



Южно-Уральский
государственный
университет

Национальный
исследовательский
университет

КРОК

Основы JS. Основы DOM-модели.

КРОК

Челябинск, ул. Карла Маркса, д. 38

Смирнов Анатолий
Технический менеджер

Кузнецов Сергей
Старший инженер-разработчик

Фоменко Алексей
Младший инженер-разработчик



- Объектная Модель Документа (DOM) – это программный интерфейс (API) для HTML и XML документов.
- DOM предоставляет структурированное представление документа и определяет то, как эта структура может быть доступна из программ, которые могут изменять содержимое, стиль и структуру документа.
- В соответствии с DOM каждый HTML-тег является объектом. Вложенные теги являются дочерними к родительскому элементу.

API (веб либо XML страница)
=
DOM + JS (язык описания скриптов)



- DOM – это представление HTML-документа в виде дерева тегов
- Теги являются узлами-элементами (или просто элементами). Они образуют структуру дерева: `<html>` – это корневой узел, `<head>` и `<body>` его дочерние узлы и т.д.

```
<meta charset="utf-8"/>
<form action="/api/create_form.php" method="post">
  Имя: <input name="firstname"><br>
  Фамилия: <input name="surname"><br>
  Отчество: <input name="lastname"><br>
  Кол-во полетов: <input name="flights"><br>
  <input type="submit" value="Добавить">
</form>
```

Имя:

Фамилия:

Отчество:

Кол-во полетов:

```
***<html> == $0
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <form action="/api/create_form.php" method="post">
      " Имя: "
      <input name="firstname">
      <br>
      " Фамилия: "
      <input name="surname">
      <br>
      " Отчество: "
      <input name="lastname">
      <br>
      " Кол-во полетов: "
      <input name="flights">
      <br>
      <input type="submit" value="Добавить">
    </form>
  </body>
</html>
```



- Каждая веб-страница, которая загружается в браузер, имеет свой собственный объект `document`, доступный из JS
- `document.body` это элемент, который включает в себя содержимое страницы.

```
> let body = document.body
```

```
< undefined
```

```
> typeof (body)
```

```
< 'object'
```



`document` метод `querySelector()` возвращает первый элемент (Element) документа, который соответствует указанному селектору или группе селекторов:

```
element = document.querySelector(selectors);
```



Селекторы такие же, как и в CSS, например:

```
document.querySelector('тег')  
document.querySelector('#идентификатор')  
document.querySelector('.класс')  
document.querySelector('[атрибут]')  
document.querySelector('[атрибут="значение"]')
```

```
<meta charset="UTF-8">
<body>
  <div class="ex">Пример 1</div>
  <span>Пример 2</span>
  <div id="someItem">Пример 3</div>
  <span class="ex">Пример 4</span>
  <br/>
  <input type="text" value="Пример 5">
</body>
```

```
document.querySelector('div')
document.querySelector('span')
document.querySelector('#someItem')
document.querySelector('span.ex')
document.querySelector('br')
document.querySelector('[type="text"]')
```

```
> document.querySelector('div')
< <div class="ex">Пример 1</div>
```

```
> document.querySelector('span')
< <span>Пример 2</span>
```

```
> document.querySelector('#someItem')
< <div id="someItem">Пример 3</div>
```

```
> document.querySelector('span.ex')
< <span class="ex">Пример 4</span>
```

```
> document.querySelector('br')
< <br>
```

```
> document.querySelector('[type="text"]')
< <input type="text" value="Пример 5">
```



- Можно комбинировать селекторы.
- querySelector будет искать элементы во всех вложенных дочерних элементах, пока не найдет нужный.

```
<meta charset="UTF-8">
<body>
  <div class="ex">
    <input type="text" value="Пример 1">
    <span>Пример 2</span>
    <div>
      <input type="text" value="Пример 3">
    </div>
  </div>
</body>
```

```
> document.querySelector('input')
< <input type="text" value="Пример 1">
```

```
> document.querySelector('div.ex input')
< <input type="text" value="Пример 1">
```

```
> document.querySelector('div.ex div input')
< <input type="text" value="Пример 3">
```



