{JS}
MASTERY

# The Ultimate ReactJS Guide

Created by **JS Mastery**

Visit ***jsmastery.pro*** for more

# The Ultimate React.js Course 🔥

We’re excited to let you know that your dream of learning and building React applications from scratch is within reach!

We’re working on a React.js course that will take you from zero to production-ready. You’ll learn state management, testing, advanced patterns, and more to build and deploy apps like a pro.

I’ll also dive into advanced concepts like performance optimization, caching, and SEO—focused on real-world, production-level practices.

If this sounds like what you're looking for, join us quickly to stay updated on course details and suggest any features you’d like to see.

# The Ultimate React.js Course 🔥

If you have any questions or would like to suggest features you'd like to see (we value your input), feel free to reach out.

You can reply to the email from which you received this guide, or drop your suggestion in the

**💡| suggestions**

And here's some good news:

You'll also have the opportunity to receive a special discount just for you! 🎉

# What's in the guide?

Learning JavaScript libraries and frameworks can be overwhelming. There are many libraries to choose from, and no proper step-by-step guides that'll teach you how to use these libraries to their fullest potential.

That's why, in this guide, you'll learn the most popular JavaScript library, used by hundreds of thousands of developers worldwide - **React.js**.

This guide covers the **complete React.js roadmap**, **JavaScript prerequisites**, **essential React.js concepts**, and **project ideas** that you can build & deploy and put up on your portfolio and get a job.

# Introduction to React.js

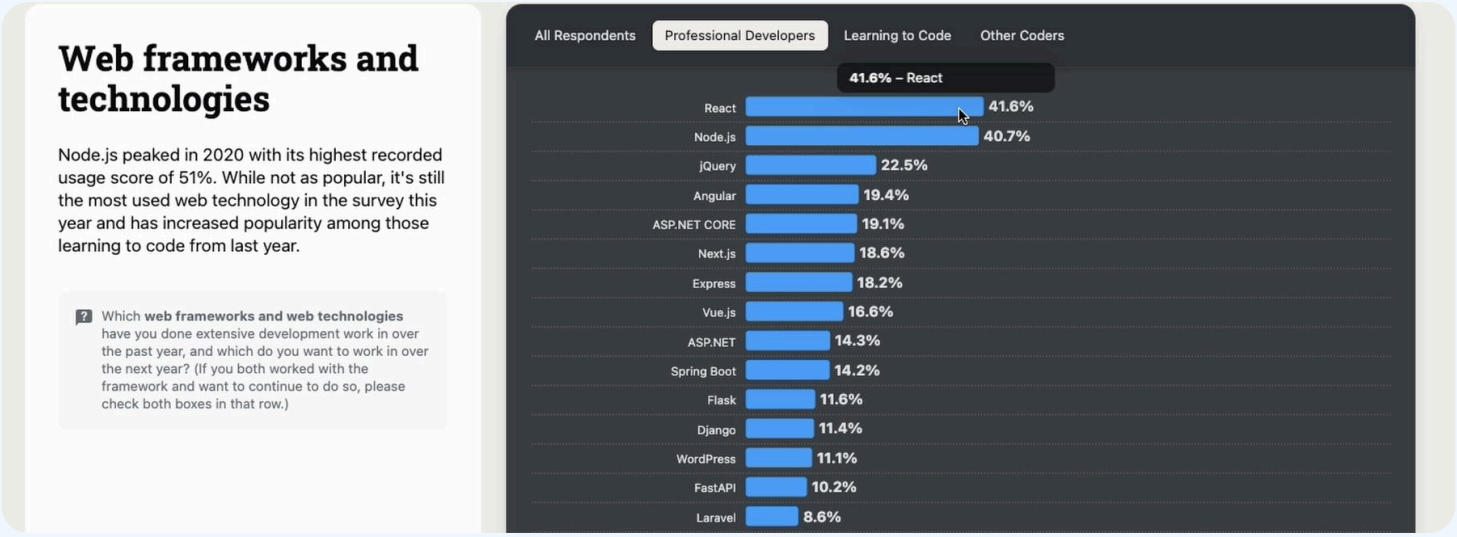
React.js is a front-end JavaScript library for building user interfaces. It was developed by Facebook and is maintained by Facebook and the open-source community.

