События / Events

JavaScript
Courses

www.courses.dp.ua

Система управления основанная на событиях

Каждая из этих вещей делает что-то, только в ответ на действия пользователя. Можно сказать каждое действие пользователя это событие, и на него нужно как-то отреагировать.

События / Events

В программировании обработка событий основана на функциях. Поскольку функции хорошо подходят для того чтобы многократно (неизвестно заранее сколько) выполнять один и тот же фрагмент кода.

События / Events

Смысл событий в JS - сказать браузеру: «когда произойдёт клик по элементу, то выполни вот эту функцию»;

События / Events

Событийная модель – *подход в программировании, когда действия программы определяются событиями, как правило действиями пользователя (мышь, клавиатура, сенсор), сообщениями от других программ и/или операционной системы.*

Событие *– действие о котором браузер должен уведомить нашу программу;*

Подписаться на событие *– указать браузеру, что «при клике нужно вызвать функцию ABC()»;*

Обработчик события *– функция которая будет вызываться при наступлении события;*

Слушать событие *– тоже самое, что и ждать наступления события.*

События / Events

Программа не может отреагировать на абсолютно все возможные события, который могут произойти.

Вариантов событий много, задача программиста выбрать нужное.

События / Events

Вариантов событий много, задача программиста выбрать нужное

HTML DOM Events

DOM: Indicates in which DOM Level the property was introduced.

Mouse Events

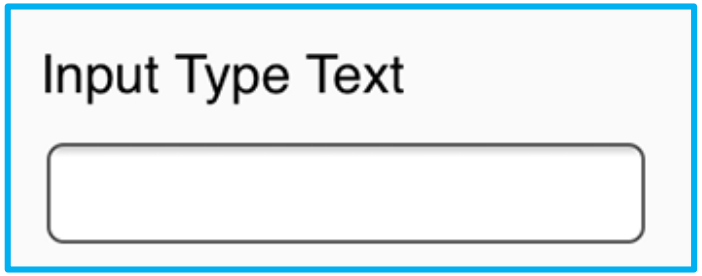
Event	Description	DOM
onclick	The event occurs when the user clicks on an element	2
oncontextmenu	The event occurs when the user right-clicks on an element to open a context menu	3
ondblclick	The event occurs when the user double-clicks on an element	2
onmousedown	The event occurs when the user presses a mouse button over an element	2
onmouseenter	The event occurs when the pointer is moved onto an element	2
onmouseleave	The event occurs when the pointer is moved out of an element	2
onmousemove	The event occurs when the pointer is moving while it is over an element	2
onmouseover	The event occurs when the pointer is moved onto an element, or onto one of its children	2
onmouseout	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children	2
onmouseup	The event occurs when a user releases a mouse button over an element	2

Keyboard Events

Event	Description	DOM
onkeydown	The event occurs when the user is pressing a key	2
onkeypress	The event occurs when the user presses a key	2
onkeyup	The event occurs when the user releases a key	2

http://www.w3schools.com/jsref/dom_obj_event.asp

События возможные для одних элементов, могут не существовать для других



Поддерживает ввод с клавиатуры, события «фокус» и «потеря фокуса».

Не поддерживает ввод с клавиатуры, и событий «фокус» и «потеря фокуса» для него тоже быть не может.

Однако есть набор событий который поддерживают все элементы: клик, наведение курсора мыши и т.д.

Подписка на события

Как сказать браузеру какую функцию и когда вызывать?

Через атрибуты HTML-элементов:

```
1  <html>
2      <head>
3          <script>
4              function on_event(){
5                  alert("Функция вызвана!");
6              }
7          </script>
8      </head>
9      <body>
10         <p onclick="on_event();" onmouseover="on_event();">Text Text Text</p>
11     </body>
12 </html>
```

Недостаток - JavaScript код в HTML-разметке.
А как быть с теми элементами которые появятся потом?

Как сказать браузеру какую функцию и когда вызывать?

Через свойства объектов входящих в дерево документа

```
<html>
    <head>
        <script>
            var func = function(){
                alert("Зафиксировано событие");
            }

            window.onload = function(){
                var p = document.querySelector("p");
                p.onclick = func;
            }
        </script>
    </head>
    <body>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit</p>
    </body>
</html>
```

Недостаток - можно подключить максимум один обработчик события.

