WEBFEJLESZTÉS 2. – AJAX

Horváth Győző

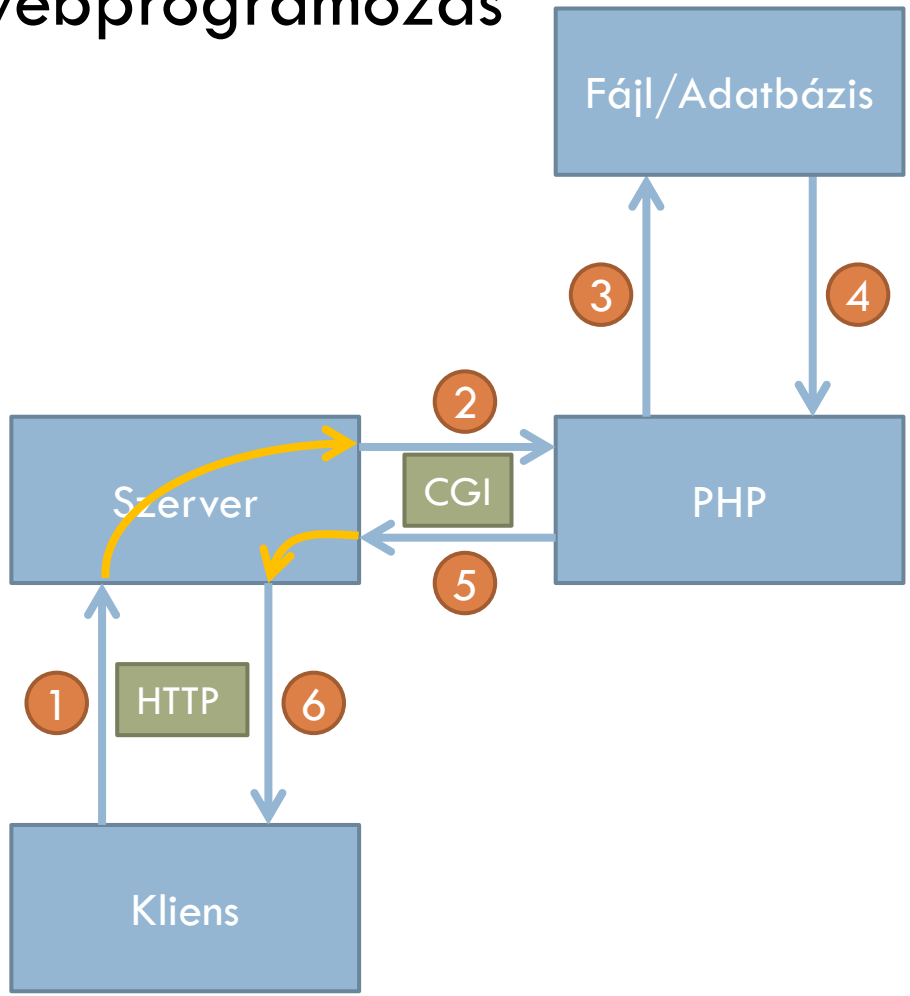
Egyetemi adjunktus

1117 Budapest,
Pázmány Péter sétány 1/C, 2.420
Tel: (1) 372-2500/1816

Ismétlés

Ismétlés

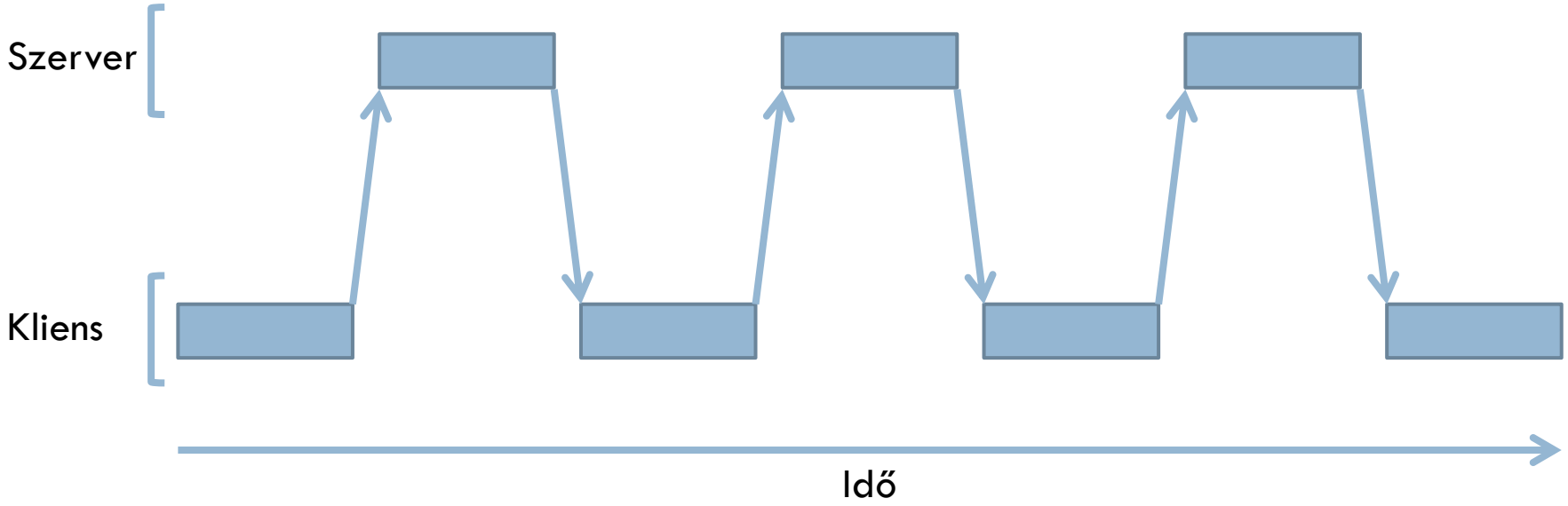
- Dinamikus szerveroldali webprogramozás
- Output (HTML generálás)
- Input (link, űrlap)
- Adattárolás: fájlok
- Kódszervezés
- Munkamenet-kezelés
- Hitelesítés
- Fájlfeltöltés



AJAX

Hagyományos oldalak

- HTTP kapcsolat
- Kérés-válasz
- Mindig a böngésző kezdeményez
- Feldolgozás: kliensen vagy szerveren
- Időben picit eltolva



Következmény

- Állandóan frissül a böngészőbeli oldal
- Szaggatott folyamat
- „Villog”
- Kényelmetlen
- Oldal tetejére ugrik
- Néha feleslegesen nagy adattömeg közlekedik

Megoldás

- A kapcsolatfelvétel a szerverrel szükséges
- Csak a szükséges adatok továbbítása a háttérben, a teljes oldal újratöltése nélkül
- → Remote scripting
  - iframe
  - Java applet
  - XMLHttpRequest objektum
- 90-es évek végi technológiák

