# JavaScript Style Guide Task

## 1. Naming Conventions

Airbnb emphasizes semantic naming to ensure code is self-documenting.

- **CamelCase:** Used for variables, functions, and instances (e.g., `let userProfile`).
- **PascalCase:** Strictly reserved for constructors, classes, and React components (e.g., `class UserAccount`).
- **Screaming Snake Case:** Used for constants that are globally defined and exported (e.g., `const API_KEY`).
- **Leading Underscores:** Avoided for "private" properties; the guide suggests relying on scope or modern private class fields instead.
- **Acronyms:** Always capitalize acronyms fully or not at all (e.g., `getHTTPResponse`, not `getHttpResponse`).

## 2. Formatting & Syntax

Consistency in formatting prevents "visual noise" and allows developers to focus on logic rather than style.

- **Indentation:** Soft tabs (2 spaces) are the mandatory standard.
- **Semicolons:** Always required. Relying on Automatic Semicolon Insertion (ASI) is discouraged as it can lead to subtle bugs.
- **Quotes:** Use single quotes (`'`) for strings. Use backticks (`` ` ``) only when using template literals for interpolation or multi-line strings.
- **Trailing Commas:** Required in multi-line objects and arrays (ES6+ feature). This makes git diffs cleaner since adding an item doesn’t modify the previous line.
- **Whitespace:** Use a single space before leading braces and around operators. Avoid padding blocks with blank lines.

## 3. Structural Standards

The guide promotes modern ES6 features over legacy patterns to reduce side effects and mutation.

- **References:** Use `const` for all references; use `let` only if you must reassign a variable. Never use `var`.
- **Objects & Arrays:** Use literal syntax (`{}` and `[]`) instead of constructors. Use the spread operator (`...`) to copy arrays or objects to prevent unintended mutation.
- **Functions:** Favor arrow functions (`=>`) for anonymous callbacks and when you need to preserve the lexical `this`.
- **Destructuring:** Mandatory for accessing properties from objects or arrays to keep code concise (e.g., `const { name } = user;`).

## 4. Code Consistency & Best Practices

Airbnb’s guide is designed to make code look like it was written by a single person, regardless of team size.

- **Modules:** Always use `import`/`export` syntax over `require`. Do not use wildcard imports (`import *`); be explicit.
- **Equality:** Always use strict equality (`===` and `!==`) to avoid type coercion issues.
- **Conditionals:** Avoid nested ternaries. If a control statement becomes too long, wrap it in parentheses and break it into multiple lines for readability.
- **Comments:** Use `/** ... */` for multi-line documentation and `//` for single-line notes. Annotate technical debt with `// FIXME:` or `// TODO:`.