

# События / Events



JavaScript  
Courses

[www.courses.dp.ua](http://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

**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 уведомляет 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

*Вариантов событий много, задача программиста выбрать нужное*

## HTML DOM Events

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

### Mouse Events

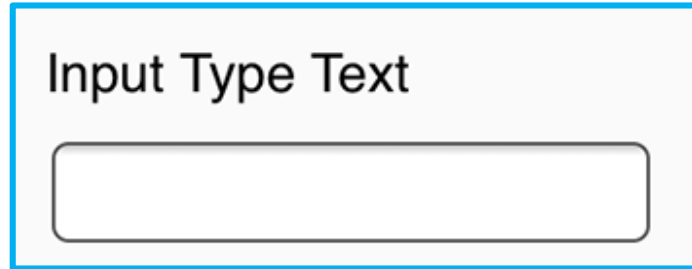
Event	Description	DOM
<a href="#">onclick</a>	The event occurs when the user clicks on an element	2
<a href="#">oncontextmenu</a>	The event occurs when the user right-clicks on an element to open a context menu	3
<a href="#">ondblclick</a>	The event occurs when the user double-clicks on an element	2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

### Keyboard Events

Event	Description	DOM
<a href="#">onkeydown</a>	The event occurs when the user is pressing a key	2
<a href="#">onkeypress</a>	The event occurs when the user presses a key	2
<a href="#">onkeyup</a>	The event occurs when the user releases a key	2

Подробнее: [http://www.w3schools.com/jsref/dom\\_obj\\_event.asp](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 указать браузеру какую функцию и когда вызывать?

```
2
3  <h1 onclick="eventListener()">Some Content</h1>
4
5  <script>
6
7
8      function eventListener(){
9          console.log('Click detected!');
10     }
11
12 </script>
13
```

*Через соответствующие атрибуты тегов*

# Как указать браузеру какую функцию и когда вызывать?

```
2
3   <h1>Some Content</h1>
4
5   <script>
6
7       let h1Tag = document.querySelector('h1');
8
9       h1Tag.onclick = function(){
10           console.log('Click detected!');
11       }
12
13   </script>
14
```

*Через свойства объектов входящих в дерево документа*

Подробнее: <https://learn.javascript.ru/introduction-browser-events>



# Как указать браузеру какую функцию и когда вызывать?

```
2
3 <h1>Some Content</h1>
4
5 <script>
6
7     let h1Tag = document.querySelector('h1');
8
9     function eventListener_1(){
10         console.log("I'm eventListener_1");
11     }
12
13     function eventListener_2(){
14         console.log("I'm eventListener_2");
15     }
16
17     h1Tag.addEventListener('click', eventListener_1);
18     h1Tag.addEventListener('click', eventListener_2);
19
20     //h1Tag.removeEventListener('click', eventListener_1);
21     //h1Tag.removeEventListener('click', eventListener_2);
22
23 </script>
24
```

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

# Вспоминаем **this**

```
2
3 <h1>Some Content</h1>
4 <p>Some P tag</p>
5
6 √ <script>
7
8     let h1Tag = document.querySelector('h1');
9     let pTag = document.querySelector('p');
10
11 √     function eventListener(){
12         console.log('this in eventListener:', this);
13     }
14
15     h1Tag.addEventListener('click', eventListener);
16     pTag.addEventListener('click', eventListener);
17
18
19 </script>
20
```

Функция обработчик становится 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 window.onload

```
3
4  ✓ window.addEventListener('load', function(e){
5      |     console.log("Event Window.onLoad", e);
6      | });
7
8  ✓ document.addEventListener('DOMContentLoaded', function(e){
9      |     console.log("Event Document.DOMContentLoaded", e);
10     | });
11
```

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

# Событие document.DOMContentLoaded

```
3
4  ✓ window.addEventListener('load', function(e){
5      |     console.log("Event Window.onLoad", e);
6      | });
7
8  ✓ document.addEventListener('DOMContentLoaded', function(e){
9      |     console.log("Event Document.DOMContentLoaded", e);
10     | });
11
```

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).

