DWA_03.5 Knowledge Check_DWA3.2

---

1. User story(ies) in Gherkin syntax for the “+” button.

1. **"Utilize single quotes for strings."**
- **Benefit:** The practice of consistently using single quotes for strings, as recommended by the Airbnb Style Guide, fosters uniformity in the codebase. This uniformity enhances code readability and minimizes the risk of syntax errors stemming from mismatched quotes, facilitating a smoother development process.

2. **"Opt for template strings over concatenation for constructing strings."**
- **Benefit:** Template strings offer a superior way to construct strings by allowing developers to directly include variables within them. This approach not only streamlines code by reducing verbosity but also enhances clarity. When variables are seamlessly integrated using template strings, it becomes easier to discern the intended string output, thereby improving code maintainability and reducing the chances of errors compared to traditional concatenation methods.

3. **"Steer clear of unary increments and decrements (`++`, `--`)."**
- **Benefit:** Unary increments and decrements can introduce confusion, particularly when used within intricate expressions or alongside other operators. Avoiding them results in more predictable code that is easier to comprehend. Developers can opt for explicit increments and decrements, which contribute to code clarity and decrease the risk of unintended side effects or bugs.

---

2. User story(ies) in Gherkin syntax for the “-” button.

1. **"Why it may cause confusion:** This guideline has the potential to be perplexing because in numerous programming languages, employing a leading underscore for private variables is a prevalent convention. Developers who are well-versed in this practice from other languages

might find it counterintuitive to deviate from using leading underscores for private variables in JavaScript, as prescribed by the Airbnb Style Guide. Following disparate conventions across different contexts can introduce confusion and compromise the consistency of the codebase."

2. **"Avoid using the `Function` constructor to generate new functions."**
- **Why it can be perplexing:** For developers who lack an in-depth understanding of JavaScript's prototype-based inheritance and the intricacies of the `Function` constructor, this rule may prove baffling. Grasping why the use of the `Function` constructor is discouraged necessitates a firm grasp of JavaScript's foundational principles. Without this knowledge, developers might be puzzled as to why a seemingly valid approach to creating functions is discouraged.

3. **"Set a reasonable limit on line lengths (e.g., 100 characters)."**
- **Why it can be perplexing:** Defining what constitutes a "reasonable limit" for line lengths can be subjective, contingent on individual preferences, team standards, and specific project requisites. Different developers may hold different opinions regarding what qualifies as a legible line length. Consequently, determining a universally applicable "reasonable limit" can pose challenges. This rule might engender debates and confusion within a team regarding the appropriate line length, particularly when no explicit project-specific guidelines are in place.