

Современные платформы прикладной разработки

ВВЕДЕНИЕ В AJAX

Технология AJAX в ASP.NET MVC

Ajax - термин, первоначально придуманный Джесси Джеймсом Гарреттом для описания технологии, которая подразумевает использование JavaScript для отправки асинхронного запроса к серверу и динамического отображения результата на странице, что избавляет от необходимости перезагружать страницу целиком.

Технология AJAX в ASP.NET MVC

Термин «Ajax» появился как акроним, образованный из **A**synchronous **J**avaScript и **X**ML. Это означает, что данные пересылались от сервера к клиенту асинхронно в формате XML.

Однако современные веб-приложения редко используют XML и, чтобы снизить трафик, используют формат JSON.

Введение в JavaScript

ЗНАКОМСТВО С AJAX

Введение в JavaScript

JavaScript (JS) — прототипно-ориентированный сценарный язык программирования. Является реализацией языка ECMAScript (стандарт ECMA-262).

История JavaScript:

1995 – начал применяться в браузерах (Brendan Eich | Netscape)

1997 – формализован в ECScript

2009 – начало использования в серверах (node.js)

2012 – Начало использования на клиентских компьютерах (Windows 8)

Основные архитектурные черты JavaScript

- динамическая типизация,
- слабая типизация,
- автоматическое управление памятью,
- прототипное программирование,
- функции как объекты первого класса,
- С-подобный синтаксис

Объявление переменных

```
let b = false;
```

```
let text = "Hello!";
```

```
var num = 10;
```

```
var str = String("Строка");
```

```
var num = Number(10);
```


Объявление функций

```
function f1 () {  
    //тело функции  
}
```

```
var f2 = function () {  
    //тело функции  
}
```

Вызов функций

```
f1 ();
```

```
f2 ();
```

```
function f1 () {  
    //тело функции  
}
```

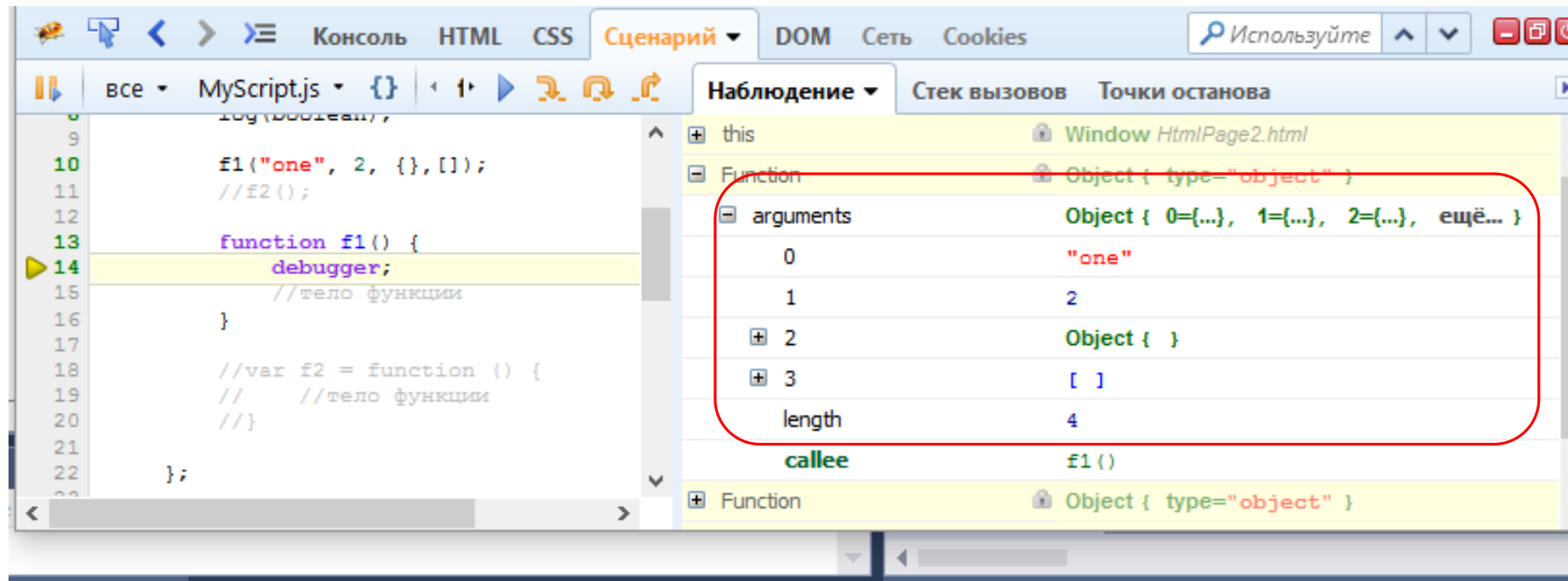
```
var f2 = function () {  
    //тело функции  
}
```

