**W04 Prepare – Q/A AI**

**JavaScript Branching**

**Q: What does branching in JavaScript mean?**

A: Branching refers to controlling the program flow based on conditions. JavaScript offers various ways to achieve this, including if...else statements, the ternary operator (?), and the switch statement.

**Q: Can you show examples of JavaScript code using the ternary operator for decisions?**

A: Sure! Here are examples using the ternary operator to check age eligibility, assign default values, and apply discounts.

**Q: What are valid conditions in JavaScript?**

A: Valid conditions are expressions that evaluate to true or false. These can involve values, comparisons, logical operators (&&, ||, !), the typeof operator, function calls, and consideration of falsy/truthy values.

**Q: Can you provide an example of branching using the typeof operator?**

A: Absolutely! This code snippet uses typeof to check the data type (string, number, etc.) of a value and executes different code blocks accordingly.

**Q: What's the simplest way to write decision-making code in JavaScript?**

A: The if statement is the most straightforward approach. It allows you to specify a code block to run only if a certain condition is true. You can optionally add an else block for handling the opposite scenario.

**Events with JavaScript**

**Q: What are events in JavaScript?**

A: Events are signals that something has happened in the web page or browser, triggered by user interactions (clicks, key presses) or browser actions (page loading, window resizing). JavaScript code can respond to these events to create dynamic and interactive web pages.

**Q: How do I write a well-programmed event handler?**

A: Here are some key principles:

* Target specific elements to avoid unintended consequences.
* Use clear function names and comments for readability.
* Access event object properties to get details about the event.
* Implement error handling to prevent unexpected behavior.
* Keep handlers focused on a single responsibility for maintainability.
* Consider performance and avoid expensive operations within handlers.

**Q: What are different ways to respond to events from the browser with JavaScript?**

A: There are three main methods:

1. **HTML Event Attributes:** Simple but limited, assigning JavaScript code directly to event attributes in HTML elements (e.g., onclick).
2. **addEventListener:** More flexible and recommended, using addEventListener on DOM elements to specify the event type and function (handler) to run.
3. **Inline Event Handlers (Not Recommended):** Inline definition within the element using on prefix (e.g., onclick). Discouraged due to poor separation of concerns.

**Q: What are examples of poorly written event handlers?**

A: Here are some anti-patterns to avoid:

* Lack of specificity (same handler for multiple elements).
* Inline event handlers (clutter HTML code).
* Large event handlers with complex logic (hard to maintain).
* No error handling (unexpected behavior or crashes).
* Performance issues (expensive operations slow down the page).

**Q: What are examples of event types that are typically responded to with JavaScript?**

A: Common event types include:

* User Interaction Events (mouse clicks, key presses, form submissions).
* Document and Window Events (page loading, resizing, scrolling).
* Focus Events (gaining or losing focus on elements).
* Error Events (like image loading failures).
* Custom Events (user-defined events for specific functionality).