***window.onload** – тоже событие, оно наступает когда документ полностью загружен браузером.*

Как сказать браузеру какую функцию и когда вызывать?

```html
<html>
    <head>
        <script>
            function handler1(){
                alert("Обработчик №1");
            }

            function handler2(){
                alert("Обработчик №2");
            }

            window.onload = function(){
                var p = document.querySelector("p");
                p.addEventListener("click", handler1);
                p.addEventListener("click", handler2);

                //Подписку можно и отменить, функцией removeEventListener()
                //p.removeEventListener("click", handler1);
                //p.removeEventListener("click", handler1);
            }
        </script>
    </head>
    <body>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit</p>
    </body>
</html>
```

*При помощи **.addEventListener()** можно на одно событие повесить множество обработчиков. А при необходимости и снять обработчик при помощи **.removeEventListener()**.*

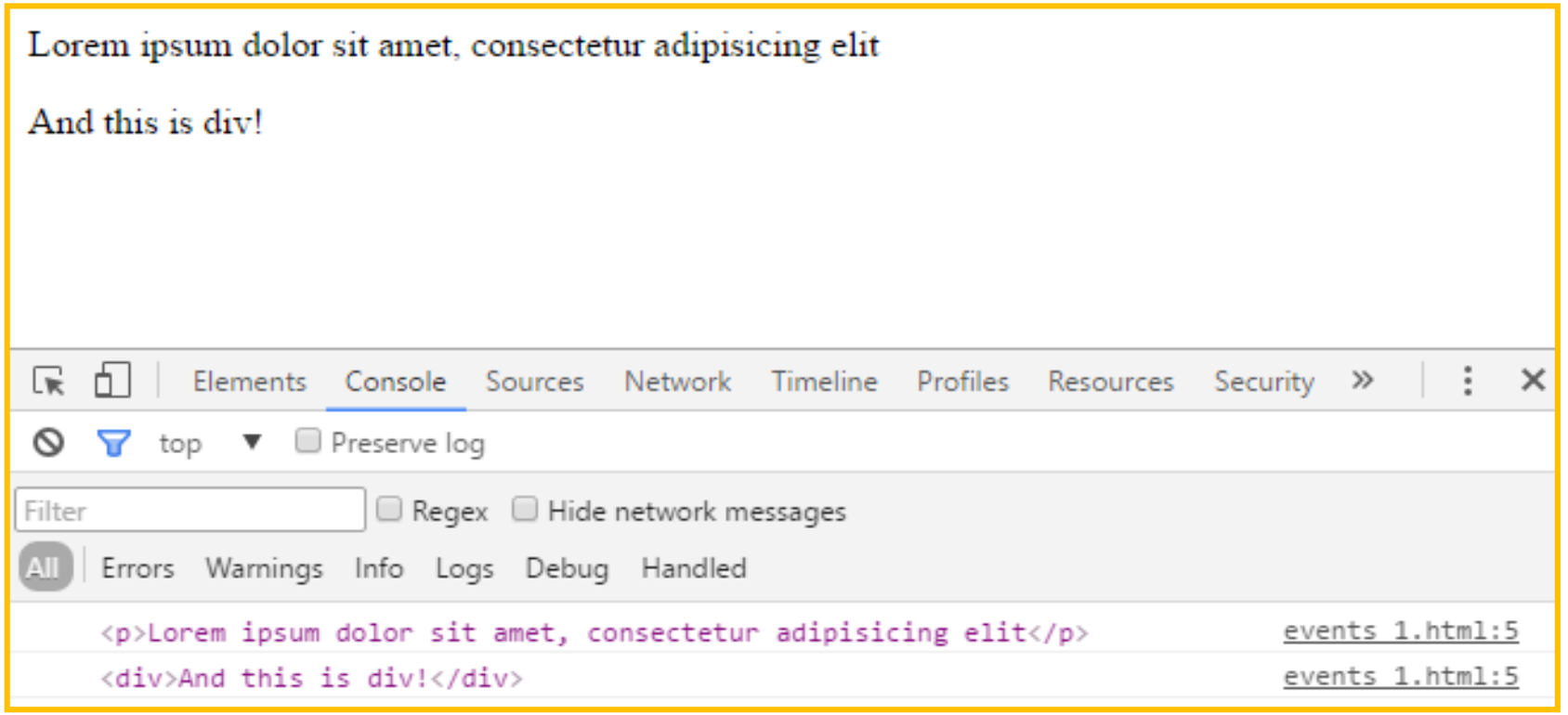
Вспоминаем this

*Функция обработчик становиться частью объекта-элемента, и вызывается как его метод. Поэтому ключевое слово **this** в обработчике ссылается на объект который вызвал обработчик события.*

```
<html>
    <head>
        <script>
            function handler(){
                console.log(this);
            }

            window.onload = function(){
                var p = document.querySelector("p");
                p.onclick = handler;
                var d = document.querySelector("div");
                d.onclick = handler;
            }
        </script>
    </head>
    <body>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit</p>
        <div>And this is div!</div>
    </body>
</html>
```

