



WEB ACADEMY

Front End за Напреднали jQuery



```
01 var band = {  
02   "name": "The Red Hot Chili Peppers",  
03   "members": [  
04     {  
05       "name": "Anthony Kiedis",  
06       "role": "lead vocals"  
07     },  
08     {  
09       "name": "Michael 'Flea' Balzary",  
10       "role": "bass guitar, trumpet, backing vocals"  
11     },  
12     {  
13       "name": "Chad Smith",  
14       "role": "drums, percussion"  
15     },  
16     {  
17       "name": "John Frusciante",  
18       "role": "Lead Guitar"  
19     }  
20   ],  
21   "year": "2009"  
22 }
```



За JQuery

5

Какво представлява JQuery:

- JQuery – библиотека надграждаща езика JavaScript
- Създаден от Джон Резиг
- безплатен open source софтуер лицензиран под MIT
- използва се в 55% от 10000 най-посещавани уеб сайта

Какво ни предоставя JQuery:

- опростява достъпа до всеки елемент на страницата
- улеснява изграждането на динамична функционалност
- DOM селекция базирана на разширени CSS селектори
- HTML Събития, AJAX и JSON, ефекти, анимации и други
- Съвместимост с широк кръг браузъри, дори и с IE6 :-)



Какво ни е нужно?

6

- За да овладеем силата на JQuery се нуждаем от:
 - Добро познаване на HTML елементите
 - Много добро познаване на CSS селекторите
 - Отлично познаване на JavaScript езика
- Как да започнем:
 - Свалете желаната от вас версия на JQuery и я заредете чрез `<script>`
 - Ако не желаете да сваляте файловете можете да ги заредите от CDN

```
<!-- Пример за зареждане на JQuery от Google Content Delivery Network -->
...
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js">
</script>
</head>
...
```



Вмъкване на JavaScript



- чрез деклариране на външен файл вътре в `<head>` тага
`<script src="file.js"></script>`
- чрез използване на `<script>` таг
`<script>`
`<!--`
 `alert('Здравейте');`
`-->`
`</script>`
- чрез директно задаване на JS код на ниво елемент
`<i onclick="alert('Здрасти');">Кликнете тук</i>`


Добрата практика изисква:

1. Функциите да се декларират чрез метод 1 или 2
2. На ниво елемент, да се прави извикване на вече дефинираните функции



Променливи



- Няколко думи за променливите:
 - Променливите са динамични и слаботипизирани
 - Декларират се чрез ключовата дума `var` следвана от името
 - Името на променливата може да съдържа букви, цифри, "_" и "\$" 
 - Името на променливата не може да започва с цифра
 - Името на променливата не може да бъде ключова дума или оператор
 - Възможно е присвояване на стойност, заедно с декларацията
 - По подразбиране всяка променлива има стойност **undefined**
- Ами константите? - JavaScript няма константи!

//Валидни имена на променливи: ninja, apples_and_oranges, var56
//Невалидни имена на променливи: 3, var, "test", if, while, 5wtx1

```
var firstVar; // Деклариране на променлива с име firstVar  
var secondVar = 1 // Деклариране и инициализиране на стойност
```



Типове данни



Типове данни поддържани от езика:

- числа - например 1, 2, 3, -3.14 и други
- низове - задават се в "кавички" или 'апострофи'
- логически (булеви) - true или false
- масиви
 - задават се чрез []
 - съдържат списък с елементи
 - елементите могат да бъдат от различни типове
 - възможно е да има вложени масиви

Типове данни

10

/ Примери за типове данни */*

var a = 3; *//Инициализиране на числова променлива:*

var b = "JavaScript iz kuwl"; *//Инициализиране на променлива с низ*

var c = **true**; *//Инициализиране на булева променлива*

//създаване на нов масив:

var empty = []; *//празен масив*

var my_data = [1, 2, 3, "hi", "bye", -2.11];

var array_of_arrays = [[1, 2, 3], [4, 5, 6], "anything else?"]; *//масив от масиви*

//Извеждане на типа на данните в променливите

alert(**typeof** a);

alert(**typeof** b);

alert(**typeof** c);



Camel Case



Historically, programmers have used three ways of joining multiple words into one variable name:

Hyphens:

first-name, last-name, master-card, inter-city.

