Справочник по AJAX

AJAX расшифровывается Asynchronous JavaScript And XML (Асинхронный JavaScript и XML). AJAX - это не новый язык программирования или разметки. AJAX - это эффективный способ совместного использования HTML, CSS, JavaScript и DOM.

AJAX - это набор технологий, которые поддерживаются веб-браузерами. AJAX использует:

1. HTML в качестве "каркаса";
2. CSS для оформления;
3. DOM для извлечения или изменения информации на странице;
4. Объект XMLHttpRequest для асинхронного обмена данными с сервером;
5. JavaScript для связи перечисленных выше технологий между собой.

Этапы выполнения AJAX запроса:

Шаг №1: Создание экземпляра объекта XMLHttpRequest.
Шаг №2: Оправка запроса на сервер методами open и send.
Шаг №3: Принятие сервером запроса, обработка и отправка ответа.
Шаг №4: Принятие ответа на клиенте с помощью responseText, responseXML (событие onreadystatechange)

XMLHttpRequest - Создание объекта

```
var xhttp;
if (window.XMLHttpRequest) xhttp = new XMLHttpRequest();
else  xhttp = new ActiveXObject("Microsoft.XMLHTTP");
```

Свойства объекта XMLHttpRequest

onreadystatechange - состояние готовности сервера.
readyState - Позволяет узнать состояние готовности сервера.
responseText - Хранит ответ сервера как строку символов.
responseXML - Хранит ответ сервера как XML файл.
status - Хранит код ответа сервера.

Методы объекта XMLHttpRequest

open(*тип_запроса*, *url*, *способ*) - создает запрос.
тип_запроса - устанавливает тип запроса (GET или POST).
url - устанавливает путь к запрашиваемому файлу.
способ - устанавливает способ выполнения запроса. При значении true запрос будет выполнен асинхронно, при false синхронно.

send('данные') - Позволяет передать данные на сервер.

setRequestHeader(*заголовок*, *значение*) - добавить HTTP заголовок к запросу.
заголовок - содержит имя заголовка
значение - содержит значение заголовка

AJAX: Запрос - С помощью методов open() и send() объекта XMLHttpRequest вы можете отправлять AJAX запросы.

```
var xhttp = new XMLHttpRequest();     // Создаем объект XMLHttpRequest
                                      // Запросим содержимое файла testfile.txt
xhttp.open('GET', testfile.txt, true);
xhttp.send();
```

Методы
open(*тип_запроса*, *url*, *способ*) - создает запрос.
тип_запроса - устанавливает тип запроса (GET или POST).
url - устанавливает путь к запрашиваемому файлу.
способ - устанавливает способ выполнения запроса. При значении true запрос будет выполнен асинхронно, при false синхронно.

send('данные') - Позволяет передать данные на сервер.

Метод GET проще и быстрее при запросах, однако POST более надежный и безопасный. Тем не менее мы рекомендуем Вам всегда использовать GET если это возможно.

Запрос GET - Передача данных

```
var x = document.getElementById('tf1').value;
var y = document.getElementById('tf2').value;
xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function()
{
      if (xhttp.readyState == 4 && xhttp.status == 200)
            document.getElementById("ajax").innerHTML = xhttp.responseText;
}
xhttp.open("GET", "add.php?x=" + x + "&y=" + y, true);
xhttp.send();
```

Кэширование результата при GET запросах.

Часто ответ полученный в результате GET запроса не является "свежим" ответом, а является копией, которая была сохранена в кэше при предыдущем обращении к серверу.

К счастью существует способ помогающий избежать этого. Необходимо, чтобы url при GET запросе всегда был уникальным. Передача при запросе в скрипт переменной со случайным значением исключит возможность получения кэшированного значения.
`url?переменная=" + Math.random()`

```
xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function()
{
      if (xhttp.readyState == 4 && xhttp.status == 200)
            document.getElementById("result").innerHTML = xhttp.responseText;
}
xhttp.open("GET", "gettime.php?x="+Math.random(), true);
xhttp.send();
```

Запрос POST

```
xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function()
{
      if (xhttp.readyState == 4 && xhttp.status == 200)
            document.getElementById("result").innerHTML = xhttp.responseText;
}
xhttp.open("POST", "testfile.txt", true);
xhttp.send();
```

Обратите внимание: чтобы отправить данные в формате, в котором передаются данные с HTML форм необходимо задать соответствующий HTTP заголовок с помощью setRequestHeader.

setRequestHeader(*заголовок*, *значение*) - добавить HTTP заголовок к запросу.
заголовок - содержит имя заголовка
значение - содержит значение заголовка

```
var x = document.getElementById('tf1').value;
var y = document.getElementById('tf2').value;

xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function()
{
      if (xhttp.readyState == 4 && xhttp.status == 200)
            document.getElementById("result").innerHTML = xhttp.responseText;
}

xhttp.open('POST', 'addpost.php', true);
xhttp.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');

var str = 'x=' + x + '&y=' + y;
xhttp.send(str);
```

AJAX: Ответ
С помощью свойства readyState Вы можете узнать состояние готовности сервера.

