



Конспект «События в JavaScript».

Раздел 1

События — действия пользователя на странице (клик по кнопке, нажатие клавиши).

Добавление обработчиков событий

```
button.addEventListener('click', function () {  
  // Инструкции  
});
```

В примере:

- `button` — элемент, на котором мы хотим «слушать» событие.
- `addEventListener()` — функция добавления обработчика события на элемент.
- `'click'` — общепринятое название события, первый параметр функции `addEventListener`. Названия всех событий можно посмотреть [здесь](#).
- Второй параметр `addEventListener` — функция-обработчик, в ней записаны инструкции, которые выполнятся, только **когда произойдёт событие**.

Обратите внимание, мы **передаём функцию, а не её вызов**. Если мы вызовем функцию, код из этой функции выполнится сразу и больше не сработает. А нам нужно, чтобы код выполнялся **асинхронно** — в момент, когда произойдёт событие.

// Так добавлять обработчик неправильно

```
button.addEventListener('click', function () {  
  console.log('Клик по кнопке');  
})();  
// Сообщение сразу же выведется в консоль
```

// А такой код верный

```
button.addEventListener('click', function () {  
  console.log('Клик по кнопке');  
});  
// Сообщение выведется, когда произойдёт событие клика
```

В примере выше мы передаём в обработчик функцию, у которой нет своего имени, она не записана в переменную. Мы создали её там же, где передаём. Такие функции, которые создаются в момент передачи и не имеют имени, называются *анонимными функциями*.

Объект event

Объект `event` — параметр функции-обработчика. Он всегда передаётся браузером в эту функцию в момент наступления события. Этот объект содержит много полезных свойств и методов.

Чтобы использовать `event`, достаточно указать этот объект параметром функции-обработчика и написать инструкции. Остальное сделает JavaScript. Среди некоторых разработчиков принято называть параметр сокращённо — `evt`, во избежание ошибок.

Действия по умолчанию

Некоторые элементы страницы имеют действия по умолчанию или дефолтные действия. Например, клик по кнопке отправления формы вызывает отправку данных этой формы на сервер, а при клике по ссылке браузер переходит по этой ссылке.

Объект `event` содержит метод, который отменяет действие элемента по умолчанию: `preventDefault()`.

```
link.addEventListener('click', function( evt ) {  
  // Отменяем действие по умолчанию  
  evt.preventDefault();  
  
  // Добавляем инструкции для события клика  
  console.log( 'Произошёл клик' );  
});
```

Клавиатурные события

У события «нажатие на клавишу» есть специальное название — `'keydown'`. Такое событие срабатывает при нажатии на **любую клавишу**. Обратите внимание, слушать это событие можно только на элементах, которые имеют состояние фокуса: поля ввода, кнопки, элементы с атрибутом `tabindex`, **документ**. При нажатии фокус должен находиться на соответствующем элементе.

Если мы хотим поймать нажатие какой-то конкретной клавиши, можно обратиться к свойству `keyCode` объекта `event`. Это свойство содержит код нажатой клавиши. Например, у `Enter` код `13`, а у `ESC` — `27`. Эти номера универсальны и одинаковы в любой раскладке. Найти код любой клавиши можно [здесь](#).

```
document.addEventListener('keydown', function( evt ) {  
    // Проверяем, что код клавиши равен 27  
    if ( evt.keyCode === 27 ) {  
        // Код отсюда выполнится только при нажатии ESC  
    }  
});
```

Кроме `keyCode` есть и другие свойства для определения нажатой клавиши. Например, `key` и `code`. Они возвращают названия клавиш, а не их номера. Эти свойства пока поддерживаются не во всех браузерах, но когда поддержка станет лучше, стоит начать использовать их вместо `keyCode` в соответствии с современным стандартом JavaScript.

Продолжить



Практикум

Курсы для новичков

Подписка

Профессии

Фронтенд-разработчик

JavaScript-разработчик

Фулстек-разработчик

Услуги

Работа наставником

Для вузов и колледжей

Для учителей

Курсы

HTML и CSS. Профессиональная вёрстка сайтов

HTML и CSS. Адаптивная вёрстка и автоматизация

JavaScript. Профессиональная разработка веб-интерфейсов

JavaScript. Архитектура клиентских приложений

React. Разработка сложных клиентских приложений

Node.js. Профессиональная разработка REST API

Node.js и Nest.js. Микросервисная архитектура

TypeScript. Теория типов

Алгоритмы и структуры данных

Паттерны проектирования

Webpack

Vite

Vue.js 3. Разработка клиентских приложений

Git и GitHub

Анимация для фронтендеров

Журнал

[Справочник по HTML](#)

[Учебник по Git](#)

[Учебник по PHP](#)

Информация

[Об Академии](#)

[О центре карьеры](#)

Остальное

[Написать нам](#)

[Мероприятия](#)

[Форум](#)

[Акции](#)

[Отзывы о курсах](#)

[Соглашение](#)

[Конфиденциальность](#)

[Сведения об образовательной организации](#)

[Лицензия № ЛО35-01271-78/00176657](#)



Участник

© ООО «Интерактивные обучающие технологии», 2013–2025