Underscore:

first_name, last_name, master_card, inter_city.

Camel Case:

FirstName, LastName, MasterCard, InterCity.



In programming languages, especially in JavaScript, camel case often starts with a lowercase letter:

firstName, lastName, masterCard, interCity.



Въвеждане на данни



В JavaScript може да се въвеждат данни по няколко начина:

- чрез диалогов прозорец - `prompt(text, default_text);`
- чрез диалог за потвърждение - `confirm(text);`
- взимане стойността на HTML елемент

```
<html>
<body>
  <input type="text" name="user_name" id="user_name" />
  <script>
    var confirm_value = confirm('Are you sure?');
    var prompt_value = prompt("Please, enter text", "Default Text...");
    var user_name_value = document.getElementById("user_name").value;
  </script>
</body>
</html>
```

- Виж още: [JavaScript HTML Input Examples](#)



Извеждане на данни



JavaScript може да извежда данни по няколко начина

- чрез системно съобщение - `window.alert()`;
- вътре в HTML документ - `document.write()`;
- вътре в HTML елемент - `innerHTML`
- в конзолата на браузъра - `console.log()`

```
<html>
<body>
  <script>
    window.alert(1 + 2);
    document.write(3 + 4);
    document.getElementById("demo").innerHTML = 5 + 6;
  </script>
  <button onclick="document.write(7 + 8)">Try it</button>
</body>
</html>
```



Оператори



JavaScript работи със следните оператори:

- оператор за присвояване =
- празен оператор ;
- аритметични оператори

- събиране +
- изваждане -
- умножение *
- деление /
- деление по модул %
- увеличаване с 1 ++
- намаляване с 1 --

- комбинирани оператори: += , -= , *=, /=
- логически оператори: <, >, <=, >=, ==, !=, ===, !==, &&, ||

Примери:

```
var x = 5;      // assign the value 5 to x
var y = 2;      // assign the value 2 to y
var z = x + y;  // assign the value 7 to z
console.log(x % y);
```

```
txt1 = "John";
txt2 = "Doe";
txt3 = txt1 + " " + txt2;
```

```
x = 5 + 5;
y = "5" + 5;
z = "Hello" + 5;
```



Проверка на условие



В JavaScript има 2 основни начина за проверка на условие:

- if оператор

- синтаксис: `if (условие) { оператори; }else{ оператори; }`
- ако условието е истина, се изпълняват операторите
- ако условието е лъжа, се изпълняват `else операторите`
- скобите `{` и `}` са важни!

- троен оператор

- синтаксис: `(условие) ? оператори; : оператори;`
- ако условието е истина, се изпълняват операторите
- ако условието е лъжа, се изпълняват операторите

Проверка на условие



```
<html>
<body>
<p>Въведете години:</p><input id="age" value="18" />
<button onclick="checkMe()">Try it</button><p id="result"></p>
<script>
function checkMe() {
    var age, voteable;
    age = Number(document.getElementById("age").value);
    if (isNaN(age)) {
        voteable = "Грешни входни данни!";
    } else {
        voteable = (age < 18) ? "Много сте млад!" : "Добре влизай!";
    }
    document.getElementById("result").innerHTML = voteable;
}
</script>
</body>
</html>
```

Избор на вариант



switch конструкция

- Позволява избор на вариант измежду няколко
- Не забравяйте **break; !!!**
- Синтаксис:

```
switch(израз) {  
    case n:  
        code block  
        break;  
    case n:  
        code block  
        break;  
    default:  
        default code block  
}
```