Вызов функций

```
f1 ("one", 2, {}, []);
```

```
function f1 () {  
    debugger;  
    //тело функции  
}
```

Вызов функций



Передача функции в качестве параметра

```
function sum(a,b) {  
    return a + b;  
}
```

```
function myMath(a,b, mathFunc) {  
    return mathFunc(a, b);  
}
```

```
var a = myMath(2, 4, sum);  
console.log(a);
```

Объявление массивов

```
var nums = ["один", "два", "три"];
```

```
var myArray = new Array();
```

```
var veqs = new Array("apple", "pear",  
"banana");
```

Функции работы с массивами

push , pop , shift - unShift - slice – join – concat – indexOf –
replace – length – sort – reverse – forEach – map - filter

Функции работы с массивами

```
var nums = ["один", "два", "три"];
var myArray = new Array();
var vegs = new Array("apple", "pear", "banana");
console.log(vergs);
vegs.push("qqq");
console.log(vergs);
console.log(nums);
console.log(nums.sort());
nums.push("четыре");
console.log(nums);
nums.pop();
console.log(nums);
nums.unshift(1);
console.log(nums);
```


Объекты в JavaScript

```
var dog = {  
    breed: "Немецкая овчарка",  
    bark: function () {  
        log("Гав-гав")  
    }  
}  
  
dog.bark();
```

Динамическое объявление свойств

```
let dog = {};  
dog.breed = "Немецкая овчарка";  
dog.bark = function () {  
    log("Гав-гав")  
};  
  
dog.bark();
```

Взаимодействие с DOM

ВВЕДЕНИЕ В JAVASCRIPT

Поиск объектов DOM

Поиск элементов DOM

getElementById
getElementsByTagName
querySelector
querySelectorAll

Поиск элементов DOM

```
var x =  
    document.getElementById("myId");
```

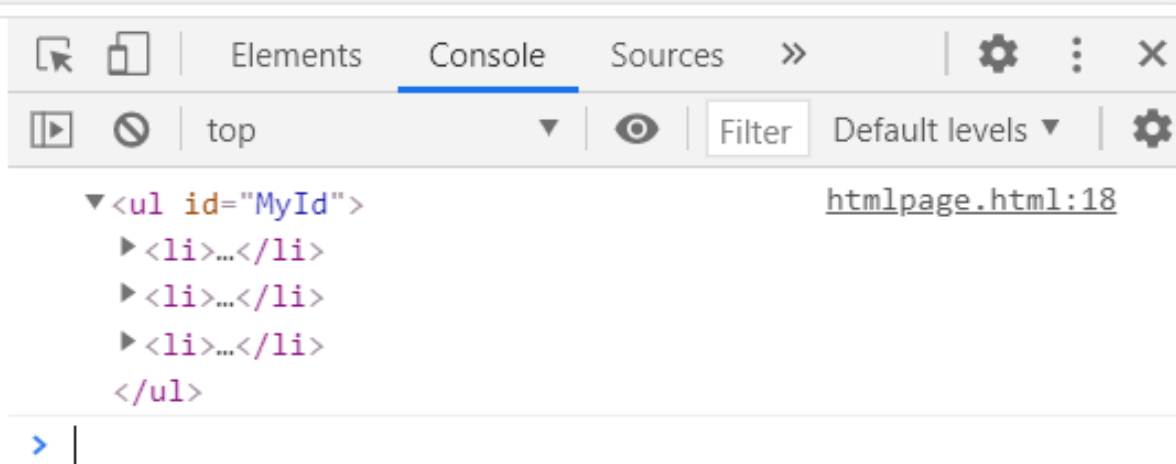
ИЛИ

```
var x =  
    document.querySelector("#myId")
```

```
<body>
  <ul id="MyId">
    <li>1</li>
    <li>2</li>
    <li>3</li>
  </ul>

  <script>
    let x = document.getElementById("MyId");
    console.log(x);
  </script>
</body>
```

