

# AJAX

# Áttekintés

- ▶ Bevezetés
- ▶ Működési elv
- ▶ AJAX-ot támogató keretrendszerek

# AJAX

## AJAX:

Asynchronous JavaScript and XML

az alábbi technológiákon alapul:

- ▶ (X)HTML, CSS
- ▶ XML
- ▶ JavaScript

- ▶ 2005-ben vált népszerűvé a Google révén (Google Suggest)

## Motiváció:

- ▶ igény az interaktív web-alkalmazásra
- ▶ a klasszikus web-alkalmazás sok szempontból nem felel meg ennek az igénynek – a teljes oldal frissítése minden kérés/válasz esetén
- ▶ a kliensek előnyben részesítenek egy böngészőben futó web-alkalmazást egy specializált desktop-alkalmazással szemben
  - ▶ nem kell kliens oldali alkalmazást telepíteni
  - ▶ könnyebb karbantartás

## Klasszikus webalkalmazás – AJAX-alapú webalkalmazás

### klasszikus webalkalmazás (szinkron)

- ▶ a kliens HTTP kérést küld a szervernek – egy web-erőforrás lekérésére (tipikusan GET vagy POST)
- ▶ a szerver feldolgozza a kérést, és előkészíti a választ
- ▶ a szerver visszaküldi a választ (tipikusan (X)HTML)
- ▶ kliens oldalon a TELJES oldal frissül (akkor is, ha annak egy részén egyáltalán nem történt változás)
- ▶ nagy (részben felesleges) adatforgalom, hosszabb várakozási idő

### aszinkron modell (AJAX)

- ▶ egy JavaScript esemény hatására HTTP kérés küldődik (aszinkron módon) a szerverre
- ▶ a szerver feldolgozza a kérést, és előkészíti a választ
- ▶ a visszaküldött (szöveges vagy XML formátumú) választ a kezelő függvény értelmezi, és ennek alapján aktualizálja az oldal megfelelő részeit

## Más alternatívák a szerverrel való aszinkron kommunikációra

- ▶ Java applet
- ▶ IFrame
- ▶ \* \* \*
- ▶ Flex (Adobe)
- ▶ Silverlight (Microsoft)
- ▶ JavaFX (Sun)

## Az XMLHttpRequest API, illetve objektum

- ▶ nem standard, de a legtöbb böngésző támogatja (böngészőfüggő eltérések)
- ▶ használható JavaScript, Jscript, VBScript-ből
- ▶ segítségével aszinkron kapcsolat hozható létre a kliens és szerver között
- ▶ a kérés feldolgozását követően a szerver válasza lehet:
  - ▶ egyszerű szöveg
  - ▶ XML
  - ▶ objektum (JSON jelöléssel megadva)

## XMLHttpRequest - folyt. 1

### Az XMLHttpRequest objektum metódusai:

- ▶ open( method, URL )
- ▶ open( method, URL, async )
- ▶ open( method, URL, async, userName )
- ▶ open( method, URL, async, userName, password )
- ▶ send( content )
- ▶ getResponseHeader( headerName )
- ▶ setRequestHeader( label, value )
- ▶ getAllResponseHeaders()
- ▶ abort()

### Az XMLHttpRequest objektum mezői:

- ▶ readyState:
  - ▶ 0 – a kérés még nincs inicializálva
  - ▶ 1 – a kérés inicializálva van
  - ▶ 2 – a kérés el lett küldve
  - ▶ 3 – a kérés feldolgozás alatt áll
  - ▶ 4 – megérkezett a válasz
- ▶ onreadystatechange – ennek értékeként kell megadni a választ kezelő függvény nevét, mely meg fog hívódni a readyState minden egyes változásakor
- ▶ status – a válasz HTTP kódja (200 = "OK")
- ▶ statusText – a HTTP válasz kódjának szöveges változata
- ▶ responseText – a válasz karaktersorozatként
- ▶ responseXML – a válasz XML formájában

### JavaScript

- ▶ a HTTP kérések küldéséért/válasz fogadásáért felelős speciális objektum lekérése (**XMLHttpRequest**)
- ▶ a kérés inicializálása (a kérés objektum segítségével):
  - ▶ a választ fogadó függvény kijelölése
    - ▶ a kérés objektum **onreadystatechange** attribútumának beállítása
  - ▶ GET (vagy POST) kérés inicializálása (**open** fg.)
  - ▶ adat elküldése (**send** fg.)
- ▶ a válasz kezelése:
  - ▶ várakozás **readyState==4**-re (illetve HTTP 200 válaszra)
  - ▶ válasz kinyerése **responseText** (vagy **responseXML**) segítségével
  - ▶ válasz feldolgozása