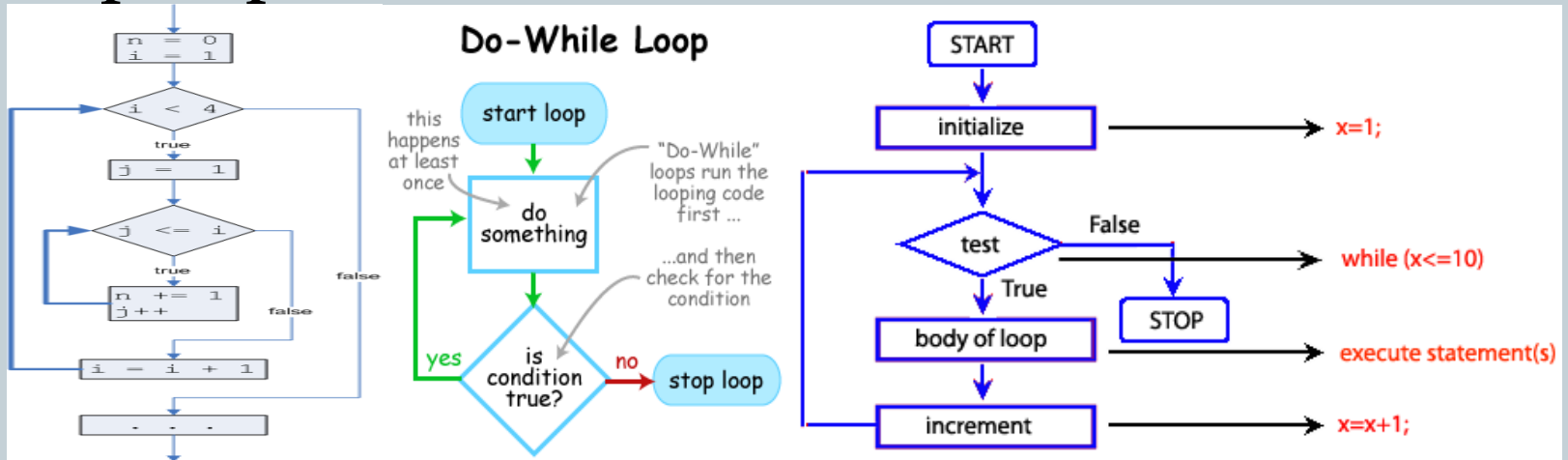
```
switch (new Date().getDay()) { Пример:  
    case 0:  
        day = "Неделя";  
        break;  
    case 1:  
        day = "Понеделник";  
        break;  
    case 2:  
        day = "Вторник";  
        break;  
    default:  
        day = "Непознат ден";  
}
```



Цикли



- оператор for - изпълнява се определен брой пъти
- оператор while - изпълнява се 0 или повече пъти
- оператор do..while - изпълнява се най-малко 1 път



Още по темата:

- http://www.w3schools.com/js/js_loop_for.asp
- http://www.w3schools.com/js/js_loop_while.asp

CSS Селектори



В CSS се използват следните основни селектори:

- елемент (таг) - селектира всички тагове от този тип
- идентификатор (id) - селектира по атрибута id
- клас -селектира по атрибута class
- групов селектор - селектира група от елементи
- Повече информация и списък с всички селектори: [ТУК](#)

Примери:

```
p { color:red; } /* селектира всички параграфи */  
#main { color:red; } /* селектира само 1 елемент с id="main" */  
.bgo{ border:1px solid red; } /* селектира елементите с class="bgo" */  
h1, h2, p { float: right; color: blue } /* селектира изброените елементи */
```



Името на клас, или ID НЕ бива да започва с цифра!



