

DOM, который построил Джек Netscape

Чистяков Денис

DOM (Document Object Model) — это
программный интерфейс, для чтения и
редактирования содержимого
HTML, XML-документов

Это дерево, узлами которого могут быть
элементы, текст, комментарии, атрибуты и т.д.

Простейший DOM

```
<!DOCTYPE html>
<html>
<head>
    <title>DOM</title>
</head>
<body>

<form class="auth form" id="auth" data-form-value="123" action="/login/">
    <input type="text" value="" name="login">
    <input type="password" value="" name="password">
    <button>Войти</button>
</form>
</body>
</html>
```

Что позволяет нам DOM?

- Искать элементы
- Обход дерева
- Изменять атрибуты узла
- Добавлять / удалять / клонировать узлы
- Обрабатывать события на узлах

A photograph of a woman with curly hair, wearing a white tank top and shorts, dancing in a dark room. She is captured in mid-motion, with her arms raised and hands open. Her shadow is cast onto a light-colored wall behind her. A dark rectangular overlay contains the text.

Поиск элементов

```
<form class="auth form" id="auth" data-form-value="123" action="/login/">
    <input type="text" value="" name="login">
    <input type="password" value="" name="password">
    <button>Войти</button>
</form>
```

getElementById

```
document.getElementById('auth') // [object HTMLFormElement]
```

```
<form class="auth form" id="auth" data-form-value="123" action="/login/">
  <input type="text" value="" name="login">
  <input type="password" value="" name="password">
  <button>Войти</button>
</form>
```

getElementsByName

```
document.getElementsByTagName('input') // [object HTMLCollection]
```

```
<form class="auth form" id="auth" data-form-value="123" action="/login/">
  <input type="text" value="" name="login">
  <input type="password" value="" name="password">
  <button>Войти</button>
</form>
```

querySelector

```
document.querySelector('#auth') // [object HTMLElement]
```

```
<form class="auth form" id="auth" data-form-value="123" action="/login/">
  <input type="text" value="" name="login">
  <input type="password" value="" name="password">
  <button>Войти</button>
</form>
```

querySelector

```
document.querySelector('.auth') // [object HTMLElement]
```

```
<form class="auth form" id="auth" data-form-value="123" action="/login/">
  <input type="text" value="" name="login">
  <input type="password" value="" name="password">
  <button>Войти</button>
</form>
```

querySelector

```
document.querySelector('[name="login"]') // [object HTMLInputElement]
```

```
<form class="auth form" id="auth" data-form-value="123" action="/login/">
  <input type="text" value="" name="login">
  <input type="password" value="" name="password">
  <button>Войти</button>
</form>
```

querySelector всегда возвращает
только первый элемент

querySelector

```
document.querySelector('input') // [object HTMLInputElement]
```

```
<form class="auth form" id="auth" data-form-value="123" action="/login/">
  <input type="text" value="" name="login">
  <input type="password" value="" name="password">
  <button>Войти</button>
</form>
```

querySelectorAll

```
document.querySelectorAll('input,button') // [object NodeList]
```

```
<form class="auth form" id="auth" data-form-value="123" action="/login/">
  <input type="text" value="" name="login">
  <input type="password" value="" name="password">
  <button>Войти</button>
</form>
```

HTMLCollection и NodeList — не массивы,
поэтому методов forEach, map и т.п.— нет!

Why does document.querySelectorAll return a StaticNodeList rather than a real Array?

