DOM API и события

Javascript
&
Browser

Проблемы с браузером

- ункциональность отличается у разных браузеров
- ы не выбираем браузер пользователя
- ы не выбираем версию браузера
- ы не можем менять настройки пользователя

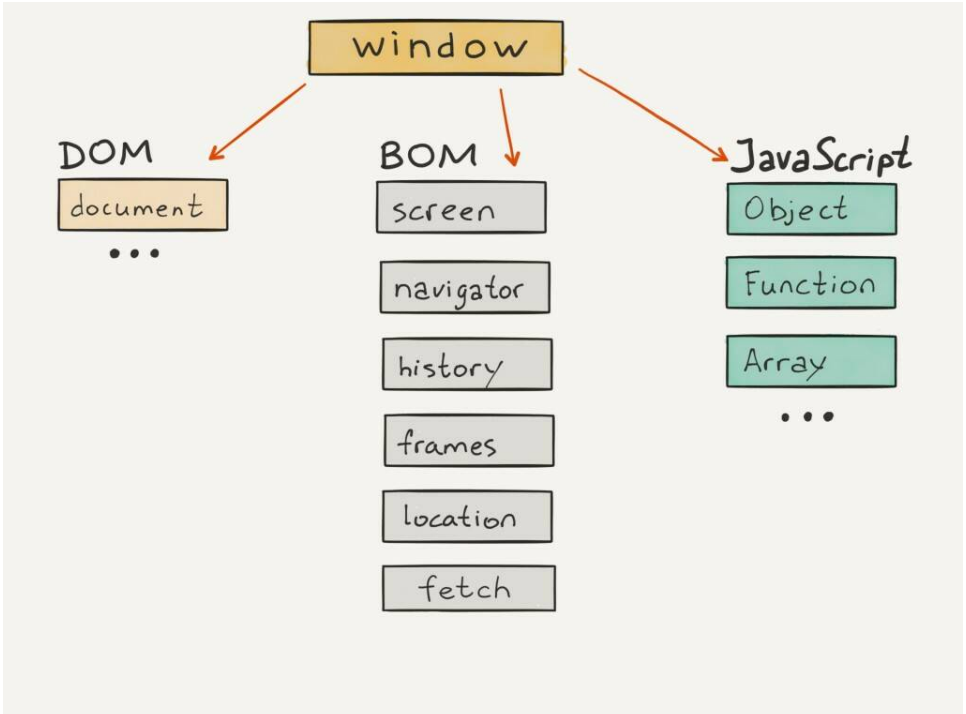
Полифил

олифил – исправляем проблемы совместимости разных
раузеров

роверяйте здесь: caniuse.com

window – глобальный объект

```
window.document === document; // true
```


window
DOM
document
...
BOM
screen
navigator
history
frames
location
fetch
JavaScript
Object
Function
Array
...

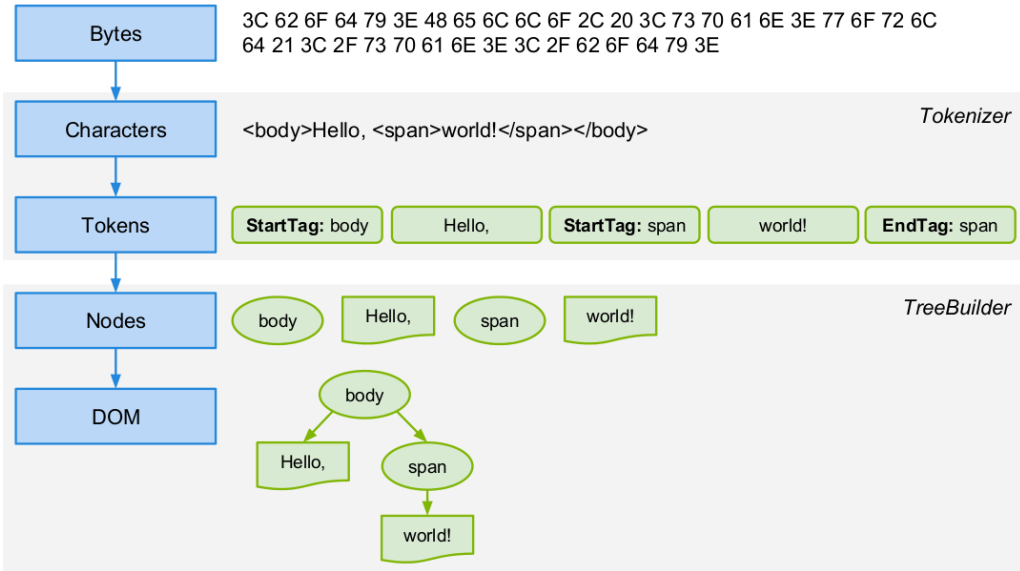
Browser Object Model (BOM)

