**NAMING CONVENTIONS**

* Component’s names should be written using pascal case:
  + Header.js
  + HeroBanner.js
  + CookieBanner.js
  + BlogListing.js
* Non-components should be written using camel case:
  + myUtilityFile.js
  + cookieHelper.js
  + fetchApi.js
* Unit test files should use the same name as its corresponding file:
  + CookieBanner.js
  + CookieBanner.test.js
  + fetchData.js
  + fetchData.test.js
* Attribute name should be camel case:
  + className
  + onClick
* Inline styles should be camel case:
  + <div style={{font-size:'1rem'}}></div>
* Variable names should be camel case. Variable names can contain number and special characters:
  + const variable = 'test';
  + let variableBoolean = true;
* CSS files should be named the same as the component:
  + CookieBanner.css
  + Header.css
* If a component requires multiple files (css, test) locate all files within component a folder
* Use .jsx or .tsx extension for React components
* **BUG AVOIDANCE**
  + Use optional chaining if things can be null
  + Use the guard pattern/prop types/typescript to ensure your passed in parameters are valid
  + Create PURE functions and avoid side-effects
  + Avoid mutating state when working with arrays
  + Remove all console.log()
  + Treat props as read-only. Do not try to modify them
* **ARCHITECTURE & CLEAN CODE**
  + No DRY violations. Create utility files to avoid duplicate code
  + Follow the component/presentation pattern where appropriate. Components should follow the single responsibility principle
  + Use Higher Order Components where appropriate
  + Split code into respective files, JavaScript, test, and CSS
  + Create a index.js within each folder for exporting. This will reduce repeating names on the imports
    - import {Nav} from './Nav.js';
    - import {CookieBanner} from './CookieBanner.js';
    - export {Nav, CookieBanner}
  + Only include one React component per file
  + Favour functionless components
  + Do not use mixins
  + No unneeded comments
  + Methods that are longer than the screen should be refactored into smaller units
  + Commented out code should be deleted, not committed
* **ES6**
  + Only use let or const
  + Favour arrow functions
  + Can the optional chain operator be used instead of an explicit null check
  + Can nullish coalescing by used instead of an explicit null comparison
* **TESTING**
  + Write tests
  + Define a quality gate using coveralls
  + Don't test more than one thing in a test
  + No logic should exist within your test code
  + Test classes only test one class
  + Code that needs to talk to a network, or, database is mocked
* **CSS**
  + Avoid Inline CSS
  + A naming convention is defined and followed (BEM, SUIT, etc..)