

Fullstack-разработка

JavaScript

События

Функции

Цели урока:

1. Узнаем, как объявить свою функцию
2. Изучим 2 новых способа писать обработчики события

Разбор ДЗ

Что получилось?

Что было трудно?

Теория

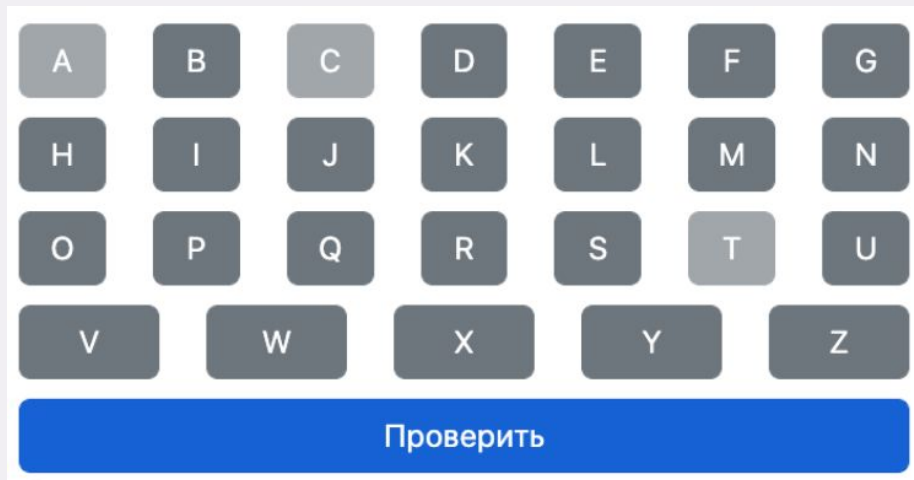
Функция как обработчик
события

Обработчик события

События возникают на **элементах**.

Чтобы отреагировать на них, мы пишем код в **обработчике события**.

```
item.addEventListener(`click`,  
function () {  
    console.log("Кликнули меня...");  
});
```

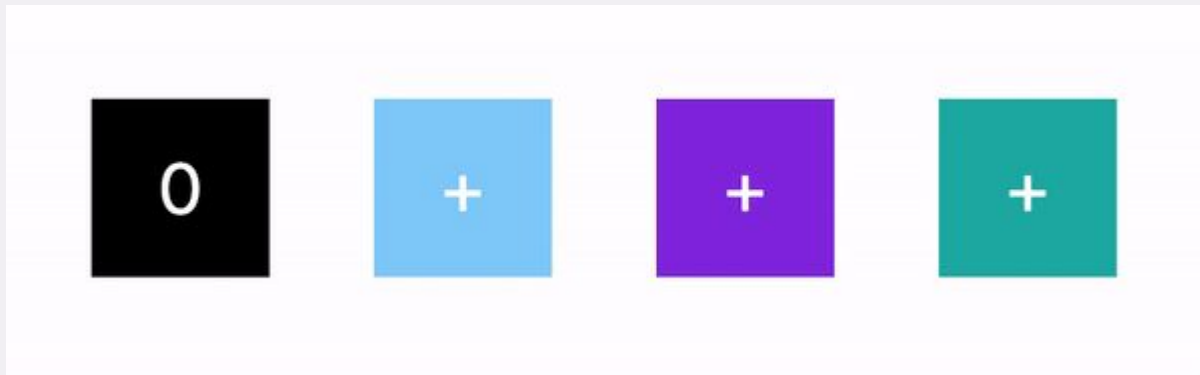


Что, если на странице
много элементов, которые
ведут себя **одинаково**?

A	B	C	D	E	F	G
H	I	J	K	L	M	N
O	P	Q	R	S	T	U
V	W	X	Y	Z		

Проверить

Сколько нужно
обработчиков событий???



Пример: клик по кнопке приводит к
одинаковому результату

Объявление функции

```
function onClick() {  
    console.log("Кликнули меня...");  
};
```

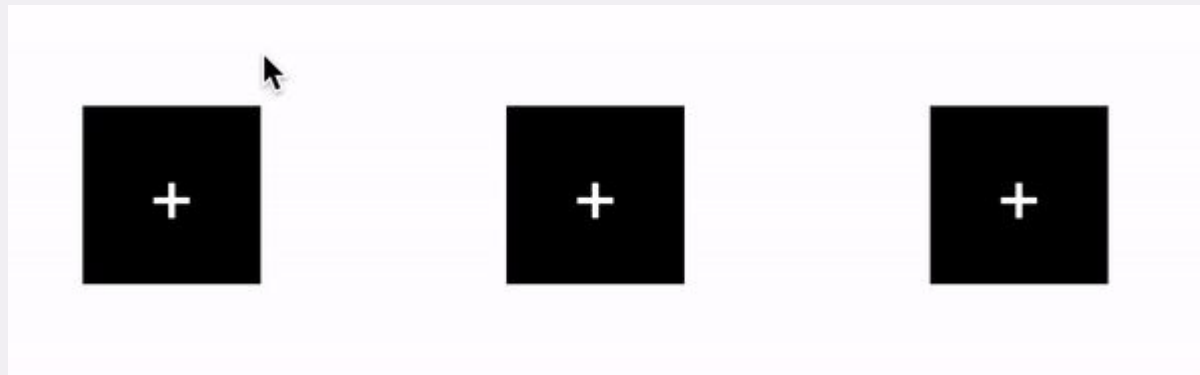
- Ключевое слово `function`
- Имя функции `onClick`
- Тело функции в фигурных скобках