- бъект navigator: информация о самом браузере
- бъект location: адресная строка и переходы по урлу
- бъект history: история переходов и переходы по истории
- бъект screen: информация об экране пользователя
- ункции alert/prompt/confirm

Document Object Model (DOM)

HTML

```
<!DOCTYPE html>
<html>
<head>

    <meta name="description" content="...">
    <title>CERN</title>
</head>
<body>
    <p>
        <a href="http://home.web.cern.ch/">CERN</a>
        is a European research organization
    </p>
</body>
</html>
```


Bytes
3C 62 6F 64 79 3E 48 65 6C 6C 6F 2C 20 3C 73 70 61 6E 3E 77 6F 72 6C
64 21 3C 2F 73 70 61 6E 3E 3C 2F 62 6F 64 79 3E
Characters
<body>Hello, world!</body>
Tokenizer
Tokens
StartTag: body
Hello,
StartTag: span
world!
EndTag: span
Nodes
body
Hello,
span
world!
TreeBuilder
DOM
body
Hello,
span
world!

Типы узлов

OOM 12 типов узлов, но мы работаем с 4

окумент – объект document

ементы – теги

кст

омментарии

ругие

Поиск элементов

document.getElementById('id')

щет элемент с заданным идентификатором

озращает первый найденный элемент

озращает null, если элемент с указанным ID не найден в кументе

document.getElementById('id')

```
<div id="hello">Hello everybody!</div>

var elem = document.getElementById('hello');

elem === hello; // true
```

getElementsByClassName('class')

щет элементы с соответствующим классом

ть не только у document, но и у любого элемента

зращает коллекцию

getElementsByClassName('class')

```
<div class="hello">
    <div class="greeting">Hello everybody!</div>
    <div class="greeting">Aloha</div>

</div>
<div class="greeting">Привет</div>

var elems = document.getElementsByClassName('greeting'); // 3 элемента

var parent = document.getElementsByClassName('hello')[0];
var elems = parent.getElementsByClassName('greeting'); // 2 элемента
```

Коллекция – array-like object

ожно итерировать

е массив: нет методов map, filter, reduce и т.д.

```
// преобразование в массив
var elemsArrayES6 = Array.from(elems); // > ES6
```

getElementsByTagName('tag')

щет элементы соответствующего тега

ть не только у document, но и у любого элемента

зращает коллекцию

getElementsByTagName('tag')

```
<div>
    <span>Hello everybody!</span>
    <span>Aloha</span>

</div>
<span>Good bye</span>

var elems = document.getElementsByTagName('span'); // 3 элемента

var parent = document.getElementsByTagName('div')[0];
var elems = parent.getElementsByTagName('span'); // 2 элемента
```

querySelectorAll('selector')

озвращает коллекцию по заданному CSS селектору

севдоселекторы тоже работают(:hover, :first-child, и т.д.)

ри невалидном селекторе выбрасывает исключение
NTAX_ERR

querySelectorAll('selector')

```
<div class="container">
    <div>Hello everybody!</div>
    <div>Aloha</div>

</div>
<div class="container">
    <div>Goodbye</div>
    <div>Aloha</div>
</div>

document.querySelectorAll('.container div:first-child');
// NodeList [ <div>Hello everybody!</div>, <div>Goodbye</div> ]
```

querySelector(selector)

озвращает первый найденный элемент по заданному
лектору

ыбрасывает исключение SYNTAX_ERR в случае передачи
валидного селектора

querySelector('selector')

```
<div class="greeting">Goodbye</div>
<div class="container">
    <div class="greeting">Hello everybody!</div>

    <div class="greeting">Aloha</div>
</div>

document.querySelector('.greeting');
// <div class="greeting">Goodbye</div>

document.querySelector('.container .greeting');
// <div class="greeting">Hello everybody!</div>
```

closest(selector)

озвращает ближайший родительский элемент (или сам емент), который соответствует заданному CSS-селектору