CSS Селектори



В CSS също така, често се използват и други селектори:

Selector	Example	Example description
<u><i>element,element</i></u>	div, p	Selects all <div> elements and all <p> elements
<u><i>[attribute]</i></u>	[target]	Selects all elements with a target attribute
<u><i>[attribute=value]</i></u>	[target=_blank]	Selects all elements with target="_blank"
<u><i>[attribute~=value]</i></u>	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
<u><i>[attribute =value]</i></u>	[lang =en]	Selects all elements with a lang attribute value starting with "en"
<u><i>[attribute^=value]</i></u>	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
<u><i>[attribute\$=value]</i></u>	a[href\$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
<u><i>[attribute*=value]</i></u>	a[href*="WA"]	Selects every <a> element whose href attribute value contains the substring "WA"
<u><i>:not(selector)</i></u>	:not(p)	Selects every element that is not a <p> element
<u><i>:nth-child(n)</i></u>	p:nth-child(2)	Selects every <p> element that is the second child of its parent
<u><i>:nth-of-type(n)</i></u>	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
<u><i>:only-of-type</i></u>	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
<u><i>:only-child</i></u>	p:only-child	Selects every <p> element that is the only child of its parent
<u><i>:optional</i></u>	input:optional	Selects input elements with no "required" attribute
<u><i>:required</i></u>	input:required	Selects input elements with the "required" attribute specified



JQ Синтаксис



Запомнете:

- Синтаксис: `$(selector).action()`
- `$` - задава, че работим с JQuery
- `selector` - представява заявка, чрез която указваме кои елементи искаме да изберем
- `action` - представлява действие или метод, който искаме да осъществим върху избраните елементи

// JQuery примери:

```
$(this).hide();           // Скрива текущия елемент  
$("p").hide();            // Скрива всички <p> елементи, т.е. вси параграфи.  
$(".test").hide();        // Скрива всички елементи, които имат class="test".  
$("#test").hide();        // Скрива елемента с id="test".
```



\$(document).ready



Използването на `$(document).ready()` е добра практика, която ни подсигурява, че :

- програмния код няма да се изпълни преди документа да се е заредил напълно
- няма да работим с елементи, които още не са създадени
- няма да опитваме да вземем размера на изображение, което още се зарежда, т.е. размера му не е известен

```
// Пълен начин на конструкцията  
$(document).ready(function(){
```

```
    // jQuery кода стои тук ...
```

```
});
```

```
// Кратък начин на конструкцията  
$(function(){
```

```
    // jQuery кода стои тук...
```

```
});
```



jQuery Селектори



Какво трябва да запомним за селекторите:

- Указват, кои елементи искаме да селектираме
- Задават се под формата на низове чрез " или '
- Аналогични са със CSS селекторите
- Синтаксис:

```
// Примери за синтаксис на селектори  
$('selector')... // вместо синтаксис пишем  
$(this)... // this е указател към текущия обект
```

Вижте още:

- [пълен списък със селектори](#)
- [тестване на селектори](#)



jQuery Селектори



- Често използвани селектори

Selector	Example	Selects
<u>*</u>	<code>\$("*")</code>	All elements
<u>#id</u>	<code>\$("#lastname")</code>	The element with id="lastname"
<u>.class</u>	<code>\$(".intro")</code>	All elements with class="intro"
<u>.class.class</u>	<code>\$(".intro,.demo")</code>	All elements with the class "intro" or "demo"
<u>element</u>	<code>\$("p")</code>	All <p> elements
<u>el1,el2,el3</u>	<code>\$("h1,div,p")</code>	All <h1>, <div> and <p> elements
<u>:first</u>	<code>\$("p:first")</code>	The first <p> element
<u>:last</u>	<code>\$("p:last")</code>	The last <p> element
<u>:even</u>	<code>\$("tr:even")</code>	All even <tr> elements
<u>:odd</u>	<code>\$("tr:odd")</code>	All odd <tr> elements



jQuery Селектори



- Често използвани селектори

Selector	Example	Selects
<u>*</u>	<code>\$("*")</code>	All elements
<u>#id</u>	<code>\$("#lastname")</code>	The element with id="lastname"
<u>.class</u>	<code>\$(".intro")</code>	All elements with class="intro"
<u>.class.class</u>	<code>\$(".intro,.demo")</code>	All elements with the class "intro" or "demo"
<u>element</u>	<code>\$("p")</code>	All <p> elements
<u>el1,el2,el3</u>	<code>\$("h1,div,p")</code>	All <h1>, <div> and <p> elements
<u>:first</u>	<code>\$("p:first")</code>	The first <p> element
<u>:last</u>	<code>\$("p:last")</code>	The last <p> element
<u>:even</u>	<code>\$("tr:even")</code>	All even <tr> elements
<u>:odd</u>	<code>\$("tr:odd")</code>	All odd <tr> elements



