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?

Direct DOM mutations mean making small changes to the existing elements on a web page, while replacing HTML means completely changing the entire content of the page.

Benefits of direct DOM mutations over replacing HTML are:

- Faster: Direct DOM mutations are quicker because they only change the specific parts that need to be updated while replacing HTML requires rebuilding the entire page, which takes more time.
- Less memory usage: Direct DOM mutations work with the existing elements, so they use less memory compared to replacing HTML, which creates a whole new set of elements.
- Preserves user input: Direct DOM mutations keep the user's input, like what they've typed or selected, intact. Replacing HTML would remove any input the user has made.
- More control: Direct DOM mutations allow you to make specific changes to elements without affecting other parts of the page. This gives you more control over how the page behaves.
- Works better with other tools: Direct DOM mutations are usually more compatible with other tools or libraries used in web development. Replacing HTML can sometimes cause conflicts or issues with those tools.

- Better user experience: Direct DOM mutations can provide a smoother experience for users because they don't require refreshing the whole page. You can update specific parts without disrupting the rest of the page.

Overall, direct DOM mutations are faster, use less memory, preserve user input, give you more control, work well with other tools, and offer a better user experience compared to replacing HTML.

---

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

Frameworks allow developers to focus on application logic and user experience, enhancing productivity and code maintainability.

- DOM manipulation: Frameworks offer high-level APIs that abstract away the complexities of directly manipulating the DOM. This simplifies tasks such as adding, modifying, or removing elements from the webpage, allowing developers to focus on the desired changes without getting bogged down in low-level details.
- Event handling: Frameworks provide convenient APIs for managing event handling, making it easier to respond to user interactions like mouse clicks, keyboard presses, or touch events. They abstract away the intricacies of event registration, delegation, and propagation, allowing developers to create interactive and responsive web applications with less effort.

---

3. What essence do JavaScript frameworks elevate?

- Increased productivity: Frameworks can help developers to be more productive by providing pre-written code and libraries that can be reused. This can save

developers a lot of time and effort, which can be used to focus on more creative and challenging aspects of the development process.

- Improved code quality: Frameworks can help to improve the quality of code by providing a consistent coding style and conventions. This can make code more maintainable and easier to debug.
- Increased performance: Frameworks can help to increase the performance of web applications by providing a way to optimize the code and reduce the number of HTTP requests. This can make web applications feel more responsive and user-friendly.

---

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

- Using interfaces and abstract classes: Frameworks can hide the implementation details of certain classes and components, and only expose the essential functionality of the user
- Using modules: Frameworks can organise code into reusable units and abstract away the implementation details of certain feature.

---

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

- Grasp the framework's core concepts and principles.
- Understand the framework's architecture and design patterns.
- Familiarize yourself with the framework's API and documentation.
- Learn and follow the framework's best practices and conventions.
- Apply practical knowledge through projects and hands-on experience.
- Engage with the community for insights, help, and staying updated.
- Continuously stay updated with the latest versions and trends in the framework.