озращает null, если нет элемента, который соответствует лектору

ри невалидном селекторе выбрасывает исключение NTAX_ERR

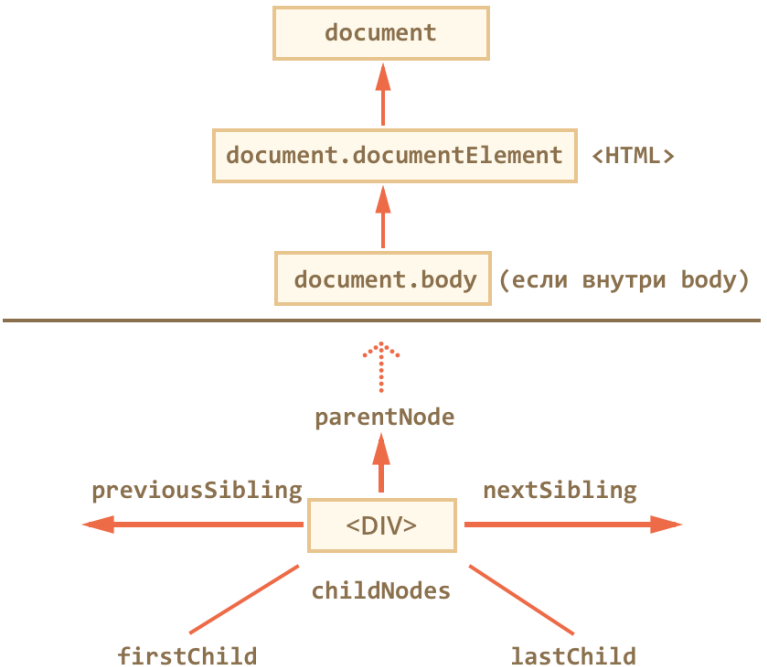
е доступен в IE и старых версиях браузеров

closest(selector)

```
<div id="block" title="Я - блок">
    <a href="#">Я ссылка в никуда</a>
    <a href="http://site.ru">Я ссылка на сайт</a>

    <div>
        <div id="too"></div>
    </div>
</div>

var div = document.querySelector("#too");

div.closest("#block"); //<div id="block" title="Я - блок">
div.closest("div"); //Сам <div id="too">
div.closest("a"); //null
```

Внутреннее устройство поисковых методов

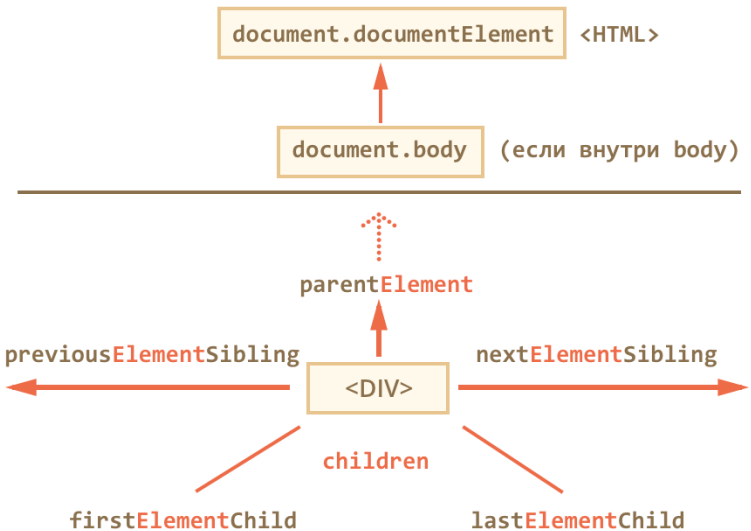
Навигация по узлам



Навигация по узлам

```
<div id="container">
    <div>Hello</div>
    Aloha

    <div>Привет</div>
</div>

var container = document.getElementById('container');

console.log(container.childNodes); // <div>Hello</div> Aloha <div>Привет</div>
console.log(container.firstChild); // <div>Hello</div>
console.log(container.lastChild); // <div>Привет</div>
console.log(container.firstChild.parentNode); // <div id="container">...</div>
console.log(container.firstChild.nextSibling); // Aloha
console.log(container.lastChild.previousSibling); // Aloha
```

Навигация по элементам



Навигация по элементам

```
<div id="container">
    hi
    <div>Hello</div>

    Aloha
    <div>Привет</div>
</div>

var container = document.getElementById('container');

console.log(container.children); // <div>Hello</div> <div>Привет</div>
console.log(container.firstElementChild); // <div>Hello</div>
console.log(container.lastElementChild); // <div>Привет</div>
console.log(container.firstElementChild.parenElement); // <div id="container">...
console.log(container.firstElementChild.nextElementSibling); // <div>Привет</div>
console.log(container.lastElementChild.previousElementSibling); // <div>Hello</di
```