Събития



- Събитията (Events) представляват действие на потребителя. Най-често използвани събития са:

- `change()`
- `click()`
- `dblclick()`
- `focus()`
- `hover()`
- `keyup()`
- `keydown()`
- `keypress()`
- `mouseup`
- `resize()`

- Повече информация: [тук](#)

// Примери за събития

```
$("#p").click(function(){  
    $(this).hide();  
});
```

```
$("#p").dblclick(function(){  
    $(this).hide();  
});
```

```
$("#input").focus(function(){  
    $(this).css("background-color",  
    "#cccccc");  
});
```


DOM Методи



Взимане/Промяна на данни:

- `text()` - връща/задава текстовото съдържание на елемента
- `html()` - връща/задава HTML кода на елемента
- `val()` - връща/задава стойността на елемента
- `attr()` - връща/задава стойността на атрибута
- `data()` - връща стойността на data атрибута

```
$("#btn1").click(function(){  
    alert("Text: " + $("#test").text());  
    alert("HTML: " + $("#test").html());  
});  
$("#btn2").click(function(){  
    alert($("#w3s").attr("href"));  
    alert("Value: " + $("#test").val());  
}); // Пример за взимане на данни
```

```
$("#btn1").click(function(){  
    $("#test").text("new text");  
    $("#test").html("<b>HTML</b>");  
});  
$("#btn2").click(function(){  
    $("#id").attr("href","http://abv.bg");  
    $("#test").val('new value');  
}); // Пример за задаване на данни
```



DOM Методи



Добавяне на данни:

- `append()` - вмъква съдържание в края на елемента
- `prepend()` - вмъква съдържание в началото на елемента
- `after()` - вмъква съдържание след елемента
- `before()` - вмъква съдържание преди елемента

```
$(document).ready(function(){
    $("#btn1").click(function(){
        $("img").before("<b>Before</b>");
        $("img").after("<i>After</i>");
    });
    $("#btn_addNewListItem").click(function(){
        $("ol").append("<li>Appended item</li>");
        $("ol").prepend("<li>Prepended item</li>");
    });
});
```

DOM Методи



Премахване на данни:

- `remove()` - премахва избрания елемент
- `empty()` - премахва "децата" на избрания елемент

```
$(document).ready(function(){  
    $("#div1").remove();  
    $("#div1").empty();  
});
```

Управление на CSS класове

- `addClass()` - добавя CSS клас към избрания елемент
- `removeClass()` - премахва CSS клас, от избрания елемент
- `toggleClass()` - превключва CSS клас към елемента
- `css()` - задава CSS дефиниции на елемента. Има 2 форми:
 - `css("propertyname","value");`
 - `css({"propertyname":"value","propertyname":"value",...});`



DOM Методи



<!-- Пример за работа с CSS -->

```
<style>
.important {
  font-weight: bold;
  font-size: xx-large;
}
.blue { color: blue; }
</style>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("h1, h2, p").addClass("blue");
    $("div").addClass("important");
  });
});
</script>
```



<!DOCTYPE html>

```
<html>
<head>

</head>
<body>
  <h1>Heading 1</h1>
  <h2>Heading 2</h2>

  <p>This is a paragraph.</p>
  <p>This is another paragraph.</p>
  <div>This is important text!</div>
  <button>Add classes</button>
</body>
</html>
```

