DOM events model: cross-browser support, mobile specific, bubbling, capturing, preventDefault.

Существует несколько способов навешивать обработчики событий на элемент:

***DOM Level 0***

<button onclick="alert('hello!');"> Say Hello! </button>

Такой способ не рекомендуется использовать, так как в нем перемешивается html и js и такой метод лишает приложение гибкости.

***DOM Level 1***

document.getElementById('myButton').onclick = function(){

alert('Hello!');

}

Недостаток такого метода в том, что он позволяет навешивать всего 1 обработчик на 1 элемент

***DOM Level 2 – Event listeners***

Наилучший способ:

element.addEventListener(<event-name>, <callback>, <use-capture>);

*<event-name (string)>* – название события (click, mousedown, touchstart, transitionEnd, etc.)

*<callback (function)>* функция, которая вызовется при срабатывании события. В параметры передается объект-событие, которое в своих свойствах содержит данные о событии

*<use-capture (boolean)>* - нужно ли вызывать колбек функцию на фазе capture.

По возможности нужно использовать делегирование событий (вместо того чтоб навешивать обработчик на все дочерние элементы, навесить 1 на их родителя).

var list = document.querySelector('ul');

list.addEventListener('click', function(event) {

var target = event.target;

while (target.tagName !== 'LI') {

target = target.parentNode;

if (target === list) return;

}

// Do stuff here

});

Хорошей практикой является удаление обработчиков, после того как они выполнят свое предназначение

element.removeEventListener(<event-name>, <callback>, <use-capture>);

### The Event Object

### Event Object создается при первом срабатывании события и распространяется по DOM следуя распространению события.

### Этот объект содержит такие свойства:

*type (string)* – название события;

*target (node)* – элемент на котором сработало событие ;

*currentTarget (node)* – элемент, на котором событие срабатывает в ходе распространения;

*bubbles (boolean)* – определяет находится ли распространение на стадии bubbling;

*preventDefault (function*) – предотвращает поведение браузера по умолчанию;

*stopPropagation (function)* – предотвращает срабатывание всех колбеков следующих далее по цепочке всплытия. (не предотвращает на данном элементе);

*stopImmediatePropagation (function)* предотвращает срабатывание текущего и всех колбеков следующих далее по цепочке всплытия;

*cancelable (boolean)* – определяет можно ли предотвратить поведение по умолчанию с помощью [event.preventDefault](http://www.smashingmagazine.com/2013/11/12/an-introduction-to-dom-events/" \l "preventing-default)();

*defaultPrevented (boolean)* – было ли вызвано preventDefault  на объекте;

*isTrusted (boolean)* – определяет было ли событие вызвано самим браузером, а не через JavaScript;

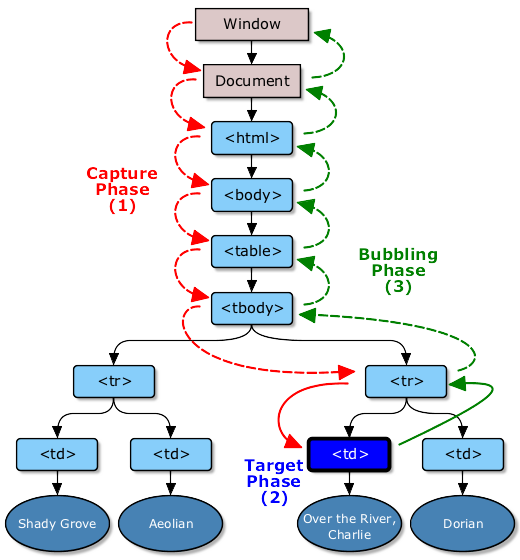
*eventPhase (number)* none (0), capture (1), target (2) or bubbling (3);

*timestamp (number)* – время, когда произошло событие.