React.js is a phenomenal library that is easy to understand, has excellent cross-platform support, has a fantastic community, and is one of the most loved libraries out there.

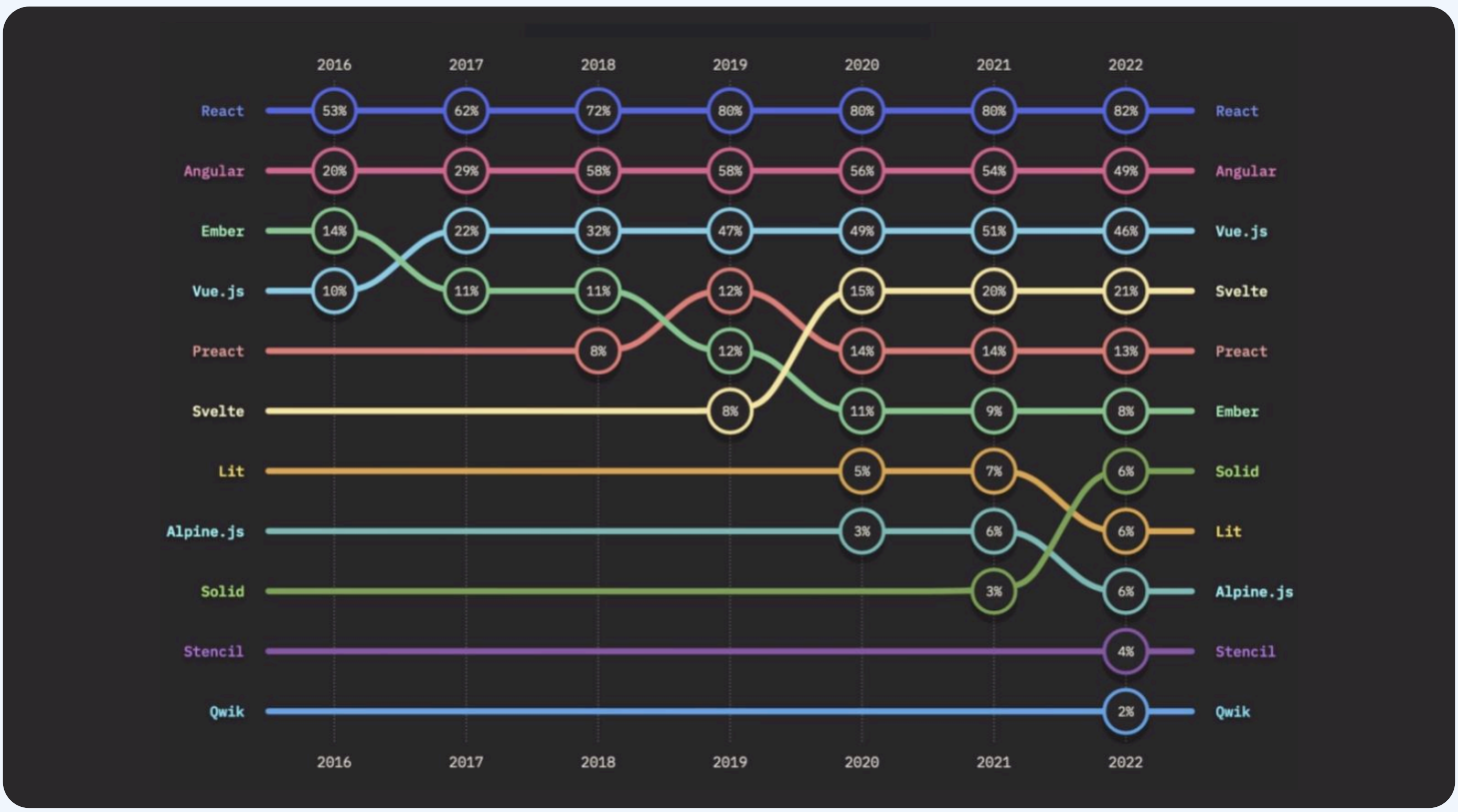
Of Course, React isn't alone. Frameworks like Vue.js, Svelte, Astro, and Angular are also making waves. But when it comes to popularity, React dominates. Just look at the numbers:

- **41.6% of developers** on the Stack Overflow Developer Survey chose React.js.