Взаимовръзки

31

- Възможно е да се предвижваме през елементите на DOM дървото и да селектираме елементи чрез взаимовръзките помежду им.
- С jQuery можете да селектирате лесно ancestors (предци), descendants (наследници), siblings (братя и сестри) и др. членове на "семейното дърво"

ПРИМЕРНО ДЪРВО



Разяснения по примера:

- <div> е родител на и дядо на
- е родител на и дете на <div>
- е родител на , дете на и насл. на <div>

Взаимовръзки

32

- Методи за родители
 - `parent()`
 - `parents()`
 - `parentsUntil`
- Методи за наследници
 - `children()`
 - `find()`
- Методи за братя/сестри
 - `siblings()`
 - `next()`, `nextAll()`, `nextUntil()`
 - `prev()`, `prevAll()`, `prevUntil()`
- Други методи:
 - `first()`, `last()`
 - `filter()`, `not()`, `eq()`

```
<!-- Избираме ul родителя на span -->  
<script>  
$(document).ready(function(){  
    $("span").parents("ul").css({  
        "color": "red", "border": "2px solid red"  
    });  
});  
</script>
```

div (great-grandparent)

ul (grandparent)

li (direct parent)

span



JQuery Ефекти



Методи за създаване на ефекти:

- hide()
- show()
- toggle()
- fadeIn()
- fadeOut()
- fadeToggle()
- fadeTo()
- slideDown()
- slideUp()
- slideToggle()
- animate()
- stop()

```
$(selector).hide(speed,callback);  
$(selector).show(speed,callback);  
$(selector).toggle(speed,callback);  
$(selector).fadeIn(speed,callback);  
$(selector).fadeOut(speed,callback);  
$(selector).fadeToggle(speed,callback);  
$(selector).fadeTo(speed,opacity,callback);  
$(selector).slideDown(speed,callback);  
$(selector).slideUp(speed,callback);  
$(selector).slideToggle(speed,callback);  
$(selector).animate({params},speed,callback);  
$(selector).stop(stopAll,goToEnd);
```

```
$("#button").click(function(){  
    $("#div1").fadeTo("slow", 0.15);  
});
```



Обхождане на елементи



Елементи най-често се обхождат чрез 2 метода:

- `$(selector).each()` - служи за обхождане на JQuery обекти
- `$.each()` - служи за обхождане на всякаква колекция елементи, без значение дали са обект или масив

Синтаксис:

- `$.each(array_or_object, callback_function)`
- `$(selector).each(callback_function)`

```
$( "div" ).each(function( index, element ) { // element == this
    $( element ).css( "backgroundColor", "yellow" );
    if ( $( this ).is( "#stop" ) ) {
        $( "span" ).text( "Stopped at div index #" + index );
        return false;
    }
});
```



Change colors

Stopped at index #4



Обхождане на елементи



Няколко думи за callback функциите:

- могат да бъдат безименни (т.нар. "лямбда") функции
- параметрите на функцията зависят от типа на данните

Синтаксис на функцията според данните:

- `$.each(array, function(indexInArray, value))`
- `$.each(object, function(propertyName, value))`
- `$(selector).each(function(index, element))`

```
$.each([ 52, 97 ], function( index, value )  
{  
  alert( index + ": " + value );  
});
```

// Вляво: Пример за работа с масив
// Вдясно: Пример за работа с обект

```
var obj = {  
  "flammable": "inflammable",  
  "duh": "no duh"  
};  
$.each( obj, function( key, value ) {  
  alert( key + ": " + value );  
});
```



- AJAX - съкр. от Asynchronous JavaScript And XML. AJAX не е програмен език или библиотека - той е технология
- Позволява обмен на данни със сървъра, както и да се зарежда съдържание, без презареждане на страницата
- Комбинация от: xHTML, CSS, DOM, XMLHttpRequest
- Формат за пренос на данни: HTML, XML, JSON, текст ...
- Можем да направим AJAX и без JQuery, но е доста магическо, трудоемко и в зависимост от брауъра :)
- Вижте още: Подробен AJAX наръчник

