В императивной реализации с использованием jQuery разработчик напрямую управляет DOM пошагово. Для кнопки назначается обработчик события клика, и при его срабатывании код явно добавляет или удаляет CSS-класс у абзаца. Такой подход требует от разработчика вручную описывать, *как именно* должен изменяться интерфейс в ответ на действия пользователя.

В отличие от этого, декларативная реализация в React сосредоточена на описании того, *как должен выглядеть* пользовательский интерфейс в зависимости от текущего состояния. Вместо прямого изменения DOM компонент хранит данные интерфейса в состоянии с помощью хука useState. При изменении состояния React автоматически выполняет повторный рендер компонента и обновляет DOM так, чтобы он соответствовал новому описанию интерфейса.

Подход React соответствует концепциям, описанным в главе 1, в частности разделам **«Declarative UI structures»** и **«Data changes over time»**. Пользовательский интерфейс рассматривается как функция от состояния: при изменении данных интерфейс обновляется предсказуемо и без ручных DOM-операций.

Декларативная модель значительно лучше масштабируется в сложных приложениях. По мере роста количества элементов и взаимодействий императивный код становится труднее поддерживать и анализировать. React упрощает этот процесс за счёт централизации логики интерфейса в компонентах и состоянии, делая приложения более читаемыми, поддерживаемыми и удобными для отладки.