



INTRODUCTION, OVERVIEW

Introduction to React

A popular JavaScript library for creating dynamic user interfaces

Bhavesh Garg





MASTERING REACT TOPICS

Agenda

Explore Key Topics in Mastering React

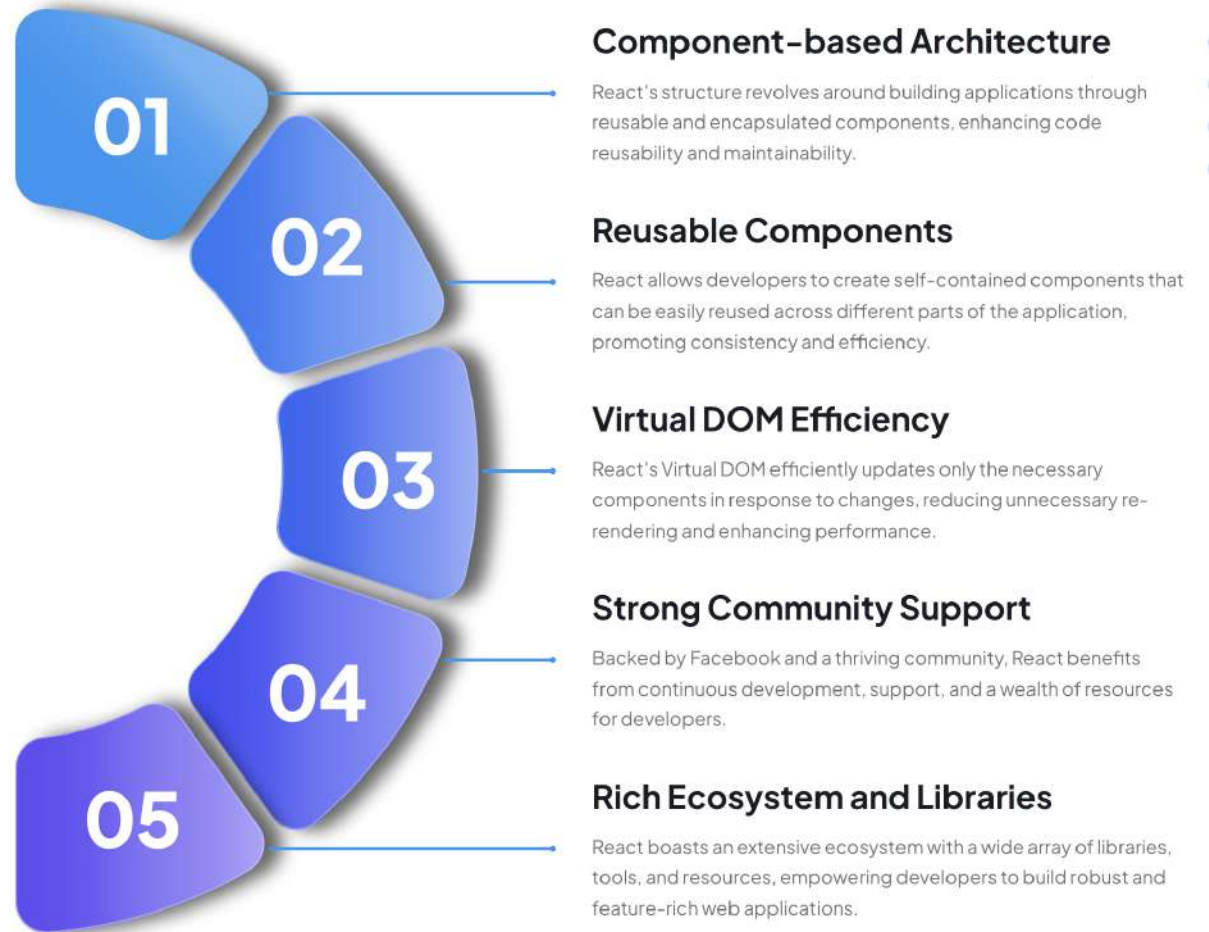
- 01 Why React is so popular?
- 02 JSX Syntax
- 03 CSS Frameworks
- 04 Package Bundlers
- 05 React Hooks
- 06 Additional Topics



TECHNOLOGY INSIGHTS

Why React is so Popular?