0 - Запрос не инициализирован;
1 - Установлено подключение к серверу;
2 - Запрос получен;
3 - Обработка запроса;
4 - Обработка запроса закончена и ответ готов.

Во время обработки запроса свойство readyState поочередно принимает значения от 0 до 4.

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function()
{
      if (xhttp.readyState == 0){}
      if (xhttp.readyState == 1){}
      if (xhttp.readyState == 2){}
      if (xhttp.readyState == 3){}
```

```
      if (xhttp.readyState == 4)
      {
            console.log(xhttp.responseText);
            console.log(xhttp.responseXML);
      }
}
xhttp.open('GET', 'state.php', true);
xhttp.send();
```

---

status - свойство

С помощью свойства status Вы можете узнать код ответа на Ваш запрос.
Код 200 означает - запрос обработан успешно, код 404 означает - страница не существует.

```
xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function()
{
      if (xhttp.readyState == 4 && xhttp.status == 200)
      {
            alert("OK!");
      }
      else if(xhttp.status == 404)
      {
            alert("Error 404");
      }
}
xhttp.open("GET", "testfile.php", true);
xhttp.send();
```

Принятие ответа сервера responseXML и responseText

Для того, чтобы принять ответ сервера используйте свойства объекта XMLHttpRequest responseText и responseXML.

responseText - Хранит ответ сервера как строку символов.
responseXML - Хранит ответ сервера как XML файл.

Свойство responseText

Если запрошенные у сервера данные не являются XML данными, то для считывания ответа сервера используйте свойство responseText.

```
xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function()
{
      if (xhttp.readyState == 4 && xhttp.status == 200)
            document.getElementById("result").innerHTML = xhttp.responseText;
}
xhttp.open("GET", "testfile.txt", true);
xhttp.send();
```

Свойство responseXML

Если Вы ожидаете получить от сервера XML данные используйте для считывания ответа сервера свойство responseXML.

```
xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function(){
      if (xhttp.readyState == 4 && xhttp.status == 200){
            var r = xhttp.responseXML;

            document.getElementById('ajax').innerHTML =
            "Содержимое первого тэга name в XML файле: "
            + r.getElementsByTagName('name')[0].childNodes[0].nodeValue;

            var dep = r.getElementsByTagName('dept');
            var cont = "Содержимое всех тэгов dept в XML файле: <br>";
            for (var i = 0; i < dep.length; i++)
            {
                  cont += dep[i].childNodes[0].nodeValue + "<br>";
            }
            document.getElementById('ajax1').innerHTML = cont;
      }
}
xhttp.open("GET", "test.xml", true);
xhttp.send();
```

AJAX: JSON / XML

JSON расшифровывается JavaScript Object Notation.

Формат JSON может использоваться с AJAX вместо XML. Поддержка формата JSON встроена в JavaScript. Синтаксис JSON - включает в себя следующие структуры:

Массив

```
[ значение1, значение2, значениеN ]
```

Объект

```
{ имя1:значение1, имя2:значение2, имяN:значениеN }
```

Литералы: (Строка; Число; Логическое значение; Значение null).

```
{ имя1:"строка", имя2:1000, имя3:true, имя4:false, имя5:null }
```

Пример JSON

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

Пример XML

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

Использование JSON

```
xhttp = new XMLHttpRequest();
xhttp.open('GET', 'bookbase.json', true);
xhttp.send();
xhttp.onreadystatechange = function()
{
      if (xhttp.readyState == 4)
      {
      // Принятое содержимое json файла
      // должно быть вначале обработано функцией eval
      var json = eval( '(' + xhttp.responseText + ')' );