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.*

# Информация о событии

```
2
3  <h1>Some Content</h1>
4
5  <script>
6
7      let h1Tag = document.querySelector('h1');
8
9      function eventListener(e){
10         console.log('Event info:', e);
11     }
12
13     h1Tag.addEventListener('click', eventListener);
14
15 </script>
16
```

*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.*

Подробнее: <https://learn.javascript.ru/mouse-events-basics>

Подробнее: <https://learn.javascript.ru/keyboard-events>

```

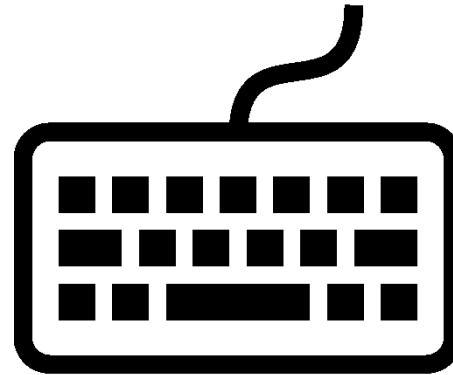
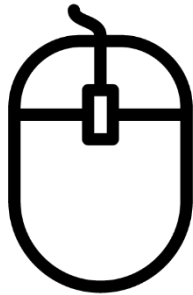
▼ MouseEvent ⓘ
  altKey: false
  bubbles: true
  button: 0
  buttons: 0
  cancelBubble: false
  cancelable: true
  clientX: 83
  clientY: 17
  ctrlKey: false
  currentTarget: null
  defaultPrevented: false
  detail: 1
  eventPhase: 0
  fromElement: null
  isTrusted: true
  isTrusted: true
  layerX: 83
  layerY: 17
  metaKey: false
  movementX: 0
  movementY: 0
  offsetX: 75
  offsetY: 9
  pageX: 83
  pageY: 17
  ▶ path: Array[5]
    relatedTarget: null
    returnValue: true
    screenX: 2003
    screenY: 102
    shiftKey: false
  ▶ sourceCapabilities: InputDeviceCapabilities
  ▶ srcElement: p
  ▶ target: p
    timeStamp: 1314.7900000000002
  ▶ toElement: p
    type: "click"
  ▶ view: Window
    which: 1
    x: 83
    y: 17
  ▶ __proto__: UIEvent