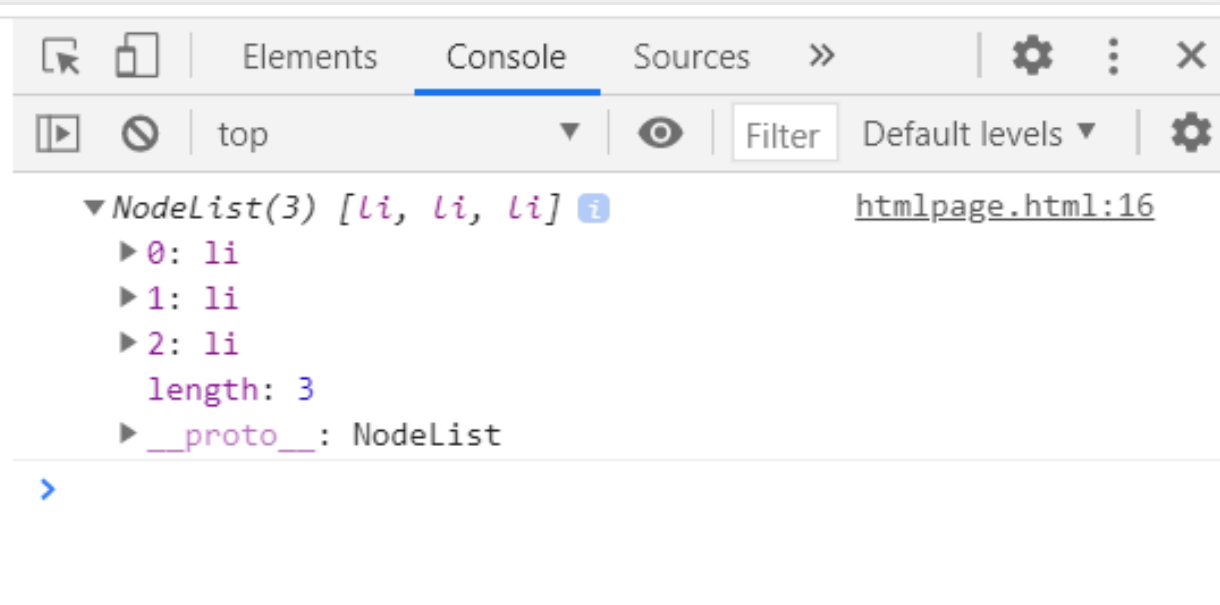
- 1
- 2
- 3




```
<body>
  <ul id="MyId">
    <li>1</li>
    <li>2</li>
    <li>3</li>
  </ul>

  <script>
    let x = document.querySelectorAll("li");
    console.log(x);
  </script>
</body>
```

- 1
- 2
- 3



Изменение объектов DOM

Основные задачи манипулирования элементами DOM

Добавление | Изменение | Удаление

Изменение стиля

Пример манипуляции элементами DOM

```
<body>
  <div id="MyId" class=""></div>

  <script>
    let x = document.querySelector("#MyId");
    x.innerHTML = "Добавленный текст";
    x.className = "badge";
    x.classList.add("bg-primary");
  </script>
</body>
```

Пример манипуляции элементами DOM

```
function insertList() {  
    let myDiv = document.querySelector("#MyId");  
  
    let list = document.createElement('ul');  
    list.className = "bg-danger";  
    list.innerHTML = '<li>1</li><li>2</li>';  
  
    let listItem = document.createElement('li');  
    listItem.innerText = '3';  
    list.appendChild(listItem);  
  
    myDiv.appendChild(list);  
};
```

Обработка событий

Способы привязки событий

Декларативное связывание
Программное связывание

Декларативное связывание

```
<button id="btnShow" onclick="buttonHandler(1)">  
    Show  
</button>  
  
<script>  
    function buttonHandler(id) {  
        alert("Button "+id+" pressed");  
    }  
</script>
```

