**Регламент по написанию кода на React**

**в рамках тестового задания**

1. Структура папок вашего проекта должна иметь примерно следующий вид:

.

└── src/

├── components/

│ ├── App/

│ │ ├── index.js

│ │ ├── App.jsx

│ │ ├── Filler.jsx

│ │ └── styles.css

│ ├── Container/

│ │ ├── index.js

│ │ ├── Container.jsx

│ │ └── styles.css

│ └── Button/

│ ├── index.js

│ ├── Button.jsx

│ └── styles.css

├── hooks/

│ └── useCustomHook.js

├── store/

| ├── index.js

│ └── users/

│ ├── actions.js

│ ├── reducer.js

│ ├── selectors.js

│ └── types.js

├── utils/

│ └── api.js

└── …

Папки в директории src именуем в [camelCase](https://techrocks.ru/2018/08/09/most-common-programming-case-types/).

1. Каждая папка с компонентом должна содержать в себе index.js. Используя [reexport](https://babeljs.io/docs/en/babel-plugin-proposal-export-default-from) (в [Create React App](https://create-react-app.dev/) он уже вшит по умолчанию), экспортируете файл компонента наружу. Папки с компонентами, именуются в [PascalCase](https://techrocks.ru/2018/08/09/most-common-programming-case-types/), код компонентов пишем исключительно в файлах с расширением .jsx. Если вам нужно разделить логику и создать вспомогательные компоненты, используемые в вашем приложении единоразово, лучше всего их оставить в папке с главным компонентом. Предпочтение отдаем [функциональным компонентам](https://ru.reactjs.org/docs/components-and-props.html#function-and-class-components) и [хукам](https://ru.reactjs.org/docs/hooks-intro.html). Функциональные компоненты лучше объявлять с помощью [function expression](https://learn.javascript.ru/function-expressions), используя [arrow синтаксис функций](https://learn.javascript.ru/arrow-functions-basics) ES6 (короткую запись использовать по необходимости).

1. Если в вашем компоненте используется много хуков, связанных общей бизнес-логикой, то лучше их вынести в [пользовательский хук](https://ru.reactjs.org/docs/hooks-custom.html), с возвратом объекта, содержащего методы и данные, далее вызываем его в компоненте, получая из него необходимые методы и данные через ES6 [деструктуризацию](https://learn.javascript.ru/destructuring).
2. Для работы с API делаем отдельный сервис (файл с набором необходимых функции), выносим его в папку utils, со списком методов (использующих для обращения к серверу [axios](https://github.com/axios/axios) или [fetch](https://learn.javascript.ru/fetch)). Если вы используете axios, необходимо создать [axios instance](https://github.com/axios/axios#creating-an-instance), для последующего его переиспользования в методах вашего сервиса.
3. Для написания стилей, в основном, используем [sass](https://sass-scss.ru/), [stylus](https://stylus-lang.com/) (как модули) или СSS-in-JS библиотеки (например, в [material-ui](https://material-ui.com/ru/) используется [JSS](https://cssinjs.org/)). Если вы используете CSS-модули ([stylus](https://stackoverflow.com/questions/52111496/how-to-configure-stylus-support-in-a-react-js-application), [sass](https://blog.bitsrc.io/how-to-use-sass-and-css-modules-with-create-react-app-83fa8b805e5e)), то используем библиотеку [classnames](https://www.npmjs.com/package/classnames) или [bem-classnames](https://www.npmjs.com/package/bem-classnames).
4. Стили организуем по методологии [BEM](https://ru.bem.info/) (в каждом компоненте вашего приложения родительский html-элемент - это блок).
5. Использование [ESLint](https://eslint.org/) обязательно, в JS и React.js нет общего стандарта написания кода или общепринятой (достаточно строгой) конфигурации линтера, но в нашей компании мы используем [eslint-config-airbnb](https://www.npmjs.com/package/eslint-config-airbnb). Также в конфигурацию линтера добавляем правила для [хуков](https://www.npmjs.com/package/eslint-config-airbnb#eslint-config-airbnbhooks).

[Инструкция для WebStorm/PHPStorm.](https://docs.google.com/document/d/1ZtG2pIxhobhI5m4GEo00kw9Y9_6dDggGXGvJukSbzHs/edit?usp=sharing)

Для использования ESLint в VSCode достаточно установить [плагин](https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint).

1. Все пропсы должны быть типизированы (один из вариантов [prop-types](https://ru.reactjs.org/docs/typechecking-with-proptypes.html)). Прописываем для необязательных пропсов дефолтные значения, с помощью [defaultProps](https://ru.reactjs.org/docs/typechecking-with-proptypes.html#default-prop-values).
2. [Именование](https://gist.github.com/delbetu/a5285175b63f7b73b5245e7231d1974a) всех булевых переменных и свойств классов должно начинаться с is. Все имена переменных и свойств классов должны быть существительными. Все имена функций и методов классов должны начинаться с глагола (например, get, fetch и т.д.). Именование функций-обработчиков событий в React рекомендуется начинать с on или handle.
3. Рекомендуется писать [чистые функции](https://github.com/MostlyAdequate/mostly-adequate-guide-ru/blob/master/ch03-ru.md) и не мутировать объекты и массивы.