# Introduction to React.js




React has topped the State of JS survey for six consecutive years.



# Introduction to React.js

And with good reason. React is the backbone of key stacks like MERN and PERN, extends to mobile development with React Native, and integrates seamlessly with modern frameworks like Next.js for full-stack development.

In short: Learn React, and you'll have the tools to build apps for web, mobile, and beyond. It's your ticket to becoming the Swiss Army knife of developers.

Now that you understand just how popular React is, let's dive into the prerequisites for learning React.

Then we'll explore its syntax, JSX, code structure, and other essential concepts in detail.

# JavaScript prerequisites

You might be wondering, what are the prerequisites to learn such a great JavaScript library?

There's only one prerequisite and that is - JavaScript.

Do not jump straight into React.js without understanding the topics I've mentioned below.

Before learning React, you should have a good understanding of these JavaScript topics:

- Basic Syntax
- ES6+ features
- Arrow functions
- Template literals

# JavaScript prerequisites

- Array Methods
- Object property shorthand
- Destructuring
- Rest operator
- Spread operator
- Promises
- Async/Await syntax
- Import and export syntax

# React.js Roadmap

## *Basic things to learn in React.js*

- File & Folder structure
- Components
- JSX
- Props
- State
- Events
- Styling
- Conditional Rendering

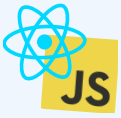
# React.js Roadmap

*Learn about React.js Hooks - the essential hooks to learn:*

- useState
- useEffect
- useRef
- useContext
- useReducer
- useMemo
- useCallback

# React.js Roadmap

*Learn how to style React components effectively*




Inline Styles

CSS

TailwindCSS


CSS Modules


Sass (optional)

# React.js Roadmap

*Then learn some of the React.js UI Frameworks*

- Material UI
- Ant Design
- Chakra UI
- React Bootstrap
- Rebass
- Blueprint
- Semantic UI React

# React.js Roadmap

*Learn to use some of the most popular React.js packages*

React Router

TanStack Query

Axios

React Hook Form

Styled Components

Storybook

Framer Motion

# React.js Roadmap

*Learn how to manage state with state management tools*

- ⭐  Redux
-  MobX
-  Hookstate
-  Recoil
-  Akita

# React.js Roadmap

*More things to learn after learning React.js fundamentals*

- ⭐  Next JS
-  Gatsby
-  TypeScript
-  React Native
-  Electron

# React.js Roadmap

*Learn to test your React.js applications with some of these libraries/frameworks*

- ⭐  Jest
- ⭐  Testing Library
-  Cypress
-  Enzyme
-  Jasmine
-  Mocha

# React.js Roadmap

*Learn to deploy your React.js applications*

- ⭐  Vercel
- ⭐  Netlify
-  Firebase
-  Github Pages
-  Heroku
-  Render

# React.js Setup

Before starting a React project, make sure you have Node.js installed.

Node.js is a JavaScript runtime that enables running JavaScript outside of the browser, which is necessary for React development tools to work on your machine.

Go to nodejs.org and download the LTS version, as it's stable and well-supported.

You can install Node.js using a bash command or package installer, depending on your OS.

Along with Node.js, you'll also get npm (Node Package Manager). It lets you add libraries and frameworks to your project quickly and easily,

# React.js Setup

keeping your tools & code updated & saving time.

Once Node is installed, you'll need a place to write your code. I recommend **WebStorm**—an IDE designed for React.js development with features like error reporting and Git integration.

While Git isn't mandatory to start, it's essential for any developer, especially React.js developers. Download Git from **here**, and check out my Git YT tutorial on using Git & GitHub efficiently.


**Link to the video**

# React.js Concepts

## Components

React JS is a **component-based** front-end library which means that all parts of a web application are divided into small components.

A component is a small piece of the User interface. Every React.js application is a tree of components.