Функция-обработчик

```
item1.addEventListener(`click`, onClick);  
item2.addEventListener(`click`, onClick);  
item3.addEventListener(`click`, onClick);
```

- Элементы
- Имя события click
- Имя функции onClick

1 функция –
несколько
обработчиков



Пример: клик по кнопке **меняет элемент**

Цель события

```
function onClick(evt) {  
    let item = evt.target;  
  
};
```

Свойство объекта события `evt.target`
хранит элемент, на котором возникло
событие

Цель события

```
function onClick(evt) {  
    let item = evt.target;  
    item.classList.add(`item_active`);  
};
```

Используйте **evt.target**, когда нужно **изменять элемент**, на котором произошло событие

classList.toggle

```
function onClick(evt) {  
    let item = evt.target;  
    item.classList.toggle('item_active');  
};
```

Ещё один метод, который пригодится в работе.

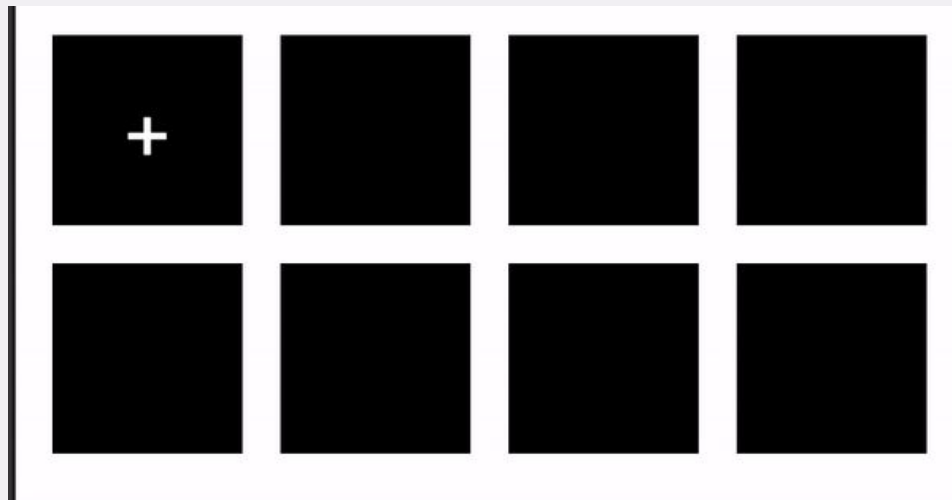
toggle добавляет класс, если его нет,
и удаляет класс, если он есть

Практика

Цель события

Теория

Проверка цели события



Пример: что, если элементов нет на странице, когда мы вешаем обработчик события?

Существует 2 способа
повесить обработчик
события:

1. На элемент
2. На контейнер элемента –
делегирование

Обработчик на элемент

```
function onClick(evt) {  
    let item = evt.target;  
    item.classList.toggle(`item_active`);  
};
```

Возьмём ту же самую функцию...

```
item1.addEventListener(`click`, onClick);  
item2.addEventListener(`click`, onClick);  
item3.addEventListener(`click`, onClick);
```

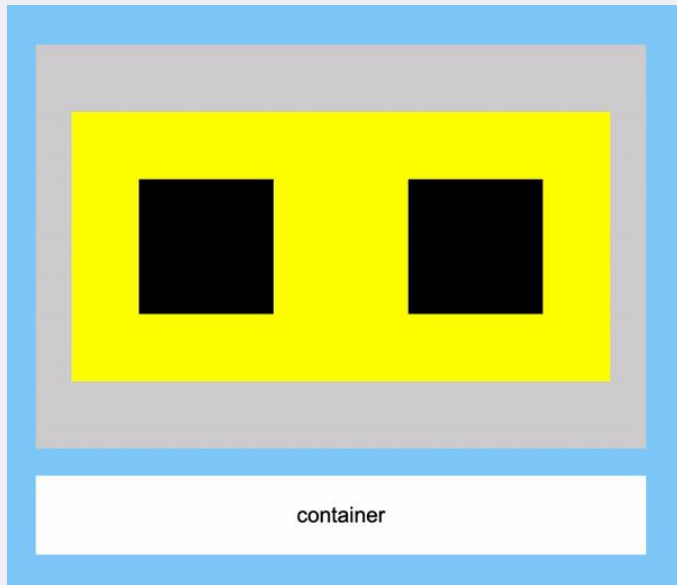
Обработчик на контейнер

```
function onClick(evt) {  
    let item = evt.target;  
    item.classList.toggle(`item_active`);  
};
```

И повесим её на **контейнер**!

```
container.addEventListener(`click`, onClick);
```

При делегировании
цель события может
быть любым
элементом внутри
контейнера.

