Tutorial: How to Use dotenv in React for Beginners

Rivai Fansuri Nasution · Follow
3 min read · Jun 29, 2024

Introduction

In modern web development, managing environment variables is crucial for configuring applications across different environments like development, testing, and production. **dotenv** is a popular tool that simplifies the process of loading environment variables into your Node.js applications, including React applications.

In this tutorial, we'll walk through the steps of setting up and using **dotenv** in a React project, aimed at beginners who are new to managing environment variables.

In modern web development, managing environment variables is crucial for configuring applications across different environments like development, testing, and production. **dotenv** is a popular tool that simplifies the process of loading environment variables into your Node.js applications, including React applications.

In this tutorial, we'll walk through the steps of setting up and using **dotenv** in a React project, aimed at beginners who are new to managing environment variables.

Before we begin, make sure you have the following installed on your machine:

- Node.js (with npm or yarn)
- A code editor (e.g., VS Code)

Step 1: Setting Up a React Project

If you haven't already set up a React project, you can do so using Create React App, which provides a quick way to bootstrap a new React application:

```
npx create-react-app dotenv-react-example
cd dotenv-react-example
```

Step 2: Installing dotenv

Next, install dotenv as a dependency in your React project. dotenv helps load environment variables from a `.env` file into `process.env`:

```
npm install dotenv
# or
yarn add dotenv
```

Step 3: Creating a .env File

Create a `.env` file in the root of your React project. This file will store your environment-specific variables. For example:

```
REACT_APP_API_KEY=your_api_key_here
REACT_APP_BASE_URL=https://api.example.com
```

Step 4: Loading Environment Variables

In your React application's entry point (typically `index.js` or `App.js`), import and configure dotenv to load variables from the `.env` file:

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import dotenv from 'dotenv';

dotenv.config();

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Step 5: Accessing Environment Variables

Now you can access your environment variables anywhere in your React components using `process.env.VARIABLE_NAME`. For example, to use the API key in your component:

```
import React from 'react';

const App = () => {
  const apiKey = process.env.REACT_APP_API_KEY;

  return (
```

```
      <div>
        <h1>Using dotenv in React</h1>
        <p>API Key: {apiKey}</p>
      </div>
    );
};

export default App;
```

Step 6: Handling Different Environments

Remember to create separate `.env` files for each environment (e.g., `.env.development`, `.env.production`) to manage variables specific to development, testing, and production environments.

Conclusion

In this tutorial, you've learned how to integrate and use dotenv in a React project to manage environment variables effectively. This setup ensures that your sensitive configuration data remains secure and flexible across different deployment environments.

Using dotenv simplifies the process of configuring and accessing environment variables in your React applications, making it an essential tool for modern web development workflows.

Now you're ready to handle environment-specific configurations confidently in your React projects using dotenv!

Reactjs | Dotenv | Front End Development | Javascript Development | React Environment

Follow

Written by Rivai Fansuri Nasution

2 Followers

Lorem ipsum dolor sit amet consectetur adipisicing elit. Vel amet quod perspiciatis rerum error maxime rem eveniet, ullam ducimus fugit nostrum quibusdam quis

More from Rivai Fansuri Nasution

Rivai Fansuri Nasution

Introduction to Class Variance Authority (CVA) in React

Class Variance Authority (CVA) is a utility for managing CSS class names based on various conditions. It helps create consistent and...

Jun 13 3

Rivai Fansuri Nasution

Using Day.js with React

Day.js is a minimalist JavaScript library for manipulating dates, similar to Moment.js, but smaller in size and with better performance...

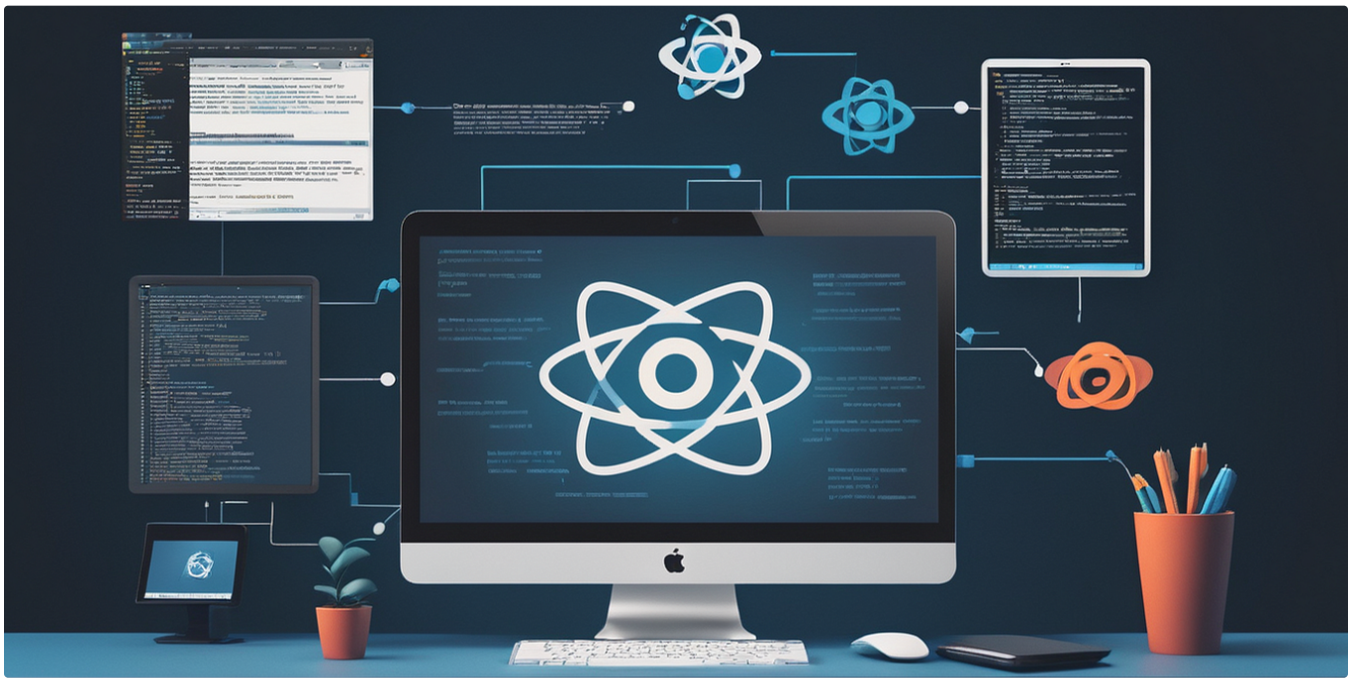
Jun 13 1

Rivai Fansuri Nasution

Fixing “React-Leaflet Marker Files Not Found” Error in Your Project

If you’ve encountered issues with missing marker files when using React-Leaflet, this guide will help you resolve the problem. The error...

Jul 16



Rivai Fansuri Nasution

How to Use clsx in React

Managing classes in React can sometimes get cumbersome, especially when you need to conditionally apply classes. The clsx library provides...

Jun 29 1

See all from Rivai Fansuri Nasution