

DWA_12 Knowledge Check

To complete this Knowledge Check, ensure you have worked through all the lessons in **Module 12: Declarative Abstractions**.

To prepare for your session with your coach, please answer the following questions. Then download this document as a PDF and include it in the repository with your code.

1. What are the benefits of direct DOM mutations over replacing HTML?

Reduced Network Traffic: Even if just a small piece of the HTML has changed, you usually need to fetch and transmit the full document when replacing HTML. You may deliver smaller, more targeted changes thanks to direct DOM mutations, which lowers network traffic and speeds up load times, especially for larger documents.

Fine Control of Components: Direct DOM modifications provide you more fine-grained control over how the page is modified. Individual items, properties, or styles can be accurately changed without changing unrelated areas of the page. Web applications may be made to be more dynamic and responsive thanks to this granularity, which allows for a more optimized and personalized user experience.

2. What low-level noise do JavaScript frameworks abstract away?

Event Handling: Event handling is made simpler by frameworks by including tools for event delegation and event listeners. Developers may concentrate on the logic of handling events rather than having to deal with the low-level intricacies of registering and managing events because they manage event binding, unbinding, and event propagation.

State management: Many JavaScript frameworks include tools for controlling the state of an application. They offer simple APIs to manage data and prompt the re-rendering of components or views when the state changes, abstracting away the difficulties of tracking and updating state. State management is made easier by this abstraction, which also aids in the development of more enduring systems.

3. What essence do JavaScript frameworks elevate?

Modularity and Reusability: JavaScript frameworks encourage the use of modular and reusable code architectures. They promote the usage of parts, modules, or other organizational structures that support code encapsulation and separation. This modularity enables developers to reuse components across various portions of the program or in upcoming projects, as well as making codebase management and maintenance easier.

Cross-Browser Compatibility: JavaScript frameworks frequently take care of cross-browser compatibility issues, making sure that programs function uniformly across various browsers and versions. Developers may write code without being concerned about unique browser variations due to the way they handle browser inconsistencies, offer polyfills for functionality that are absent, and abstract away the complexity of browser-specific APIs.

4. Very broadly speaking, how do most JS frameworks achieve abstraction?

The majority of JavaScript frameworks achieve abstraction by offering a layer of code that hides the underlying intricacies of the web platform and gives developers a streamlined and consistent interface to work with.

5. What is the most important part of learning a JS framework?

Understanding a JavaScript framework's fundamental ideas and tenets is crucial to learning it. Here are a few illustrations:

Key Features and APIs: Become familiar with the framework's key features and APIs. Learn how to handle data binding, maintain state, use components, deal with templates or JSX, handle routing, and handle events. You can use the framework to create dependable and scalable apps by comprehending these essential concepts.

Documentation and Tutorials: Reading and referring to the framework's official documentation is advised. The principles, features, APIs, and best practices of the framework are all covered in this useful material. You can also better comprehend the framework's application in practical situations by taking tutorials and online courses that are tailored to it.