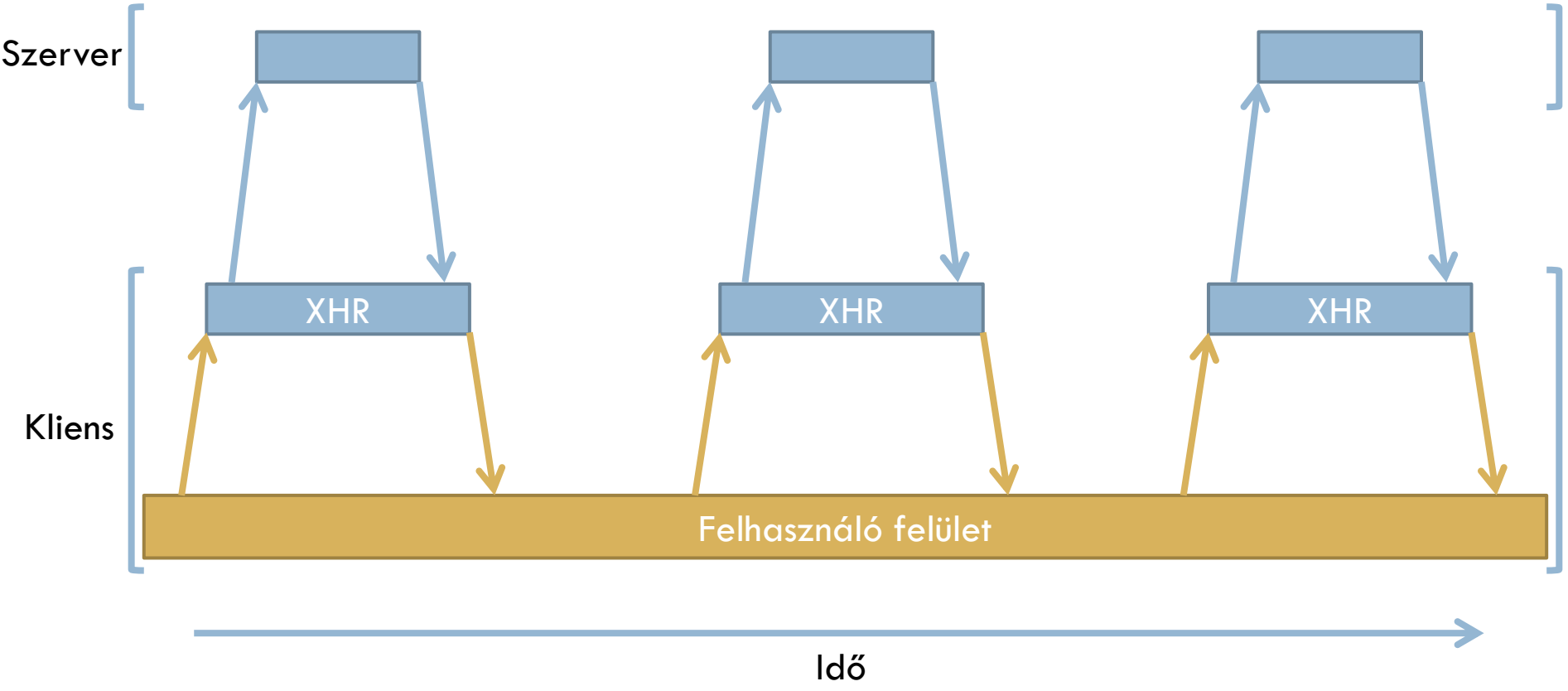
Szélesebb körű elterjedtség

- Sok tényező együttállása
  - Egyre nagyobb internetpenetráció
  - Egyre több webes alkalmazás
  - Szélesebb társadalmi rétegek kapcsolódnak be
  - Nagyobb igények a webes alkalmazások iránt
  - Úttörő, innovatív vállalatok (Google – GMail, GMaps, GDocs, stb.)
- Nevet kap az XMLHttpRequest-es technológia
- → AJAX (2005, Jesse James Garrett)

AJAX

- Aszinkron JavaScript és XML
- Nem új technológia
- Meglévő, kiforrott, szabványos technológiák együttese
  - HTML, XHTML, CSS – megjelenítés
  - DOM – dinamikus felhasználói felület, interakció
  - XML, XSLT – adatleíró formátum
  - XMLHttpRequest – aszinkron kliens-szerver adatátvitel
  - JavaScript – ezeket irányító programozási nyelv

AJAX – oldalkiszolgálás



AJAX-os oldal tulajdonságai

- A felhasználói felület folyamatosan használható
- Nincs szaggatottság, villogás, ugrálás
- A szerverrel való kommunikáció a háttérben történik
- Aszinkron módon, azaz párhuzamosan a többi eseménnyel
- Csak a szükséges adatok közlekednek a szerver és kliens között

AJAX hívás

- HTTP kommunikáció
  - url megadása
  - GET és POST adatok küldése
  - válasz feldolgozása
- Nem a böngésző végzi a HTTP kapcsolat kialakítását, hanem programból vezéreljük
- → XMLHttpRequest objektum

XMLHttpRequest: metódusok

- open("method", "URL", async): hívási paraméterek beállítása
- send([data]): kérés küldése (opc. data a kéréstörzsben)
- abort(): kérés leállítása.
- getAllResponseHeaders(): HTTP fejlécek visszaadása szövegként
- getResponseHeader("fejléc"): adott fejléc értéke
- setRequestHeader("fejléc", "érték"): kérésfejléc beállítása

XMLHttpRequest: tulajdonságok

- readyState: a kérés aktuális státusza
  - 0 = uninitialized
  - 1 = loading
  - 2 = loaded
  - 3 = interactive (néhány adat érkezett)
  - 4 = complete
- status: a HTTP válasz státuszkódja, pl. 200
- statusText: a HTTP válasz státusza szövegesen, pl. OK
- responseText: a szerverről érkezett válasz szövegként
- responseXML: ha a válasz XML dokumentum volt, akkor annak XML DOM dokumentuma.

