**General Coding Standards:**

- Use Prettier Code Formatter with these specifications for javascript
```
{
  "printWidth": 120,
  "tabWidth": 2,
  "useTabs": false,
  "semi": true,
  "singleQuote": true,
  "trailingComma": "none",
  "bracketSpacing": true,
  "arrowParens": "avoid",
  "requirePragma": false,
  "insertPragma": false,
  "proseWrap": "preserve"
}
```
- Use prettier to format html and css files as well with their default settings
- Use **camelCase** for all variable names
- Always check if any code you've written breaks any test. npm test on local first
- Readability
- Let's aim for code coverage > 75%

**Testing:**

- Use descriptions that represent the test in a direct and concise way
- Put the test methods for a specific method in the code into a describe block in Jest

**Javascript:**

- Ensure whether var is being used appropriately (maybe const is needed)
- Change to ES6 variables (var, let, const)
- Make the code more Robust like checking the var types: use typeof/instance of, use isNaN to check for NaN, use isFinite to check for Infinity, use BigInt for possible overflow
- Handle runtime errors: use try/catch to catch errors
- Make code faster: reduce amount of code, use profiling tools to optimize hotspots, use object pool (maybe)
- Use ES6 functions (arrow functions)
- Function name should represent its functionality
- Variable name should represent functionality
- Check if there is unnecessary code and remove it, but test it thoroughly(for example, warmup exercise had unnecessary code to support very old browsers)
- If functions can be made more modular, go for it!
- If any online resources are being used (example: if it is loading jquery file online each time) then make it local (download the file, change the paths, etc)
- Document as and when you go through code.
- Use JS Docs

**CSS:**

- Use semantic HTML substituting elements like <div> for <section> and <span> for <em> for example.
    - Use custom elements (ex: <form-field>) to create your own semantic elements.
    - Use ARIA attributes to add semantic meaning to non-semantic elements.
- Keep all colors is hsl format
- Constants should be defined beforehand

**Usability:**

- Provide feedback and comments to the code
- Always show status and progress for long task
- Catch failures with analysis of the problems
- Do a usability test with the users
- Let the users know what’s happening