SkillUp

JavaScript. События

Cernomschi Vitali | 2019

События мыши:

- click – происходит, когда кликнули на элемент левой кнопкой мыши
- contextmenu – происходит, когда кликнули на элемент правой кнопкой мыши
- mouseover – возникает, когда на элемент наводится мышь
- mousedown и mouseup – когда кнопку мыши нажали или отжали
- mousemove – при движении мыши

События на элементах управления:

- submit – посетитель отправил форму <form>
- focus – посетитель фокусируется на элементе, например нажимает на <input>

Клавиатурные события:

- keydown – когда посетитель нажимает клавишу
- keyup – когда посетитель отпускает клавишу

События документа:

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

События CSS:

- transitionend – когда CSS-анимация завершена.

Назначение обработчиков событий

Использование атрибута HTML

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

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <script>
        function countRabbits()  {
            for(var i=1; i<=3; i++) {
                alert("Кролик номер " + i);
            }
        }
    </script>
  </head>
  <body>
    <input type="button" onclick="countRabbits()" value="Считать кроликов!"/>
  </body>
</html>
```

Использование свойства DOM-объекта

```
<input id="elem" type="button" value="Нажми меня" />
<script>
   elem.onclick = function()  {
      alert( 'Спасибо' );
   };
</script>
```

Функция по событию

```
function sayThanks()  {
    alert( 'Спасибо!' );
}
elem.onclick = sayThanks;
```

Доступ к элементу через this

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

Недостаток назначения через свойство

```
elem.onclick = function()  {
    alert( ‘Пятница, 18-50' );
};

elem.onclick = function()  {
    alert( ‘Добрый вечер' );
};

elem.onclick = function()  {
    alert( ‘Лекция по Frontend’ );
};
```

addEventListener

```
element.addEventListener(event, handler[, phase]);
```

- event - имя события, например click
- handler - ссылка на функцию, которую надо поставить обработчиком
- phase - необязательный аргумент, «фаза», на которой обработчик должен сработать.

removeEventListener

```
// передать те же аргументы, что были у addEventListener
element.removeEventListener(event, handler[, phase]);
```

Удаление требует именно ту же функцию

```
elem.addEventListener( "click" , function()  {
    alert('Спасибо!')
});
// ....
elem.removeEventListener( "click", function()  {
alert('Спасибо!')
});
```

```
function ThankYou()  {
    alert( 'Спасибо!' );
}

input.addEventListener("click", ThankYou);
// ....
input.removeEventListener("click", ThankYou);
```

```
<input id="elem" type="button" value="Нажми меня“ />
<script>
  function handler1()  {
    alert('Спасибо!');
  };
  function handler2()  {
    alert('Спасибо ещё раз!');
  }
  elem.onclick = function()  {
    alert("Привет");
  };
  elem.addEventListener("click", handler1); // Спасибо!
  elem.addEventListener("click", handler2); // Спасибо ещё раз!
</script>
```

addEventListener работает всегда, а DOM-свойство – нет

Порядок обработки событий

Главный поток

- В каждом окне выполняется только **один *главный* поток**, который занимается выполнением JavaScript, отрисовкой и работой с DOM.

- Он выполняет команды **последовательно**, может делать только одно дело одновременно и блокируется при выводе модальных окон, таких как alert.

Очередь событий

- **Когда происходит событие, оно попадает в очередь.**
- **Иногда события добавляются в очередь сразу пачкой.**

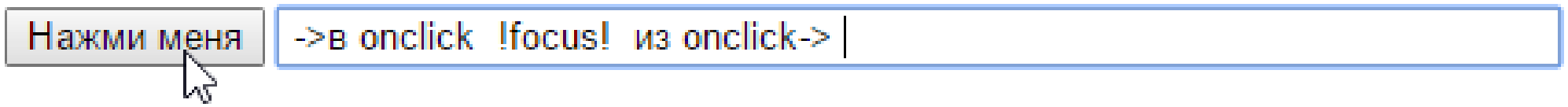
```
<textarea rows="8" cols="40" id="area">Кликни меня </textarea>
<script>
    area.onmousedown = function(event)  {
        this.value += "mousedown\n";
        this.scrollTop = this.scrollHeight;
    };
    area.onmouseup = function(event)  {
        this.value += "mouseup\n";
        this.scrollTop = this.scrollHeight;
    };
    area.onclick = function(event) {
        this.value += "click\n";
        this.scrollTop = this.scrollHeight;
    };
</script>
```

Результат

Вложенные (синхронные) события