Метод querySelector можно вызывать не только у document, а у любого элемента

```
<meta charset="UTF-8">
<body>
  <div id="div1">
    <span>Пример 1</span>
  </div>
  <div id="div2">
    <span>Пример 2</span>
  </div>
</body>
```

```
> let elem = document.querySelector('#div1')
< undefined
> elem.querySelector('span')
< <span>Пример 1</span>
```

```
> let span2 = document.querySelector('#div2').querySelector('span')
< undefined
> span2
< <span>Пример 2</span>
```




- Метод `querySelectorAll()` возвращает все элементы (Elements) документа, которые соответствуют указанному селектору или группе селекторов.

```
<meta charset="UTF-8">
<body>
  <div class="ex">Пример 1</div>
  <span>Пример 2</span>
  <div id="someItem">Пример 3</div>
  <span class="ex">Пример 4</span>
  <br/>
  <input type="text" value="Пример 5">
</body>
```

```
document.querySelectorAll('div')
document.querySelectorAll('span')
document.querySelectorAll('.ex')
```

```
> document.querySelectorAll('div')
< ▶ NodeList(2) [div.ex, div#someItem]
```

```
> document.querySelectorAll('span')
< ▶ NodeList(2) [span, span.ex]
```

```
> document.querySelectorAll('.ex')
< ▶ NodeList(2) [div.ex, span.ex]
```



- Свойство `innerHTML` позволяет считать содержимое элемента в виде HTML-строки или установить новый HTML.
- Новое значение HTML необходимо передавать в виде строки и оно заменит текущее содержимое элемента.

```
<meta charset="UTF-8">
<body>
  <div id="div1">
    <span>Пример 1</span>
  </div>
</body>
```

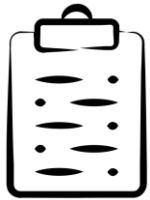
```
> let elem = document.querySelector('#div1');
< undefined
> elem.innerHTML
< '\n      <span>Пример 1</span>\n    '
```

```
> elem.innerHTML = '<b>Меняем разметку</b>';
< '<b>Меняем разметку</b>'
```

localhost/render/example2.html

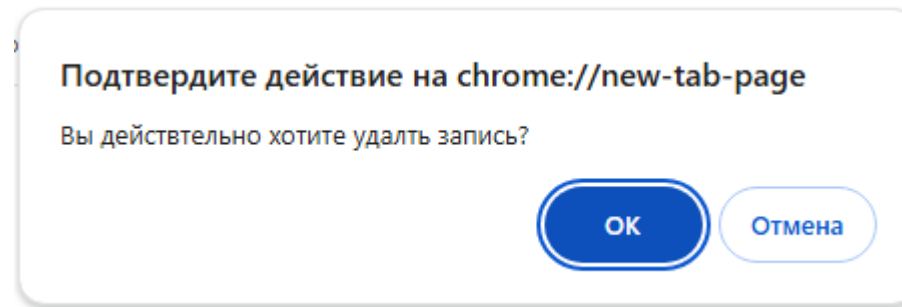
Меняем разметку

```
..<html> == $0
  ▶ <head> ... </head>
  ▼ <body>
    ▼ <div id="div1">
      <b>Меняем разметку</b>
    </div>
  </body>
</html>
```



Для подтверждения действия можно использовать функцию `confirm('сообщение')`.

Она синхронно ожидает действие пользователя. Если пользователь нажал «Ок», вернет `true`, если «Отмена», вернет `false`.



«Ок»

```
> confirm('Вы действительно хотите удалить запись?')  
true
```

«Отмена»

```
> confirm('Вы действительно хотите удалить запись?')  
false
```



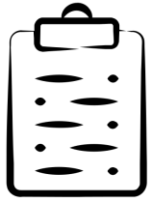
Для вывода сообщения пользователю можно использовать функцию `alert('сообщение')`:

```
> alert('Внимание!')  
← undefined
```

Подтвердите действие на chrome://new-tab-page

Внимание!

OK



Событие – это сигнал от браузера о том, что что-то произошло. Все DOM-узлы подают такие сигналы, например:

- `click` – происходит, когда кликнули на элемент левой кнопкой мыши (на устройствах с сенсорными экранами оно происходит при касании).
- `contextmenu` – происходит, когда кликнули на элемент правой кнопкой мыши.
- `mouseover` / `mouseout` – когда мышь наводится на / покидает элемент.
- `mousedown` / `mouseup` – когда нажали / отжали кнопку мыши на элементе.
- `mousemove` – при движении мыши.



