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?

1. **Efficiency**: Direct DOM mutations are more efficient as they avoid the need to recreate the entire HTML structure, resulting in better performance.

2. **State Preservation**: Preserve the state of elements, including input values, event listeners, and scroll positions, avoiding the need for resetting or reconfiguring them.

3. **Granular Control**: With direct DOM mutations, you can selectively update specific elements or properties, providing fine-grained control over the changes in the DOM.

4. **Animation and Transition**: Allow for smoother animations and transitions by directly manipulating properties and styles.

5. **Integration with External Libraries**: Facilitate seamless integration with external libraries or frameworks that require direct access to the DOM, enabling interactions with third-party APIs or lower-level browser functionalities.

_____

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

- JavaScript frameworks abstract away low-level **details** and **complexities** in web development, allowing us to focus on higher-level application logic.

- They handle tasks such as **DOM manipulation**, **event handling**, **data binding**, **AJAX** and **networking**, **state management**, and **component abstraction**.
- Abstraction increases **productivity**, **code reusability**, and **maintainability** by providing **higher-level abstractions** and **APIs**.

- As Developers, we can concentrate on **application logic** and **user experience** without getting caught up in low-level implementation details.

_____

3. What essence do JavaScript frameworks elevate?

- JavaScript frameworks elevate web development by **enhancing productivity**, **promoting code reusability**, **supporting scalability**, **optimizing performance**, **ensuring cross-browser compatibilit**y, **fostering a strong community** and **ecosystem**, and **emphasizing maintainability**.

- They provide **tools**, **abstractions**, and **conventions that simplify complex tasks**, **enable modular** and **reusable code**, **optimize performance**, **handle cross-browser inconsistencies**, and **enforce structured** and **maintainable code organization**.

- JavaScript frameworks empower us to **focus on delivering value** and **solving business problems while building robust** and **efficient web applications**.

_____

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

- The most important part of learning a JavaScript framework is understanding its core concepts and principles, including grasping the framework's recommended architecture, design patterns, and how it manages state and data flow.

- Familiarizing yourself with the framework's core API and features, such as components, directives, and data binding, is crucial.

- In addition, it is important to familiarise yourself with the tools, development workflow, and debugging techniques offered by the framework.

- Engaging with the framework's ecosystem and community resources, working on real-world projects, and cultivating adaptability and continuous learning are also important aspects of the learning process.

- Overall, mastering the core concepts of a JavaScript framework enables effective utilization of its capabilities and empowers you to build high-quality applications efficiently.

_____

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

- The most important part of learning a JavaScript framework is understanding its core concepts and principles, which include architecture, components, data management, routing, tooling, testing, and community engagement.

- By grasping these fundamentals, you can effectively utilize the framework's capabilities, build scalable applications, handle data flow, navigate between views, optimize performance, debug effectively, and leverage community resources.

- Ultimately, this understanding empowers you to develop robust and maintainable applications using the framework.