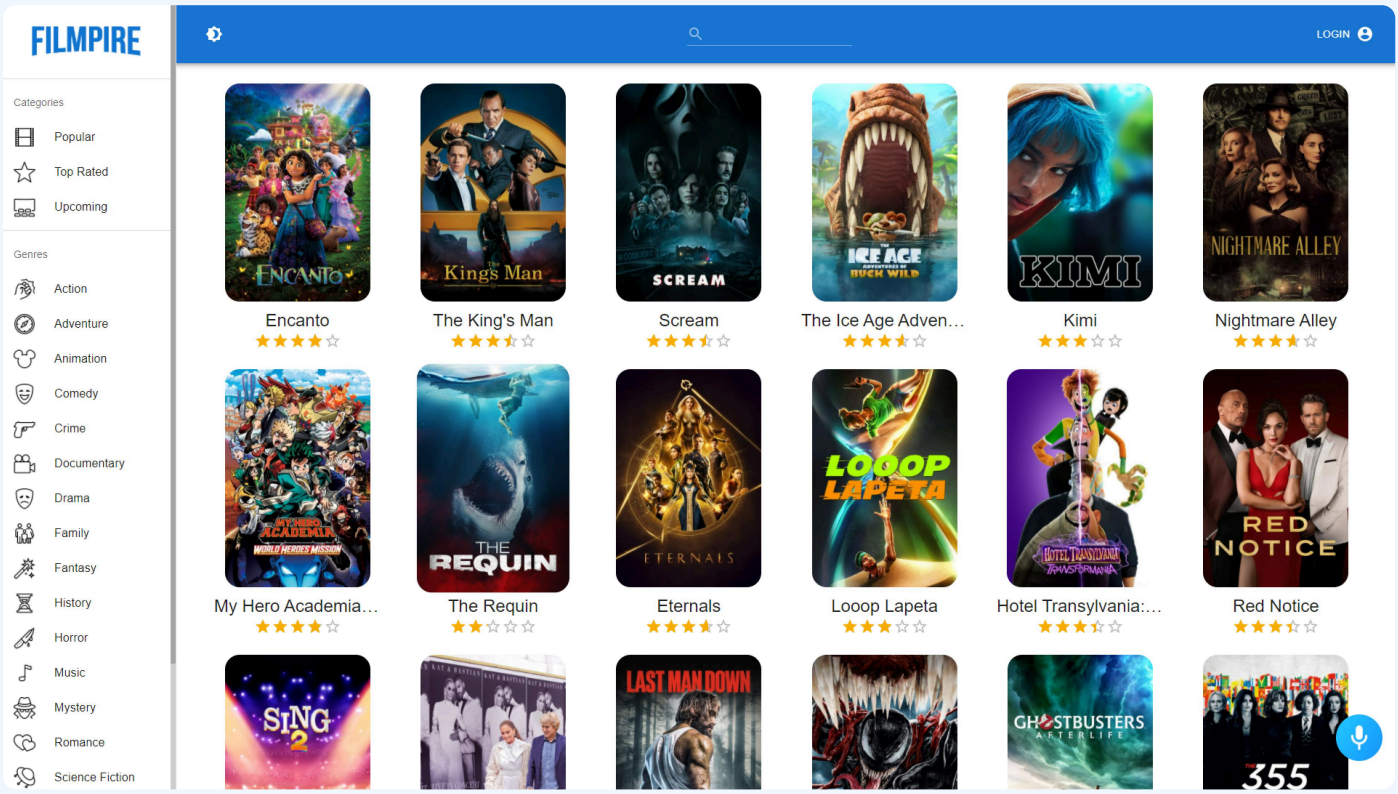
Components let you split the UI into independent, reusable parts.

So when you're building an application with React, you'll build independent and reusable components, and then you'll combine them to build a full fledged web application.

