# Управление состояниями

### Цель работы

Получить навыки работы с декларативным походом в программировании, а так же с умением убирать избыточные состояния компонента.

### Задания для выполнения

1. Ознакомьтесь с принципами декларативного подхода:

<https://react.dev/learn/reacting-to-input-with-state>

1. Создайте квиз, проверяющий знания React, используя:

<https://www.schoolsw3.com/quiztest/quiztest_REACT.php>

https://ru.w3docs.com/quiz-start/osnovy-react

https://itproger.com/test/react-js

и убрав избыточные состояния (https://react.dev/learn/choosing-the-state-structure)

1. Загрузить созданную страницу на GitHub в репозиторий tera, используя формат в названии Фамилия (латинскими буквами)\_9.

### Методические указания

Когда вы разрабатываете взаимодействия с пользовательским интерфейсом, вы, вероятно, думаете о том, как *изменяется* пользовательский интерфейс в ответ на действия пользователя. Рассмотрите форму, которая позволяет пользователю отправлять ответ:

* Когда вы вводите что-либо в форму, кнопка “Отправить” **становится включенной.**
* Когда вы нажимаете “Отправить”, и форма, и кнопка **становятся недоступными,** и появляется счетчик **.**
* Если сетевой запрос выполнен успешно, форма **скрывается,** и**появляется сообщение “Спасибо”.**
* Если сетевой запрос завершается неудачно, **появляется** сообщение об ошибке, и форма **снова становится включенной**.

В **императивном программировании** вышесказанное напрямую соответствует тому, как вы реализуете взаимодействие. Вы должны написать точные инструкции для управления пользовательским интерфейсом в зависимости от того, что только что произошло. Вот еще один способ подумать об этом: представьте, что вы едете рядом с кем-то в машине и указываете ему поворот за поворотом, куда ехать.

Они не знают, куда вы хотите попасть, они просто следуют вашим командам. (И если вы неправильно укажете направление, вы окажетесь не в том месте!) Это называется *императивом*, потому что вы должны “командовать” каждым элементом, от счетчика до кнопки, сообщая компьютеру, *как* обновить пользовательский интерфейс.

Обязательное управление пользовательским интерфейсом работает достаточно хорошо для отдельных примеров, но в более сложных системах управлять им становится экспоненциально сложнее. Представьте себе обновление страницы, полной различных форм, подобных этой. Добавление нового элемента пользовательского интерфейса или нового взаимодействия потребует тщательной проверки всего существующего кода, чтобы убедиться, что вы не внесли ошибку (например, забыли что-то показать или скрыть).

React был создан для решения этой проблемы.

В React вы не управляете пользовательским интерфейсом напрямую — это означает, что вы не включаете, отключаете, не показываете и не скрываете компоненты напрямую. Вместо этого вы **объявляете, что хотите показать,** а React определяет, как обновить пользовательский интерфейс.

**Полезные ссылки:**

https://react.dev/learn/reacting-to-input-with-state

https://react.dev/learn/choosing-the-state-structure