Свойства узлов

Тип узла – nodeType

редставлено числом

его типов 12, но используем 2

- ELEMENT_NODE

- TEXT_NODE

стальные

Тип узла – nodeType

```
<div id="container">
    <span>Hello everybody!</span>


    Aloha!
</div>

var container = document.getElementById('container');
console.log(container.firstChild.nodeType); // 1
console.log(container.lastChild.nodeType); // 3
```

node.tagName

озвращает HTML-тег элемента в UPPERCASE

лько для элемента

node.nodeName

ля элемента вернет tagName

ля TEXT_NODE – строку "#text"

element.innerHTML

одержимое элемента в виде строки

оступен на чтение и запись

егда осуществляется перезапись

ли в innerHTML записывается тег script – он не будет
полнен

element.outerHTML

одержит HTML элемент целиком

ри записи: в DOM исходный элемент замещается на овый элемент

зменить outerHTML элемента невозможно

outerHTML и innerHTML

```
let div = document.querySelector('div');
console.log(div); // <div>Привет</div>

div.innerHTML = 'Hi!'
console.log(div); // <div>Hi!</div>


div.outerHTML = '<h1>Aloha</h1>'
console.log(div) // <div>Hi!</div>
```

data/nodeValue

одержимое текстового узла или комментария

ожно изменять

ля некоторых типов узлов nodeValue равно null, поэтому
я более ожидаемого поведения лучше использовать data

data/nodeValue

```
<!-- Комментарий -->
<!DOCTYPE html>


console.log(document.firstChild.data); // Комментарий
console.log(document.firstChild.nodeValue); // Комментарий

console.log(document.lastChild.data); // undefined
console.log(document.lastChild.nodeValue); // null
```

Атрибуты

```
<img src="img.png" width="500" height="600" class="super-image">
```

Атрибуты

ement.[get/has/set/remove]Attribute

tributes – коллекция всех атрибутов элемента

hidden

ли true – элемент не виден на экране

ли false – элемент виден

Классы

assName – в виде строки

assList – объект для работы с классами

assList.[add/remove] – добавить/удалить класс

assList.toggle – переключает класс

assList.contains – проверяет есть ли класс

data-*

зволяют хранить дополнительную информацию в андартных элементах HTML

жно использовать только латинские буквы, дефис (-), оеточие (:) и подчёркивание (_)

```
<div data-index-number="123" data-parent="cars"></div>
```

data-*

е data-* атрибуты доступны в объекте dataset

ta- отбрасывается, а остаток переводится в camelCase

ример: data-user-location будет доступно в
ement.dataset.userLocation

Создание узлов

```
//Создание элемента
var element = document.createElement(tagName);

//Создание текстового узла
var textNode = document.createTextNode(text);
```

Добавление узлов

appendChild

```
var newLi = document.createElement('li');
newLi.innerHTML = '3';
// newLi.appendChild(document.createTextNode('3'))


list.appendChild(newLi);
```

insertBefore

```
var newLi = document.createElement('li');
newLi.innerHTML = '3';
// newLi.appendChild(document.createTextNode('3'))


list.insertBefore(newLi, list.children[1]);
```

Удаление узлов

rentElem.removeChild(elem) – удаляет elem из детей
rentElem

rentElem.replaceChild(newElem, elem) – удаляет elem из
тей parentElem и вставляет на его место newElem

removeChild

```
var list = document.getElementById('list');

list.removeChild(list.firstChild);
```

replaceChild

```
var list = document.getElementById('list');

var newLi = document.createElement('li');

newLi.innerHTML = '3';

list.replaceChild(newLi, list.firstChild);
```

elem.cloneNode

оздаёт копию текущего элемента

ли в качестве аргумента передать true, то создаст убокую копию элемента, включая атрибуты и одэлементы

События

События мыши

ck – на элемент кликнули левой кнопкой мыши

ntextmenu – на элемент кликнули правой кнопкой мыши

ouseover/mouseout – курсор навели/увели на элемент

ousedown/mouseup – кнопку мыши нажали (down)/
пустили (up)

ousemove – при движении курсора мыши над элементом

События пальцев

uchstart – элемента коснулись

uchmove – по элементу провели пальцем

uchend – касание закончилось и палец убрали

uchcancel – палец переместился на интерфейс браузера
и тач-событие нужно отменить

