**Подсказка**

* Установка зависимостей. Установите пакет «redux» в проект, с которым будете работать.
* Reducer. Создайте файл reducer.js. В нем определите объект начального состояния хранилища (initialState). Он будет основан на уже имеющихся в приложении состояниях, созданных с помощью хука useState() (field, currentPlayer и т.д.). Также создайте саму функцию-редьюсер. Для этого нужно понять, какие действия можно будет совершать с хранилищем. Проанализируйте приложение (как в нем меняется состояние, как с ним взаимодействует пользователь) и определите список действий (actions). Например, 'SET\_CURRENT\_PLAYER', 'SET\_FIELD','RESTART\_GAME' и т.д.

const initialState =

{

*// ...*

};

export const reducer = (state = initialState, { type, payload }) => { switch (type)

{

case 'SET\_CURRENT\_PLAYER':

return {

*// ...*

};

case 'SET\_FIELD':

*// ...*

case 'RESTART\_GAME':

return initialState;

default:

return state;

}

};

* Store. Создайте файл store.js. Используя createStore() из пакета «redux» создайте новое хранилище. При вызове функции не забудьте передать в нее редьюсер. Экспортируйте полученный объект хранилища.
* Связка React с Redux.

1. Уберите проброс пропсов из корневого компонента Game (убираем передачу состояний, которые теперь находятся в store — status, currentPlayer и т. д.). Также уберите проброс этих пропсов во всех компонентах-посредниках.
2. В конечных компонентах, где использовалось это состояние (компоненты Information и Field), необходимо заменить получение из пропсов на получение из Store (вспоминаем из урока, как получить текущее состояние хранилища). **Подсказка:** Метод store.getState().
3. В конечных компонентах убираем вызов всех старых функций изменения состояния useState(). Вместо этого во всех местах используем способ, которым обновляется хранилище (вызов dispatch() с необходимым действием).
4. Но теперь у нас возникает проблема. Вызовы dispatch() не приводят к обновлению данных на странице. Чтобы решить такую проблему, нужно вспомнить, что компонент ререндерится при изменении своего состояния. Зная этот факт, мы можем использовать useState(), чтобы в нужный момент запустить ререндер. Также вам может понадобиться [метод store.subscribe()](https://redux.js.org/api/store#subscribelistener), который принимает коллбек и вызывает его каждый раз после того, как состояние хранилища было изменено.