Назначить обработчик события можно либо в соответствующем атрибуте, либо непосредственно в JS

```
<meta charset="UTF-8">
<body>
  <div id="div1">Пример 1</div>
  <div onclick="console.log('Пример 2')">Пример 2</div>
  <div onclick="myFunc()">Пример 3</div>

  <script>
    const item = document.querySelector('#div1')
    item.onclick = function() {
      console.log('Пример 1')
    }
    function myFunc() {
      console.log('Пример 3')
    }
  </script>
</body>
```

Пример 1

Пример 2

Пример 3



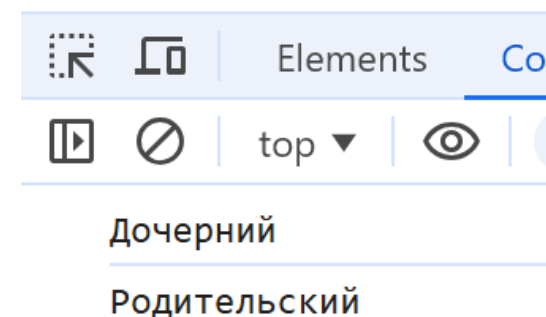


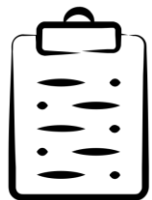
Если на все элементы наложены события, например, onclick, по умолчанию они будут вызваны, начиная с самого дочернего.

- Браузер проверяет, имеет ли элемент, который был фактически нажат, обработчик события onclick, зарегистрированный на нем в фазе всплытия, и запускает его, если это так.
- Затем он переходит к следующему непосредственному родительскому элементу и выполняет то же самое, затем следующее и так далее, пока не достигнет элемента <html>.

```
<meta charset="UTF-8">
<body>
  <div onclick="console.log('Родительский')">
    <div onclick="console.log('Дочерний')">
      Текст
    </div>
  </div>
</body>
```

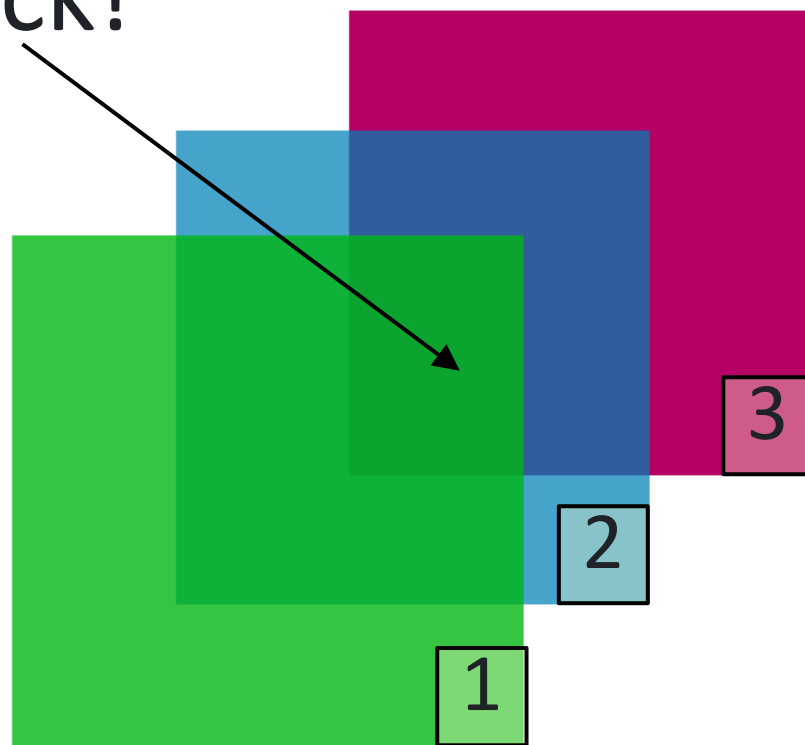
Текст



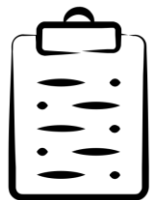


По умолчанию используется «всплытие», от дочернего к родительскому

Click!



