https://artzolin.ru/javascript-theory/javascript-events/

**Руководство по событиям в JavaScript**

Работа JavaScript основана на событийной модели — это значит, что для того, чтобы запустить какой-либо код, должно произойти событие. Даже код, который был написан в файле и не привязан к какому-либо событию, будет обработан после того, как произойдёт событие, которое сообщит браузеру, что код был загружен.

**Событие** — это то, что происходит, когда пользователь взаимодействует с веб-страницей, например, когда он щелкает ссылку или кнопку, вводит текст в поле <input> или <textarea> , делает выбор в поле <select>, нажимает клавишу на клавиатуре, перемещает указатель мыши, отправляет форму и т. д. В некоторых случаях сам браузер может инициировать такие события, например после окончательной загрузки страницы.

Событию можно назначить обработчик, то есть функцию, которая сработает, как только событие произошло.

**Функция-обработчик, или просто обработчик,** — это функция, которая вызывается браузером при наступлении события.

**События можно разделить на четыре основные группы** — события мыши, события клавиатуры, события формы и события документа/окна. Рассмотрим их подробнее.

1. **События Mouse**

Событие мыши запускается, когда пользователь щелкает какой-то элемент, перемещает указатель мыши на элемент и т. д. Вот некоторые наиболее важные события мыши и их обработчик.

**- Событие onclick**

Событие click происходит, когда пользователь нажимает на элемент на веб-странице. Часто это элементы формы и ссылки. Вы можете обработать событие click с помощью обработчика onclick.

Следующий пример покажет вам предупреждающее сообщение при нажатии на элементы.

<button type="button" onclick="alert('Вы нажали кнопку!');">Кнопка</button>

<a href="#" onclick="alert('Вы нажали на ссылку!');">Ссылка</a>

- **Событие oncontextmenu**

Событие contextmenu возникает, когда пользователь нажимает правую кнопку мыши на элементе, чтобы открыть контекстное меню. Вы можете обработать событие contextmenu с помощью обработчика oncontextmenu.

В следующем примере будет показано предупреждение при щелчке правой кнопкой мыши по элементам.

<button type="button" oncontextmenu="alert('Вы нажали правую кнопку!');">Кнопка</button>

<a href="#" oncontextmenu="alert('Вы щелкнули правой кнопкой мыши ссылку!');">Ссылка</a>

**- Событие onmouseover**

Событие mouseover происходит, когда пользователь наводит указатель мыши на элемент. Вы можете обработать событие mouseover с помощью обработчика onmouseover.

Следующий пример покажет вам предупреждающее сообщение при наведении мыши на элементы.

<button type="button" onmouseover="alert('Вы поместили указатель мыши на кнопку!');">Кнопка</button>

<a href="#" onmouseover="alert('Вы поместили указатель мыши на ссылку!');">Ссылка</a>

**-Событие onmouseout**

Событие mouseout происходит, когда пользователь перемещает указатель мыши за пределы элемента. Вы можете обработать событие mouseout с помощью обработчика onmouseout.

Следующий пример покажет вам предупреждающее сообщение, когда происходит событие mouseout.

<button type="button" onmouseout="alert('Вы убрали указатель с кнопки!');">Кнопка</button>

<a href="#" onmouseout="alert('Вы убрали указатель с ссылки!');">Ссылка</a>

1. **События Keyboard**

Событие клавиатуры вызывается, когда пользователь нажимает или отпускает клавишу на клавиатуре. Вот некоторые наиболее важные события клавиатуры и их обработчик событий.

**-Событие onkeydown**

Событие нажатия клавиши происходит, когда пользователь нажимает клавишу на клавиатуре. Вы можете обработать событие keydown с помощью обработчика onkeydown.

Следующий пример покажет вам предупреждающее сообщение, когда происходит событие keydown.

<input type="text" onkeydown="alert('Вы нажали клавишу внутри ввода текста!')">

<textarea onkeydown="alert('Вы нажали клавишу внутри текстовой области!')"></textarea>

**-Событие onkeyup**

Событие keyup происходит, когда пользователь отпускает клавишу на клавиатуре. Вы можете обработать событие keyup с помощью обработчика onkeyup.