```

# Информация о событии

*Разные события – разные объекты с информацией о них.*

*<= MouseEvent*



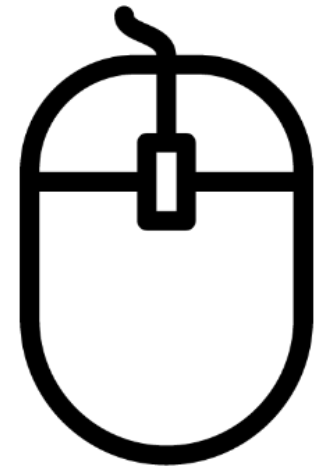
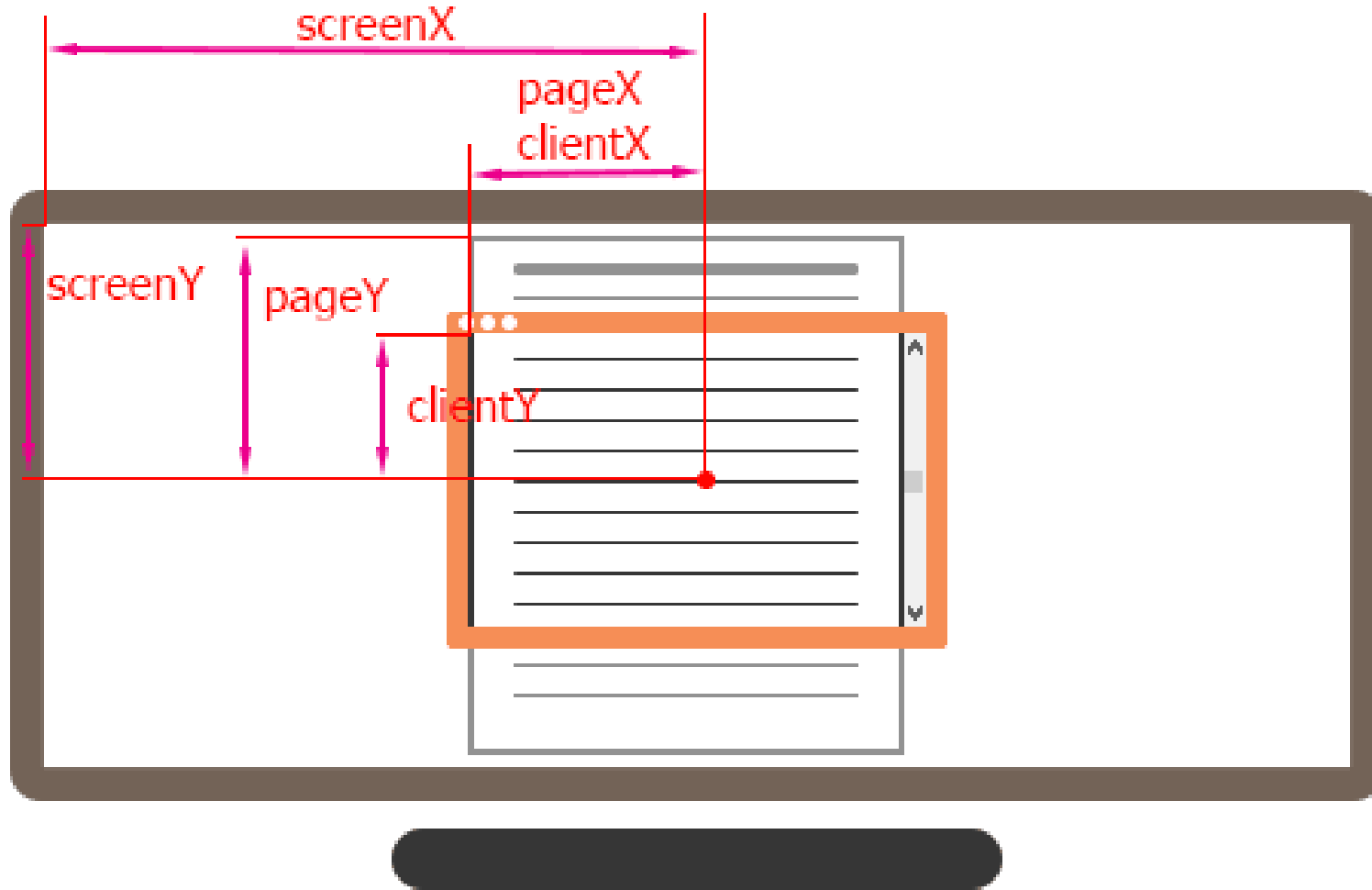
*KeyboardEvent =>*

```

▼ KeyboardEvent ⓘ
  altKey: false
  bubbles: true
  cancelBubble: false
  cancelable: true
  charCode: 97
  code: "KeyA"
  ctrlKey: false
  currentTarget: null
  defaultPrevented: false
  detail: 0
  eventPhase: 0
  isTrusted: true
  isTrusted: true
  keyCode: 97
  keyIdentifier: "U+0041"
  keyLocation: 0
  location: 0
  metaKey: false
  ▶ path: Array[5]
    repeat: false
    returnValue: true
    shiftKey: false
  ▶ sourceCapabilities: InputDeviceCapabilities
  ▶ srcElement: input
  ▶ target: input
    timeStamp: 2079.225
    type: "keypress"
  ▶ view: Window
    which: 97
  ▶ __proto__: UIEvent

