

Южно-Уральский
государственный
университет

Национальный
исследовательский
университет

КРОК

Основы JS. Запросы к серверу. Основы DOM-модели.

КРОК

Челябинск, ул. Карла Маркса, д. 38

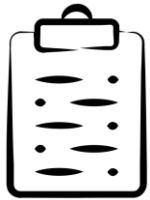
Смирнов Анатолий
Технический менеджер

Антонов Сергей
Тимлид группы разработчиков,
старший инженер-разработчик

Кузнецов Сергей
Инженер-разработчик



- Очень распространённой задачей в современных веб-сайтах и приложениях является получение отдельных элементов данных с сервера для обновления разделов веб-страницы без необходимости загрузки всей новой страницы
- JavaScript может отправлять сетевые запросы на сервер и подгружать новую информацию по мере необходимости



- Fetch API предоставляет интерфейс JavaScript для работы с запросами и ответами HTTP.
- Он также предоставляет глобальный метод `fetch()`

```
fetch('http://example.com', {  
  // Опции  
})  
  .then(function(response) {  
    return response.json();  
  })  
  .then(function(data) {  
    console.log(data);  
  });
```

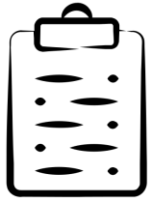


Для представления ответа от сервера в Fetch API применяется интерфейс Response. Свойства:

- ok: хранит булево значение, которое указывает, завершился ли запрос
- status: хранит статусный код ответа
- statusText: хранит сообщение статуса, которое соответствует статусному коду
- ...

```
fetch('http://example.com')  
  .then(function(response) {  
    console.log(response)  
  })
```

```
▼ Response {type: 'basic', url: 'http://loca  
  ► body: ReadableStream  
    bodyUsed: true  
  ► headers: Headers {}  
    ok: true  
    redirected: false  
    status: 200  
    statusText: "OK"  
    type: "basic"  
    url: "http://localhost/render/api.php"
```



Методы:

- `text()` для текста
- `json()` для текста в формате JSON
- `blob()` для бинарных объектов (изображения, аудио и т.д.)
- ...

```
fetch('http://example.com')  
  .then(function(response) {  
    return response.json();  
  })  
  .then(function(data) {  
    // data – объект из JSON  
  });
```



Для установки метода используется свойство `method` в опциях `fetch`. По умолчанию GET

```
fetch('https://httpbin.org/put', {  
  method: 'PUT'  
})  
.then(function(response) {  
  return response.json()  
})  
.then(function(data) {  
  console.log(data)  
})
```

```
▼ {args: {...}, data: '', files: {...}, form: {...}, headers: {...},  
  ► args: {}  
    data: ""  
    ► files: {}  
    ► form: {}  
    ► headers: {Accept: '*/*', Accept-Encoding: 'gzip, deflate'  
      json: null  
      origin: "88.206.74.184"  
      url: "https://httpbin.org/put"  
    ► [[Prototype]]: Object
```



Для отправки тела используется свойство `body` в опциях `fetch`

```
let data = {  
  key1: 'value1'  
}  
fetch('https://httpbin.org/post', {  
  method: 'POST',  
  body: JSON.stringify(data)  
})  
.then(function(response) {  
  return response.json()  
})  
.then(function(data) {  
  console.log(data)  
})
```

```
▼ {args: {...}, data: '{"key1":"value1"}', files: {...}, form: {...}  
  ► args: {}  
  data: "{\\"key1\\":\\"value1\\"}"  
  ► files: {}  
  ► form: {}  
  ► headers: {Accept: '*/*', Accept-Encoding: 'gzip, deflate,  
  ► json: {key1: 'value1'}  
  origin: "88.206.74.184"  
  url: "https://httpbin.org/post"  
  ► [[Prototype]]: Object
```



Для установки заголовков используется свойство `headers` в опциях `fetch`

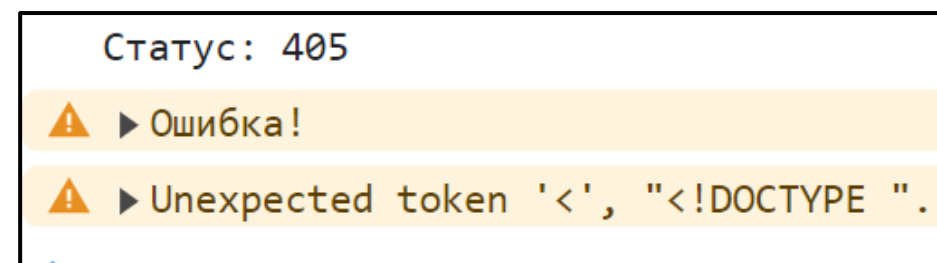
```
let data = {
  key1: 'value1'
}
fetch('https://httpbin.org/post', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(data)
})
.then(function(response) {
  return response.json()
})
.then(function(data) {
  console.log(data)
})
```

```
▼ {args: {...}, data: '{"key1":"value1"}', files: {...}, form: {...}
  ► args: {}
  data: "{\\"key1\\":\\"value1\\"}"
  ► files: {}
  ► form: {}
  ▼ headers:
    Accept: "*/*"
    Accept-Encoding: "gzip, deflate, br"
    Accept-Language: "ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7"
    Content-Length: "17"
    Content-Type: "application/json"
    Host: "httpbin.org"
    Origin: "http://localhost"
```




