Coding Standards

Owned by Fiona Zhang •••
Last updated: Oct 27, 2023• See how many people viewed this page

This page documents the rules and standards of our code.

Folder Structure:

```
vue-app/
|
├── public/
|   ├── logo/
|   └── index.html
|
├── src/
|   ├── assets/
|   |   ├── animals/
|   |   |   └── animal icons/
|   |   ├── audio/
|   |   |   ├── Anita/
|   |   |   ├── Feliks/
|   |   |   ├── Fiona/
|   |   |   ├── Vincent/
|   |   |   └── Leo/
|   |   ├── font files/
|   |   ├── css files/
|   |   ├── gif tutorials/
|   |   └── all pictures/
|   |
|   ├── pages/
|   |   ├── Account.vue
|   |   ├── Challenge1.vue
|   |   ├── Challenge2.vue
|   |   ├── Finish.vue
|   |   ├── Lobby.vue
|   |   ├── Playground.vue
|   |   ├── Settings.vue
|   |   ├── Test1.vue
|   |   ├── Test2.vue
|   |   ├── Test3.vue
|   ├── App.vue
|   |
|   └── utils/ (JavaScript utility files)
```

Coding conventions

- **File Naming:**
  - Always opt for descriptive file names which mirror the function or component.
- **Component Naming:**
  - Use Camel Case for naming components.
    - *Example:* `MyComponentName`
- **Code Formatting:**
  - Ensure 2 spaces are used for indentation.

- **Components:**
  - Favour stateless functional components for their simplicity and better performance over class components.
- **Separation of Concerns:**
  - Strive to maintain a distinct boundary between logic and presentation.
    - *For instance,* `Checker.js` should only handle submission validation, whereas `speech.js` should deal with voice messages.
- **Global Variables:**
  - Centralise global variables, such as student data, scores, and settings in `store.js`.
- **Styling:**
  - Avoid inline styles. Instead, utilise `StyleSheet` or `styled-components`.
  - Refrain from using units like `rem` or `px`. Adopt `vw`, `vh`, and `%` to ensure compatibility across all device screens.
  - For recurring styles, define them in `global.css` to reduce redundancy.
- **Platform Availability:**
  - As a web application, our platform can be accessed on any device equipped with a browser.
- **Error Handling:**
  - Incorporate comprehensive error handling, paying special attention to asynchronous tasks and API interactions.
- **Callbacks:**
  - Eschew deeply nested callbacks; use async/await or `Promise.all()` to manage parallel tasks more efficiently.
- **Figma Naming:**
  - Space out words and numerically group similar items.
    - *Example:* `button_exit`, `button_restart`.
- **Reusability:**
  - Extract commonly used utility and game logic functions into distinct JavaScript files to prevent code duplication.