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?

There are several benefits to using direct DOM mutations over replacing HTML:

- **Performance:** Direct DOM mutations are more efficient than replacing HTML. Replacing HTML requires re-rendering the entire content, which can be resource intensive.
- **Fine-grained control:** With direct DOM mutations you have granular control over the specific changes you want to make. This level of control allows for precise updates and avoids unnecessary reflows or repaints.
- **Compatibility with external modifications:** Direct DOM mutations work well where the DOM may be modified by external scripts or libraries.
- **Optimised updates:** By making targeted changes, you can optimise the updates to minimise the impact on the browser's rendering pipeline.
- **Reduced bandwidth usage:** When you replace HTML, you typically need to send the entire content over the network, even if only a small part has changed. With direct DOM mutations, you can send and receive only the necessary data required to make the specific changes.

While direct DOM mutations offer these benefits, it's important to consider the specific use case and the trade-offs involved. In some scenarios, replacing HTML may still be the preferred approach, especially when dealing with large-scale updates or when the underlying DOM structure needs significant modifications.

---

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

Some of the common low-level noise that JavaScript frameworks handle includes:

- **DOM manipulation:** Frameworks abstract away the complexities of directly manipulating the DOM.
- **Event handling:** Frameworks often provide abstractions for handling events in a cross-browser compatible way.
- **State management:** Many frameworks provide abstractions for managing application state. They offer features like reactive data binding, two-way data binding, or state management patterns (Redux, Vuex) that simplify the process of managing and synchronising data between components or views.
- **Routing:** Frameworks often provide routing capabilities to enable single-page application (SPA) development.
- **Cross-browser compatibility:** JavaScript frameworks strive to abstract away the differences and inconsistencies across various web browsers.
- **Build process and tooling:** Many frameworks come with built-in build processes and tooling to simplify development workflows.

By abstracting away these low-level complexities, JavaScript frameworks enable developers to focus on building applications and implementing business logic more efficiently. They promote code reusability, maintainability, and provide a higher level of productivity compared to writing everything from scratch or dealing with low-level details manually.

---

3. What essence do JavaScript frameworks elevate?

JavaScript frameworks elevate several key essences:

- Abstraction
- Productivity
- Structure and Organisation
- Maintainability
- Reusability
- Performance

---

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

Most JavaScript frameworks achieve abstraction by providing higher-level APIs, conventions, and patterns that encapsulate complex functionality and hide low-level implementation details.

---

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

The most important part of learning a JavaScript framework is gaining a solid understanding of its core concepts and principles. While specific frameworks may have unique features and syntax, grasping the fundamental concepts will provide a strong foundation for effectively working with any framework. Here are some key aspects to focus on when learning a JavaScript framework:

- Architecture and Design Patterns
- Component Composition
- Templating and Rendering
- State Management
- Routing and Navigation
- Tooling and Development
- Community and Documentation

Learning JavaScript frameworks is an ongoing process. As frameworks evolve, new features and patterns may emerge, and staying updated with the latest developments in the framework’s ecosystem is crucial. Continuously practising and building projects using the framework will help solidify your understanding and proficiency over time.