Для обработки ошибок применяется метод catch

```
fetch('https://httpbin.org/post', { // post
  method: 'PUT' // put
})
.then(function(response) {
  console.log('Статус: ' + response.status)
  return response.json()
})
.then(function(data) {
  console.log(data)
})
.catch(function(error) { // Обработка ошибок
  console.warn('Ошибка!')
  console.warn(error.message)
});
```



```
fetch('https://httpbin.org/post', { // post
  method: 'PUT' // put
})
.then(function(response) {
  console.log('Статус: ' + response.status)
  if (response.status === 200) {
    return response.json()
  }
  return response.text()
})
.then(function(data) {
  console.log(data)
})
.catch(function(error) {
  console.warn('Ошибка!')
  console.warn(error.message)
});
```

```
fetch('https://httpbin.org/post', { // post
  method: 'PUT' // put
})
.then(function(response) {
  console.log('Статус: ' + response.status)
  if (response.status === 200) {
    return response.json()
  }
  return response.text()
})
.then(function(data) {
  console.log(data)
})
.catch(function(error) {
  console.warn('Ошибка!')
  console.warn(error.message)
});
```

Статус: 405

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```



- 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 «детьми» родительского элемента.

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('#ИД')  
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">
```

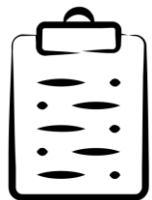


Метод querySelector можно вызывать не только у document, а у любого элемента

```
<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>
```



- document метод querySelectorAll() возвращает все элементы (Elements) документа, которые соответствует 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>
```

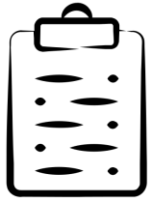
```
> let elem = document.querySelector('#div1');
< undefined
> elem.innerHTML
< '\n      <span>Пример 1</span>\n    '
```