# Components explanation

Let's take an example to represent what are

React.js components:

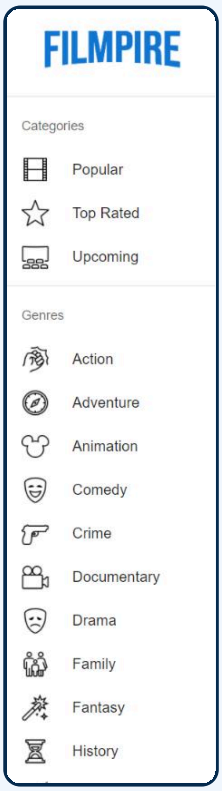



filmpire.netlify.app

This website is entirely built in React.js. So imagine we're building this website. How would we make it?

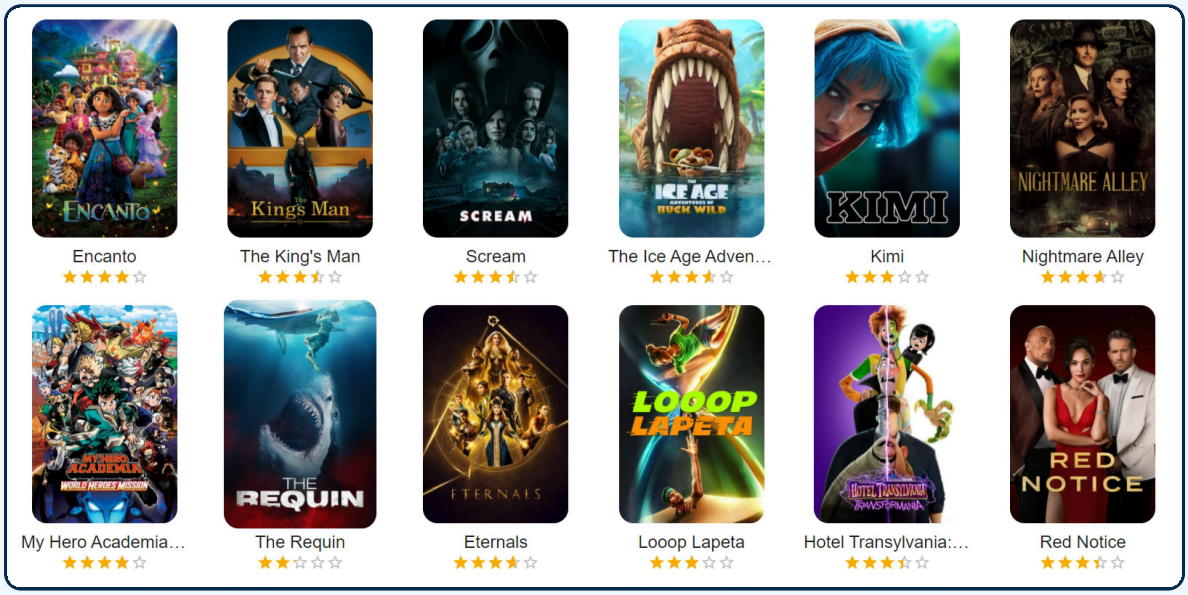
# Components explanation

Firstly we'll split the User Interface into small components like **Sidebar**, **Search bar**, and **Movies**, including several **single movie** components with names and ratings.

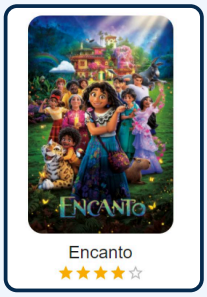



Sidebar


Search bar




Movies




Single movie component

# Components explanation

In React, there are two types of components - Functional Components & Class Component

```
import React, {Component} from 'react'

class Example extends Component {
    render() {
       return <div>Hello World!</div>
    }
}
```

Class-based Component

# Components explanation

If you don't fully understand, how to use classes, what are the class methods, and what does 'extends' means, don’t you worry at all. Class based are not being used at all anymore and they were replaced by their simpler counterparts ↓

```
import React from ‘react’

const Example = () => {

  return <div>Hello World!</div>

}
```

Functional Component

# Components explanation

That's it! This is a React Component. You can see how easy it is.

You might be thinking, why are we writing HTML when returning something.

```
import React from 'react'

const Example = () => {

  return <div>Hello World!</div>

}
```

This tag syntax is neither a string nor HTML.
It is called **JSX**.

