Style Guide

1. File Structure

1.1. Hierarchy

1.1.1. The file structure is divided into Backend and Frontend folders, respectively.

1.1.2. Files that perform backend or frontend functions should exist in their respective paths and be referenced there accordingly.

1.1.3. Duplicate files should not exist – it is preferable to merge them and restructure dependencies to accommodate this.

1.2. Source Files

1.2.1. One script should unify frontend and backend functionality to form an entry point into the code.

2. Formatting

2.1. Escaped Characters

2.1.1. Special escape sequences (‘\n’) are preferred to numeric ones (‘\0A’).

2.2. Imports

2.2.1. All imports should be used or pruned.

2.3. Special Characters

2.3.1. For the remaining non-ASCII characters, either the actual Unicode character (e.g. ∞) or the equivalent hex or Unicode escape (e.g. \u221e) is used, depending only on which makes the code **easier to read and understand**. Use comments to clarify when using a raw Unicode escape.

2.4. Whitespace

2.4.1. Spaces are preferred to tabs.

2.4.2. Separate logical portions of code should use whitespace liberally for readability.

2.4.3. Multiple-definition lines (e.g. a=1,b=2,c=3…;) should be separated by newlines for readability.

3. Language Features

3.1. CSS

3.1.1. Styles should be applied via classes, in a separate file. (**Not inline)**

3.1.2. Flexbox should be used when possible for a responsive web page.

3.1.3. Media queries and relative size units should be used for the same reason.

3.2. HTML

3.2.1. When possible, use descriptive elements (i.e. not a div for everything).

3.2.2. Use inclusive practices so that pages are accessible to all users (e.g. E-Readers) via alt-texts, Aria, etc.

3.3. Javascript

3.3.1. *Const* should always be used over *let* or *var* when possible.

3.3.2. '===' or an appropriate function (e.g. assertEquals) should always be used over '==' when possible.

3.3.3. ';' is encouraged for readability, although not required by the language.

3.3.4. Arrow functions ('=>') are preferred for readability.

3.3.5. Avoid the use of globals, or ensure **strict** use of *Const* so that they cannot be reassigned. It is preferable to define local copies of an unchanging variable than to have an accidental global variable change propagate to all instances.

4. Naming

4.1. File Names

4.1.1. All file names, except React components by convention, are camelCase.

4.1.2. Filenames may contain '-' or '_' but no other punctuation.

4.2. Variable Names

4.2.1. All variable names are camelCase.

4.2.2. Variable names may contain '-' or '_' but no other punctuation.

4.2.3. Variable names must be descriptive. Verboseness is preferred when it improves readability.

4.2.4. If there is a conflict, prefer industry standard naming conventions when using an outside library.

5. Policies

5.1. Comments

5.1.1. Comments should be used liberally and accompany all compositions of logic that are not ***instantly*** comprehensible.

5.1.2. Comments should take up their line *before* the code they discuss, rather than alongside it or after it.

5.1.3. Prefer single–line comments to multi-line, if the comment only takes a single line (i.e. use '//' before '/*' if you can).

5.1.4. Please, no silly or unnecessary comments.

5.2. Nesting Depth

5.2.1. Do not nest more than 3 levels deep. If your code nests this far, you should refactor it.