В следующем примере будет показано предупреждающее сообщение при возникновении события keyup.

<input type="text" onkeyup="alert('Вы отпустили клавишу внутри ввода текста!')">

<textarea onkeyup="alert('Вы отпустили клавишу внутри textarea!')"></textarea>

**-Событие onkeypress**

Событие нажатия клавиши происходит, когда пользователь нажимает клавишу на клавиатуре, с которой связано символьное значение. Например, такие клавиши, как Ctrl, Shift, Alt, Esc, клавиши со стрелками и т. д., не будут генерировать события keypress, но будут генерировать события keydown и keyup.

Вы можете обработать событие нажатия клавиши с помощью обработчика onkeypress. В следующем примере будет показано предупреждающее сообщение при возникновении события нажатия клавиши.

<input type="text" onkeypress="alert('Вы нажали клавишу внутри ввода текста!')">

<textarea onkeypress="alert('Вы нажали клавишу внутри textarea!')"></textarea>

1. **События Form**

Событие формы вызывается, когда элемент управления получает или теряет фокус, или когда пользователь изменяет значение элемента управления формы, например, вводя текст в поле ввода, выбирает любой параметр и т. д. Вот некоторые наиболее важные события формы и их обработчик.

**-Событие onfocus**

Событие focus происходит, когда пользователь фокусируется на элементе на веб-странице. Вы можете обработать событие фокуса с помощью обработчика onfocus.

Следующий пример выделит фон ввода текста желтым цветом, когда он получит фокус.

<script>

function highlightInput(elm){

elm.style.background = "yellow";

}

</script>

<input type="text" onfocus="highlightInput(this)">

<button type="button">Button</button>

Значение ключевого слова this в обработчике события относится к элементу, на котором есть обработчик (то есть, куда в данный момент доставляется событие).

**-Событие onblur**

Событие blur происходит, когда пользователь убирает фокус с элемента формы или окна. Вы можете обработать событие размытия с помощью обработчика onblur.

Следующий пример покажет вам предупреждающее сообщение, когда элемент ввода текста теряет фокус.

<input type="text" onblur="alert('Ввод текста теряет фокус!')">

<button type="button">Submit</button>

Чтобы отвести фокус от элемента формы, сначала щелкните внутри него, затем нажмите клавишу табуляции на клавиатуре, наведите фокус на что-то другое или щелкните за его пределами.

**-Событие onchange**

Событие change происходит, когда пользователь изменяет значение элемента формы. Вы можете обработать событие изменения с помощью обработчика onchange.

В следующем примере будет показано предупреждающее сообщение при изменении параметра в поле выбора.

<select onchange="alert('Вы изменили выбор!');">

<option>Select</option>

<option>Male</option>

<option>Female</option>

</select>

**-Событие onsubmit**

Событие submit происходит только тогда, когда пользователь отправляет форму на веб-странице. Вы можете обработать событие submit с помощью обработчика onsubmit.

В следующем примере будет показано предупреждение при отправке формы на сервер.

<form action="action.php" method="post" onsubmit="alert('Данные формы будут отправлены на сервер!');">

<label>First Name:</label>

<input type="text" name="first-name" required>

<input type="submit" value="Submit">

</form>

1. **События Document/Window**

События также инициируются в ситуациях, когда страница загружена или когда пользователь изменяет размер окна браузера. Вот некоторые наиболее важные события document/window и их обработчик событий.

**-Событие onload**

Событие load происходит после завершения загрузки веб-страницы в веб-браузере. Вы можете обработать событие загрузки с помощью обработчика события onload.

Следующий пример покажет вам предупреждающее сообщение, как только страница завершит загрузку.

<body onload="window.alert('Страница загружена успешно!');">

<h1>This is a heading</h1>

<p>This is paragraph of text.</p>

</body>

**-Событие onunload**

Событие unload происходит, когда пользователь покидает текущую веб-страницу. Вы можете обработать событие unload с помощью обработчика onunload.