```

# Позиция курсора мыши в объекте MouseEvent



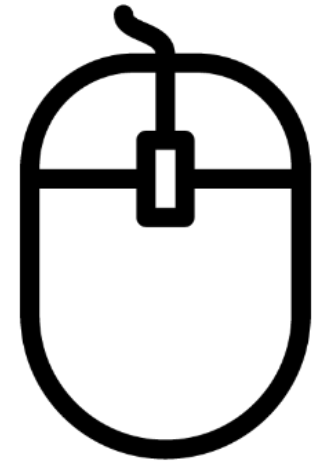
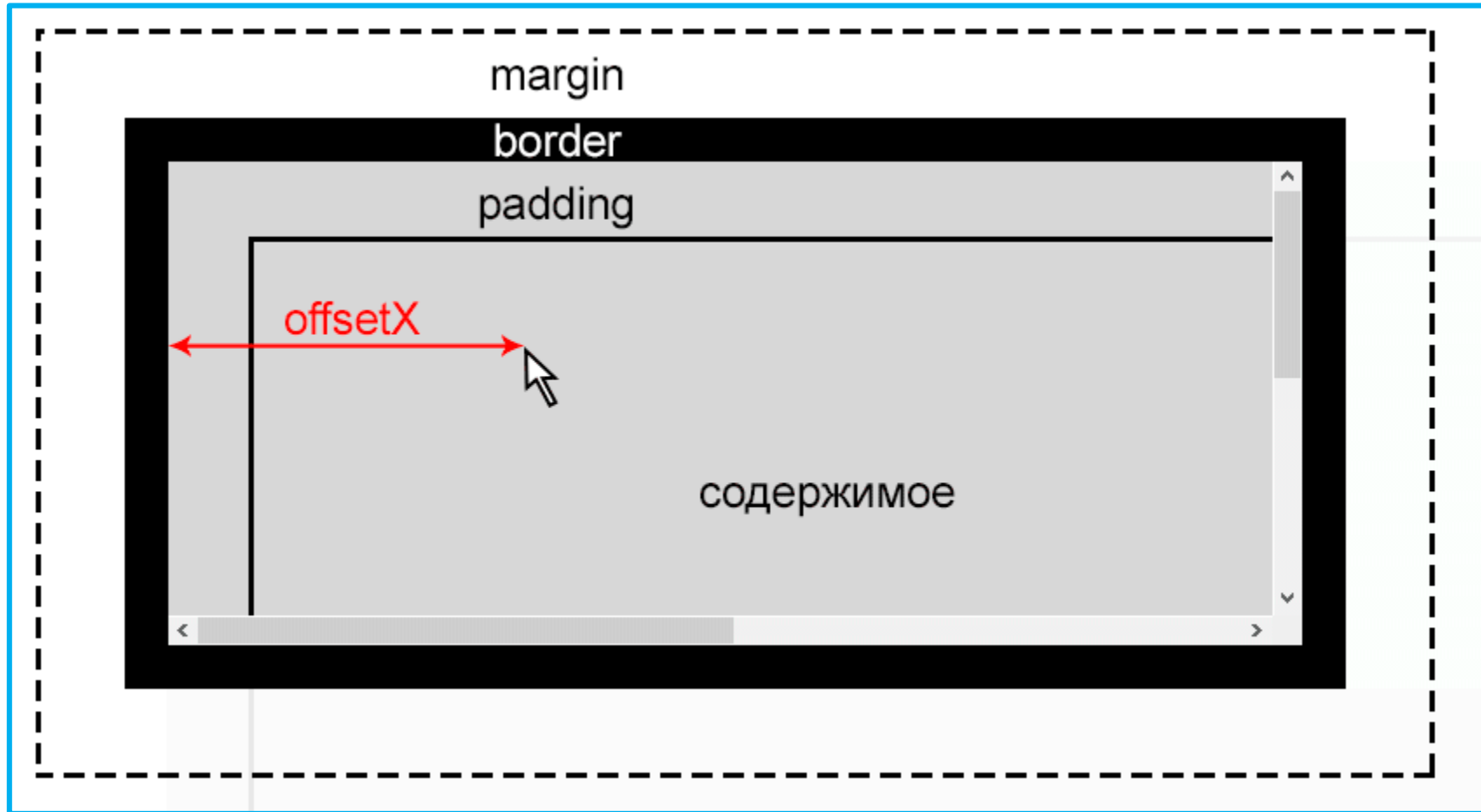
Подробнее:

[https://www.w3schools.com/jsref/event\\_screenx.asp](https://www.w3schools.com/jsref/event_screenx.asp)

[https://www.w3schools.com/jsref/event\\_pagex.asp](https://www.w3schools.com/jsref/event_pagex.asp)

[https://www.w3schools.com/jsref/event\\_clientx.asp](https://www.w3schools.com/jsref/event_clientx.asp)

