# Введение в браузерные события

**Событие** – это сигнал от браузера о том, что что-то произошло. Все DOM-узлы подают такие сигналы (хотя события бывают и не только в DOM)

## События мыши:

**click** – происходит, когда кликнули на элемент левой кнопкой мыши (на устройствах с сенсорными экранами оно происходит при касании).

**contextmenu** – происходит, когда кликнули на элемент правой кнопкой мыши.

**mouseover / mouseout** – когда мышь наводится на / покидает элемент.

**mousedown / mouseup** – когда нажали / отжали кнопку мыши на элементе.

**mousemove** – при движении мыши.

## События на элементах управления:

**submit** – пользователь отправил форму <form>.

**focus** – пользователь фокусируется на элементе, например нажимает на <input>.

## Клавиатурные события:

**keydown и keyup** – когда пользователь нажимает / отпускает клавишу.

## События документа:

**DOMContentLoaded** – когда HTML загружен и обработан, DOM документа полностью построен и доступен.

## CSS events:

**transitionend** – когда CSS-анимация завершена.

И др.

## Обработчики событий

* Обработчик может быть назначен прямо в **разметке**, в атрибуте, который называется on<событие>.

<input value="Нажми меня" onclick="alert('Клик!')" type="button">

<input type="button" **onclick**="countRabbits()" value="Считать кроликов!">

* Можно назначать обработчик, используя **свойство DOM-элемента on<событие>.**

<input id="elem" type="button" value="Нажми меня!">

<script>

elem.**onclick** = function() {

alert('Спасибо');

};

</script>

**Так как у элемента DOM может быть только одно свойство с именем onclick, то назначить *более одного обработчика так нельзя*.**

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> // Нажми меня

**!** Не используйте **setAttribute** для обработчиков.

Такой вызов работать не будет:

document.body.**setAttribute**('onclick', function() { alert(1) });

**!** Регистр DOM-свойства имеет значение.

* Специальные методы:

**element.addEventListener(event, handler[, options]); -** листенер

***event*** - Имя события, например "click".

***Handler*** - Ссылка на функцию-обработчик.

***Options***- Дополнительный объект со свойствами:

* + ***once***: если true, тогда обработчик будет автоматически удалён после выполнения.
  + ***capture***: фаза, на которой должен сработать обработчик, подробнее об этом будет рассказано в главе Всплытие и погружение. Так исторически сложилось, что options может быть false/true, это то же самое, что {capture: false/true}.
  + ***passive****:* если true, то указывает, что обработчик никогда не вызовет preventDefault(), подробнее об этом будет рассказано в главе Действия браузера по умолчанию.

Для удаления обработчика следует использовать removeEventListener:

**element.removeEventListener(event, handler[, options]);**

Удаление требует именно ту же функцию (не анонимную, не стрелочную).

Обратим внимание – если функцию обработчик не сохранить где-либо, мы не сможем её удалить. Нет метода, который позволяет получить из элемента обработчики событий, назначенные через addEventListener.

**!** Метод addEventListener позволяет добавлять несколько обработчиков на одно событие одного элемента.

Существуют события, которые нельзя назначить через DOM-свойство, но можно через addEventListener.

Например, таково событие **DOMContentLoaded** и **transitionend**.

## Некоторые свойства объекта event:

*event.type* - тип события,например, "click".

*event.target* - это «целевой», самый глубокий элемент, на котором произошло событие

*event.target.id* - id целевого элемента

*event.target.className* - все классы целевого элемента одной строкой

*event.target.classList* - классы целевого элемента в виде коллекции

*event.currentTarget -* элемент, на котором сработал обработчик. Значение – обычно такое же, как и у this, но если обработчик является функцией-стрелкой или при помощи bind привязан другой объект в качестве this, то мы можем получить элемент из event.currentTarget.

*event.clientX / event.clientY* - координаты курсора в момент клика относительно **окна**, для событий мыши.

*event.* *offsetX/ event.* *offsetY*- координаты курсора в момент клика относительно **элемента**, для событий мыши.

*event.altKey*- зажатая клавиша **Alt**.

*event. shiftKey*- зажатая клавиша **Shift**.

*event.ctrlKey*- зажатая клавиша **Ctrl**.

## Отмена действий браузера по умолчанию

Чтобы отменить действие браузера по умолчанию, используйте **event.preventDefault()** или **return false**. Второй метод работает, только если обработчик назначен через **on<событие>.**

Опция **passive: true** для **addEventListener** сообщает браузеру, что действие по умолчанию не будет отменено. Это очень полезно для некоторых событий на мобильных устройствах, таких как touchstart и touchmove, чтобы сообщить браузеру, что он не должен ожидать выполнения всех обработчиков, а ему следует сразу приступать к выполнению действия по умолчанию, например, к прокрутке.

Если событие по умолчанию отменено, то значение **event.defaultPrevented** становится **true**, иначе **false**.

## Объект-обработчик: handleEvent

Мы можем назначить обработчиком не только функцию, но и объект при помощи addEventListener. В этом случае, когда происходит событие, вызывается метод объекта handleEvent. К примеру:

<button id="elem">Нажми меня</button>

<script>

elem.addEventListener('click', {

**handleEvent(event)** {

alert(event.type + " на " + event.currentTarget);

}

});

</script>