События / Events

*Функция обработчик становиться частью объекта-элемента, и вызывается как его метод. Поэтому ключевое слово **this** в обработчике ссылается на объект который вызвал обработчик события.*



События

onLoad,
onDOMContentLoaded

Событие onLoad

*Событие **onload** (объекта **window**) срабатывает тогда когда загружен HTML документ и все подключаемые файлы, в т.ч изображения.*

```
3       window.onload = function(event){
4           console.log("Load complete");
5       });
```

Событие DOMContentLoaded

*Событие **DOMContentLoaded** доступно для объекта **document** через **.addEventListener()** и срабатывает тогда когда загружен HTML документ и JS файлы (завершилась ли загрузка изображений и css-файлов неважно).*

```
3    document.addEventListener("DOMContentLoaded", function(event) {
4        console.log("DOM fully loaded and parsed");
5    });
```

Информация о событии

Информация о событии

Чтобы обработать событие, недостаточно знать о том, что это – «клик» или «нажатие клавиши». Могут понадобиться детали: координаты курсора, введённый символ и другие, в зависимости от события.

Браузер может дать много полезной информации о событии, для этого он создаёт объект, в свойства которого записывает детали произошедшего события. И передаёт этот объект функции обработчику события.

https://developer.mozilla.org/en-US/docs/Web/API/MouseEvent

Информация о событии

```
1  <html>
2      <head>
3          <script>
4              function handler(e){ console.dir(e); }
5
6              window.onload = function(){
7                  document.querySelector("button").onclick = handler;
8              }
9          </script>
10     </head>
11     <body>
12         <button>Click Me!</button>
13     </body>
14 </html>
```

Браузер записывает информацию о событии в объект т.н. «объект события», который передаётся первым аргументом в функцию обработчик события. Если она принимает параметры, т.к. это является необязательным.

Информация о событии

Разные события – разные объекты с информацией о них.

В зависимости от типа события, объект с детальной информацией о событии содержит разные наборы полей, например: для событий мыши он содержит координаты курсора, а события клавиатуры он содержит данные о нажатых клавишах.

https://developer.mozilla.org/en-US/docs/Web/API/MouseEvent

https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent

Информация о событии

Разные события – разные объекты с информацией о них.

```
<html>
    <head>
        <script>
            function handler(e){ console.dir(e); }

            window.onload = function(){
                document.querySelector("p").onclick = handler;
                document.querySelector("input").onkeypress = handler;
            }
        </script>
    </head>
    <body>
        <p>Text Text Text</p>
        <input type="text">
    </body>
</html>
```