Следующий пример покажет вам предупреждение при попытке покинуть страницу.

<body onunload="alert('Вы уверены, что хотите покинуть эту страницу??');">

<h1>Это заголовок</h1>

<p>Это абзац текста.</p>

</body>

**-Событие onresize**

Событие resize происходит, когда пользователь изменяет размер окна браузера. Событие resize также происходит в ситуациях, когда окно браузера свернуто или развернуто.

Вы можете обработать событие resize с помощью обработчика onresize. В следующем примере будет показано предупреждающее сообщение при изменении размера окна браузера до новой ширины и высоты.

<p id="result"></p>

<script>

function displayWindowSize() {

var w = window.outerWidth;

var h = window.outerHeight;

var txt = "Window size: width=" + w + ", height=" + h;

document.getElementById("result").innerHTML = txt;

}

window.onresize = displayWindowSize;

</script>

**Полный список обработчиков событий** приводится ниже. Выполнение кода обработчика запускается, когда выполнено условие:

* **onabort** – прерывание воспроизведения;
* **onafterprint** – закончена печать;
* **onautocomplete** – выполнено автозаполнение формы;
* **onautocompleteerror** – ошибка при автозаполнении формы;
* **onbeforeprint** – подготовка к печати;
* **onbeforeunload** - документ выгружен;
* **onblur** – потеря фокуса;
* **oncancel** – отмена действия;
* **oncanplay** - можно начать воспроизведение указанного медиа-файла;
* **oncanplaythrough** - можно начать воспроизведение указанного медиа-файла без необходимости остановки для буферизации;
* **onchange** – изменение значения;
* **onclick** – клик на элементе;
* **onclose** – закрытие чего-либо;
* **oncontextmenu** – открытие контекстного меню;
* **oncopy** – выполнено копирование;
* **oncuechange** - изменение метки в элементе **track**;
* **oncut** – выполнено вырезание контента;
* **ondblclick** – двойной клик на элементе;
* **ondrag** - перетаскивание элемента;
* **ondragend** - перетаскивание элемента завершено;
* **ondragenter** - элемент перетаскивается в допустимую целевую зону;
* **ondragexit** – выход из режима перетаскивания;
* **ondragleave** - элемент оставляет допустимую цель;
* **ondragover** - элемент перетаскивается по допустимой целевой точке;
* **ondragstart** – начало операции перетаскивания;
* **ondrop** – перетаскиваемый элемент отпущен;
* **ondurationchange** - изменение длины носителя;
* **onemptied** – файл неожиданно стал недоступен;
* **onended** – воспроизведение завершено;
* **onerror** – произошла ошибка;
* **onfocus** – установка фокуса на элементе;
* **onhashchange** - изменение привязки части адреса;
* **oninput** – начало ввода данных;
* **oninvalid** – элемент поврежден;
* **onkeydown** – нажата клавиша;
* **onkeypress** - нажата клавиша и затем отпущена;
* **onkeyup** – отпущена клавиша;
* **onload** – элемент загружен;
* **onloadeddata** – загружены данные файла;
* **onloadedmetadata** – загружены метаданные файла;
* **onloadstart** – начало загрузки элемента;
* **onmessage** - появление сообщения;
* **onmousedown** – нажата клавиша мыши;
* **onmouseenter** – указатель мыши наведен на элемент;
* **onmouseleave** – указатель мыши покинул элемент;
* **onmousemove** - указатель мыши перемешен над элементом;
* **onmouseout** - указатель мыши перемещается из элемента;
* **onmouseover** - указатель мыши перемещается по элементу;
* **onmouseup** - кнопка мыши отпускается над элементом;
* **onmousewheel (onwheel)** – использовано колесико мыши;
* **onoffline** – браузер запущен в оффлайн режиме;
* **ononline** – браузер запущен в онлайн режиме;
* **onpagehide** - пользователь переходит от страницы;
* **onpageshow** - пользователь переходит на страницу;
* **onpaste** – выполнена вставка контента;
* **onpause** – приостановка воспроизведения;
* **onplay** – начало воспроизведения;
* **onplaying** – воспроизведения файла;
* **onpopstate** - изменение истории окна;
* **onprogress** – получение метаданных файла;
* **onratechange** - изменение скорости воспроизведения;
* **onreset** – выполнен сброс данных;
* **onresize** – изменение размера элемента;
* **onscroll** – прокрутка содержимого элемента;
* **onsearch** – выполнен поиск;
* **onseeked** - поиск закончился;
* **onseeking** – поиск активен;
* **onselect** – выбор некоторого текста или значения;
* **onshow** – показ элемента;
* **onsort** – выполнение сортировки;
* **onstalled** - браузер не может получить мультимедийные данные по любой причине;
* **onstorage** – обновлено веб-хранилище;
* **onsubmit** – подтверждение отправки данных формы;
* **onsuspend** – остановка извлечения метаданных;
* **ontimeupdate** – изменение позиции (времени) воспроизведения файла, то есть перемотка файла;
* **ontoggle** - пользователь открывает или закрывает элемент **details**;
* **onunload** – загрузка выполнена, после чего произошло закрытие документа;
* **onvolumechange** – громкость изменена;
* **onwaiting** – ожидание восстановления воспроизведения.