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

localhost/render/example2.html

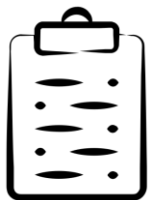
Меняем разметку

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



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



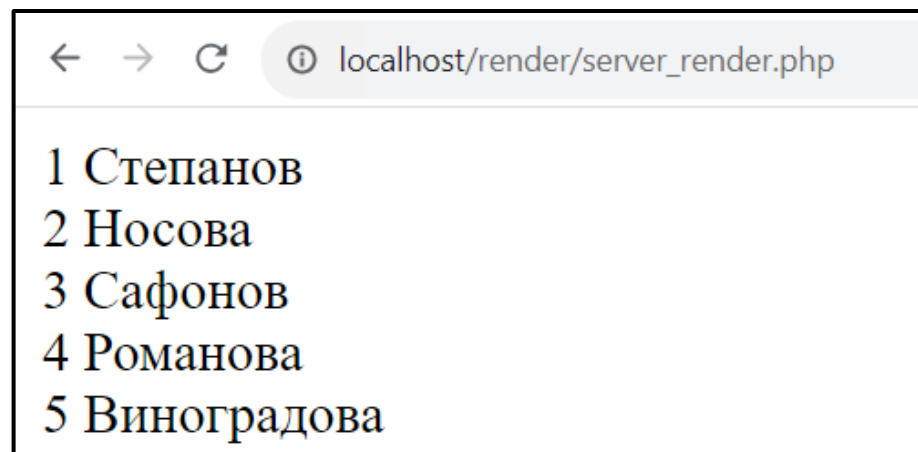


- Серверный рендеринг генерирует полный HTML страницы на сервере в ответ на навигацию. Это позволяет избежать дополнительных проходов для получения данных и шаблонов на клиенте, так как это выполняется до того, как браузер получает ответ
- Рендеринг на стороне клиента (CSR) означает рендеринг страниц непосредственно в браузере с использованием JavaScript. Вся логика, сбор данных, построение DOM-модели и маршрутизация обрабатываются на клиенте, а не на сервере

```
<?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);

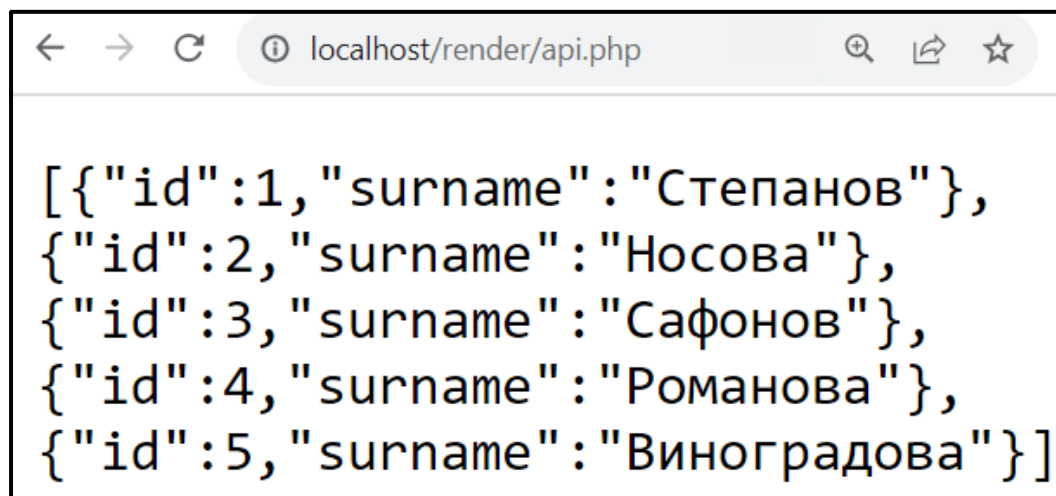
    while ($row = $stmt->fetch()) {
        echo $row['id']. " " . $row['surname'] . "<br/>";
    }
?>
```




```
<?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.

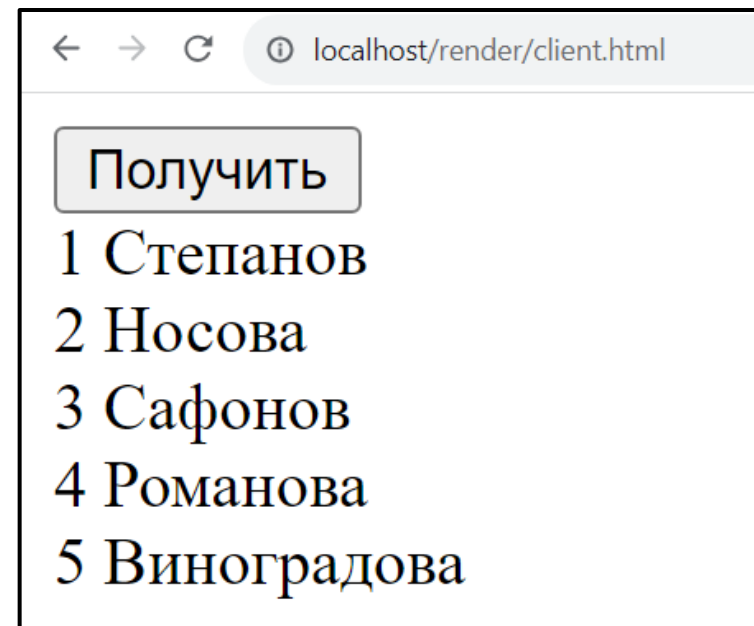
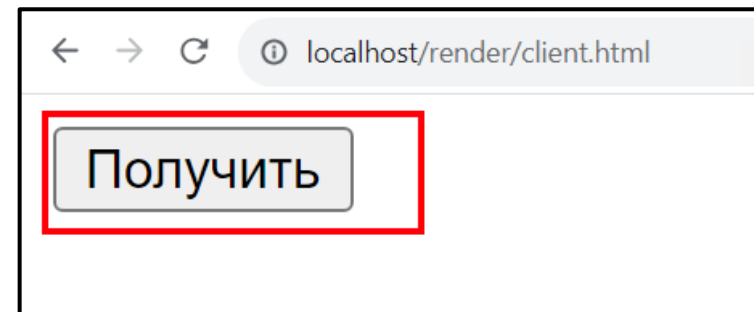
```
[{"id":1,"surname":"Степанов"},
{"id":2,"surname":"Носова"},
{"id":3,"surname":"Сафионов"},
{"id":4,"surname":"Романова"},
{"id":5,"surname":"Виноградова"}]
```

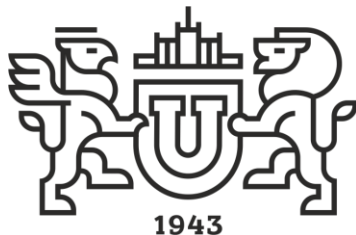
```
<meta charset="UTF-8">
<button id="sendButton" onclick="getUsers()">Получить</button>
<div id="container"></div>
```

```
<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