https://developer.mozilla.org/en-US/docs/Web/API/MouseEvent

https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent

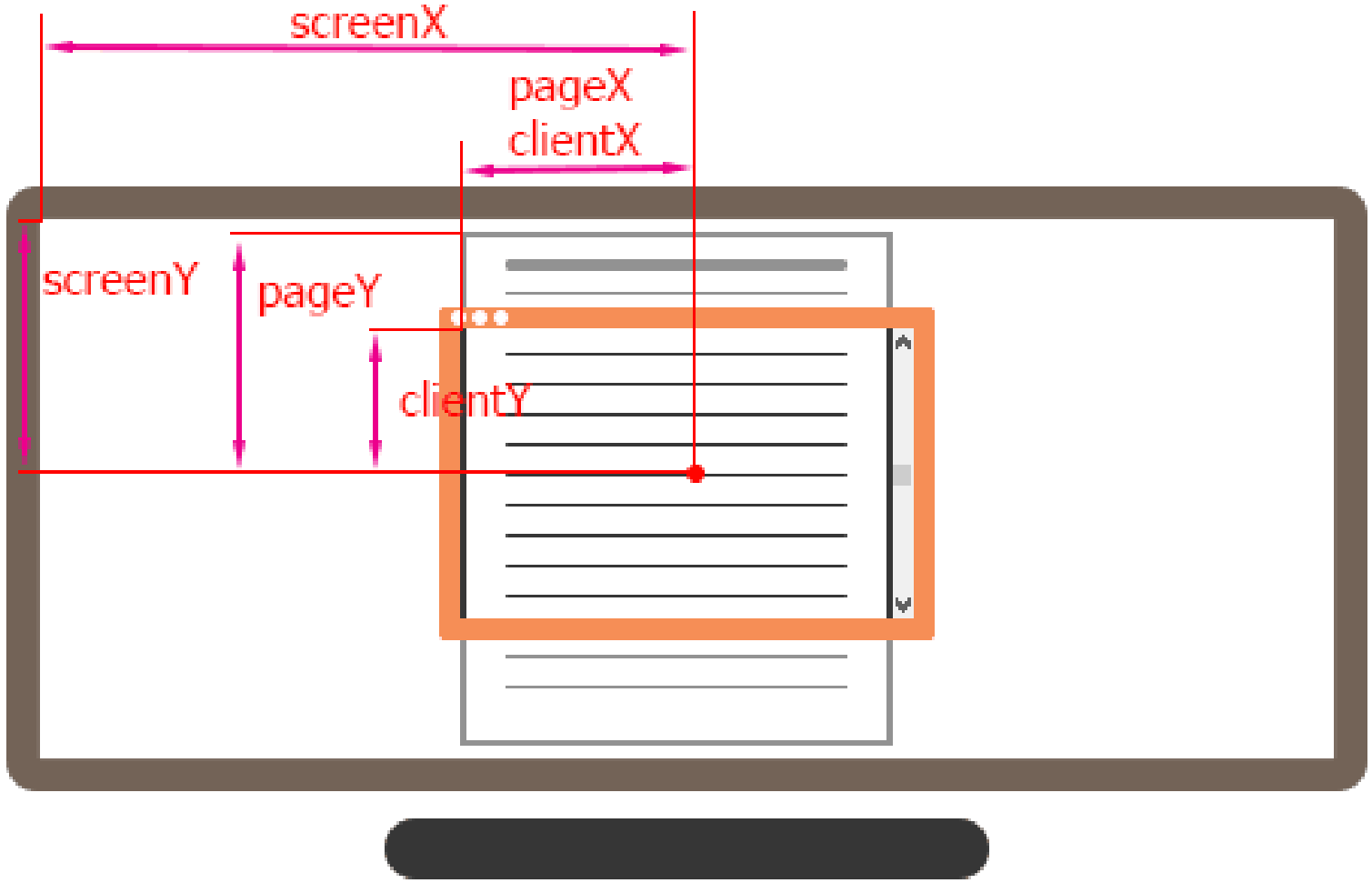
Информация о событии

Разные события – разные объекты с информацией о них.