- **Предимства:**

- Няма нужда от презареждане на страницата
- Времето за зареждане се скъсяват значително
- Файловете се зареждат веднъж, което намаля броя заявки към сървъра
- Чрез Java Script променливи може да се запази текущото състояние

- **Недостатъци:**

- Не се поддържа от стари браузъри, и някои мобилни телефони
- При натискане на Back бутона на браузъра приложението не се връща в предишното състояние, а в състоянието, което е при зареждането му
- При запазване на отметка в браузъра се запазва отметка към приложението в началното му състояние, а не към текущото състояние
- Повечето търсачки не изпълняват код на JavaScript, поради което не цялото съдържание на сайтове с Ajax се индексира коректно

АJAX Методи



- Методът load() е прост, но мощен АJAX метод
- взема данни от сървъра и ги вмъква в елемент
- Синтаксис: \$(selector).load(URL,data,callback);

// Съдържание на demo_test.txt

<h2>jQuery and AJAX is FUN!!!</h2>

<p id="p1">This is some text in a paragraph.</p>

```

$("button").click(function(){
    $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr){
        if(statusTxt == "success")
            alert("External content loaded successfully!");
        if(statusTxt == "error")
            alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
});
    
```

АJAX Методи



- AJAX методите `get()` и `post()` се използват за да се изпрати заявка за изискваните данни към сървъра
- Синтаксис:
`$.get(URL,callback);`
`$.post(URL,data,callback);`

```
$("#button").click(function(){
    $.post("demo_test_post.asp",
    { // Тук се задават параметрите
        name: "Donald", city: "Duckburg"
    }, // Име на параметър: Стойност
    function(data, status){
        alert("D: " + data + "\nS: " + status);
    });
}); //Данните се изпращат в data
```

```
$("#button").click(function(){
    $.get(
        "demo_test.asp",
        function(data, status){
            alert("D: " + data + "\nS: " + status);
        });
});
// callback е функцията, която чете
// върнатите от сървъра данни
```

AJAX Методи

Method	Description
<u>\$.ajax()</u>	Performs an async AJAX request
<u>\$.ajaxPrefilter()</u>	Handle custom Ajax options or modify existing options before each request is sent and before they are processed
<u>\$.ajaxSetup()</u>	Sets the default values for future AJAX requests
<u>\$.ajaxTransport()</u>	Creates an object that handles the actual transmission of Ajax data
<u>\$.get()</u>	Loads data from a server using an AJAX HTTP GET request
<u>\$.getJSON()</u>	Loads JSON-encoded data from a server using a HTTP GET request
<u>\$.getScript()</u>	Loads (and executes) a JavaScript from a server using an AJAX HTTP GET request
<u>\$.param()</u>	Creates a serialized representation of an array or object (can be used as URL query string for AJAX requests)
<u>\$.post()</u>	Loads data from a server using an AJAX HTTP POST request
<u>ajaxComplete()</u>	Specifies a function to run when the AJAX request completes
<u>ajaxError()</u>	Specifies a function to run when the AJAX request completes with an error
<u>ajaxSend()</u>	Specifies a function to run before the AJAX request is sent
<u>ajaxStart()</u>	Specifies a function to run when the first AJAX request begins
<u>ajaxStop()</u>	Specifies a function to run when all AJAX requests have completed
<u>ajaxSuccess()</u>	Specifies a function to run when an AJAX request completes successfully
<u>load()</u>	Loads data from a server and puts the returned data into the selected element
<u>serialize()</u>	Encodes a set of form elements as a string for submission
<u>serializeArray()</u>	Encodes a set of form elements as an array of names and values



```
{ "employees": [  
  { "firstName": "John", "lastName": "Doe"},  
  { "firstName": "Anna", "lastName": "Smith"},  
  { "firstName": "Peter", "lastName": "Jones"}  
]}
```

JSON



Какво представлява JSON?