1. Событие 1
2. Событие 2
3. Событие 3



В каждое событие передается объект event, который содержит информацию о текущем событии. Структура каждого event для каждого события отличается, необходимо обращаться к документации и стандарту.

```
<meta charset="UTF-8">
<body>
  <div onclick="console.log(event)">
    Текст
  </div>
</body>
```

```
▼ PointerEvent {isTrusted: true, pointerId: 1, wi
  isTrusted: true
  altKey: false
  altitudeAngle: 1.5707963267948966
  azimuthAngle: 0
  bubbles: true
  button: 0
  buttons: 0
  cancelBubble: false
  cancelable: true
  clientX: 34
  clientY: 19
  composed: true
  ctrlKey: false
  currentTarget: null
  defaultPrevented: false
  detail: 1
  eventPhase: 0
  fromElement: null
```

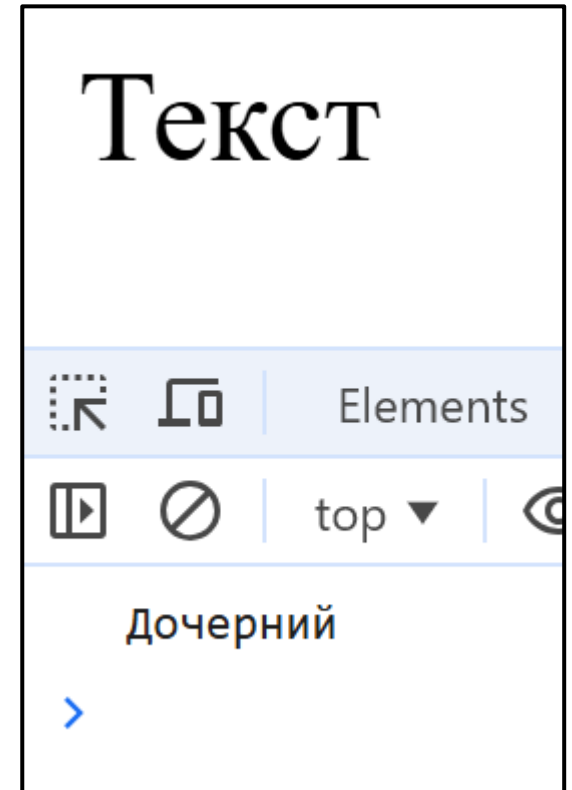


Помимо свойств у event есть и методы. Самые часто используемые:

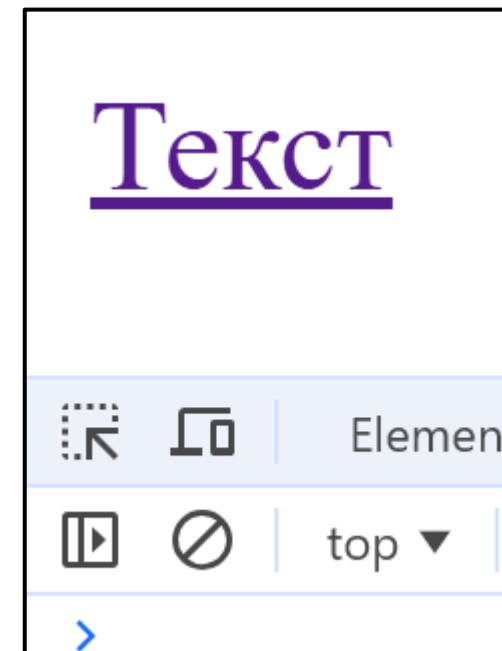
- `stopPropagation()` – остановить всплытие события. Если у родительского элемента есть события того же типа, то он не будут вызваны
- `preventDefault()` – остановить действие по умолчанию (переходы по ссылке, меню на ПКМ, отмена печати и т.д.)

```
<meta charset="UTF-8">
<body>
  <div id="div1">
    <div id="div2">
      Текст
    </div>
  </div>
</div>
<script>
  document.querySelector('#div1').onclick = () => {
    console.log('Родительский');
  }

  document.querySelector('#div2').onclick = (event) => {
    // Событие на div1 не уйдет
    event.stopPropagation();
    console.log('Дочерний');
  }
</script>
</body>
```



```
<meta charset="UTF-8">
<body>
  <a href="https://developer.mozilla.org/">
    Текст
  </a>
  <script>
    document.querySelector('a').onclick = (event) => {
      // Переход по ссылке не произойдет
      event.preventDefault();
    }
  </script>
</body>
```





В HTML тегах можно задавать пользовательские атрибуты, которые начинаются с data-.*.