# JSX - JavaScript XML

JSX is a syntax extension to JavaScript. It is used in React to describe what the UI should look like. JSX may remind you of a template language, but it comes with the full power of JavaScript.

JSX produces React "elements". JSX forms the core syntax of React..

```
return (
  <div className="greetings">
    <h1> Hello, {prop.name} </h1>
  </div>
)
```

# JSX - JavaScript XML

There are a few differences between HTML & JSX, although generally it's incredibly similar. Some of the differences are:

Writing className instead of class,

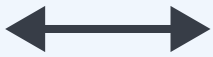

Because the class is a reserved keyword in JavaScript. Since we use JSX in React, the extension of JavaScript, we have to use '**className**' instead of the **class** attribute.

# JSX - JavaScript XML

Same as class, there's also one more reserved keyword of JavaScript that is used in HTML. That is the '**for**' attribute used with the *<label>* element.

So, to define **for** attribute in JSX, you do it as 'htmlFor'

```
<label htmlfor="">
```



```
<label for="">
```

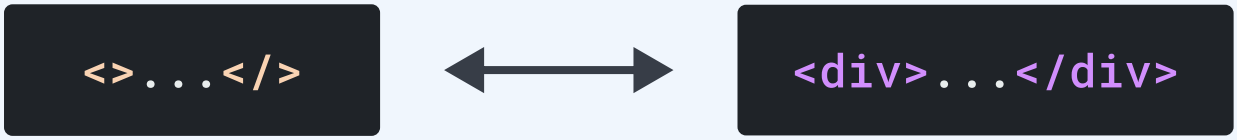
```
return (
  <form className="form">
    <label htmlFor="name">Name</label>
    <input type="text" id="name" />
  </form>
)
```

Example

# JSX - JavaScript XML

One of the major differences between HTML and JSX is that in JSX, you must return a single parent element, or it won't compile.

You can use '**React fragments**' instead of **divs**




You can also use **divs** instead of React fragments, it's not necessary to use a particular, but using '**React fragments**' makes the code more readable.

# JSX - JavaScript XML

You can implement JavaScript directly in JSX. To use JavaScript expressions, we use curly brackets `{...}`

```
return (
  <div>
    <h1> Hello, {prop.name} </h1>
    <p> {3 + 7} is ten </p>
  </div>
)
```

Whereas in HTML, you need a script tag or an external JavaScript file to implement JavaScript

# What are Props?

To make our components accept different data, we can use props. Props are arguments passed into React components. They are passed to components via HTML attributes.

Props is just a shorter way of saying properties.

We use props in React to pass data from one component to another (*from a parent component to child components*), But you can't pass props from a child component to parent components.

Data from props is read-only and cannot be modified by a component receiving it from outside.

# What is State?

State is like a React component's brain. It holds information about the components that can change over time.

A State is a plain JavaScript object used by React to represent a piece of information about the component's current situation.

It's managed in the component (*just like any variable declared in a function*).

The state object is where you store property values that belongs to the component. When the state object changes, the component re-renders.

State data can be modified by its own component, but is private (cannot be accessed from outside)

# What is Events?

An event is an action that could be triggered as a result of the user action or a system-generated event. For example, a mouse click, pressing a key, and other interactions are called events.

Handling events with React elements is similar to handling events on DOM elements. There are just some syntax differences.

- React events are named using camelCase, rather than lowercase.

- With JSX you pass a function as the event handler, rather than a string.

# How to add Events?

```
const handleClick = () => {...}

return (
  <button onClick={handleClick}>
    Explore More
  </button>
)
```

# What are React.js Hooks?

Hooks are a new addition to React in version 16.8 that allows you to use state and other React features, like lifecycle methods. Using hooks makes your code cleaner.

Hooks don't work inside classes — they let you use React without classes.

Hooks let you always use functions instead of having to constantly switch between functions, classes, higher-order components, and render props.

Nowadays, hooks are widely used. So you should start using it as well. You can also create your own Hooks to reuse stateful behavior between different components.

# React.js Hooks?

Hooks in React are special functions that allow you to access React features like state management and lifecycle methods in functional components.