XMLHttpRequest: események

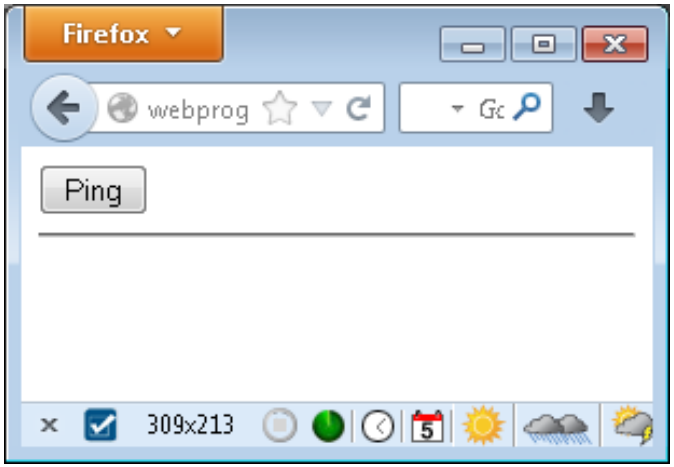
- readystatechange: a readyState állapot változásainál hívódik meg

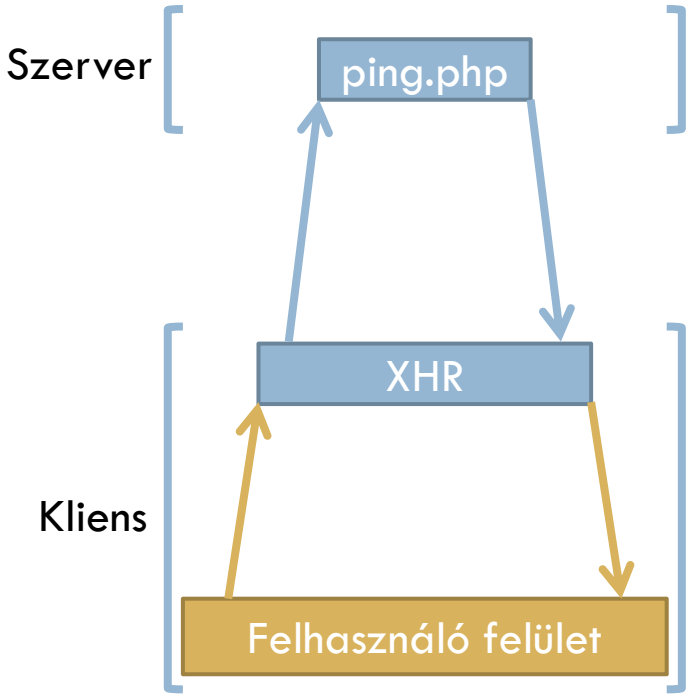
Példa lépésről lépésre

Kódevolúció

Példa

- Ping
  - szerver megszólítása
  - válaszban az aktuális idő érkezik vissza
- Felhasználói felület
  - gomb
  - div – kimenet





Hagyományos megoldás


```php
<?php
print_r($_GET);
print_r($_POST);
$ido = date('Y.m.d. G:i:s');
?>
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>AJAX példa</title>
  </head>
  <body>
    <form action="pingphp.php" method="get">
    <input type="submit" id="gomb" value="Ping">
    </form>
    <hr>
    <div id="output">
      <?php echo $ido; ?>
    </div>
  </body>
</html>
```