Это позволяет хранить внутри тегов некоторые данные с привязкой к этому тегу.

```
<article  
  id="electriccars"  
  data-columns="3"  
  data-index-number="12314"  
  data-parent="cars">  
  ...  
</article>
```

```
<meta charset="UTF-8">
<body>
  <div class="parent"/>
    <script>
      const data = ['one', 'two', 'three'];
      const domList = data.map((value, index) =>
        `<div data-index="${index}"
          onclick="console.log('Нажали ${index}')"
          class="main">
            ${value}
          </div>`
      );

      const parent = document.querySelector('.parent');
      parent.innerHTML = domList.join('');
    </script>
    <style>
      .main {
        margin: 5px;
        background: yellow;
      }
    </style>
  </body>
```

JS. data-атрибут элементов html

one

two

three

```
Elements Console Sources Network Performance Memory Application Se
<html>
  > <head> </head>
  > <body>
    > <div class="parent"> == $0
      <div data-index="0" onclick="console.log('Нажали 0')" class="main"> one </div>
      <div data-index="1" onclick="console.log('Нажали 1')" class="main"> two </div>
      <div data-index="2" onclick="console.log('Нажали 2')" class="main"> three </div>
      > <style> </style>
    </div>
  </body>
</html>
```

one

two

three

```
Elements Console So
top ▼ | Filter
Нажали 1
Нажали 2
>
```

```
<script>
  const data = [
    {
      id: 1111, value: 'one'
    },
    {
      id: 2222, value: 'two'
    },
    {
      id: 3333, value: 'two'
    },
  ];
  const domList = data.map(item =>
    `<div data-id="${item.id}"
      class="main">
        ${item.value}
      </div>`
  );

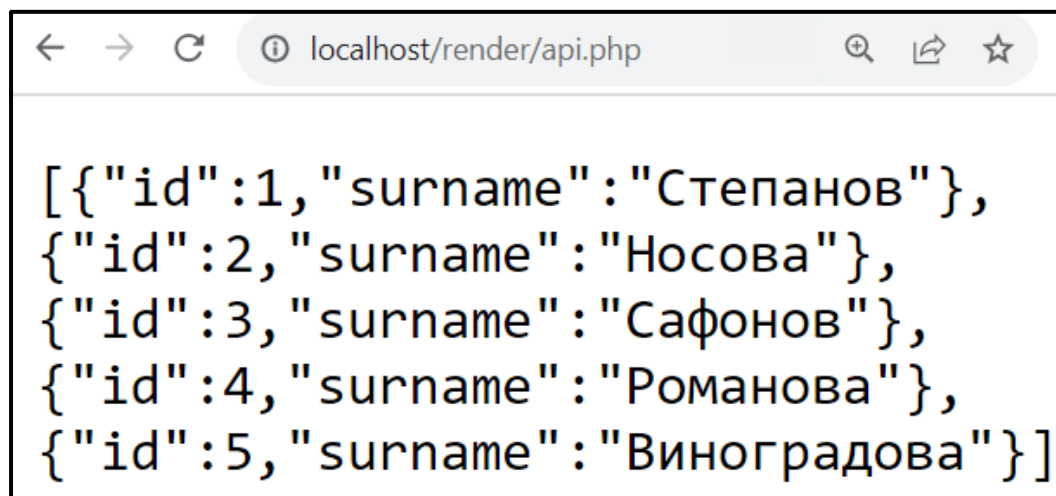
  const parent = document.querySelector('.parent');
  parent.innerHTML = domList.join('');
</script>
```




```
<?php
    $dsn = "pgsql:host=localhost;port=5432;dbname=postgres;";
    $pdo = new PDO($dsn, 'postgres', '1');

    $sql = "select id, surname from users limit 5";
    $stmt = $pdo->query($sql);

    $result_list = $stmt->fetchAll(PDO::FETCH_ASSOC);
    header('Content-Type: application/json');
    echo json_encode($result_list, JSON_UNESCAPED_UNICODE);
?>
```



The screenshot shows a web browser window with the address bar displaying 'localhost/render/api.php'. The main content area shows a JSON array of five objects, each containing an 'id' and a 'surname'. The surnames are in Cyrillic: Степанов, Носова, Сафионов, Романова, and Виноградова.