- съкращение от JavaScript Object Notation
- лесна за използване алтернатива на XML
- подмножество на JavaScript синтаксиса
- спецификация създадена от Дъглас Крокфорд
- поддържа се от всички съвременни уеб браузъри

Ами JSON файловете?

- тяхното файловото разширение е ".json"
- техният MIME тип е "application/json"



```
{ "employees": [  
  { "firstName": "John", "lastName": "Doe"},  
  { "firstName": "Anna", "lastName": "Smith"},  
  { "firstName": "Peter", "lastName": "Jones"}  
]}
```

JSON vs XML



Прилики:

- и двата формата са описателни и лесно четими от хора
- и двата формата са йерархични - стойност със стойности
- и двата формата се извличат чрез XMLHttpRequest
- и двата формата могат да бъдат използвани от различни програмни езици

Разлики:

- JSON е по кратък
- JSON не използва тагове
- JSON е по бърз за четене и писане
- JSON се използва най-често при AJAX заявки
- XML се обработва от XML парсер, а JSON от JS функция




```
{ "employees": [  
  { "firstName": "John", "lastName": "Doe" },  
  { "firstName": "Anna", "lastName": "Smith" },  
  { "firstName": "Peter", "lastName": "Jones" }  
]}
```

JSON Синтаксис



Синтактични правила:

- дефинициите се задават чрез "name" : "value"
- дефинициите се разделят чрез "," (запетая)
- [квадратни скоби] представляват масив
- {фигурни скоби} представляват обект
- задължително се използват "кавички"

JSON стойности могат да бъдат:

- числа (цели и реални числа)
- низове (зададени в кавички)
- логически (true или false), null
- масиви (зададени в [квадратни скоби])
- обекти (зададени във {фигурни скоби})



```
{ "employees": [
  { "firstName": "John", "lastName": "Doe" },
  { "firstName": "Anna", "lastName": "Smith" },
  { "firstName": "Peter", "lastName": "Jones" }
]}
```

JSON Синтаксис



Примери:

"firstName": "John"

{ "firstName": "John", "lastName": "Doe" } // Обект с 2 полета

"employees": [// Масив от 3 елемента тип Object

{ "firstName": "John", "lastName": "Doe" },

{ "firstName": "Anna", "lastName": "Smith" },

{ "firstName": "Peter", "lastName": "Jones" }

]

// Достъпване на елемент. (И двата варианта са верни и допустими)

employees[0].firstName + " " + employees[0].lastName; // Връща John Doe

employees[0]["firstName"] + " " + employees[0]["lastName"]; // Като горното

// Промяна стойността на елемент (И двата варианта са верни и допустими)

employees[0].firstName = "Gilbert";

employees[0]["firstName"] = "Gilbert";



```
{ "employees": [
  { "firstName": "John", "lastName": "Doe" },
  { "firstName": "Anna", "lastName": "Smith" },
  { "firstName": "Peter", "lastName": "Jones" }
]}
```

JSON Методи



Често използвани методи за работа с JSON:

- `JSON.parse(text)` - конвертира JSON текст в JS обект
- `JSON.stringify(value)` - превръща стойност в JSON текст

```
var text = '{ "emp" : [ ' +
  '{ "op1": "Дака", "op2": "Енев" }, ' +
  '{ "op1": "Гали", "op2": "Пътева" }, ' +
  '{ "op1": "Деси", "op2": "Енева" } ] }';

var obj = JSON.parse(text);
```



```
<p id="demo"></p>
<script>
var d = document;
d.getElementById("demo").innerHTML =
obj.emp [1].op2;
</script>
```

Старите браузъри, които не поддържат `JSON.parse()`, биха могли да използват функция `eval()`, която изпълнява всеки JavaScript код. Тази JavaScript функция е сериозна дупка в сигурността, затова **НЕ Я ИЗПОЛЗВАЙТЕ!** ⚡



ВЪПРОСИ

46