Exploring the Key Factors Driving React's Popularity



JSX Syntax

Understanding JSX in React Development





01

Combining HTML with JavaScript

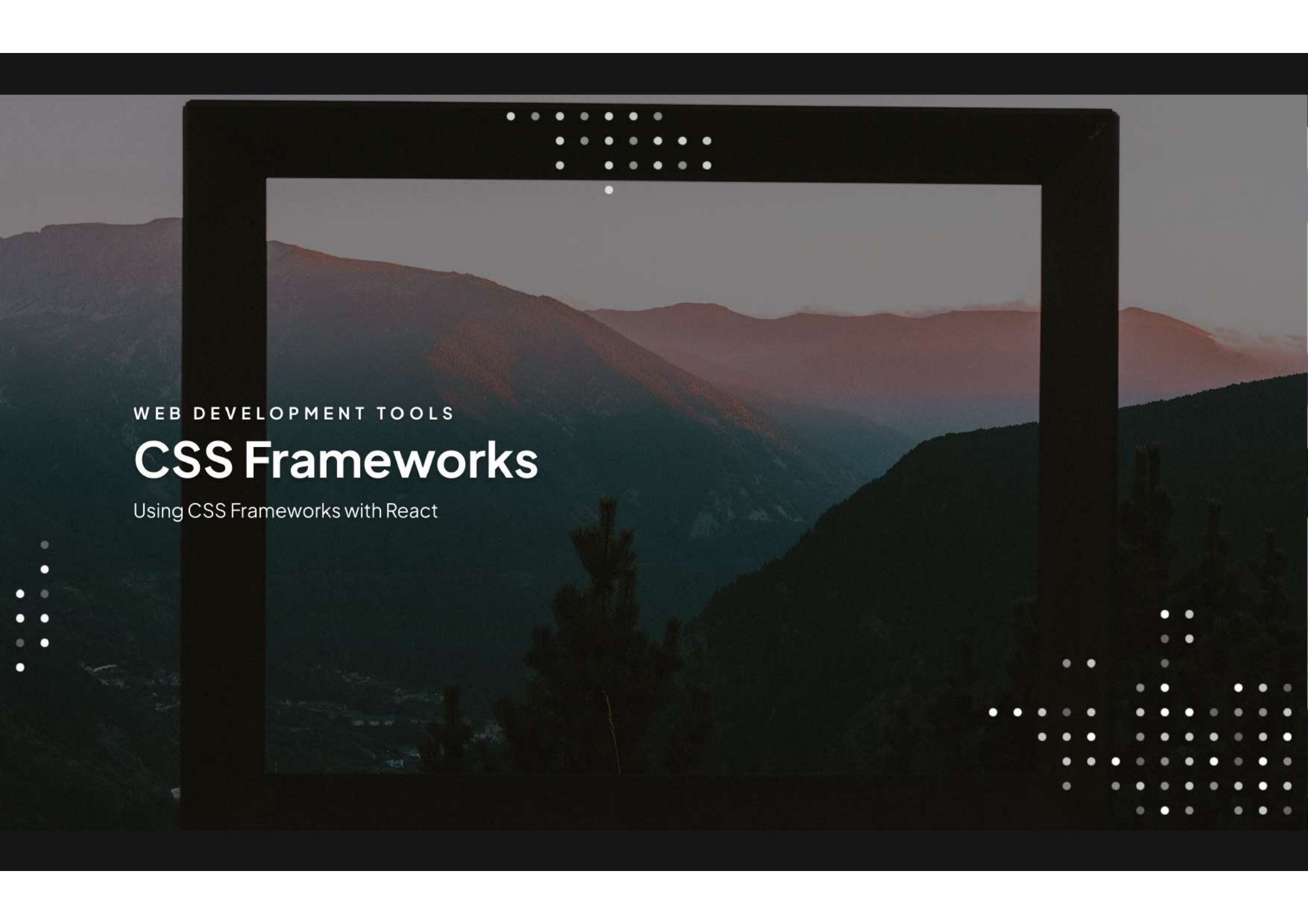
JSX is a syntax extension for JavaScript that combines HTML structure and JavaScript logic, making it easier to create UI components in React.

The background of the slide is a photograph of a city skyline, likely New York City, with various skyscrapers visible. In the foreground, there is a large, black, industrial-looking billboard structure. The billboard itself is a large white rectangle in the center of the frame. The number '02' is centered on the billboard in a blue font. Below it, the title 'Simplifying Rendering Logic' is written in a larger blue font. Underneath the title, a paragraph of text explains that JSX allows developers to write cleaner code for UI elements. The overall aesthetic is clean and modern, with a focus on the technical topic of rendering logic in React.