There are various hooks for different tasks, such as

`useState`

→ for managing state

`useEffect`

→ for handling side effects like data fetching

`useContext`

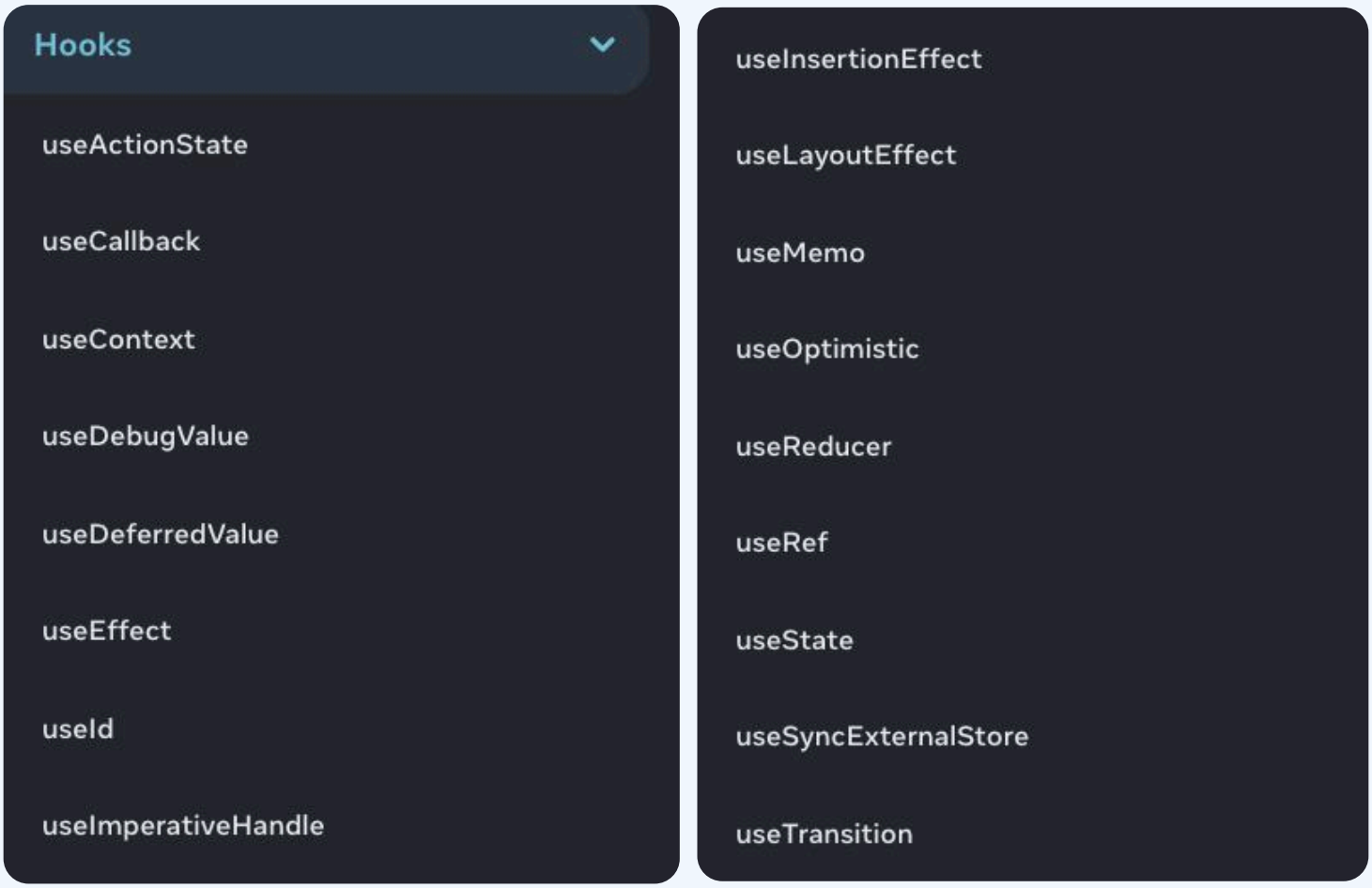
→ for sharing data across components

`useCallback`

→ for optimizing callback functions

# React.js Hooks?

And there are many more....




