Vue.js Training

by TBSCG

INTRODUCTION to VUE.JS

1. Brief History
2. Features
3. Comparison with Other Frameworks

Brief History of Vue.js

Vue.js was created by ***Evan You***, a former ***Google engineer***, in 2014.

Dissatisfied with the limitations and complexities of other frameworks, he set out to build a framework that offered the best parts of Angular (data binding) and React (component-based architecture) but in a lighter, more manageable package. Vue started as a personal project and quickly gained popularity due to its simplicity, flexibility, and powerful features.

Over the years, it has grown significantly, backed by a strong community and a dedicated core team.

Features of Vue.js - part 1

Reactive Data Binding:

Vue's reactive data binding allows developers to bind data to the DOM and automatically update it whenever the underlying data changes. This makes it easy to keep the user interface in sync with the application's state.

Component-Based Architecture:

Vue promotes building UIs using components, which are self-contained, reusable pieces of code that manage their own state and rendering logic. This modular approach helps in organizing and maintaining the codebase.

Features of Vue.js - part 2

Directives

Vue provides special tokens in the template language, like ***v-bind*** and ***v-if***, that offer a clean way to apply logic and effects to the DOM.

Single-File Components (SFCs)

Vue's single-file components allow developers to encapsulate the HTML, JavaScript, and CSS of a component in a single ***.vue*** file much like Web Components. This keeps all related code together, making it easier to manage and understand.

Features of Vue.js - part 3

Vue Router

The official router for Vue.js enables developers to build single-page applications (SPAs) with ease, offering features like nested routes, dynamic routing, and route guards.

Vuex

Vuex is a state management library tailored for Vue.js applications. It provides a centralized store for all the components in an application, ensuring a single source of truth for the state, much like the Context API or Redux for React.

Features of Vue.js - part 4

Vue CLI

Vue CLI is a command-line tool that helps in quickly scaffolding, developing, and deploying Vue applications. It provides a rich ecosystem of plugins and presets to streamline development.

Transition Effects:

Vue includes built-in directives for adding transition effects when elements are inserted, updated, or removed from the DOM.

Comparison with React - part 1

Similarities

Component-Based Architecture: Both React and Vue.js use a component-based architecture, making it easy to reuse code and build complex user interfaces.

Virtual DOM: Both frameworks utilize a Virtual DOM for efficient rendering and updating of the user interface.

Reactive Data Binding: Both offer reactive data binding, though they implement it differently.

Ecosystem: Rich ecosystems with state management libraries, routing solutions, and extensive documentation.

Differences

Template Syntax: Vue.js uses a more HTML-like template syntax, which is easier for those familiar with HTML. React uses JSX, a syntax extension that combines JavaScript and HTML.

Reactivity System: Vue.js has a built-in reactivity system that tracks dependencies automatically. React relies on state and props to manage reactivity, requiring more explicit management.

Comparison with React - part 2

Performance

Rendering Performance: Both have similar performance for most use cases, though React Fiber (React 16+) has optimized update handling.

Initial Load: Vue.js can be slightly faster to load initially due to its smaller size, but this difference is minimal with modern bundling and tree-shaking techniques.

Ecosystem & Learning Curve

Learning Curve: Vue.js is often considered easier to learn, especially for beginners, due to its simpler and more intuitive syntax. React may require a deeper understanding of JavaScript and JSX.

Ecosystem: React has a larger ecosystem with more third-party libraries and tools. Vue's ecosystem is growing rapidly, with official support for state management (Vuex) and routing (Vue Router).

Comparison with Angular- part 1

Similarities

Component-Based Architecture: Both Angular and Vue.js are built around components, facilitating modular development.

Reactive Programming: Both support reactive programming, though Angular uses RxJS extensively for this purpose.

Differences

Size and Complexity: Angular is a full-fledged framework with a steep learning curve and a larger bundle size. Vue.js is more lightweight and easier to pick up.

Learning Curve: Vue.js is generally easier to learn, while Angular's extensive features and complexity require more time to master.

Comparison with Angular- part 2

Integration and Flexibility

Integration: Vue.js is more flexible and easier to integrate into existing projects. Angular's monolithic nature can make integration more challenging.

Flexibility: Vue.js allows for more flexible project structures and gradual adoption, while Angular enforces a specific project structure and setup.

Community and Adoption

Community: Both have strong communities, but Angular's enterprise backing by Google gives it a more established presence in larger organizations.

Adoption: Angular is often favored in large-scale enterprise applications due to its comprehensive nature, while Vue.js is popular in startups and companies valuing flexibility and ease of use.

Why use Vue.js?

User-Friendly: Intuitive syntax and comprehensive documentation make it accessible for beginners.

Incremental Adoption: Can be used for small parts of a project or the whole application, making it easy to integrate into existing projects.

Efficient: Lightweight and fast, ensuring good performance even in

Flexible: Suitable for both simple and complex projects, adaptable to various use cases. complex applications.

Community Support: A growing and active community providing support, plugins, and extensions.

Official Tools: Well-maintained official tools like Vuex for state management and Vue Router for routing.

Questions?

Please feel free to ask any questions.

Cheat Sheet for Vue Directives and Template Syntax

- ***v-bind*** Binds an attribute to an expression. Example: <img v-bind:src="imageSrc">
- ***v-model*** Creates two-way binding on an input, textarea, or select element. Example: <input v-model="message">
- ***v-if*** Conditionally renders an element. Example: <p v-if="seen">Now you see me</p>
- ***v-else*** Specifies an alternative block to be rendered when v-if is false. Example: <p v-else>Now you don't</p>
- ***v-show*** Toggles the visibility of an element. Example: <p v-show="isVisible">I am visible</p>
- ***v-for*** Renders a list of items. Example: <li v-for="item in items">{{ item.text }}</li>
- ***v-on*** Attaches event listeners. Example: <button v-on:click="doSomething">Click me</button>
- ***v-bind:class*** Binds a class to an element. Example: <div v-bind:class="{ active: isActive }"></div>
- ***v-bind:style*** Binds inline styles. Example: <div v-bind:style="{ color: activeColor, fontSize: fontSize + 'px' }"></div>
- ***v-pre*** Skips compilation for this element and all its children. Example: <span v-pre>{{ raw }}</span>
- ***v-cloak*** Keeps the element cloaked until the Vue instance is ready. Example: <div v-cloak>{{ message }}</div>
- ***v-once*** Renders the component once. Example: <span v-once>This will never change: {{ message }}</span>