

# События JavaScript

# События

Взаимодействие JS с HTML осуществляется посредством событий (events), которые сигнализируют что в документе или окне браузера что-то произошло.

События соответствуют определенным действиям, которые выполняет пользователь или сам браузер, и имеют имена вроде click, load. Функция, выполняемая в ответ на событие, называется обработчиком события (event handler), или слушателем события (event listener). Имена таких функций начинаются с префикса “on”: например обработчик события click имеет имя onclick.

Назначать события можно несколькими способами:

# HTML-обработчики событий

Каждому событию, поддерживаемому конкретным элементом, можно назначить обработчик, указав специальный атрибут HTML. Например, для обработки щелчка по кнопке можно использовать следующий код:

```
<input type="button" value="Click me!" onclick="alert("Clicked!")" >
```

Таким образом, при клике на кнопку появится нужное нам сообщение.

Обработчик события также может вызывать функцию, указанную в другом м

Например:

```
<script>
```

```
function showMessage() {
```

```
  alert("Clicked!");
```

```
}
```

```
</script>
```

```
<input type="button" value="Click me!" onclick="showMessage()" >
```

# Традиционный способ обработки события:

Традиционный способ обработки события происходит через назначение функции свойству обработчика события.

```
const btn = document.getElementById("myBtn");  
  
btn.onclick = function() {  
  
    alert("Hello!");  
  
}
```

# Events

Для удаления/назначения обработчиков событий как правило используются методы `addEventListener()` / `removeEventListener()`. Они есть у всех элементов. Каждый метод может принимать два обязательных аргумента - имя обрабатываемого события и функцию-обработчик .

```
var btn = document.getElementById('myBtn');
```

```
btn.addEventListener('click', alMessage, false);
```

```
function alMessage() {
```

```
    alert('Hello!');
```

```
}
```

# Типы событий

- 1) События пользовательского интерфейса. Это общие событие браузера.
- 2) События изменения фокуса. Генерируются когда элемент теряет или получает фокус.
- 3) События мыши. Генерируются при выполнении каких-либо действий на странице при помощи мыши.
- 4) События колесика. Генерируются при использовании колесика мыши.
- 5) События редактирования текста. Генерируются при вводе текста в документ.
- 6) События клавиатуры. Генерируются при выполнении каких-либо действий на странице при помощи клавиатуры.

7) События композиции. Генерируются при вводе символов в редакторе метода ввода.

8) События изменения DOM-структуры. Генерируются при изменении базовой DOM-структуры.

9) События изменения имен. Генерируются при изменении имен элементов или атрибутов. Эти события очень устарели и реализованы не во всех браузерах.

В дополнение к этим событиям так же доступны HTML5 события и фирменные события DOM , BOM (они обычно определяются исходя из требований разработчиков).



# События пользовательского интерфейса.

Они не всегда связаны с действиями пользователя.

load - генерируется для объекта window при завершении загрузки страницы, для элементов img, object после завершения их загрузки

abort - генерируется для объекта object , если пользователь останавливает загрузку, а элемент загружен не полностью

error - генерируется для объекта window, если возникает JS ошибка, для объекта object, img - если их невозможно загрузить

select - генерируется если пользователь выделяет один или несколько символов в текстовом поле (input, textarea)

# События пользовательского интерфейса.

resize - генерируется для объекта window или фрейма при изменении его размеров

scroll - генерируется для любого элемента с полосой прокрутки, когда пользователь его прокручивает.

# Задание:

Попробуйте назначить обработчик события на загрузку страницы.  
После - на скролл.

# События мыши и колесика мыши

События мыши (mouse events) используются в веб-разработке чаще любых других, потому что большинство действий в браузере выполняется при помощи мыши.

`click` - генерируется, когда пользователь щелкает основной кнопкой мыши или нажимает клавишу Enter.

`dblclick` - генерируется, когда пользователь щелкает дважды основной кнопкой мыши

`mousedown` - генерируется когда пользователь нажимает любую кнопку мыши

`mouseenter` - генерируется при наведении указателя мыши на элемент

`mouseleave` - генерируется при смещении указателя мыши, находящегося на элементе

`mousemove` - генерируется при перемещении указателя мыши на элементе

`mouseout` - генерируется при перемещении указателя мыши , находящегося на одном элементе, в область другого элемента

`mouseover` - генерируется при наведении указателя мыши на элемент

`mouseup` - генерируется когда пользователь отпускает кнопку мыши.

События мыши поддерживаются всеми элементами страницы.

# Доступ к элементу через this

Внутри обработчика события `this` ссылается на текущий элемент, то есть на тот, на котором, как говорят, «висит» (т.е. назначен) обработчик.

```
<button onclick="alert(this.innerHTML)">Нажми меня</button>
```

# Объект event

Когда генерируется DOM-событие, все релевантные данные сохраняются в объекте event. Они включают базовые сведения, такие как целевой элемент и тип события, а также любые другие данные о конкретном событии. Например, для события мыши сохраняются сведения о позиции мыши, а для события клавиатуры - сведения о нажатых клавишах. Объект event поддерживают все браузеры, но по-разному.

# Отмена действия браузера

Основной способ — это воспользоваться объектом `event`. Для отмены действия браузера существует стандартный метод `event.preventDefault()`.



# События клавиатуры

Событие `keydown` происходит при нажатии клавиши, а `keyup` — при отпускании.

На современных устройствах есть и другие способы «ввести что-то». Например, распознавание речи или Копировать/Вставить с помощью мыши.

Поэтому, если мы хотим корректно отслеживать ввод в поле `<input>`, то одних клавиатурных событий недостаточно. Существует специальное событие `input`, чтобы отслеживать любые изменения в поле `<input>`. И оно справляется с такой задачей намного лучше.

# Задание 1

попробуем при помощи обработчика событий скрывать выпадающее меню при наведении на любой из элементов навигации (пункты меню).

## Задание 2

Создать 2 кнопки со значениями удалить и добавить. При нажатии кнопки добавить на страницу добавляется элемент с каким-либо текстом. Добавлять можно сколько угодно раз. При нажатии кнопки удалить - удаляется последний добавленный элемент.

# Задание 3

## Создание геометрических фигур

Реализовать функцию `drawFigure`, которая принимает в качестве аргументов: название геометрической фигуры, цвет заливки, положение от верхнего края страницы (css свойство `top`) и положение от левого края страницы (css свойство `left`), а также радиус, если это круг, сторону, если квадрат и ширину и высоту, если прямоугольник. Вызывать функцию при нажатии на соответствующие клавиши, например круг - Q, квадрат - W и тд

Пример использования: `drawFigure('круг', 'red', 100, 200, 40);` // рисует красный круг с радиусом 40px и на расстоянии 100px от верхнего 200px от левого края