

ОСНОВЫ JS (8)

События

Для реакции на действия посетителя и внутреннего взаимодействия скриптов существуют **события**.

Событие – это сигнал от браузера о том, что **что-то** произошло.

Обработка событий (основные)

- События мыши =>(клик, наведение)
- События клавиатуры
(нажатие клавиш)
- События формы
(отправка/ввод/фокус)
- События браузера
(изменение размера окна)
- Страница
(готовность DOM - "ready")

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

События на элементах управления и клавиатурные:

submit – посетитель отправил форму <form>

focus – посетитель фокусируется на элементе, например нажимает на <input>

keydown – когда посетитель нажимает клавишу

keyup – когда посетитель отпускает клавишу

Браузер

`.resize()`

Изменения размеров окна браузера.

`.scroll()`

Обработчик "прокрутки" элементов документа.

Документ

- **DOMContentLoaded** – означает, что все DOM-элементы разметки уже созданы, можно их искать, вешать обработчики, создавать интерфейс, но при этом, возможно, ещё не догрузились какие-то картинки или стили.
- **load** – страница и все ресурсы загружены, используется редко, обычно нет нужды ждать этого момента.

Обработчики

Для того, чтобы скрипт
реагировал на действия
необходимо повесить
"обработчик"
(handler)

Обработчик

HTML

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

HTML + JS

```
<input type="button" id="button" value="Я кнопка.  
Не кликай меня!" />  
<script>  
    button.onclick = function() {  
        //передача по id <==  
        alert( 'Ты меня кликнул... Теперь повтори!' );  
    };  
</script>
```

При вызове обработчика объект события event будет передан ему первым аргументом.

```
function doSomething(event) {
```

```
// event будет содержать объект события  
}
```

```
element.onclick = doSomething;
```

```
////!! Имя функции для обработчика пишется  
БЕЗ ()
```

```
//=> element.onclick = doSomething(); //ОШИБКА
```

Современный подход

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

event

Имя события, например click

handler

Ссылка на функцию, которую надо поставить обработчиком.

phase

True – фаза перехвата.

False – фаза всплытия

Подробности фаз позже!

Особенности addEventListener

```
<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>
```

Плюс на минус дают меч

- `addEventListener` – можно повесить сколько угодно обработчиков.

Для удаления нужно сохранять ссылку и вызывать `removeEventListener` с теми же аргументами

- `on(click)` – перезаписывается при добавлении обработчика

Легче удалять. Старый подход

Всплытие

Основной принцип всплытия:

При наступлении события обработчики *сначала* срабатывают на самом вложенном элементе, затем на его родителе, затем выше и так далее, вверх по цепочке вложенности.

Пример

```
<style>  
  body * {  
    margin: 10px;  
    border: 1px solid blue;  
  }  
</style>
```

```
<form onclick="alert('form')">FORM  
  <div onclick="alert('div')">DIV  
    <p onclick="alert('p')">P</p>  
  </div>  
</form>
```

Всплытие

Всплывают *почти* все события.

Ключевое слово в этой фразе – «почти».

Например, событие **focus** не всплывает.

Немного о «предотвращении стандартного поведения»