      //Далее мы можем спокойно использовать данные
      document.getElementById('wrap').style.dislay = 'block';
      document.getElementById('res1').innerHTML = json.bookbase[1].title;
      document.getElementById('res2').innerHTML = json.bookbase[1].author;
      document.getElementById('res3').innerHTML = json.bookbase[1].cost;
      document.getElementById('res4').innerHTML = json.bookbase[1].quantity;
      }
}
```

AJAX: jQuery

Создание AJAX запросов с помощью jQuery (библиотека JavaScript позволяющая ускорить написание кода).

$("селектор").**load**(*url, данные, функция*)

> *селектор* - выбирает элемент, в котором будет отображен результат AJAX запроса.
>
> *url* - адрес, на который будет отправлен AJAX запрос.
>
> *данные* - является необязательным параметром. Данные, которые будут переданы вместе с запросом.
>
> *функция* - является необязательным параметром. Функция, которая будет вызвана после выполнения запроса.

Способ удаленного использования библиотеки предоставленный компанией Google.

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
```

Пример чтение текстового файла

```
$(document).ready(function(){
      $("#but1").click(function(){
            $("#but1").load("testfile.txt");
      });
});
```

Пример передачи параметров в php файл

```
$(document).ready(function(){
	$("#but1").click(function(){
		$("#but1").load("add.php", "x=4&y=5");
	});
});
```

Сопровождающие функции

Функция **ajaxSend**() - выполнит переданный в нее код во время отправления AJAX запроса.

Функция **ajaxComplete** - выполнит переданный в нее код, когда выполнение AJAX запроса будет завершено, при этом неважно произошла при этом ошибка или нет.

Функция **ajaxSuccess** - выполнит переданный в него код, если выполнение AJAX запроса будет завершено успешно.

Функция **ajaxError** - выполнит переданный в него код, если выполнение AJAX запроса будет завершено с ошибкой.

```
$(document).ready(function(){
	$("#par1").ajaxSend(function(){
		$("#img1").css("display", "block");
	});
	$("#par1").ajaxComplete(function(){
		$("#img1").css("display", "none");
	});
	$("#par1").ajaxError(function(){
		alert("Выполнение AJAX запроса завершено с ошибкой.");
	});
	$("#par1").ajaxSuccess(function(){
		alert("AJAX запрос успешно выполнен!");
	});
	$("#but1").click(function(){
		$("#par1").load("add.php", "x=10&y=100");
	});
});
```

Низкоуровневые AJAX запросы - предоставляюn более широкую функциональность, но более сложны в использовании

Синтаксис:

```
	$.ajax({
		url:"add.php",
		data:"x=4&y=5",
		success:function(result){
			$("#par1").html(result);
		}
	});
```

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script type="text/javascript"
             src="http://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
        <script type="text/javascript" src="script.js"></script>
    </head>
    <body>
        <h1>AJAX TEST</h1>
        <form>
             <label>Name:</label>
             <input type="text" value="" id="textBox" placeholder="enter name">
             <input type="button" value="Send" id="buttonSend">
        </form>
        <h3>Result</h3>
        <div id="showResult"></div>
    </body>
</html>
```

server.php

```
<?php
$value = $_GET['value'];
$jsonResponse = '{"data":{';
$jsonResponse .= '"name":"'.$value.'",';
$jsonResponse .= 'type:"строка"';
$jsonResponse .= '}}';
echo $jsonResponse;
```

script.js

```
var Events = (function() {
    return {
        onButtonClick: function() {
            var value = document.getElementById('textBox').value;

            var xhttp;
            if(window.XMLHttpRequest){
                xhttp = new XMLHttpRequest();
            }else{
                xhttp = new ActiveXObject("Microsoft.XMLHTTP");
            }

            xhttp.onreadystatechange = function () {
                if(xhttp.readyState == 4 && xhttp.status == 200){
                    console.log("VALUE:", xhttp.responseText);
                    var jsonResponse = eval('(' + xhttp.responseText + ')');
                    document.getElementById('showResult').innerHTML = jsonResponse.data.name;
                }
            };
            xhttp.open('GET', 'server.php?value='+encodeURI(value), true);
            xhttp.send();
        }
    }
}());
```

script.js

```
(function () {
    $(document).ready(function(){
        $("#buttonSend").click(function(){
            var value = $("#textBox").val();
            // $("#showResult").load("server.php", "value="+value);
            $.ajax({
                url: "server.php",
                data: "value="+value,
                success: function(result){
                    var jsonResponse = eval('(' + result + ')');
                    $("#showResult").html(jsonResponse.data.name);
                }
            });
        });
    });
}());
```