02

Simplifying Rendering Logic

By using JSX, developers can write cleaner and concise code to define the UI elements and their behavior, simplifying the rendering process in React applications.



WEB DEVELOPMENT TOOLS

CSS Frameworks

Using CSS Frameworks with React



How to Use Any CSS Framework in Your Project

Step-by-step guide to integrating CSS frameworks in React



Step 1

Install the desired CSS framework using npm or yarn to manage dependencies efficiently.

Step 2

Integrate the CSS file into your React project by importing it.
For example, with Tailwind CSS, you can refer to the documentation here:

[Tailwind CSS Installation Guides.](#)

Follow the documentation above.

Step 3

Ensure that the CSS framework is correctly linked to your project to leverage its styling capabilities effectively.

Step 4

Test the implementation by applying styles from the CSS framework to your project elements and components.

Step 5

Optimize the usage of the CSS framework by customizing styles and leveraging its features to enhance your project's visual appeal and functionality.

Step 6

Regularly update the CSS framework to access new features, bug fixes, and performance enhancements that can benefit your project.

02

REACT PACKAGE BUNDLERS

Package Bundlers

Understanding Package Bundlers in React





BUILD TOOLS COMPARISON

Differences between Vite and Create React App

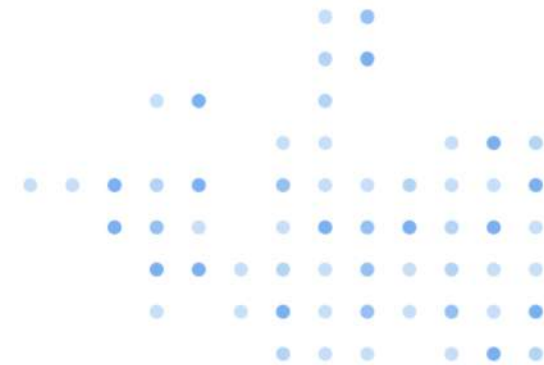
Comparing Build Tools for React Applications

Vite

1. Fast development speed using ES modules and esbuild,
2. Flexible, easy customization with **vite.config.js**.
3. Highly responsive with fast HMR.
4. Growing, supports Rollup plugins.

Create React App

1. Slower development due to Webpack bundling.
2. Zero-config out of the box, requires ejecting for customization.
3. Smooth but slower rebuilds and HMR.
4. Robust, primarily through community packages.





INTRODUCTION, OVERVIEW

React Hooks

An Overview of React Hooks



Focus on useState and useEffect

A diagram illustrating the components of the `useState` function signature. At the top, the text "state variable" is on the left and "default value" is on the right. Below "state variable" is a downward-pointing arrow. Below "default value" is a downward-pointing arrow. In the center, the code `const [state, setState] = useState(false);` is shown. Below this code is an upward-pointing arrow. At the bottom, the text "function that changes state" is displayed.

useState is used to manage state in functional components, allowing for the storage and updating of component-specific data.

useEffect is utilized to manage side effects in functional components, enabling actions like data fetching, subscriptions, or manually changing the DOM.

```
useEffect(() => {
  console.log('all the time');
});
```

first render AND update

```
useEffect(() => {
  console.log('only once');
}, []);
```

first render ONLY

```
useEffect(() => {
  console.log(`on ${variable} update`);
}, [variable]);
```

update ONLY

REACT HOOKS EXAMPLES

Demonstrations on Implementing Hooks

Illustrating the Practical Use of useState and useEffect Hooks in React

useState Implementation

Defines a state variable 'count' initialized to 0 and a function 'setCount' to update its value.



useEffect Example

Executes a side effect: setting the document title to include the current 'count' value, with a dependency on 'count'.



Walkthrough of Chrome Developer Tools for Debugging

Mastering React: A Comprehensive Guide to Building Dynamic Web Applications



Step 1

Inspect and manipulate elements within a webpage using Chrome Developer Tools.

Step 2

Step 3

Walkthrough of Chrome Developer Tools for Debugging

Mastering React: A Comprehensive Guide to Building Dynamic Web Applications

Step 1



Step 2

Strategies for debugging JavaScript code to identify and resolve errors efficiently.

Step 3

Walkthrough of Chrome Developer Tools for Debugging

Mastering React: A Comprehensive Guide to Building Dynamic Web Applications

Step 2



Step 3

Monitoring and analyzing network requests to optimize performance and troubleshoot issues.