### HTML

- ▶ JavaScript kód betöltése
- ▶ a kérést generáló HTML elem/esemény kijelölése

## Kérés objektum lekérése

```
var xmlhttp;
function getRequestObject()
{
    if (window.XMLHttpRequest)
    { // IE7+, Firefox, Chrome, Opera, Safari
        return(new XMLHttpRequest());
    }
    else if (window.ActiveXObject)
    { // IE6, IE5
        return(new ActiveXObject("Microsoft.XMLHTTP"));
    }
    else
    { // a böngésző nem támogatja egyik típusú objektumot sem
        return(null);
    }
}
```

## Kérés inicializálása

```
function sendRequest(url)
{
    xmlhttp=getRequestObject();
    // a választ kezelő handler beállítása:
    xmlhttp.onreadystatechange=handleResponse;
    xmlhttp.open("GET",url,true);
    xmlhttp.send(null);
}
```

### az *open* és *send* függvények paraméterei:

- ▶ open paraméterei: metódus (GET, POST, PUT), szerver-oldali erőforrás URL-je, true=aszinkron kéreseküldés
- ▶ send paraméterei: POST adat (GET esetében null)

## A válasz kezelése

```
function handleResponse() {
  if(xmlhttp.readyState==4) {
    // A szerverről érkező válasz kinyerése (responseText adattag értéke)
    alert(xmlhttp.responseText); }
}
```

egyszerű példa (szerver oldali alkalmazás nélkül):

lásd: ajaxExample.htm, ajaxExample.js

## GET, POST pl., szerver-oldalon PHP

- ▶ AjaxSuggest.htm, clienthint.js, (gethint.php)
- ▶ AjaxSuggest\_post.htm, clienthint\_post.js, (gethint\_post.php)

Servlet-es pl.

ajaxTest.htm, map-elés: /showTime.do

## megjegyzések

annak elkerülése, hogy a böngésző a cache-ből töltsse be a kért URL-t

- ▶ válasz fejlécének beállítása:  
Response.CacheControl = "no-cache";  
Response.AddHeader("Pragma", "no-cache");  
Response.Expires = -1;  
... IE-ben nem mindig működik ...
- ▶ változó érték (pl. véletlen szám vagy az aktuális dátum) küldése az URL-ben

POST-al küldött adatok esetén:

- ▶ a *send* metódus paramétereként adjuk meg a küldött adatokat (pl. `send("val1=ertek1&valt2=ertek2")`)
- ▶ küldés előtt header-információ beállítás(ok)ra van szükség:  
`http.setRequestHeader("Content-type", "application/x-www-form-urlencoded");`

## XML alapú válasz feldolgozása

- ▶ a kérésobjektum `responseXML` mezője XML-ként tartalmazza a választ
- ▶ ennek feldolgozása XML DOM segítségével történik

Servlet-es pl.

setuserxml.htm, map-elés: /XMLResponse.do

ha a válasz JSON jelöléssel megadott objektum

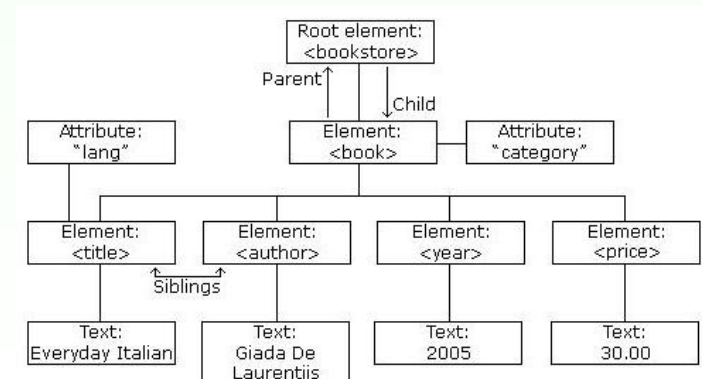
- ▶ a JSON kifejezést tartalmazó szöveget a `responseText` mezőből nyerjük ki
- ▶ a kifejezés kiértékelhető az **eval** függvény segítségével, vagy egy specializált JSON feldolgozóval
- ▶ lásd még: <http://www.json.org/>

pl. (a választ egy statikus állományból olvassa)

ajaxExample\_JSON.htm, ajaxExample\_JSON.js

## XML DOM

- ▶ XML DOM: XML Document Object Model – az XML dokumentumok feldolgozásához biztosít egy standard API-t.
- ▶ a DOM az XML dokumentumot egy fa-szerkezet formájában ábrázolja, melynek csomópontjai az elemek, attribútumok, illetve szövegrészek.