If you want to go deeper, check out my React.js Pro course, where I dive into each hook with practical examples, use cases, and insights.

# Project Ideas


SaaS Landing Page


Awwwards Website Clone

3D Portfolio

Apple Landing Page

Real-Time Dashboard

Social Media Platform

Nike Shoes Landing Page


AI SaaS Application

# Project Ideas

Real Estate App

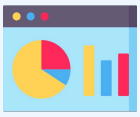
Cryptocurrency App

Travel Companion App

ECommerce Web Shop

Voice Assistant News App

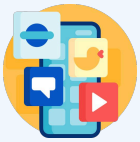
Portfolio Website



Voice Powered Budget Tracker

Blog App with CMS

# Project Ideas



Social Media Web App

Modern UI/UX Website

Chat App

Video Chat App

Progressive Web Apps

3d T-Shirt Ecommerce App

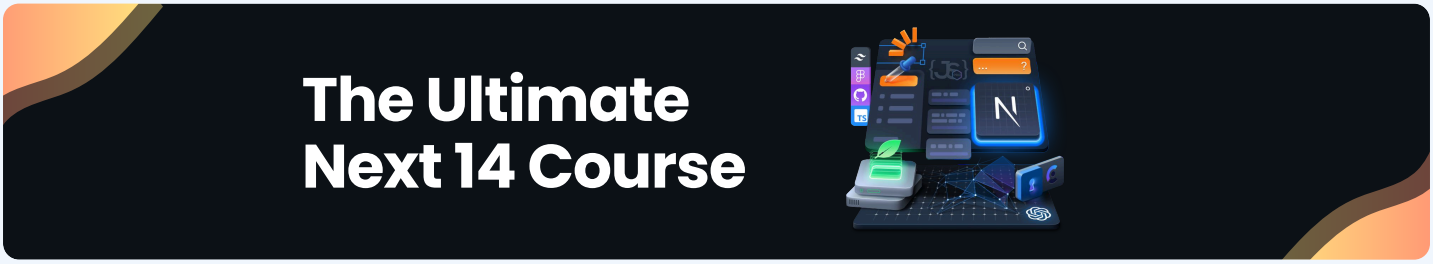
Google Search Clone

Premium Landing Page

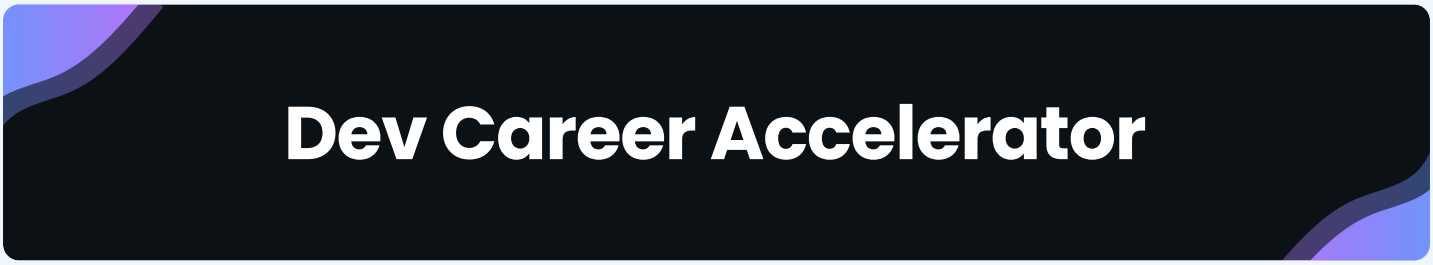
# The End

Congratulations on reaching the end of our guide! But hey, learning doesn't have to stop here.

If you're eager to dive deep into something this specific and build substantial projects, our **special course on Next.js** has got you covered.




If you're craving a more personalized learning experience with the guidance of expert mentors, we have something for you — [Dev Career Accelerator](#).




If this sounds like something you need, then don't stop yourself from leveling up your skills from junior to senior.

Keep the learning momentum going. Cheers! 🚀