onclick

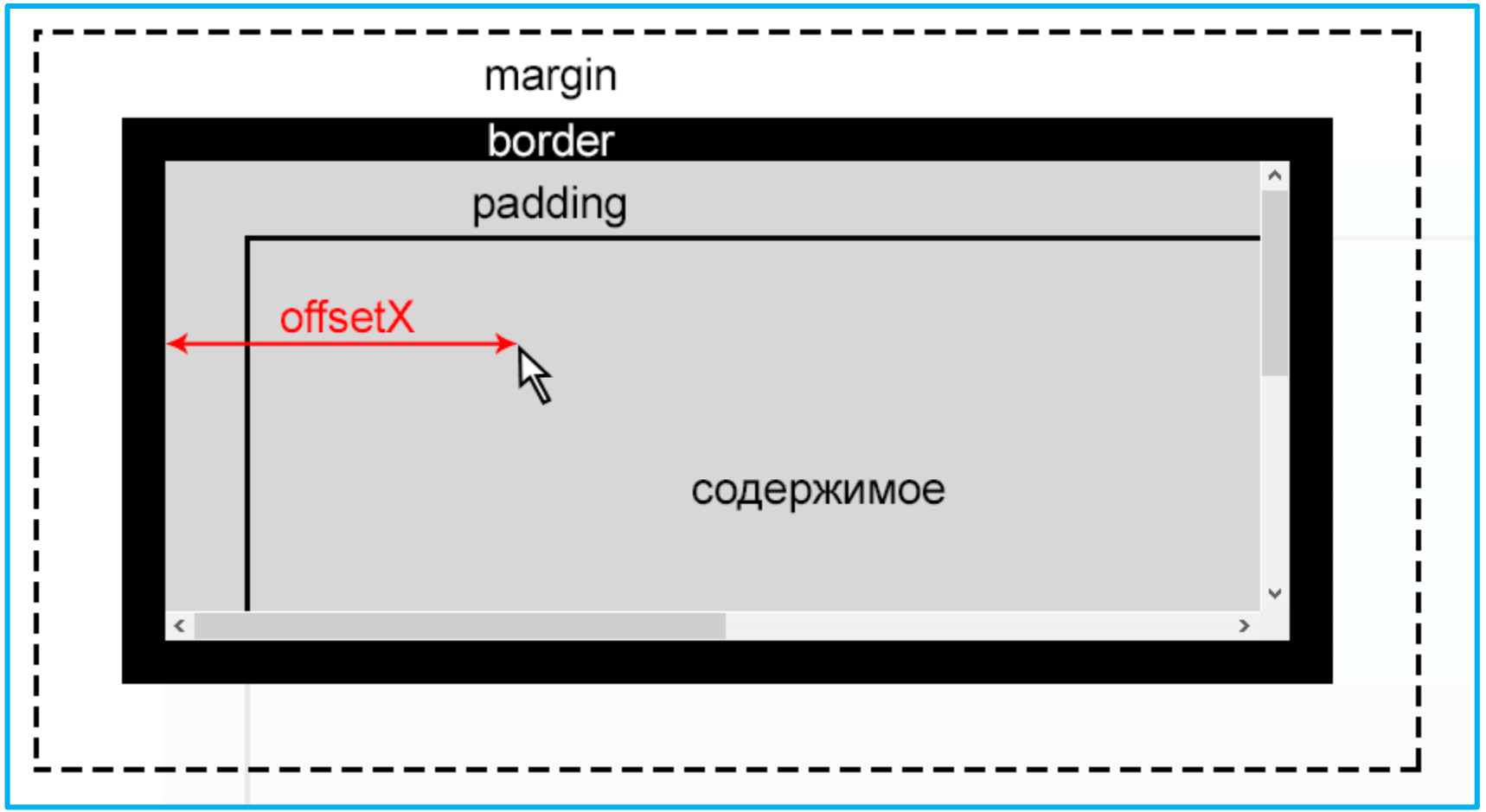
onkeypress

Позиция курсора мыши в объекте события



http://javascript.ru/tutorial/events/properties#koordinaty-myshi:-clientx-y-pagex-y

Позиция курсора мыши в объекте события



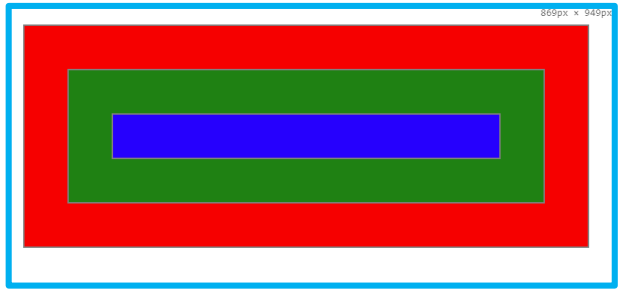
http://javascript.ru/tutorial/events/properties#koordinaty-myshi:-clientx-y-pagex-y

«Всплытие» и «Перехват» событий

«Всплытие» и «Перехват» событий

Воспользуйтесь заготовкой:

./source/ex01.html



```
14      <script>
15          window.onload = function(){
16
17              var divs = document.querySelectorAll("div");
18
19              divs.forEach(function(tag){
20                  tag.addEventListener("click", function(e){
21                      console.log(this.style.backgroundColor);
22                  });
23              });
24
25
26          }
27      </script>
```

?!?

Что мы увидим в консоли после клика по синему блоку?

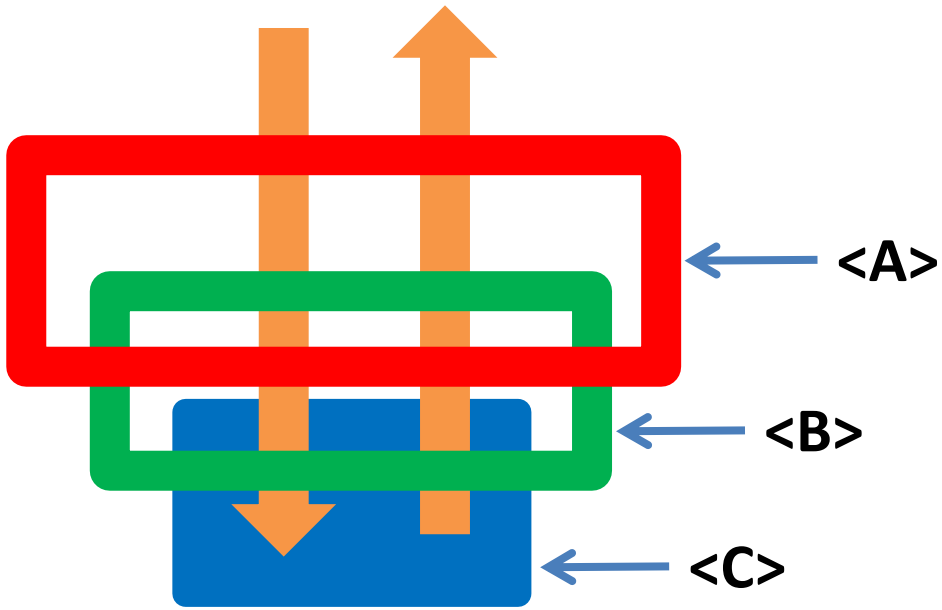
e.target

*Свойство **.target** (объекта события) содержит ссылку на объект инициатор события, т.е. например тот элемент по которому произошел клик.*

```
14      <script>
15          window.onload = function(){
16
17              var divs = document.querySelectorAll("div");
18              divs.forEach(function(tag){
19                  tag.addEventListener("click", function(e){
20                      console.log(this.style.backgroundColor + ", Target: " +
                        e.target.style.backgroundColor);
21                  }, false);
22              });
23
24
25
26          }
27      </script>
```