Итерирование по коллекциям

```
// Вариант №1
var collection = document.querySelectorAll('input,button');
for (var i = 0, len = collection.length; i < len; i++) {
    var elem = collection[i];
}
```

Итерирование по коллекциям

```
// Вариант №2
var elems = document.querySelectorAll('input,button');

var elemsList = Array.prototype.slice.call(elems);
elemsList.forEach(function(elem) {
    ...
});
```

Итерирование по коллекциям

```
// Вариант №3
var elemsList = Array.from(document.querySelectorAll('input,button'));
elemsList.forEach(function(elem) {
    ...
});
```



Атрибуты и свойства

*Attribute

```
var form = document.getElementById('auth');
```

```
form.getAttribute('action'); // '/login/'
```

```
form.hasAttribute('method'); // false
form.setAttribute('method', 'POST');
form.hasAttribute('method'); // true
form.getAttribute('method'); // 'POST'
form.removeAttribute('method');
form.hasAttribute('method'); // false
```

data-атрибуты

```
form.getAttribute('data-form-value'); // '123'
```

```
form.dataset.formValue; // '123'  
form.dataset.hasOwnProperty('formValue'); // true
```

```
form.dataset.fooBarBazBaf = 'boo'; // data-foo-bar-baz-baf="boo"  
form.hasAttribute('data-foo-bar-baz-baf'); // true
```

```
<form class="auth form" id="auth" data-form-value="123" action="/login/">  
  <input type="text" value="" name="login">  
  <input type="password" value="" name="password">  
  <button>Войти</button>  
</form>
```

Атрибуты !=== свойства

```
form.getAttribute('action'); // "/"  
form.getAttribute('method'); // null  
form.getAttribute('id'); // "auth"
```

```
form.action; // "https://yandex.ru/login/"  
form.method; // "get"  
form.id; // "auth"
```

Классы

```
form.className; // 'auth form'  
form.className += ' login-form'; // 'auth form login-form'
```

```
form.classList.add('login-form');  
form.classList.item(1); // 'form'  
form.classList.item(2); // 'login-form'  
form.classList.contains('login-form'); // true  
form.classList.remove('login-form');
```

Chalkaholic Digital

Создание элементов



createElement + appendChild

```
var elem = document.createElement('span');
```

```
elem.className = 'error';  
  
elem.setAttribute('id', 'auth-error');  
elem.setAttribute('status', 'auth-error');  
elem.textContent = 'Введен неверный логин или пароль';
```

```
document.body.appendChild(elem);
```

cloneNode

```
var clone = elem.cloneNode(true);
```

```
clone.id = 'mail-error';
clone.textContent = 'Не удалось отправить письмо';
```

```
document.body.appendChild(clone);
```



События

Все DOM-элементы могут реагировать на те или
иные события

Можно навешивать обработчики в HTML

```
<form class="auth form" id="auth" data-form-value="123"  
      action="/login/" onsubmit="submitHandler()">  
  
    <input type="text" value="" name="login">  
    <input type="password" value="" name="password">  
    <button onclick="clickHandler()">Войти</button>  
</form>
```

Почему это плохо?

- Нельзя навешать более одного обработчика на узел
- JS и HTML становятся тесно связаны
- При изменении HTML необходимо всегда помнить про атрибуты с обработчиками
- Нельзя динамически навешивать обработчики

addEventListener

```
var form = document.getElementById('auth');
form.addEventListener('submit', submitHandler);
```

```
var button = document.querySelector('button');
button.addEventListener('click', function(event) {
    console.log(this, event);
});
button.addEventListener('click', clickHandler);
button.addEventListener('click', yetAnotherClickHandler);
```

Все обработчики будут вызваны в порядке их
установки

Контекстом функции обработчика событий
(`this`) всегда является DOM-узел на котором
обрабатывается событие