AJAX-osítás: kellékek

ping.html

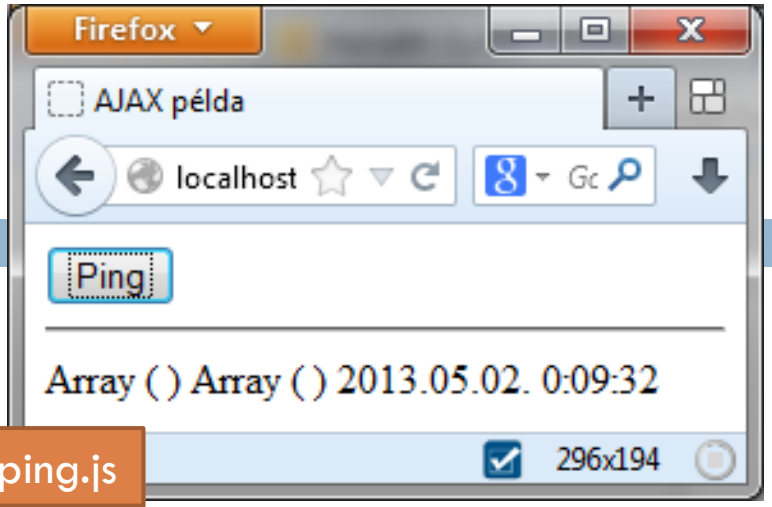
```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>AJAX példa</title>
    <script type="text/javascript" src="ping.js"></script>
  </head>
  <body>
    <input type="button" id="gomb" value="Ping">
    <hr>
    <div id="output"></div>
  </body>
</html>
```

ping.php

```
print_r($_GET);
print_r($_POST);

echo date('Y.m.d. G:i:s');
```

Szinkron megoldás

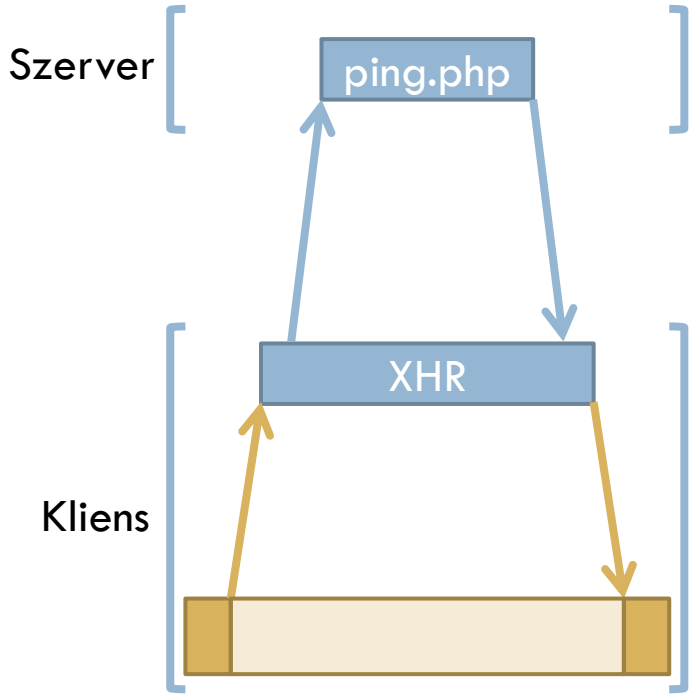


ping.js

```
//Segédfüggvények
function $(id) {
  return document.getElementById(id);
}
//Oldal betöltésekor lefutó függvény
function init() {
  $('gomb').onclick = ping;
}
window.addEventListener('load', init, false);
//A gomb lenyomásakor lefutó függvény
function ping() {
  var xhr = new XMLHttpRequest();
  xhr.open('GET', 'ping.php', false);
  xhr.send(null);
  $('output').innerHTML = xhr.responseText;
}
```

Szinkron megoldás

- Szinkron = akkor folytatódik a szkript futása, ha a válasz megérkezett (ld. alert)
- Ha sokára érkezik válasz → egy ideig nem használható a felület
- Szaggatott élmény
- → Aszinkronitás



Aszinkron kommunikáció

- Az elküldést követően a szkript továbbfut
- Eseményen keresztül értesülünk a válasz megérkezéséről
- → readyState tulajdonság változik
- → readystatechange eseménykezelőt kell írnunk

