В рамках лабораторной работы я провел сравнение императивного и декларативного подхода. Для императивного подхода был использован jQuery. В этом случае код старается изменить непосредственно пользовательский интерфейс. Я вручную выбрал элемент DOM, настроил для него переключатель и задал команду браузеру на переключение класса CSS. Следовательно, я, как разработчик пошагово реализовал переход между различными состояниями пользовательского интерфейса.

Для реализации декларативного подхода я использовал соответственно React. В нем вместо прямого управления DOM я описал, как должен выглядеть пользовательский интерфейс в зависимости от текущего состояния. Я задал окончательный вид, а React сам реализовал всю остальную часть и обновляет UI. Также нет необходимости в пошаговой реализации(как и отмечено в книге).

Возможность повторного рендеринга React позволяет избежать лишней работы и менять DOM быстро и стабильно. Под повторным рендерингом я подразумеваю возможность обновления переменной IsHighlighted с помощью кнопки, что и запускает сам повторный рендеринг(в книге – Re-rendering). С помощью этого функционала React выполняет то, что описано в книге, как “diffing and patching” – то есть сравнение нового Виртуального DOM со старым и применяет все необходимые изменения к реальному.

Также важно отметить, что в отличии от императивного подхода, декларативный подход гораздо лучше и проще масштабируется для более сложных вариантов интерфейса. В Больших приложениях использование императивного подхода вынуждает отслеживать каждый DOM вручную, что создает лишнюю нагрузку и добавляет риск нарушения работы некоторых элементов интерфейса. Так как “Performance matters”, данный подход является категорически нежелательным. Поэтому способность React минимизировать взаимодействие с DOM делает его лучшим выбором для современных приложений, содержащих в себе большое количество данных. Так как он сильно уменьшает нагрузку и соответственно снижает риски.