## jellemzők:

- ▶ az XML DOM (Document Object Model for XML) – objektum modellt definiál az XML dokumentumhoz
- ▶ az XML DOM platform- illetve nyelvfüggetlen
- ▶ az XML DOM standard hozzáférésmódot biztosít az XML dokumentumokhoz (olvasás, módosítás)
- ▶ az XML DOM W3C standard

## hozzáférés az egyes csomópontokhoz:

- ▶ `getElementsByTagName("tag-nev")` metódus segítségével – csomópontok listáját téríti vissza
- ▶ `parentNode`, `firstChild`, `lastChild` mezőket használva
- ▶ gyökér elem: `document.documentElement`

## információ az illető csomópontról az alábbi mezőkben:

- ▶ `nodeName`
  - ▶ `nodeValue`
  - ▶ `nodeType`
- 
- ▶ egy elem *attributes* mezője az attribútumokat tartalmazza map formájában (`NamedNodeMap`)

## AJAX keretrendszerek

### AJAX keretrendszer

- ▶ AJAX-ot használó web-alkalmazás fejlesztését segítő eszköz

### Keretrendszer típusok

- ▶ közvetlen AJAX-keretrendszerek
- ▶ közvetett AJAX-keretrendszerek – magasszintű programozási nyelven (pl. Java, Python) írt kód JavaScript-é lesz fordítva (pl. GWT)
- ▶ AJAX komponens-keretrendszerek
  - ▶ kész komponenseket kínál fel (pl. fülekkel (tab) ellátott lapok, naptár, fa-nézet, drag-and-drop lehetőség)
- ▶ AJAX-ot támogató funkciókkal ellátott szerver oldali keretrendszerek

### Néhány népszerűbb AJAX (komponens)-keretrendszer

- ▶ jQuery
- ▶ MooTools
- ▶ Prototype, Script.aculo.us
- ▶ YUI Library
- ▶ ASP.NET AJAX
- ▶ Spry framework (Adobe)
- ▶ Dojo Toolkit
- ▶ Ext JS

# jQuery

- ▶ JavaScript függvénykönyvtár (elég népszerű, nyílt forráskódú, ingyenes)
- ▶ tömör szintaxis ("write less, do more")

## letöltés (tömörített vagy olvasható formában):

[http://docs.jquery.com/Downloading\\_jQuery#Download\\_jQuery](http://docs.jquery.com/Downloading_jQuery#Download_jQuery)

## használata:

- ▶ a letöltött .js állomány HTML oldalba való beszurása
- ▶ jQuery kód script elembe ágyazva (tipikusan a head elembe)

# jQuery - szintaxis

## alapvető szintaxis:

`$(szelektor).művelet()`

- ▶ \$ – jQuery szintaxis része
- ▶ szelektor – kiválasztja a megfelelő HTML eleme(ke)t
- ▶ művelet – a kiválasztott eleme(ke)n végrehajtásra kerülő művelet (action)

## annak elkerülésére, hogy a jQuery kód az oldal betöltése előtt hajtódjon végre:

```
$(document).ready(function(){  
    // jQuery függvény...  
});
```

# jQuery – szelektorok

## CSS szelektorok HTML elemek, attribútumok kiválasztására (néhány példa):

- ▶ `$("p")` – az összes p elem kiválasztása
- ▶ `$(".p.piros")` – a class="piros" stílusosztályhoz tartozó p elemek kiválasztása
- ▶ `$("#p#elso")` – p elem, melynek id attribútuma "elso"
- ▶  `$("a[href]")` – az összes href attribútummal rendelkező elem

## szelektorok (referencia):

- ▶ [http://www.w3schools.com/jquery/jquery\\_ref\\_selectors.asp](http://www.w3schools.com/jquery/jquery_ref_selectors.asp)

# jQuery – AJAX függvények

- ▶ `$(selector).load(url,data,callback)` – (távoli) adat betöltése a kiválasztott elemekbe
- ▶ `$.ajax(options)` – adat betöltése egy XMLHttpRequest objektumba
- ▶ `$.get(url,data,callback,type)` – adat betöltése HTTP GET segítségével
- ▶ `$.post(url,data,callback,type)` – adat betöltése HTTP POST segítségével
- ▶ `$.getJSON(url,data,callback)` – JSON kifejezéssel megadott adat betöltése HTTP GET segítségével
- ▶ `$.getScript(url,callback)` – (távoli) JavaScript állomány betöltése és végrehajtása

## bővebb leírás:

<http://api.jquery.com/category/ajax/>

## Példák - jQuery + AJAX

AjaxSuggest\_jquery.htm, gethint\_post.php

két egyszerű (szerver oldali programot nem igénylő) példa:

- ▶ egyszerű\_csere.htm, ajax\_szoveg.txt
- ▶ egyszerű\_JSON\_pl.htm, resp\_JSON