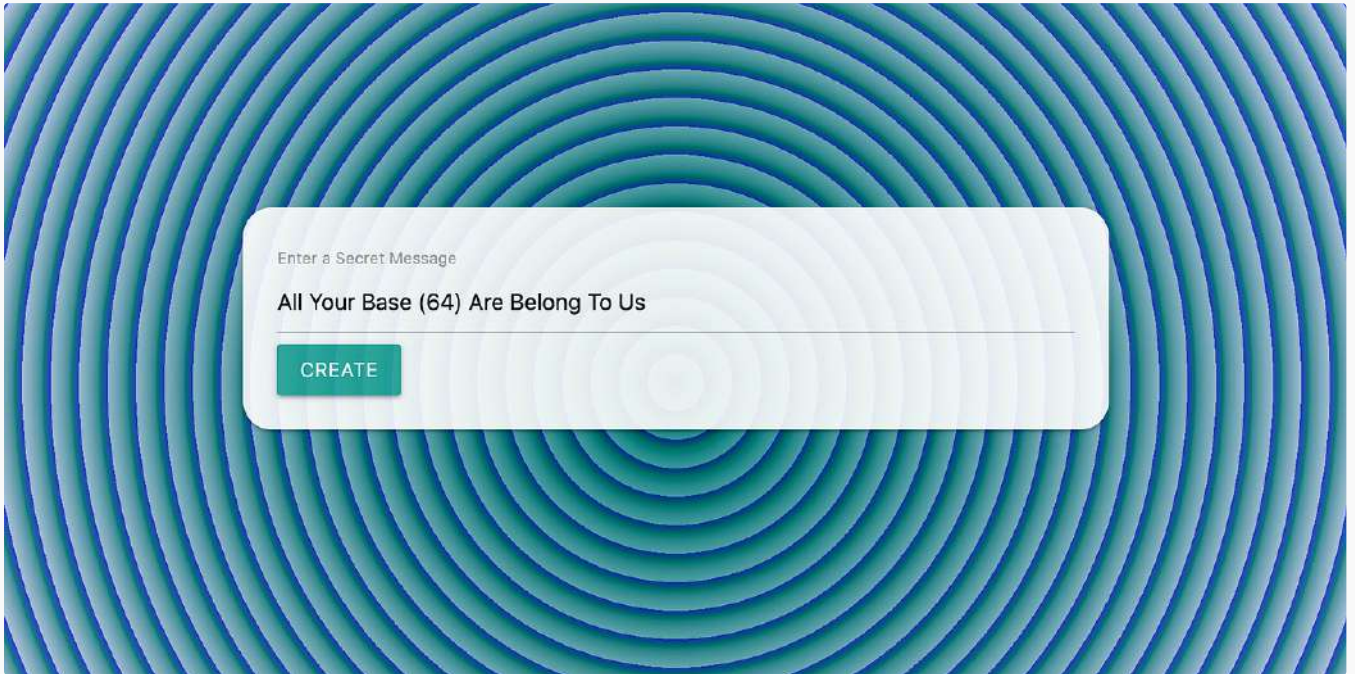
JavaScript 3D Modeling with Three.js

Getting Started

6 min read · Apr 30



22

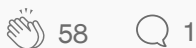


Jawara Gordon

Unleash the Power of JavaScript Base 64 Encoding

I've been pleasantly surprised by how much Stephen Grider has taught me about JavaScript. He teaches the second half of "The Modern...

6 min read · Jan 7



58


1



See all from Jawara Gordon

Recommended from Medium



 Gabriel Rossetto


Creating a Simple Telegram Bot using Node.js: A Step-by-Step Guide

In this step-by-step guide, we'll walk through the process of creating a simple Telegram bot using Node.js. By the end of this tutorial...

3 min read · Jul 10

 135



 cengkuru michael

How to Update Node.js to Any Version: A Step-by-Step Guide

Introduction

2 min read · Aug 3

 26





Lists



Stories to Help You Grow as a Software Developer

19 stories · 608 saves



New_Reading_List

174 stories · 215 saves



Productivity

232 stories · 221 saves



Abdullah Muhammad in Stackademic

Vanilla JS & GitHub Pages for Easy Deployment

This article assumes the reader is familiar with basic Web Development and how GitHub works.

7 min read · Aug 14



3




Roman Glushach

Comparing NPM, YARN, and PNPM Package Managers: Which One is Right for Your Distributed Project to...

If you are a Frontend or Full Stack developer, you probably have used one or more of these package managers: NPM, YARN, and PNPM. They are...

9 min read · Jun 28

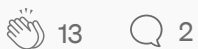


 Aishwarya Parab

Deploying Vite Deploying Vite App to GitHub Pages

Looking for a free hosting platform to deploy your Vite app? Look no further than GitHub Pages!

2 min read · Jul 12



```
components > react vite.jsx > ...
1  import React from 'react';
2
3  const Home = () => {
4    return (
5      <div>
6        <h1>Welcome to the Home Page!</h1>
7        <p>This is the content of the Home page.</p>
8      </div>
9    );
10  };
11
12  export default Home;
13
```



Alousiusk

React Router with Vite: A Powerful Combination for Modern Web Development

Introduction

5 min read · Jul 26



67



1

[See more recommendations](#)