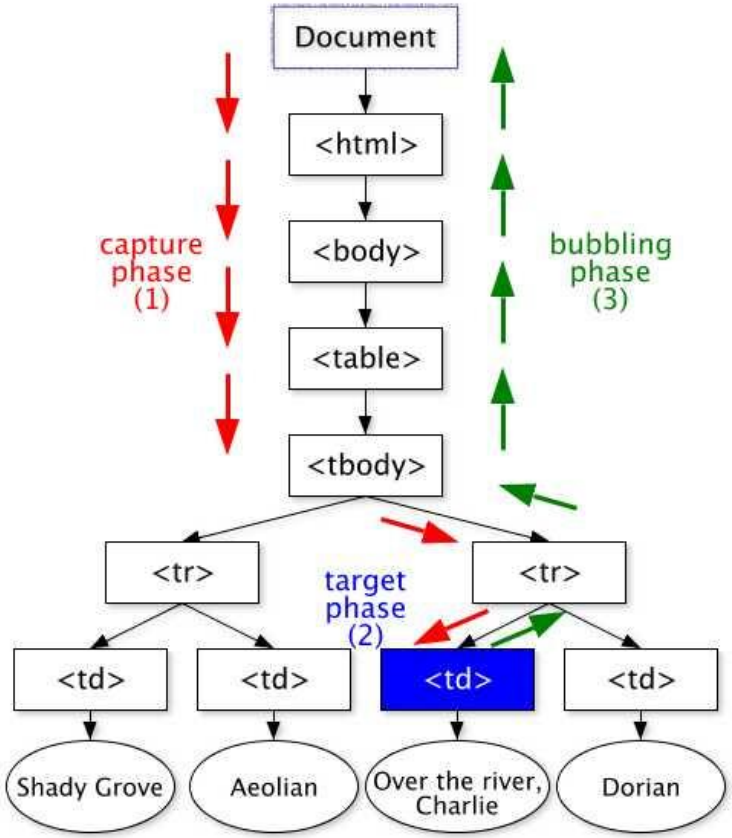
```
blue, Target: blue
green, Target: blue
red, Target: blue
```

«Всплытие» и «Перехват» событий



*При наступлении события обработчики сначала срабатывают на самом «верхнем» элементе, постепенно спускаясь к «цели» события (этап: **перехвата**), а потом обратно поднимается к самому «верхнему» элементу (этап: **всплытия**). По сути обработчик события может быть вызван два раза для каждого из тегов.*

«Всплытие» и «Перехват» событий



e.eventPhase *– в объекте с информацией о событии содержит информацию о фазе обработки события.*

«Всплытие» и «Перехват» событий

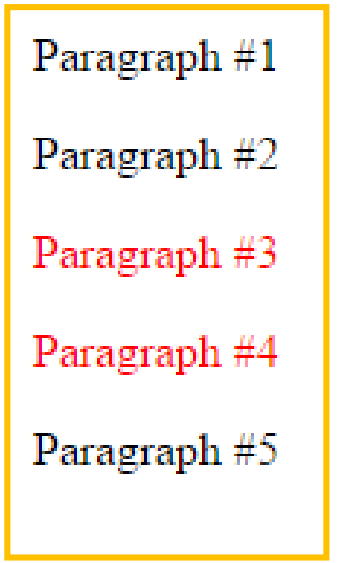
```
14    <script>
15        window.onload = function(){
16
17            var divs = document.querySelectorAll("div");
18            divs.forEach(function(tag){
19                tag.addEventListener("click", function(e){
20                    console.log("UP " + this.style.backgroundColor + ",
                      Target: " + e.target.style.backgroundColor + ", Phase" +
                       e.eventPhase);
21                }, false);
22            });
23
24            divs.forEach(function(tag){
25                tag.addEventListener("click", function(e){
26                    console.log("Down " + this.style.backgroundColor + ",
                      Target: " + e.target.style.backgroundColor + ", Phase" +
                       e.eventPhase);
27                }, true);
28            });
29
30        }
31    </script>
```