Отмена стандартного поведения

`e.preventDefault()` // браузер стоять

Отмена дальнейшей обработки события

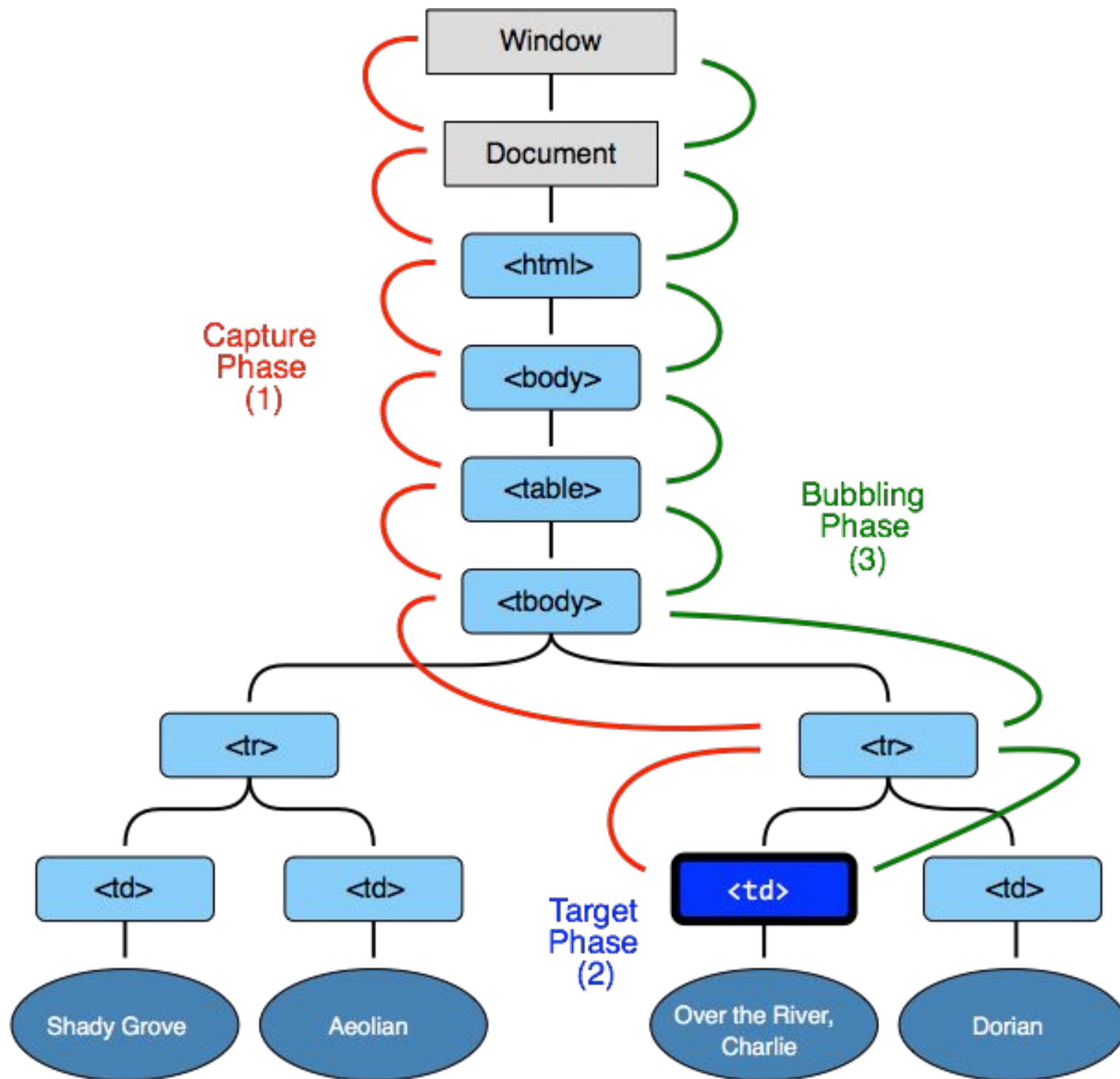
`e.stopPropagation()` // событие не всплывать

```
<form id='myform' action=''><input type='submit'  
  value='Обновить!'></form>
```

```
<script type='text/javascript'>  
alert('page loaded or reloaded');//  
var t=document.getElementById('myform');  
t.addEventListener('click',handler,false)
```

```
function handler(e) {  
  e.preventDefault() // браузер стоять  
  e.stopPropagation() //событие не всплывать  
  console.log(e.target); alert('Checked!');  
  return false // Завершение  
}
```

```
elem.addEventListener('submit', handler,  
false);// Можно и click.. В чем разница?  
</script>
```



Погружение

```
var elems =  
document.querySelectorAll('form,div,p');  
  
for (var i = 0; i < elems.length; i++) {  
    elems[i].addEventListener("click", highlightThis,  
true);  
}  
  
function highlightThis() {  
    this.style.backgroundColor = 'yellow';  
    alert(this.tagName);  
    this.style.backgroundColor = '';  
}
```

Задания

- learn.javascript.ru/introduction-browser-events
Спрятать при клике
Спрятаться
Какие обработчики сработают?
- Написать обработчик используя **this**, который **меняет фон элемента**, по которому прошел клик. **Цвета сменяются поочередно.**
(Не менее 7 цветов в наборе)
Повесить обработчик на **все** выбранные элементы (`<p></div>`). Элементы должны быть вложены друг в друга

Задания

- <https://learn.javascript.ru/event-bubbling>

Изучить интерактивные примеры

Сделать обработчики погружения, которые модифицирует обработчики всплытия при установленной галочке в checkbox.

!!!Вывод только обработчиками всплытия!!!

- По умолчанию:

Привет из (элемент)

- При галочке

(элемент) говорит тебя прощай

Скрипт не должен "ломаться"