Aszinkron megoldás


```
var xhr;
function ping() {
  xhr = new XMLHttpRequest();
  xhr.open('GET', 'ping.php', true);
  xhr.addEventListener('readystatechange', pingKezelo, false);
  xhr.send(null);
}
function pingKezelo() {
  if (xhr.readyState == 4 && xhr.status == 200) {
    $('output').innerHTML = xhr.responseText;
  }
}
```

- Globális xhr objektum → problémás → paraméter

XHR objektum paraméterként

- Névtelen függvény segítségével

```
function ping() {
  var xhr = new XMLHttpRequest();
  xhr.open('GET', 'ping.php', true);
  xhr.addEventListener('readystatechange', function () {
    pingKezelo(xhr);
  }, false);
  xhr.send(null);
}
function pingKezelo(xhr) {
  if (xhr.readyState == 4 && xhr.status == 200) {
    $('output').innerHTML = xhr.responseText;
  }
}
```

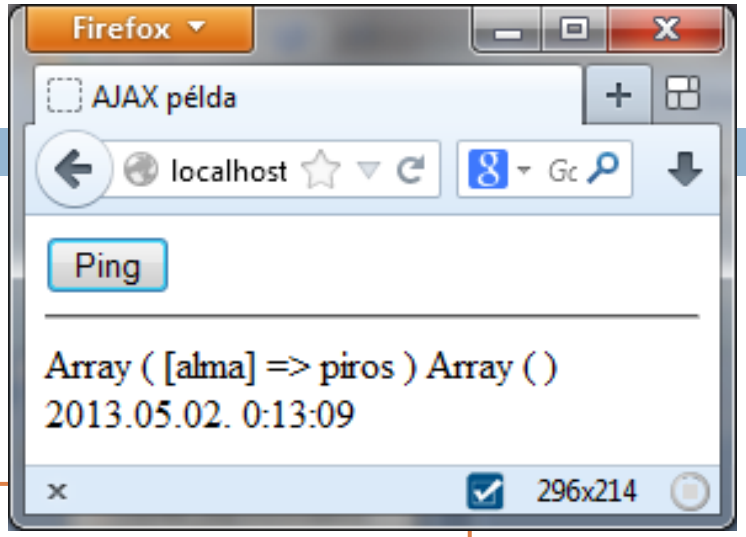
XHR objektum paraméterként

- A lényegi részt még jobban leválasztva

```
function ping() {
  var xhr = new XMLHttpRequest();
  xhr.open('GET', 'ping.php', true);
  xhr.addEventListener('readystatechange', function () {
    if (xhr.readyState == 4 && xhr.status == 200) {
      pingKezelo(xhr);
    }
  }, false);
  xhr.send(null);
}
function pingKezelo(xhr) {
  $('output').innerHTML = xhr.responseText;
}
```

GET paraméterek

- Kérésszöveg az URL-ben

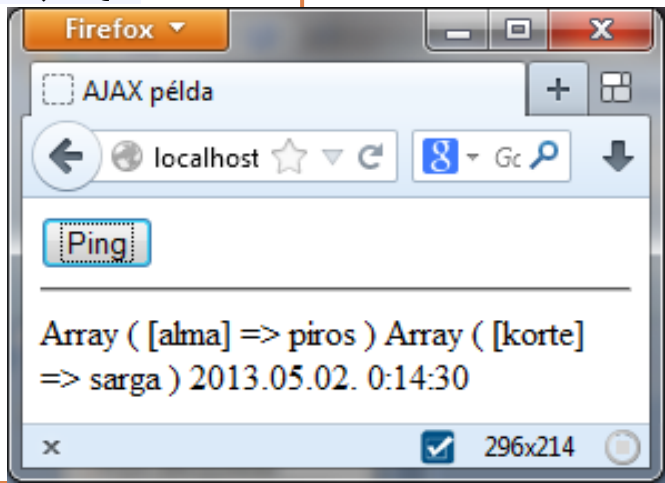



```
function ping() {
  var xhr = new XMLHttpRequest();
  xhr.open('GET', 'ping.php?alma=piros', true);
  xhr.addEventListener('readystatechange', function () {
    if (xhr.readyState == 4 && xhr.status == 200) {
      pingKezelo(xhr);
    }
  }, false);
  xhr.send(null);
}
function pingKezelo(xhr) {
  $('output').innerHTML = xhr.responseText;
}
```