Функция обработчик события может обрабатывать события как на этапе всплытия ***.addEventListener("click", func())*** *так и на этапе перехвата* ***.addEventListener("click", func(), true)****.*

https://developer.mozilla.org/ru/docs/Web/API/Event/eventPhase

Зачем нужно всплытие?

```html
<html>
    <head>
        <script>
            function handler(e){
                e.target.style.color = "red";
            }

            window.onload = function(){
                document.querySelector("div").onclick = handler;
            }
        </script>
    </head>
    <body>
        <div>
            <p>Paragraph #1</p>
            <p>Paragraph #2</p>
            <p>Paragraph #3</p>
            <p>Paragraph #4</p>
            <p>Paragraph #5</p>
        </div>
    </body>
</html>
```



Родительский элемент может обрабатывать событие за всех потомков.

Зачем нужно всплытие?

Разные типы элементов – разные события, но….

```
1  <html>
2      <head>
3          <script>
4              function on_key(){ console.log("key"); }
5          </script>
6      </head>
7      <body onkeypress="on_key()">
8          <p>
9          Duis neque mi, rhoncus in urna quis, congue pr
           molestie et dolor eu eleifend. Aliquam sed jus
           Suspendisse non efficitur justo. Donec feugiat
           tincidunt non vitae lorem. Mauris vitae conseq
10         </p>
11     </body>
12 </html>
```

Когда элементов ввода на странице нет, но нужно получать информацию с клавиатуры.

Всплытие/Перехват можно остановить

```
14      <script>
15          window.onload = function(){
16
17              var divs = document.querySelectorAll("div");
18              divs.forEach(function(tag){
19                  tag.addEventListener("click", function(e){
20                      console.log(this.style.backgroundColor + ", Target: " +
                        e.target.style.backgroundColor);
21                      e.stopPropagation();
22                  }, false);
23              });
24
25          }
26      </script>
```

.stopPropagation() *– останавливает всплытие событий.*

Действие по умолчанию

Действия по умолчанию

У некоторых элементов есть встроенная реакция на событие, или по другому действие по умолчанию.

Например:

1. *Для ссылок действие по умолчанию переход на другую страницу;*
2. *Для кнопок внутри формы действие по умолчанию – отправить форму на сервер;*
3. *Двойной клик по тексту – выделяет его фрагмент.*

и т.д.

Отмена действия по умолчанию

```
1  <html>
2      <head>
3          <script>
4              function handler(e){
5                  e.preventDefault();
6              }
7
8              window.onload = function(){
9                  document.querySelector("a").onclick = handler;
10             }
11         </script>
12     </head>
13     <body>
14         <a href="http://www.itc.ua">This is a link to ITC.ua</a>
15     </body>
16 </html>
```

.***preventDefault()*** *– (метод объекта с информацией о событии) отменяет действие по умолчанию (если такое предусмотрено).*

Не путайте!

.preventDefault() *– отменяет действие по умолчанию (как то переход по ссылке, отправка формы и т.д.).*

.stopPropagation() *– останавливает всплытие события, т.е. после вызова этой функции элементы-родители уже не получат уведомление о событии.*

Немного практики #1

События на практике

Воспользуемся заготовкой: **./source/ex02.html**

Реализуем функцию установки метки на карту по клику

События на практике

```
25    <script>
26
27        window.onload = function(){
28            var map = document.querySelector("div");
29            map.onclick = function(e){
30                var tag = document.createElement("span");
31                tag.style.position  = "absolute";
32                tag.style.top       = e.offsetY - 16 + "px";
33                tag.style.left      = e.offsetX - 16 + "px";
34                tag.className       = "fa fa-2x fa-bullseye";
35                tag.style.color     = "red";
36                map.appendChild(tag);
37                console.dir(e);
38            }
39        }
40
41    </script>
```

Реализуем функцию установки метки на карту по клику.

Немного практики #2

События и математика

Воспользуйтесь заготовкой **./source/ex03.html**

Сделаем интенсивность цвета на изображении в зависимости от того как близко курсор к центру изображения.

Демо: **./source/ex03_demo.html**

События и математика

```
<script>
    function dist(x1, y1, x2, y2){
        return Math.sqrt(Math.pow(x2-x1, 2)+Math.pow(y2-y1, 2));
    }

    window.onload = function(){
        var image       = document.querySelector("img");
        var center_x        = image.width / 2;
        var center_y        = image.height / 2;
        var full_distance   = center_x < center_y ? center_x : center_y;

        image.onmousemove = function(e){
            var distance    = dist(e.offsetX, e.offsetY, center_x, center_y);
            distance = distance / full_distance;
            distance = Math.floor((distance > 1 ? 1 : distance)* 100);

            image.style.webkitFilter = "blur(" + distance/10 +"px) grayscale(" + distance + "%)";
            image.style.filter = "blur(" + distance/10 +"px) grayscale(" + distance + "%)";
        }
    }
</script>
```

Сделаем интенсивность цвета на изображении в зависимости от того как близко курсор к центру изображения.