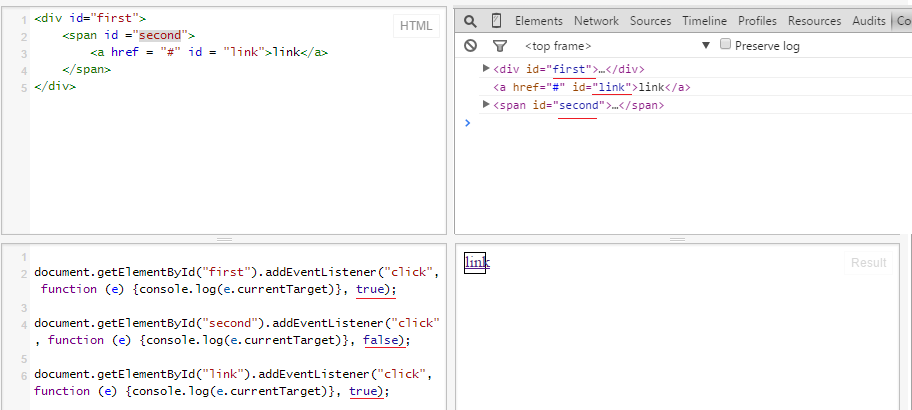
Поведение браузера, когда на каком-то элементе срабатывает событие.

1. Event setup
2. Calculate parent node path
3. Execute Capture Phase handlers
4. Execute DOM Level 1 event handler
5. Execute Bubble Phase handlers
6. Execute default browser behavior

2 фазы распространения события по документу: capture и bubble.



Основное назначение стадии capture – выстроить путь к элементу, на котором произошло событие, по которому в дальнейшем будет проходить стадия bubbling





### Stopping Propagation

stopPropagation – метод объекта события, который позволяет прервать цепочку распространения события (на текущем элементе событие сработает)

child.addEventListener('click', function(event) {

event.stopPropagation();

});

parent.addEventListener('click', function(event) {

// If the child element is clicked

// this callback will not fire

});

Для того, чтоб предотвратить вызов колбека и на текущем элементе следует использовать event.stopImmediatePropagation()

child.addEventListener('click', function(event) {

event.stopImmediatePropagation();

});

child.addEventListener('click', function(event) {

// If the child element is clicked

// this callback will not fire

});

**Prevent The Browser’s Default Behavior**

anchor.addEventListener('click', function(event) {

event.preventDefault();

// Do our own thing

});

В основном используется для того, чтоб предотвратить переход по ссылке при клике на нее, но с помощью этого метода можно предотвращать и такое стандартное поведение браузера как, например, скрол страницы при нажатии пробела.

**Cross Browser Event Handling in JavaScript**

Кросс браузерный Event Handler

|  |  |
| --- | --- |
| t | var EventHandler = {      bind:function(el, ev, fn){          if(window.addEventListener){ // modern browsers including IE9+              el.addEventListener(ev, fn, false);          } else if(window.attachEvent) { // IE8 and below              el.attachEvent('on' + ev, fn);          } else {              el['on' + ev] = fn;          }      },        unbind:function(el, ev, fn){          if(window.removeEventListener){              el.removeEventListener(ev, fn, false);          } else if(window.detachEvent) {              el.detachEvent('on' + ev, fn);          } else {              elem['on' + ev] = null;          }      },        stop:function(ev) {          var e = ev || window.event;          e.cancelBubble = true;          if (e.stopPropagation) e.stopPropagation();      }  } |

**Всевозможные события и их поддержка браузерами** <http://www.quirksmode.org/dom/events/>

(blur, change, click, contextmenu, copy, cut, dblclick, error, focus, focusin, focusout, hashchange, keydown, keypress, keyup, load, mousedown, mouseenter, mouseleave, mousemove, mouseout, paste, resize, scroll, select, submit, unload, wheel)

**События для mobile**

https://api.jquerymobile.com/category/events/

Links

<http://blog.bitovi.com/a-crash-course-in-how-dom-events-work/>

<http://www.smashingmagazine.com/2013/11/12/an-introduction-to-dom-events/>

<https://api.jquerymobile.com/category/events/>

<http://www.quirksmode.org/mobile/>

<http://www.anujgakhar.com/2013/05/22/cross-browser-event-handling-in-javascript/>