# Позиция курсора мыши в объекте MouseEvent



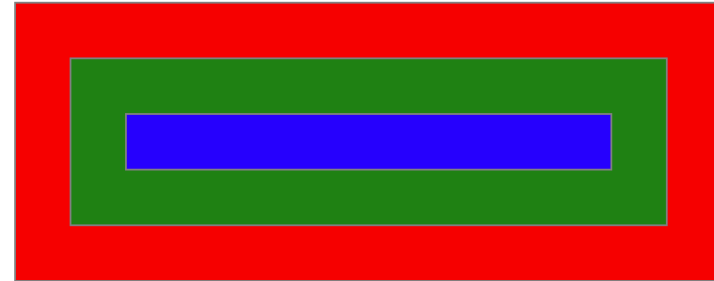
Подробнее: [https://www.w3schools.com/jsref/event\\_offsetx.asp](https://www.w3schools.com/jsref/event_offsetx.asp)

«Всплытие» и «Перехват» событий

# «Всплытие» и «Перехват» событий

Воспользуйтесь заготовкой:

[./source/example\\_1](#)



```
17
18     function eventListener(e){
19         |     console.log(this.id);
20     }
21
22     document.addEventListener('DOMContentLoaded', () => {
23         |
24         document.querySelectorAll('div').forEach(item => {
25             |     item.addEventListener('click', eventListener);
26         });
27     });
28
29
```

Что мы увидим в консоли после клика по синему блоку?

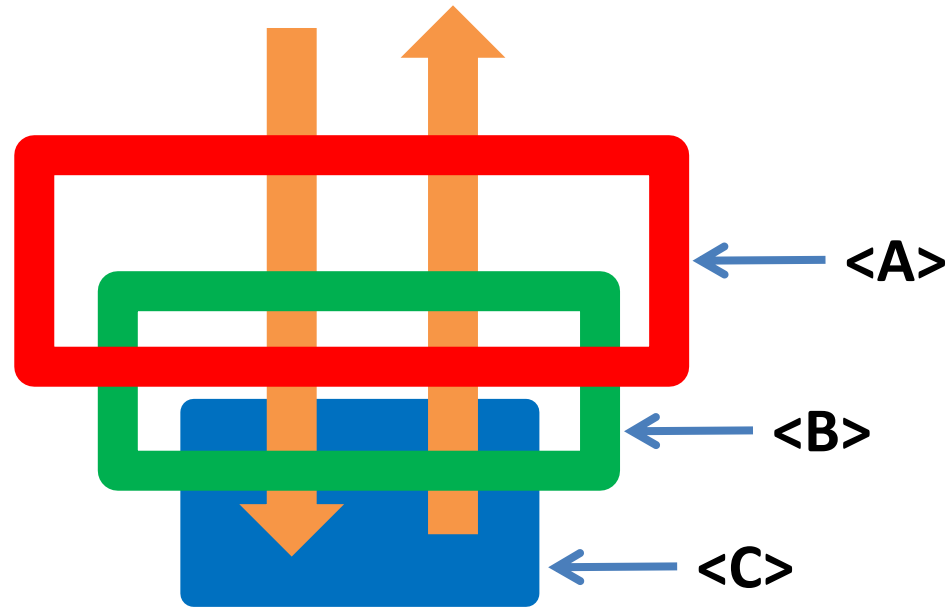
# e.target

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

```
17
18     function eventListener(e){
19         |     console.log(this.id, e.target.id);
20     }
21
22     document.addEventListener('DOMContentLoaded', () => {
23         |
24         document.querySelectorAll('div').forEach(item => {
25             |     item.addEventListener('click', eventListener);
26         });
27     });
28
29
```