Т.е. узел относительно которого был вызван
`addEventListener`

Объект события event

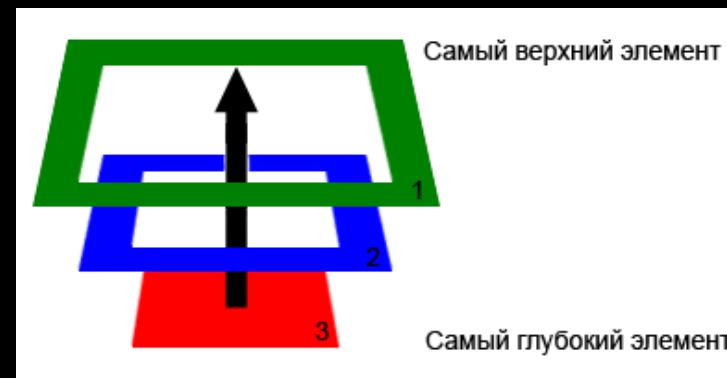
- target — узел на котором фактически произошло событие
- altKey — была ли нажата клавиша alt во время события
- ctrlKey — была ли нажата клавиша ctrl во время события
- shiftKey — была ли нажата клавиша shift во время события

A close-up photograph of a fluffy, light-colored cat with large, expressive blue eyes. The cat is looking directly at the camera with a slightly curious expression. The background is a soft, out-of-focus teal color.

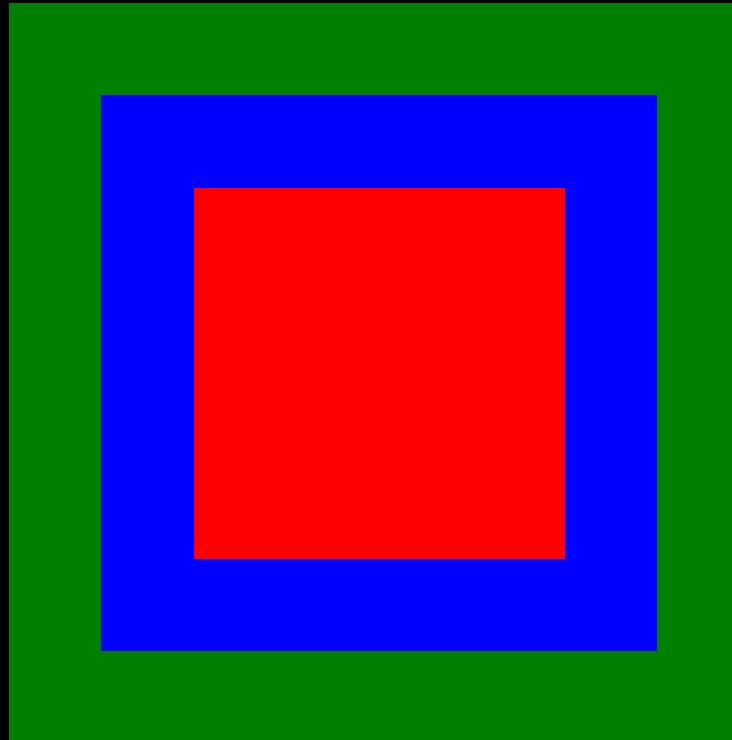
Существует два подхода

- Всплытие событий (Bubbling)
- Перехват событий (Capturing)

Всплытие событий (Bubbling)



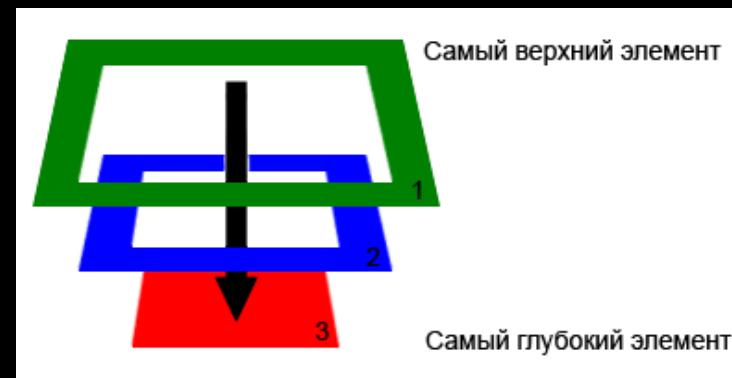
Всплытие событий (Bubbling)