Другие события

bmit – отправка формы

cus – фокус на элементе

yup/keydown – печать на клавиатуре

size – изменение размеров окна

nsitionend – завершение css анимации

Ещё больше событий

Назначение обработчика событий

```
<li onclick="alert('привет');">Щелкни меня</li>
```

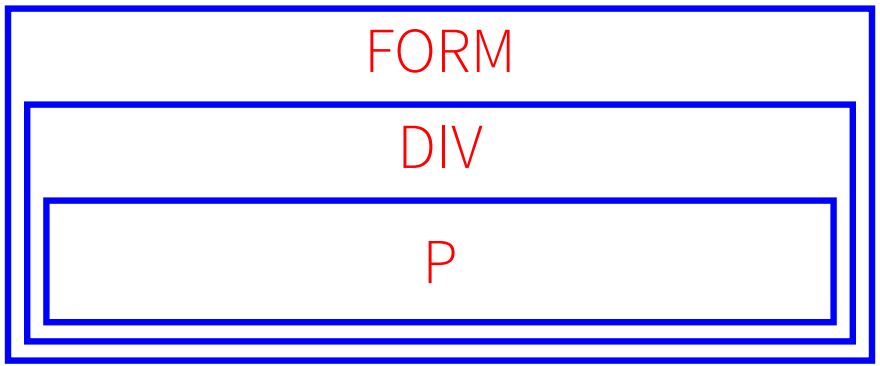
```
var li = document.getElementByTagName('li')[0];
li.onclick = function () { alert('hello'); }

li.addEventListener('click', function(event) {
    alert('aloha');
});
```

Удаление обработчика событий

```
var li = document.getElementByTagName('li')[0];
li.addEventListener('click', function(event){ alert('привет'); });


// не работает
li.removeEventListener('click', function() { alert('привет'); });

var li = document.getElementByTagName('li')[0];
var onClickHandler = function(event) {
    alert('привет');
}
li.addEventListener('click', onClickHandler);
li.removeEventListener('click', onClickHandler);
```

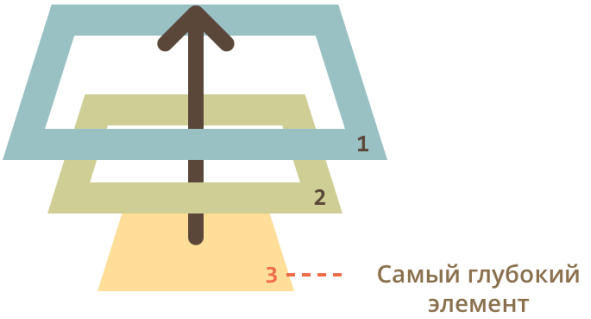
Всплытие событий

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

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



Всплытие событий

Сначала события срабатывают на самом вложенном элементе, затем на его родителе, и так далее, вверх до window

