**Best practice to name JS variables**

**1. Use Descriptive and Meaningful Names**

Choose names that clearly describe the purpose or content of the variable.

* ✅ Good: userAge, totalPrice, isLoggedIn
* ❌ Bad: x, data, temp

**Why?** This makes the code easier to read and understand for others (or yourself in the future).

**2. Follow CamelCase for Variables and Functions**

Use camelCase for variable and function names, starting with a lowercase letter.

* ✅ Good: userAge, getUserData
* ❌ Bad: Userage, get\_user\_data

**Why?** CamelCase is a widely accepted convention in JavaScript, improving consistency and readability.

**3. Avoid Reserved Words**

Do not use JavaScript reserved words or keywords as variable names.

* ❌ Bad: var let = 5;
* ✅ Good: letValue = 5;

**Why?** Reserved words have predefined meanings and will cause errors.

**4. Be Consistent with Naming Conventions**

Use consistent patterns to name similar types of variables:

* Boolean values: Start with is, has, or can to indicate true/false values.
  + ✅ Good: isLoggedIn, hasPermission
* Functions: Use verbs to indicate actions.
  + ✅ Good: calculateTotal, fetchData

**Why?** This makes the purpose of variables clear at a glance.

**5. Avoid Abbreviations**

Avoid unnecessary or cryptic abbreviations unless they are universally understood.

* ✅ Good: userAddress, priceInDollars
* ❌ Bad: usrAdd, pID

**Why?** Full words are easier to understand and reduce ambiguity.

**6. Use Singular and Plural Forms Appropriately**

Name variables in the singular or plural form depending on their content:

* ✅ Good: user (single object), users (array or collection)
* ❌ Bad: userList (ambiguous if used inconsistently)

**Why?** This keeps your code intuitive and reflective of the data structure.

**7. Avoid Prefixes Like my or tmp**

Prefixes like my or tmp should be avoided unless absolutely necessary.

* ✅ Good: currentUser, temporaryValue
* ❌ Bad: myVar, tmpData

**Why?** These prefixes add no meaningful context and can cause confusion.

**8. Use Constants for Unchanging Values**

Use UPPER\_SNAKE\_CASE for constants to distinguish them from variables.

* ✅ Good: MAX\_LIMIT, API\_URL
* ❌ Bad: maxLimit, apiUrl

**Why?** This clarifies that the value should not change during execution.

**9. Avoid Magic Numbers and Use Descriptive Names**

Replace "magic numbers" with named variables or constants.

* ❌ Bad: if (score > 70) { ... }
* ✅ Good: const PASSING\_SCORE = 70; if (score > PASSING\_SCORE) { ... }

**Why?** It makes your code easier to modify and understand.

**10. Context Matters**

Use names that are appropriate to their scope and context:

* For global variables, use more descriptive names to avoid collisions.
  + ✅ Good: appConfig
  + ❌ Bad: config
* For local variables, shorter but meaningful names are acceptable.
  + ✅ Good: i (loop index)

**Why?** This balances readability and practicality.