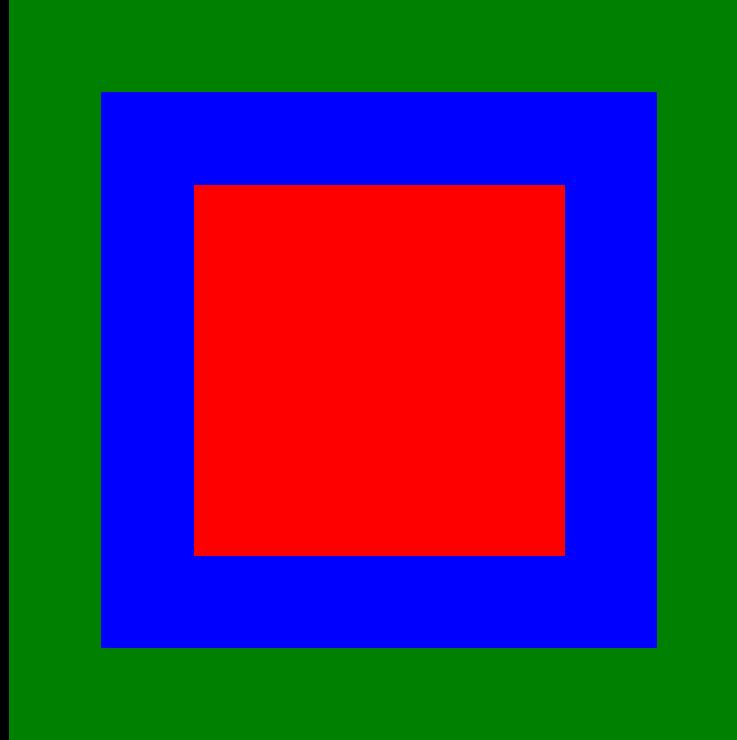
Всплытие событий (Bubbling)

```
var button = document.querySelector('button');
button.addEventListener('click', clickHandler, false);
button.addEventListener('click', yetAnotherClickHandler, false);
```

Перехват событий (Capturing)



Перехват событий (Capturing)

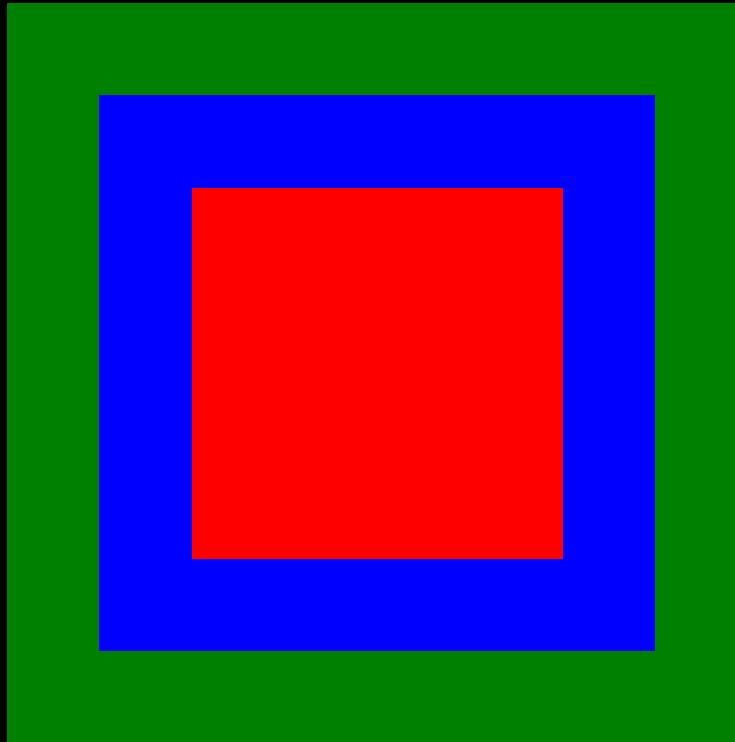


Перехват событий (Capturing)

```
var button = document.querySelector('button');
button.addEventListener('click', clickHandler, true);
button.addEventListener('click', yetAnotherClickHandler, true);
```