Программное связывание

```
<button id="btnShow">  
    Show  
</button>
```

```
<script>  
    function buttonHandler() {  
        alert("Button pressed");  
    };  
    let btn = document.querySelector("#btnShow");  
    btn.addEventListener("click", buttonHandler);  
</script>
```

Программное связывание

```
<button id="btnShow">
```

Show

```
</button>
```

```
<script>
```

```
let btn = document.querySelector("#btnShow");
```

```
btn.addEventListener("click", function () {
```

```
    let id = this.attributes["id"].value;
```

```
    alert("Button "+id+" pressed");
```

```
});
```

```
</script>
```

Программное связывание

```
<script>
  document
    .querySelector("#btnShow")
    .addEventListener("click", function () {
      let id = this.attributes["id"].value;
      alert("Button "+id+" pressed");
    });
</script>
```

Недостатки «сырого» JavaScript

- Работа с сырым JavaScript-кодом весьма затруднительна. Различные браузеры имеют свои возможности и ограничения, которые значительно усложняют написание **кросс-браузерного** JavaScript-кода.
- Навигация и манипулирование HTML **DOM** ("объектная модель документа", Document Object Model) также очень трудозатратны и сложны.

Избежать всего вышеперечисленного можно, используя библиотеки JavaScript

jQuery

ВЗАИМОДЕЙСТВИЕ С DOM

jQuery был выпущен Джоном Резигом в 2006. Это одна из самых популярных библиотек JavaScript благодаря простому, но мощному механизму взаимодействия с HTML DOM.

Microsoft, в свою очередь, добавил несколько функций к кодовой базе JQuery, обеспечил официальную поддержку и включил в ASP.NET MVC как часть шаблона проектов по умолчанию.

Библиотека jQuery определяет единственную глобальную функцию с именем `jQuery()`. Эта функция используется настолько часто, что библиотека определяет также глобальное имя `$`, как сокращенный псевдоним этой функции. Эти два имени – все, что библиотека jQuery добавляет в глобальное пространство имен.

• Возможности jQuery

- Доступ к элементам документа посредством DOM
- Изменение внешнего вида документа
- Изменение контента документа
- Ответ на действия пользователя
- Анимация изменений в документе
- Получение информации с сервера без обновления страницы
- Упрощение общих задач JavaScript, таких как итерация и манипулирование массивами.

Доступ к элементам документа посредством DOM

Поиск тэгов <div> :
\$('div')

Поиск элемента с id="MyID" :
\$('#MyID') \$('div #MyID')

Поиск элемента класса MyClass:
\$('.MyClass') \$('a.MyClass')

Поиск элемента <a> внутри элемента id="MyID"
\$('#MyID a')

Цепочки команд

`$('селектор').метод1().метод2()`

Изменение внешнего вида документа

```
$(“селектор”).attr(“название”, “значение”)  
$(“селектор”).prop(“название”, “значение”)  
$(“селектор”).removeAttr(“название”)  
$(“селектор”).removeProp(“название”)
```

```
$(“селектор”).css(“стиль”, “значение”)  
$(“селектор”).css({  
    “стиль” : “значение”,  
    “стиль” : “значение”})
```

```
$(“селектор”).addClass(“имя”)  
$(“селектор”).removeClass(“имя”)  
$(“селектор”).toggleClass(“имя1”, “имя2” )
```

Изменение внешнего вида документа

Выделить цветом и быстро отобразить все скрытые элементы `<p>`, имеющие класс «details»:

```
$("p.details").css("background-color","yellow")  
                .show("fast");
```

Изменить класс элемента:

```
$("ul >li:first").addClass("active")
```

Изменение контента документа

Добавить элемент внутри другого элемента:

`.append()` `.appendTo()` `.prepend()` `.prependTo()`

Добавить элемент рядом с другим элементом:

`.after()` `.insertAfter()` `.before()` `.insertBefore()`

Заменить элемент или его содержимое:

`.html()` `.text()` `.replaceAll()` `.replaceWith()`

Создание нового элемента

```
$ ( '<a/>', {  
    html: "ссылка",  
    "class": "btn btn-default",  
    href: "xxxx"  
} ) ;
```

Обработка событий

`.bind('событие', делегат)`

```
$("#foo").bind("click", function () {  
    alert("текст сообщения");  
});
```