Немного практики #3

Воспользуйтесь заготовкой **./source/ex04.html**

Сделаем эффект увеличения блока на котором находиться курсор, с добавлением элементов (в котором цена показывается с 20% скидкой) (и возвратом к первоначальному состоянию при уходе курсора с элемента): **./source/ex04_demo.html**).

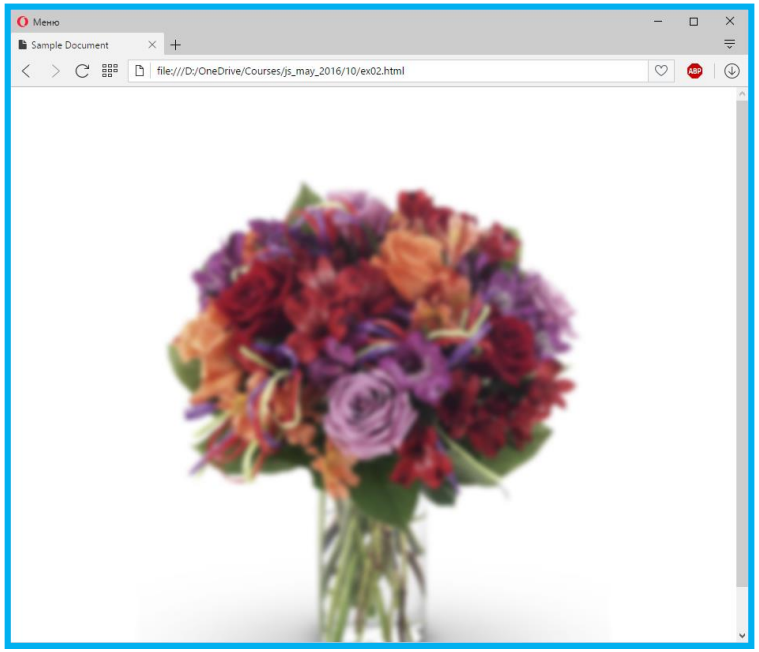
```html
<script>

window.onload = function(){

    var lis = document.querySelectorAll("ul > li");

    lis.forEach(function(element){

        element.addEventListener("mouseenter", function(e){

            var span = e.target.querySelector("span");
            var old_price = parseInt(span.innerHTML);
            span.style.textDecoration = "line-through";

            var btn = document.createElement("button");
            btn.className = "btn btn-success";
            btn.innerHTML = "Buy NOW!!!";

            e.target.appendChild(btn);

            var red_span = document.createElement("span");
            red_span.className = "badge badge-danger badge-pill";
            red_span.innerHTML = (old_price * 0.8).toFixed(0) + "$";
            red_span.style.transform = "scale(2)";
            e.target.insertBefore(red_span, btn);
        });

        element.addEventListener("mouseleave", function(e){
            e.target.querySelector("span").style.textDecoration = "none";
            e.target.querySelector("span + span").remove();
            e.target.querySelector("button").remove();
        });

    });

};

</script>
```

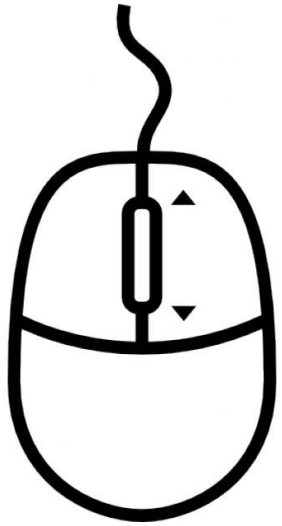
```html
<style>
    .container, h1 {
        margin-top: 10px;
    }

    h1 {
        margin-bottom: 20px;
    }

    li {
        transition: 0.5s;
        z-index: 100;
    }

    li:hover {
        z-index: 200;
        border-radius: 5px;
        background-color: #dbfffe;
        transform: scale(2);
    }
</style>
```

Сделаем эффект увеличения блока на котором находиться курсор, с добавлением элементов (и возвратом к первоначальному состоянию при уходе курсора с элемента).

Домашнее задание
/сделать

Домашнее задание #G.1





Воспользуйтесь заготовкой **./homework/hw_g1.html**

Необходимо сделать так, чтобы размытие и интенсивность цвета регулировалась при помощи колёсика мыши (например: **./homework/hw_g1_demo.html***).*

https://learn.javascript.ru/mousewheel