```
<input type="button" id="button" value="Нажми меня" />
<input type="text" id="text" size="60" />
<script>
    button.onclick = function()  {
        text.value += ' ->в onclick ';
        text.focus(); // вызов инициирует событие onfocus
        text.value += ' из onclick-> ';
    };
    text.onfocus = function()  {
        text.value += ' !focus! ';
    };
</script>
```

Результат



Исключение в IE

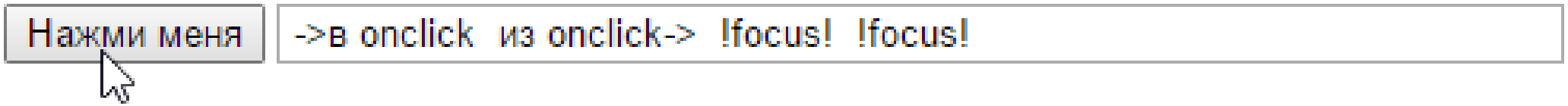
Так ведут себя все браузеры, кроме IE.

В нём событие onfocus – всегда асинхронное, так что будет сначала полностью обработан клик, а потом – фокус. В остальных – фокус вызовется посередине клика.

Делаем события асинхронными через setTimeout(...,0)

```
<input type="button" id="button" value="Нажми меня" />
<input type="text" id="text" size="60" />
<script>
    button.onclick = function()  {
        text.value += ' ->в onclick ';
        setTimeout(function()  {
            text.focus(); // сработает после onclick
        }, 0);
        text.value += ' из onclick-> ';
    };
    text.onfocus = function()  {
        text.value += ' !focus! ';
    };
</script>
```

Результат



Объект события

Свойства объекта события

```
<input type="button" value="Нажми меня" id="elem" />
<script>
  elem.onclick = function(event) {
    // вывести тип события, элемент и координаты клика
    alert(event.type + " на " + event.currentTarget);
    alert(event.clientX + ":" + event.clientY);
  }
</script>
```

Свойства объекта event:

- event.type - тип события, в данном случае click
- event.currentTarget - элемент, на котором сработал обработчик. Значение – в точности такое же, как и у this, но бывают ситуации, когда обработчик является методом объекта и его this при помощи bind привязан к этому объекту, тогда мы можем использовать event.currentTarget
- event.clientX / event.clientY - координаты курсора в момент клика (относительно окна)

Объект события доступен и в HTML

```
<input type="button" onclick="alert(event.type)" value="Тип события" />
```

References

- 1. События:
  - RU: https://learn.javascript.ru/events-and-interfaces
  - EN: https://javascript.info/events

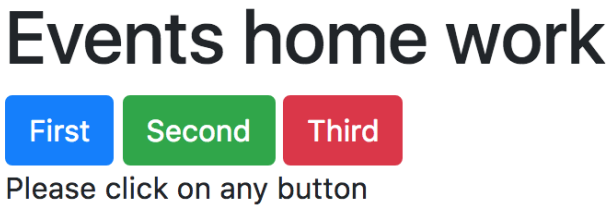
Домашнее задание — теоретическая часть

- 1. Прочитать следующие разделы
  - Русская версия учебника
    - https://learn.javascript.ru/introduction-browser-events
    - https://learn.javascript.ru/events-and-timing-depth
    - https://learn.javascript.ru/obtaining-event-object
    - https://learn.javascript.ru/event-bubbling
  - English Version
    - https://javascript.info/introduction-browser-events
    - https://javascript.info/bubbling-and-capturing

Домашнее задание — практическая часть

Создать страницу которая будет по нажатию на кнопки «First», «Second», «Third» менять текст на страницы из «Please click on any button» на «You clicked First button», «You clicked Second button» и «You clicked Third button» соответственно нажатой кнопке.

Подписка и обработка может осуществлять любым из изученных методов используя атрибут HTML, DOM-объект или addEventListener.



*__Advanced part__ сделать 2 варианта:

- на каждую кнопку применять разный метод подписки и обработки события