REACT CONCEPTS EXPLORATION

Additional Topics

Exploring Other Important React Concepts



REACT HOOKS

BASIC UNDERSTANDING OF OTHER HOOKS

Exploring Additional React Hooks for Enhanced Functionality



useContext

Allows accessing context within a functional component, enabling the sharing of values without prop drilling.

useReducer

An alternative to useState, manages complex state logic through dispatching actions and specifying reducers.

useMemo

Memoizes the result of a function, optimizing performance by caching expensive calculations until dependencies change.

useCallback

Returns a memoized callback function to prevent unnecessary re-renders in child components by memoizing functions.

React Routing

Exploring React Router for Dynamic Navigation





Using React Router

Leveraging React Router library to enable seamless navigation within React applications.



Defining Routes

Establishing routes in React applications to map specific URLs to corresponding components for dynamic content rendering.



Linking Components

Creating links to different components within the application, facilitating user interaction and navigation between various sections.

Route Example

Illustration showcasing a route configuration where the path `'/home'` is linked to the `'Home'` component for rendering content specific to the home page.

Props

Passing Data Between Components

01

Props are used to pass data from one component to another in React. This facilitates communication between components.

02

03

Props

Passing Data Between Components



01

02

In React, props are passed as attributes to child components, as shown in the example: .

03

04



Props

Passing Data Between Components

02

03

Props flow in a unidirectional manner, from parent to child components. This ensures a predictable data flow in React applications.

04

05

Props

Passing Data Between Components



03

04

Props are immutable, meaning that components receiving props cannot modify them. This enforces data consistency and helps in debugging.

05

06



Props

Passing Data Between Components



04

05

React allows the definition of prop types for components, specifying the type of data expected. This enhances code robustness and error handling.

06



Props

Passing Data Between Components

05

06

By passing props, components become more reusable and versatile, as they can dynamically render based on the data received.

Summary of Key Takeaways

Insights into Mastering React for Dynamic Web Applications



React Components Simplify UI Development

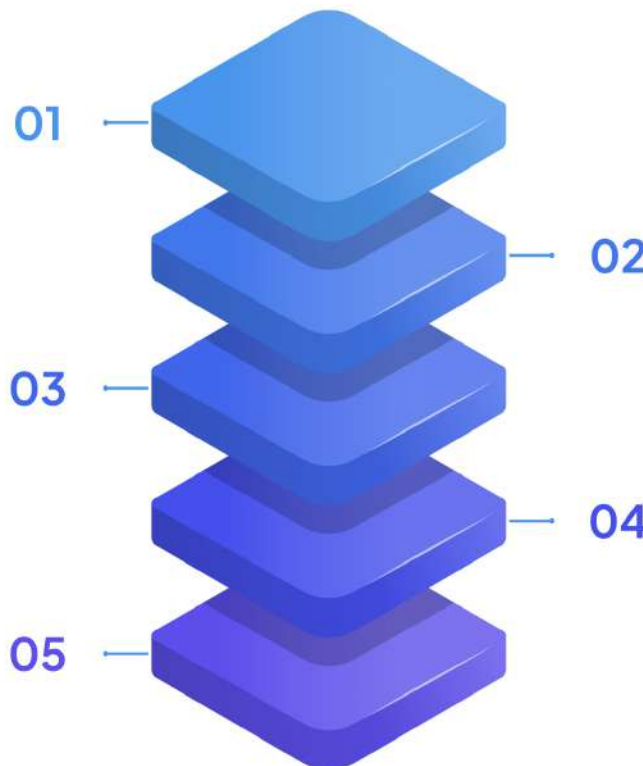
React's component-based architecture streamlines the UI development process, making it easier to manage and maintain code.

CSS Frameworks Enhance Styling

Leveraging CSS frameworks can significantly improve the visual appeal and responsiveness of web applications built with React.

useState and useEffect are Crucial Hooks

The useState and useEffect hooks in React are fundamental for managing state and side effects, respectively, enabling dynamic functionality.




JSX Combines HTML and JS for Clarity

JSX, a syntax extension for JavaScript, allows the mixing of HTML and JavaScript, enhancing code readability and maintainability.

Vite and CRA Offer Distinct Advantages

Vite and Create React App (CRA) present unique benefits and features, catering to different project requirements and developer preferences.



Thanks, everyone, for joining!
I hope you learned something new.