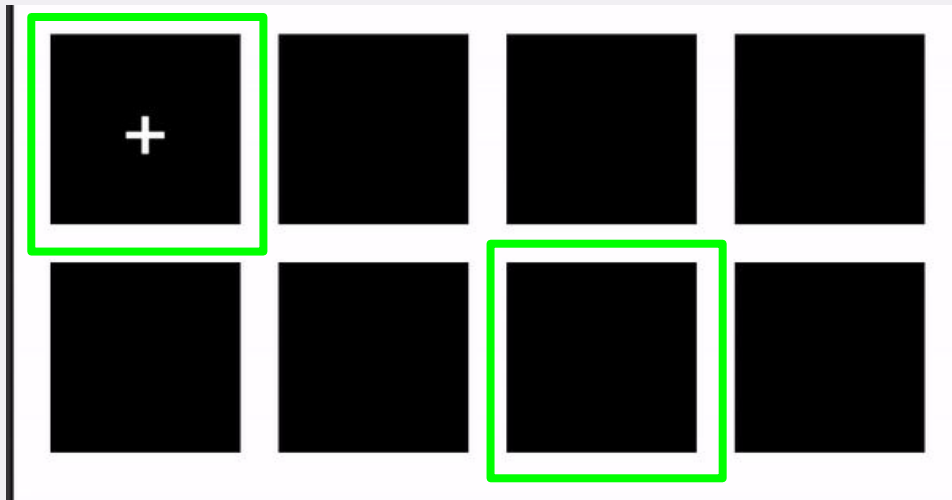


Для определения цели вместо атрибута **id**
теперь будем использовать **classList**

Проверка цели

```
function onClick(evt) {  
    let item = evt.target;  
    if (item.classList.contains('item_clickable')) {  
        item.classList.toggle('item_active');  
    }  
};
```

С помощью метода **classList.contains** можно проверить, какой перед нами элемент, и в зависимости от этого действовать по-разному



Так мы сможем в одном обработчике написать код и **для кнопки +**, и **для элемента, которого нет** на странице :)

querySelector

```
// Поиск по id
document.querySelector(`#container`);

// Поиск по class
document.querySelector(`.container`);
```

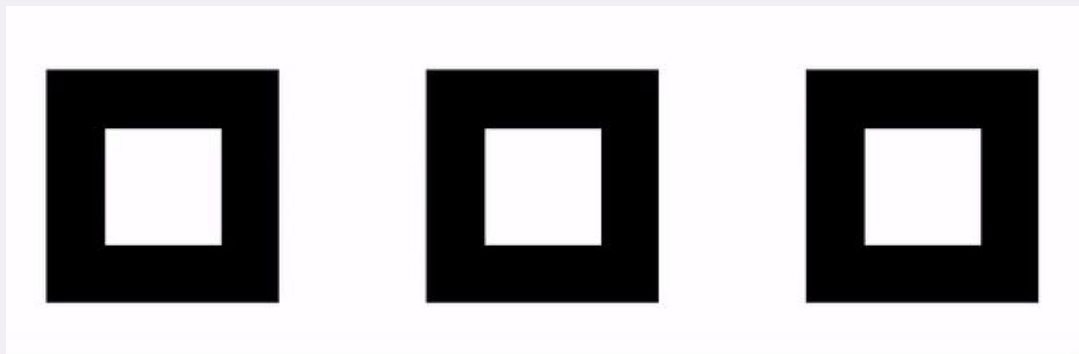
Теперь id уже не играет для нас большой роли, поэтому давайте полностью перейдём на классы

Практика

Проверка цели

Теория

Поиск контейнера



При делегировании целью может стать как белый, так и чёрный квадраты.

Что делать, если менять нужно только чёрный квадрат?

Методы для поиска

```
// Ищет по классу внутри item  
item.querySelector(`.item`);
```

```
// Ищет по классу среди контейнеров item  
item.closest(`.item`);
```

node.closest

```
function onClick(evt) {  
    let node = evt.target; // белый, чёрный, фон?  
    let black = node.closest(`.black`);  
    if (black) {                // точно нашли?  
        black.classList.toggle(`item_active`);  
    }  
}
```

Метод `node.closest` находит контейнер,
внутри которого находится **node**.

Он может и не найтись, поэтому не забывай про
дополнительное условие проверки

Практика

Подарок

Итоги

Если элементы уже есть на странице:

1. **Создай обработчик**
для одного элемента
2. **Добавь обработчик**
всем нужным элементам
3. Используй **цель события**,
если нужно менять элементы

Итоги

Если элементы появляются на странице не сразу:

1. **Создай обработчик**
для контейнера
2. **Сохрани цель** события
`evt.target`
3. **Определи цель** по её
`classList`

Цели урока:

- ✓ Узнаем, как объявить свою функцию
- ✓ Изучим 2 новых способа писать обработчики события

Домашнее задание