Variables

Variable declarations, assignments, and usage

- Always use **const** or **let** to declare variables.
- Variable names must be in camelCase.

Examples:

```
const fullName = "Ghanshyam Motwani";

const phone = 1234567890;

let a = 10;

a = a + 5;
```

Please refer to the below link for more details:

https://github.com/airbnb/javascript#variables

Comments

Comments syntax and usage

- ➢ Use `/** */` for multiline comments.
- ➢ Use `//` for single line comments.
- ➢ Make clean comments for complex logic

Comment structure for file, function, inline follow jsdoc.

Examples:

```
// is current tab
const active = true;


/**
* make() returns a new element
* based on the passed-in tag name
*/
function make(tag) {
 // ...
 return element;
}
```

Please refer to the below link for more details:

https://github.com/airbnb/javascript#comments

Functions

Function syntax and usage

- ➢ Use named function expressions instead of function declarations
- ➢ Function name must be in camelCase.
- ➢ Use default parameter syntax rather than mutating function arguments.

Examples:

```
function handleThings(param1, param2, ...paramN) {
      // ...
}
const handleThings = () => (param1, param2, paramN) => {
      // ...
}
function defaultParams(param1 = {}, param2 = '', param3 = 1) {
      // ...
}
```

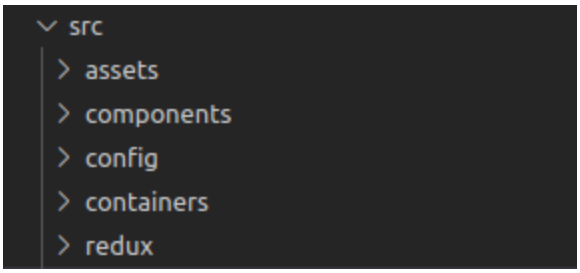
Please refer to the below link for more details:

https://github.com/airbnb/javascript#functions

File Structure

Best practices for file structure

➢ A folder and subfolder name always start with small letters.



Now, let's go over the folders one by one and understand the motivation behind them and the type of files you would store in them:

assets: This folder contains all the media assets, such as images, videos, JSON files, etc.

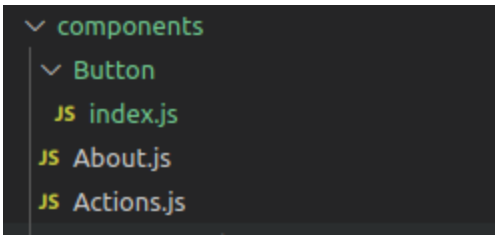
components: This folder contains all the Presentational/Stateless Components as discussed above.

config: This folder contains all the files like the site.config.js, language.config.js, etc...,

containers: The name is pretty self-explanatory. It contains all the Stateful Components as discussed above.

redux: This folder contains all the redux related files like redux/app/actions.js, redux/app/reducer.js, redux/store.js

- Component or Screen's file name must be in PascalCase, If you make a component with multiple files then the folder name of the component will be in PascalCase and the main file of that will be **index.js**.



- src folder may also contain some other folders like src/hooks, src/i18n, src/hoc, etc.. when needed.

Coding Standards for React

Best practice for tremendous React JS

GROOVY

Assurance of Excellence!

General

General points to remember for development

➢ Always use **try{...}catch() {...}** to handle unpredictable errors like errors from API calls, some parts of JS code that may return some error.

Example:

```
try {
 // Your stuff...
} catch(err) {
 // Handle errors here...
}
```

➢ Always make **readme.md** file in the root folder of the project. This file will contain Title, Short description, Installation setup, Other required steps to install the project. Readme file must be clear, formatted, and easy to understand so anyone can install the project seamlessly.

Readme Format: Coming soon.,.

➢ Must use **airbnb ESLINT + Prettier** to make sure that your code follows good standards.

➢ For asynchronous processes, always use the **async, await**, and **promise** features of JavaScript. Refer to this link for detailed knowledge of the asynchronous process.

➢ Below are some VS Code plugins that should be used by React js / React Native Developers.

- ES7 React/Redux/GraphQL/React-Native snippets
- ESLint
- Beautify
- Prettier Now
- GitLens — Git supercharged
- Git Graph

➢ Always remember the DRY (Don't Repeat Yourself) principle. It usually means refactoring code by taking something done several times and turning it into a loop or a function. DRY code is easy to change because you only have to make any change in one place.

Example:

```
let chips = ['corn', 'pita', 'potato', 'tortilla'];


// non DRY code

console.log(chips[0]);

console.log(chips[1]);

console.log(chips[2]);


// DRY code

for (let i = 0; i < chips.length; i++) {

 console.log(chips[i]);

}
```