Примечание: элемент DOM должен существовать до выполнения команды

Обработка событий

.on('событие', делегат)

```
$("#dataTable tbody tr")  
  .on("click", function (e) {  
    console.log($(this).text());  
  });
```

Метод .on появился, начиная с версии 1.7, является более предпочтительным, чем более ранний метод .bind

```
<script  
src="lib/jquery/dist/jquery.js"></script>  
<script>  
    $("#btnShow")  
        .on("click", function () {  
            let id = $(this).attr("id");  
            alert("Button "+id+" pressed");  
        });  
</script>
```

События DOM

.ready()
.load()
.unload()

События формы

.change()

.focus()

.select()

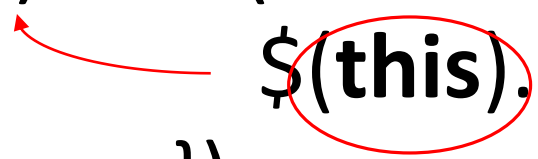
.submit()

.blur()

Ответ на действия пользователя

Щелчок на любом элементе `<p>` окрашивает его фон в серый цвет:

```
$("#p").click(function(){  
    $(this).css("background-color", "gray");  
});
```



Анимация изменений в документе

`fadeIn(), fadeOut(), fadeTo()`

`show(), hide(), toggle()`

`slideDown(), slideUp(), slideToggle()`

Растворить все элементы ``, затем показать их, затем свернуть и развернуть:

```
$("img").fadeOut().show(300).slideUp().slideToggle();
```

Манипулирование массивами

`$.each(collection, callback)`

Манипулирование массивами

```
<script>
  $(document).ready(function ()
  {
    $('button').click(function ()
    {
      $('a').text(function (n, current)
      {
        return '$' + n + ': ' + current;
      })
    })
  })
</script>
```

Коллекция содержимого
тегов <a>

Номер текущего элемента
коллекции

Текущий элемент
коллекции

Создание Ajax-запросов с помощью JQuery

JQUERY

События Ajax

.ajaxComplete()
.ajaxError()
.ajaxSend()
.ajaxStart()
.ajaxStop()
.ajaxSuccess

Вспомогательные функции

jQuery.params (или \$.params)

jQuery.serialize (или \$.serialize)

jQuery.serializeArray (или \$.serializeArray)

Вспомогательные функции

```
var params = {  
    width: 1680, height: 1050 };  
var str = jQuery.param(params);  
$("#results").text(str);
```



width=1680&height=1050

Вспомогательные функции

```
var str = $( "form" ).serialize();
```

Функция jQuery.ajax

```
$ (function () {  
    $.ajax({  
        url: 'xxx',  
        dataType: 'html',  
        data: { id: 1 }  
    }).success(function (myData) {  
        // тело обработчика  
        // принятых данных  
    });  
})
```

Сокращенные методы Ajax

jQuery.get()
jQuerygetJSON()
jQuery.getScript()
jQuery.post()
.load()

Создание Ajax-запросов с помощью JQuery

```
<html>
  <body>
    <h1>Partial Rendering Demo</h1>
    <div id="container" />
  </body>
</html>
```

```
$("#container").load('ajax_content.html')
```

Или внутри Razor

```
$("#container").load(
  '@Url("PartialMethod", "Controller")/ '+id)
```


Получение разметки с сервера без обновления страницы

```
<div id="temp"></div>
  <button id="btnShow">
    Show
  </button>
<script src="lib/jquery/dist/jquery.js"></script>
<script>
  $("#btnShow")
    .on("click", function () {
      $("#temp").load("Partial.html");
    });
</script>
```

Файл Partial.html

```
<h1>data from other file</h1>
```

Создание Ajax-запросов с помощью JQuery

```
<div id="temp"></div>
<a href="Partial.html" id="ancorLoad">Загрузить</a>
<script src="lib/jquery/dist/jquery.js"></script>
<script>
    →$( "#ancorLoad" )
      .on("click", function (e) {
        e.preventDefault();
        var url = $(this).attr("href");
        →$( "#temp" ).load(url);
      })
</script>
```

Elements Console Sources **Network** Performance Memory Application >> | ⚙️ ⋮

⛔ | 🔍 | ☐ Preserve log ☒ Disable cache | No throttling ▾ | ⬆️ ⬇️

Filter ☐ Hide data URLs **All** | XHR JS CSS Img Media Font Doc WS Manifest Other