Подробнее: <https://learn.javascript.ru/bubbling-and-capturing>

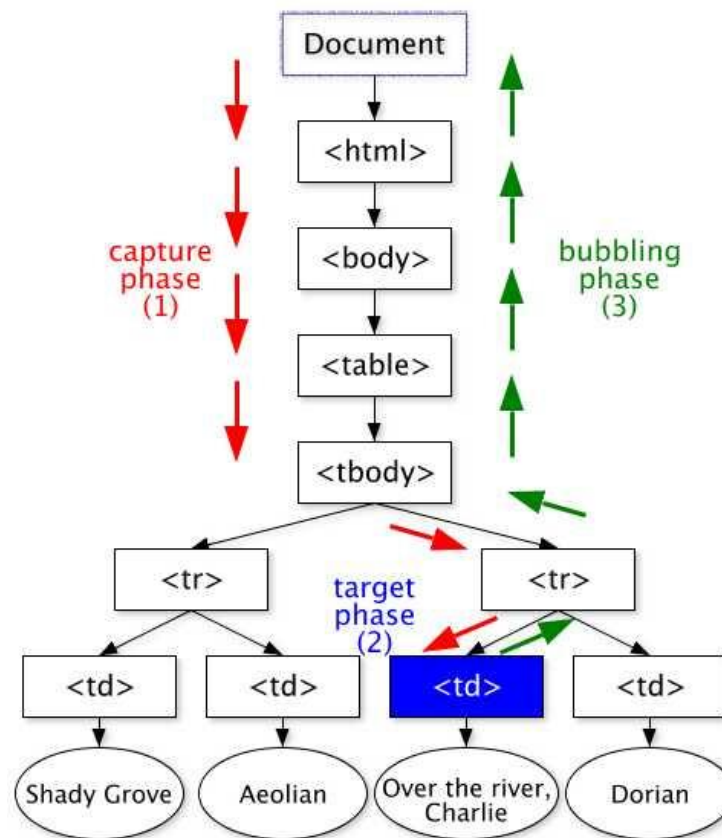
# «Всплытие» и «Перехват» событий



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

```
17
18     function eventListener(e){
19         console.log("Phase:", e.eventPhase, this.id, e.target.id);
20     }
21
22     document.addEventListener('DOMContentLoaded', () => {
23
24         document.querySelectorAll('div').forEach(item => {
25             item.addEventListener('click', eventListener, true);
26         });
27
28         document.querySelectorAll('div').forEach(item => {
29             item.addEventListener('click', eventListener, false);
30         });
31     });
32
33
```

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

## Всплытие/Перехват можно остановить

```
17
18     function eventListener(e){
19         console.log(this.id, e.target.id);
20         e.stopPropagation();
21     }
22
23     document.addEventListener('DOMContentLoaded', () => {
24
25
26         document.querySelectorAll('div').forEach(item => {
27             item.addEventListener('click', eventListener, false);
28         });
29     });
30
31
```

***`e.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

```
2
3  <a href="https://itc.ua">Site ITC.ua</a>
4
5  <script>
6
7      let aTag = document.querySelector('a');
8
9      aTag.addEventListener('click', function(e){
10         e.preventDefault();
11     });
12
13 </script>
14
```

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

# Не путайте!

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

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

Немного практики



# События и динамика

## Адресная книга

Поиск

Имя	Телефон	Адрес
Иван	+380675558823	London, United Kingdom

Используйте заготовку:

[./source/example\\_2](#)

Запустите при помощи **serve**

Выведем в  
подготовленную  
разметку данные  
пользователей  
полученные от сервиса  
<https://randomuser.me/>  
И обеспечим  
функционирование  
поиска (динамического)

Домашнее задание  
/узнать




**Небольшое введение в библиотеку jQuery, для лучшего понимания следующего занятия**

**<https://webref.ru/layout/advanced-html-css/jquery>**

Домашнее задание  
/сделать

# Домашнее задание #Н.1

## Адресная книга

Имя 	Телефон	Адрес
Иван	+380675558823	London, United Kingdom

Используйте заготовку:  
[./source/homework](#)

В качестве рабочего примера  
[./source/homework/demo.html](#)

Выведем в подготовленную разметку данные пользователей полученные от сервиса <https://randomuser.me/> Обеспечьте сортировку по имени. После загрузки список выводится в неотсортированном виде. Сортировка начинается только после нажатия кнопки (сортироваться должен именно тот список который был загружен при запуске нашего приложения). Для вашего удобства: <https://randomuser.me/api/?seed=speciallist&results=25> будет давать всегда один и тот же набор пользователей.