SkillSwap

Coding Standards

Contents

1 Introduction

This document outlines the coding standards, guidelines, and best practices to be followed by all developers working on the SkillSwap project. The purpose of these standards is to improve code readability, maintainability, and consistency across the entire codebase. Adherence to these guidelines is mandatory and will be enforced through code reviews.

2 Naming Conventions

1. Avoid using names that are too general or too wordy.

   Example

   Bad Practice: `users_who_are_active, options`

   Good Practice: `active_users, menu_options`

2. Do not use short forms of naming.

   Example

   Bad Practice: `add, mul`

   Good Practice: `addition, multiplication`

3. It is better to avoid the use of digits in variable names.

   Example

   Bad Practice: `points4user`

   Good Practice: `reward_points`

4. Use plural nouns for variables that represent a collection.

   Example

   Bad Practice: `String user[] = new String[5];`

   Good Practice: `String users[] = new String[5];`

5. Object property names should not repeat the object's context.

   Example

   Bad Practice: `const Car = { carMake: "Honda" };`

   Good Practice: `const Car = { make: "Honda" };`

6. Package names should be all lower case.

   Example

   ```
   com.skillswap.utilities
   ```

7. Capitalize all letters of an abbreviation.

   Example

   ```
   Bad Practice: http_server

   Good Practice: HTTP_server
   ```

8. Constants must be fully capitalized and follow `UPPER_SNAKE_CASE`.

   Example

   ```
   Bad Practice: Pi, MAXUSERS

   Good Practice: PI, MAX_USERS
   ```

9. Functions and local variables should be in `snake_case`.

   Example

   ```
   Bad Practice: useraccount(), studentform

   Good Practice: user_account(), student_form
   ```

10. Class names should be in `PascalCase`.

    Example

    ```
    Bad Practice: accountdetails

    Good Practice: AccountDetails
    ```

11. Global variables should be in `camelCase`.

    Example

    ```
    Bad Practice: globalvariable

    Good Practice: globalVariable
    ```

12. Boolean variables and methods should be prefixed with a verb.

Example

```
Bad Practice: visible, biscuits

Good Practice: is_visible, has_biscuits
```

Code Element	Convention	Example
Constants	UPPER_SNAKE_CASE	MAX_USERS, API_KEY
Functions & Local Variables	snake_case	user_account(), student_form
Class Names	PascalCase	AccountDetails, UserProfile
Global Variables	camelCase	globalUser, activeSession
Package Names	lowercase	com.skillswap.utilities
Abbreviations	All letters capitalized	HTTP_server

Table 1: Summary of Casing Conventions

3 Code Structure and Formatting

1. Use 4 spaces per indentation level. Do not use tabs.
2. Opening braces go on the same line as the statement.

Example

```
Bad Practice:
if (true)
{
    console.log('losing');
}

Good Practice:
if (true){
    console.log('losing');
}
```

3. Always place a space before and after binary operators (‘+‘, ‘=‘, ‘*‘, etc.), and after commas in argument lists. Never place a space before a comma, semicolon, or colon.

Example

```
Bad Practice:
const total=x*5+y;

Good Practice:
const total = x * 5 + y;
    if (a === b) {
        /* ... */
```

```
    }
```

4. Use single blank lines to separate logical sections of code.

4 Function, Class Design and Data Hiding

1. Only declare a variable as global if it is used in multiple functions.
2. Functions should not exceed 20-25 lines. Break down large functions.
3. Avoid duplication by creating reusable functions.
4. Use ‘if‘ and/or ‘else if‘ statements instead of repeated ‘else‘.
5. Class members are public by default.
6. Prefix with a single underscore (‘_‘) to indicate a protected member.

Example

```
self._internal_variable = 10
```

7. Prefix with a double underscore (‘__‘) to indicate a private member.

Example

```
self.__private_data = 20
```

5 Comments, Git, and Error Handling

1. Comments should explain the ”why,” not the ”what.” Add author initials for tracking.
2. Keep comments up-to-date with code changes.
3. Commit messages must be meaningful and descriptive.

Example

```
Bad Practice: git commit . -m "fifth update"

Good Practice: git commit . -m "user_profile() error handling
    updated"
```

4. All functions must handle exceptions properly.

Example

```
Bad Practice:
    try {
        num.toPrecision(500);
```

```
    }
    catch(err) {
        document.getElementById("demo").innerHTML = 1;
    }

Good Practice:
    try {
        num.toPrecision (500);
    }
    catch(err) {
        document.getElementById("demo").innerHTML = err.name;
    }
```

6 Language-Specific Standards

6.1 SQL Conventions

1. All SQL keywords must be capitalized.

   Example

   ‘SELECT‘, ‘FROM‘, ‘WHERE‘, ’OR’, etc.

2. Table and column names must use `snake_case`.
3. Foreign key columns should be prefixed with `fk_` (e.g., `fk_product_key`).
4. Always use parameterized queries to prevent SQL injection.

6.2 Web and API Standards

1. RESTful API endpoints must use ‘kebab-case‘.

   Example

   Bad Practice: /api/productorders, /api/userProfile

   Good Practice: /api/product-orders, /api/user-profile

2. CSS classes should also use ‘kebab-case‘.
3. Attribute values in HTML must always be enclosed in double quotes (“”‘).
4. Every CSS property must be terminated with a semicolon.

   Example

   Bad Practice: div color: red

   Good Practice: div color: red;

References

- GitHub - felixge/node-style-guide
- Code Guide by @mdo
- GitHub - bendc/frontend-guidelines
- SQL Style Guide
- Coding Standards and Guidelines - GeeksforGeeks