☐ Has blocked cookies ☐ Blocked Requests

500 ms 1000 ms 1500 ms 2000 ms

Name	× Headers Preview Response Initiator Timing Cookies
htmlpage.html	referer: https://localhost:44397/htmlpage.html
bootstrap.css	sec-ch-ua: "Google Chrome";v="89", "Chromium";v="89", ";Not A Brand";v="99"
jquery.js	sec-ch-ua-mobile: ?0
favicon.ico	sec-fetch-dest: empty
Partial.html	sec-fetch-mode: cors
	sec-fetch-site: same-origin
	user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.114 Safari/537.36
	x-requested-with: XMLHttpRequest

5 requests | 159 kB transferred

Создание Ajax-запросов с помощью JQuery

```
public IActionResult PartialMethod()  
{  
    return PartialView();  
}
```

```
public IActionResult PartialMethod()  
{  
    if(Request.Headers["x-requested-with"]  
        .ToString()  
        .ToLower()  
        .Equals("xmlhttprequest"); )  
    {  
        return PartialView();  
    }  
    return View();  
}
```

Создание Ajax-запросов с помощью JQuery

Пример асинхронной отправки формы:

```
<ul id="comments"></ul>  
<form method="post" id= "MyForm" action="@Url.Action("MyAction")">  
    ....  
    <input type="submit" value= "Отправить" />  
</form>
```

Создание Ajax-запросов с помощью JQuery

```
$(document).ready(function () {  
    $('#MyForm').submit(function (event) {  
        event.preventDefault();  
        var data = $(this).serialize();  
        var url = $(this).attr('action');  
        $.post(url, data, function (response) {  
            $('#comments').append(response);  
        });  
    });  
});
```

Работа с JSON

JQUERY

Работа с JSON

JSON обозначает **JavaScript Object Notation** и представляет собой сжатый способ представления данных. Он часто используется в приложениях, в которых много Ajax, поскольку строки JSON требуют очень мало анализа в JavaScript - можно просто передать строку JSON в функцию `eval`, и JavaScript преобразует его в граф объекта.

Работа с JSON

```
{  
  "firstName": "Иван",  
  "lastName": "Иванов",  
  "address":  
  {  
    "streetAddress": "Московское ш., 101, кв.101",  
    "city": "Ленинград",  
    "postalCode": 101101  
  },  
  "phoneNumbers":  
  [  
    "812 123-1234",  
    "916 123-4567"  
  ]  
}
```

Работа с JSON

```
public async Task<IActionResult>GetJsonBooks()  
{  
    return Json(await _context.Books.ToListAsync());  
}
```

Работа с JSON

Чтобы запросить JSON-данные, просто используется **\$.ajax()** вызов метода контроллера и определяется функция **success** для обработки ответа. Первый параметр этой функции – десериализованный объект, полученный с сервера.

Можно также использовать **\$.getJSON()**

Работа с JSON

```
$.ajax({  
    url: "/Persons/JsonPerson/"  
        + personId,  
    success: function (result) {  
        $('#FirstName').val(result.FirstName);  
        $('#LastName').val(result.LastName);  
        $('#Age').html(result.Age);  
    }  
});
```

Работа с JSON

```
<h1>JsonDemo</h1>  
  <a asp-action="GetJsonBooks" asp-controller="Books"  
    id="add-button"  
    class="btn-outline-danger">Show books</a>  
<div id="content"></div>
```

```
@section Scripts{  
  <script src="~/js/site.js"></script>  
}
```

```
$(document).ready(function () {  
    $("#add-button").click(function (e) {  
        e.preventDefault();  
        let url = $(this).attr("href");  
        $.getJSON(url, function (data) {  
            let target = $("#content");  
            let table = $("<table/>", { class: "table" });  
            $.each(data, function (pos,item) {  
                let row = makeRow(item);  
                table.append(row);  
            });  
            table.appendTo(target);  
        })  
    })  
})
```

```
function makeRow(item) {  
    return "<tr><td>" + item.bookId + "</td><td>" +  
        item.name + "</td><td>" + item.author + "</td></tr>";  
}
```


JsonDemo

Show books



1	Book 1	Author 1
2	Book 2	Author 1
3	Book 3	Author 2