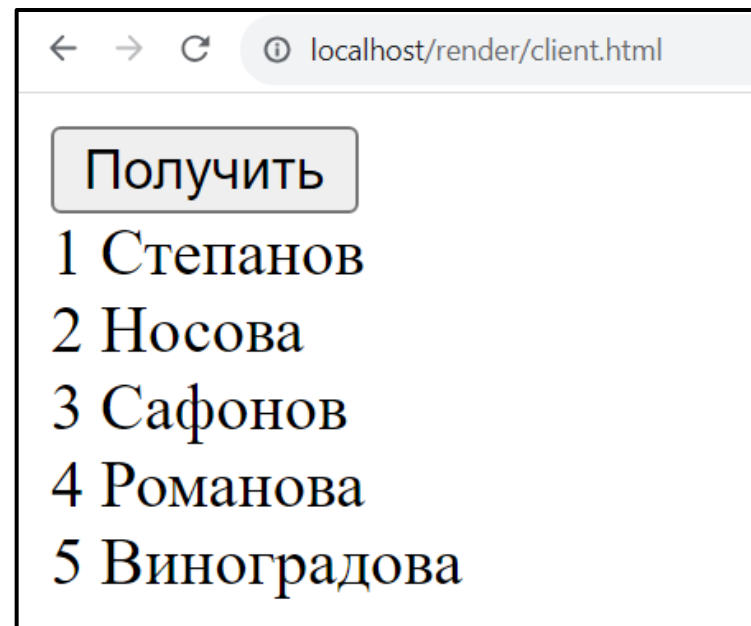
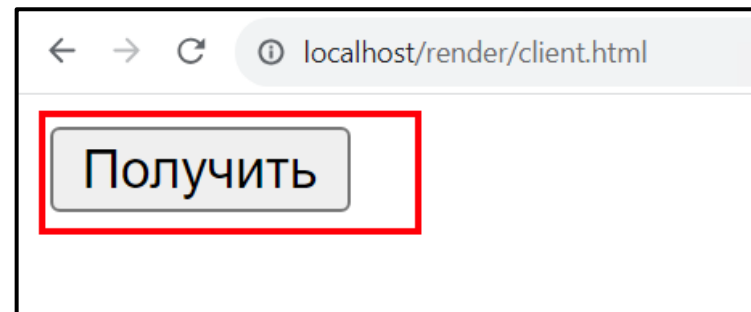
```
[{"id":1,"surname":"Степанов"},
{"id":2,"surname":"Носова"},
{"id":3,"surname":"Сафионов"},
{"id":4,"surname":"Романова"},
{"id":5,"surname":"Виноградова"}]
```

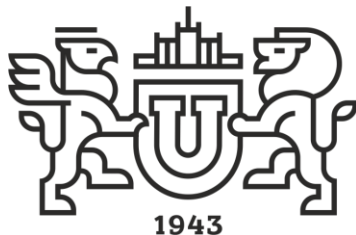
```
<meta charset="UTF-8">
<button id="sendButton" onclick="getUsers()">Получить</button>
<div id="container"></div>
```

```
<script>
const cont = document.querySelector('#container');
```

```
function onSuccess(data) {
  cont.innerHTML = '';
  for (let row of data) {
    cont.innerHTML += row['id'] + ' ' +
                      row['surname'] + '<br/>';
  }
}
```

```
function getUsers() {
  fetch('http://localhost/render/api.php')
    .then(function (result) {
      return result.json();
    })
    .then(onSuccess)
}
</script>
```





Южно-Уральский
государственный
университет

Национальный
исследовательский
университет

КРОК

Спасибо за внимание!



КРОК

Челябинск, ул. Карла Маркса, д. 38