**Тема:** Поведенческие паттерны: «Наблюдатель», «Состояние», «Хранитель».

**«Наблюдатель»**

Описание:

Создает механизм у класса, который позволяет получать экземпляру объекта этого класса оповещения от других объектов об изменении их состояния, тем самым наблюдая за ними.

Пример реализации:

По клику на любую кнопку приложения (старт одного из типов сортировки) мы отправляем сообщение всем кнопкам о том, что сейчас идёт сортировка. Каждой кнопке мы создаём обработчик данного события, который делает кнопку неактивной. Также компонент «сцена» отправляет событие, когда она закончила сортироваться. На это событие также подписываются все кнопки; по этому событию они снова становятся активными.

Исходный код:

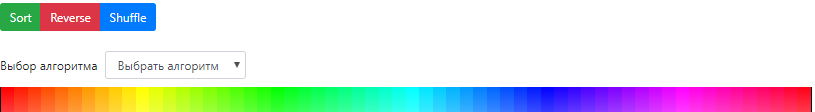
//Первая часть "наблюдателя"  
**let** buttons = document.body.querySelectorAll('.btn');  
buttons.forEach((curr) => {  
 curr.addEventListener('click', () => {  
 document.body.dispatchEvent(**new** Event('startAlgorithm', { bubbles: **true**, cancelable: **true** }));  
 });  
});  
//Вторая часть "наблюдателя". Все кнопки изменятся, когда начнётся сортировка  
document.body.addEventListener('startAlgorithm', () => {  
 buttons.forEach((curr) => {  
 curr.classList.add('disabled');  
 });  
});

Ответ на событие от компонента «сцена»

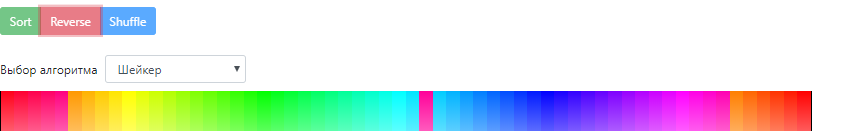
//Продолжение второй части "наблюдателя". Все кнопки вернуться в исходное состояние, когда закончится сортировка  
document.body.addEventListener('sorted', () => {  
 console.log('event "sorted" did handled');  
 buttons.forEach((curr) => {  
 curr.classList.remove('disabled');  
 });  
});

Снимки экрана:

Кнопки. Активное состояние.



В процессе сортировки.



Сортировка закончена.