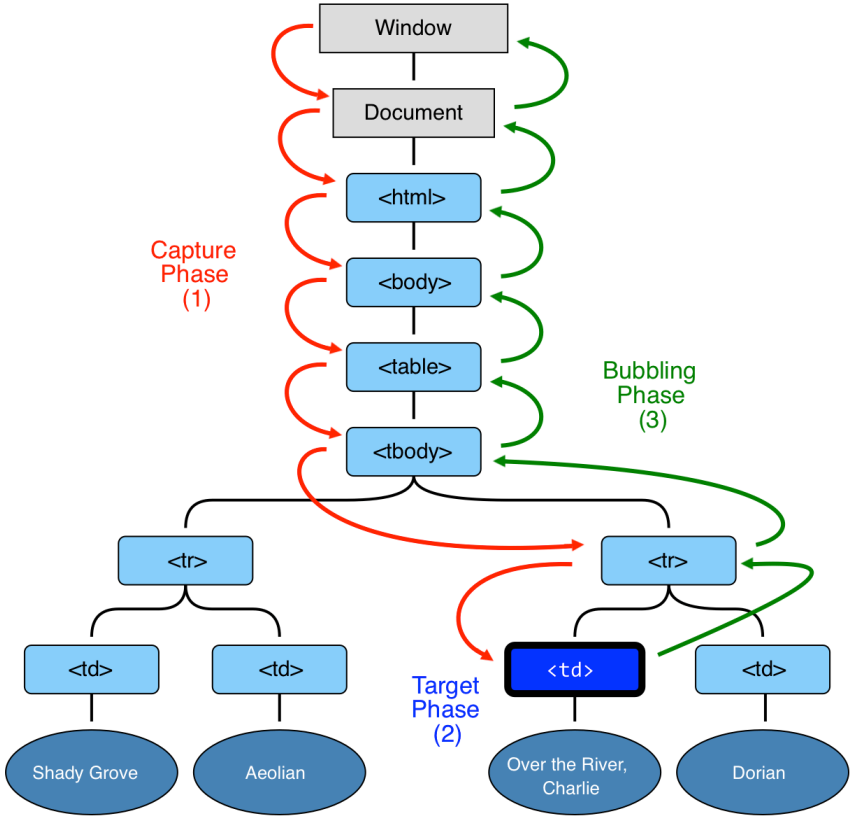


Стадии прохода события

адия перехвата (capturing stage) – событие идет сверху
из

адия цели (target stage) – событие достигло целевого
емента

адия всплытия (bubbling stage) – событие идет снизу верх


Window
Document
<html>
<body>
<table>
<tbody>
Capture Phase (1)
Bubbling Phase (3)
<tr>
<tr>
<td>
<td>
<td>
<td>
Target Phase (2)
Shady Grove
Aeolian
Over the River, Charlie
Dorian

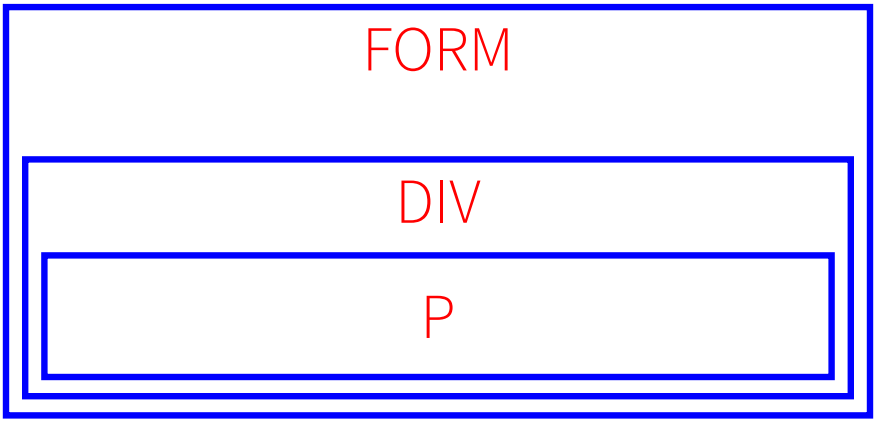
Самый глубокий элемент, который вызывает событие, называется «целевым» или «исходным» элементом и доступен как event.target

event.target

```
<form id="eventform">FORM
    <div>DIV

        <p>P</p>
    </div>
</form>
```

```
eventform.onclick = function(event) {
    alert('target = ', event.target.tagName);
    alert('this = ', this.tagName);
}
```

event.target



Работа с событием

ent.preventDefault – отменяет обработчик по-умолчанию

ent.stopPropagation – прекращает всплытие

ent.stopImmediatePropagation – прекращает всплытие и выполняет оставшиеся обработчики события

Перехват события

... обработчики не обрабатывают перехват

ement.addEventListener('event', callback, false) – обработка
стадии всплытия (дефолтное поведение)

ement.addEventListener('event', callback, true) – обработка
стадии перехвата

Делегирование событий

Делегирование событий

```
<ul>
    <li>0</li>
    <li>1</li>

    <li>2</li>
</ul>

// без делегирования

var logger = function(event) {
    console.log(event.target.innerHTML);
}


var liElements = document.getElementsByTagName('li');

for (var i = 0; i < liElements.length; i++) {
    var li = liElements[i];
```

Делегирование событий

```
<ul>
    <li>0</li>
    <li>1</li>

    <li>2</li>
</ul>

// с делегированием

var logger = function(event) {
    if(event.target.tagName === 'LI') {
        console.log(event.target.innerHTML);
    }
}


var ul = document.getElementsByTagName('ul')[0];
ul.addEventListener('click', logger);
```

Преимущества делегирования

дин обработчик вместо множества

ри добавление новых элементов не нужно добавлять им
бработчик

Полезные ссылки

пецификация DOM

niuse.com

arn.javascript.ru

DN

Вопросы