Способы назначить события:

1. **Использование атрибута HTML**

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

Атрибут HTML-тега – не самое удобное место для написания большого количества кода, поэтому лучше создать отдельную JavaScript-функцию и вызвать её там.

<script>

function countRabbits() {

for(let i=1; i<=3; i++) {

alert("Кролик номер " + i);

}

}

</script>

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

1. **Использование свойства DOM-объекта**

elem.onclick = function() {

alert('Спасибо');

};

Обработчиком можно назначить и уже существующую функцию:

function sayThanks() {

alert('Спасибо!');

}

elem.onclick = sayThanks;

Убрать обработчик можно назначением elem.onclick = null.

Фундаментальный недостаток описанных выше способов назначения обработчика – невозможность повесить несколько обработчиков на одно событие **(так как последний будет перебивать предыдущий).**

**Синтаксис добавления обработчика:**

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

**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.

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

<script>

function handler1() {

alert('Спасибо!');

};

function handler2() {

alert('Спасибо ещё раз!');

}

elem.onclick = () => alert("Привет");

elem.addEventListener("click", handler1); // Спасибо!

elem.addEventListener("click", handler2); // Спасибо ещё раз!

</script>

**Для удаления обработчика следует использовать removeEventListener:**

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

**Объект события —** это JavaScript-объект с информацией о событии. В объекте события есть как общие свойства (тип события, время события), так и свойства, которые зависят от типа события (например, на какую кнопку нажал пользователь).

Чтобы работать с объектом события, нужно добавить параметр event в объявление обработчика.

element.addEventListener('keydown', function (event) {

Объект Event описывает событие, произошедшее на странице.

event.bubbles — является ли данное событие всплывающим.

event.cancelable — является ли событие отменяемым.

event.currentTarget — указывает на элемент, на котором установлен обработчик события.

event.defaultPrevented — отменено ли поведение события по умолчанию.

event.eventPhase — указывает на фазу срабатывания события.

event.isTrusted — указывает на происхождение события, будет в значении true, если событие инициировано действиями пользователя. false — если событие инициировано из кода с помощью dispatchEvent().

event.target — ссылка на объект, которым было инициировано событие. Например, если событие произошло на поле ввода, мы получим ссылку на этот DOM элемент.

event.timeStamp — время возникновения события в миллисекундах.

event.type — тип события.

**Методы**

composedPath() — вернёт массив элементов, на которых сработает событие.

preventDefault() — предотвращает дефолтное поведение события. Если вызвать этот метод на событии клика по ссылке, то переход по ссылке не произойдёт, но событие продолжит всплытие.

stopPropagation() — предотвращает всплытие события.

stopImmediatePropagation() — делает то же самое, что и stopPropagation, но в том числе предотвращает вызов обработчиков события, которые были установлены на этом же элементе.

**Всплытие и погружение**