По умолчанию последним параметром
передается `false` и события всплывают

Bubbling + stopPropagation

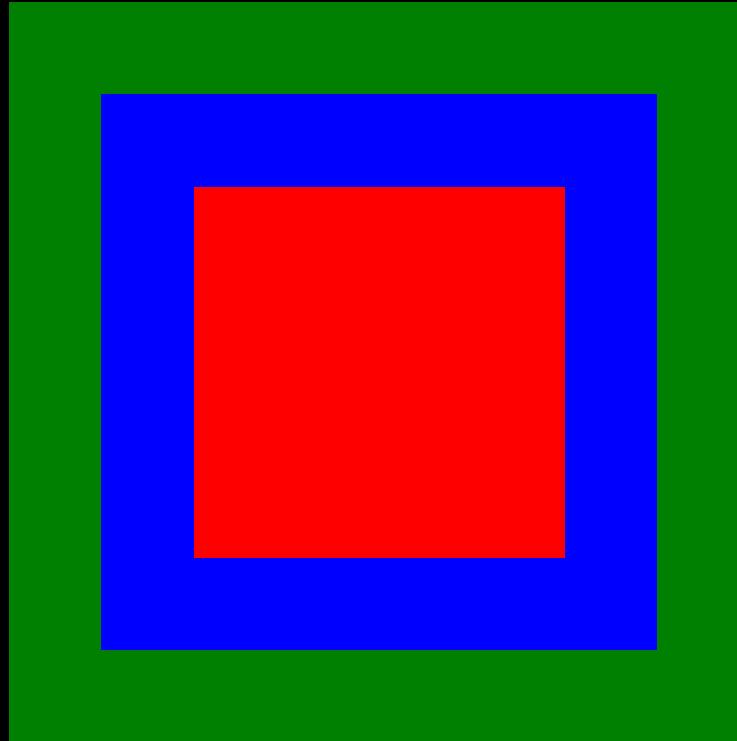


Bubbling + stopPropagation

```
var button = document.querySelector('button');

button.addEventListener('click', function(event) {
    event.stopPropagation();
}, false);
```

Capturing + stopPropagation



Capturing + stopPropagation

```
var button = document.querySelector('button');

button.addEventListener('click', function(event) {
    event.stopPropagation();
}, true);
```

`stopImmediatePropagation` также
отменяет всплытие / погружение, но также
отменяет все следующие обработчики на узле

У многих элементов есть действия по умолчанию

preventDefault

```
var form = document.getElementById('auth');

form.addEventListener('submit', function(event) {
  console.log(event.target);
  event.preventDefault();
});
```

Благодаря всплытию / погружению событий
возможно использовать такой подход, как
«Делегирование событий»

Делегирование событий

```
<ul>
  <li><a href="#maps">Карты</a></li>
  <li><a href="#market">Маркет</a></li>
  <li><a href="#news">Новости</a></li>

  <li><a href="#translate">Переводчик</a></li>
  <li><a href="#images">Картинки</a></li>
  <li><a href="#video">Видео</a></li>
  <li><a href="#music">Музыка</a></li>
  <li><a href="#more" class="more">ещё</a></li>
</ul>
```

Делегирование событий

```
document.addEventListener('click', function(event) {
    event.preventDefault();

    if (event.target.tagName === 'A') {
        if (event.target.classList.contains('more')) {
            openMorePopup();
        } else {
            console.log(event.target.href);
        }
    });
});
```

Мы научились

- Искать элементы
- Изменять атрибуты узла
- Добавлять / удалять / клонировать узлы
- Обрабатывать события на узлах
- Делегировать обработку событий родительским узлам