POST paraméterek

- Kérésszöveg az üzenettörzsben → send() metódus
- speciális kódolás beállítása kérés fejléceként

```
function ping() {
  var xhr = new XMLHttpRequest();
  xhr.open('GET', 'ping.php?alma=piros', true);
  xhr.setRequestHeader(
    'Content-Type', 'application/x-www-form-urlencoded');
  xhr.addEventListener('readystatechange', function () {
    if (xhr.readyState == 4 && xhr.status == 200) {
      pingKezelo(xhr);
    }
  }, false);
  xhr.send('korte=sarga');
}
function pingKezelo(xhr) {
  $('output').innerHTML = xhr.responseText;
}
```



XMLHttpRequest objektum

- Manapság minden böngészőben
  - new XMLHttpRequest()
- Böngészőfüggetlenítés

```
function ujXHR() {
  var xhr = null;
  try { xhr = new XMLHttpRequest(); } catch(e) {
  try { xhr = new ActiveXObject("Msxml2.XMLHTTP"); } catch(e) {
  try { xhr = new ActiveXObject("Microsoft.XMLHTTP"); } catch(e) {
        xhr = null;
  }}}
  return xhr;
}
function ping() {
  var xhr = ujXHR();
  //...
}
```

Hibakezelés


```
function ping() {
  var xhr = ujXHR();
  xhr.open('POST', 'ping.php?alma=piros', true);
  xhr.setRequestHeader('Content-Type',
    'application/x-www-form-urlencoded');
  xhr.addEventListener('readystatechange', function () {
    if (xhr.readyState == 4) {
      if (xhr.status == 200) {
        pingKezelo(xhr);
      } else {
        console.log('Hiba');
      }
    }
  }, false);
  xhr.send('korte=sarga');
}
```

Általánosítás

- AJAX hívás nagy része mindig ugyanaz
- Változó
  - url
  - metódus (GET/POST)
  - GET adatok
  - POST adatok
  - feldolgozó függvény siker esetén
  - feldolgozó függvény hiba esetén
- → ajax() függvény

Általánosítás – ajax() függvény

- Alapértelmezett értékek

```
function ajax(opts) {
  var mod    = opts.mod      || 'GET',
      url    = opts.url      || '',
      getadat  = opts.getadat  || '',
      postadat = opts.postadat || '',
      siker    = opts.siker    || function(){},
      hiba     = opts.hiba     || function(){};

  mod = mod.toUpperCase();
  url = url+'?'+getadat;
  var xhr = ujXHR();
  xhr.open(mod, url, true);
  if (mod === 'POST') {
    xhr.setRequestHeader('Content-Type',
      'application/x-www-form-urlencoded');
  }
```

Általánosítás – ajax() függvény

- Folytatás

```
xhr.addEventListener('readystatechange', function () {
    if (xhr.readyState == 4) {
      if (xhr.status == 200) {
        siker(xhr, xhr.responseText);
      } else {
        hiba(xhr);
      }
    }
  }, false);
  xhr.send(mod == 'POST' ? postadat : null);
  return xhr;
}
```

Megoldás ajax() függvénnyel


```
function ping() {
  ajax({
    mod:      'post',
    url:      'ping.php',
    getadat:  'alma=piros',
    postadat: 'korte=sarga',
    siker:    pingKezelo
  });
}
function pingKezelo(xhr, text) {
  $('output').innerHTML = text;
}
```

```
function ping() {
  ajax({
    mod:      'post',
    url:      'ping.php',
    getadat:  'alma=piros',
    postadat: 'korte=sarga',
    siker:    function (xhr, text) {
      $('output').innerHTML = text;
    }
  });
}
```

Összefoglalás

- Segédfüggvény
  - getXHR()
  - ajax()
- Alkalmazás
  - deklaratív
  - az ajax() függvény paraméterezése

Válaszformátumok

AJAX I/O

Kérés (input)

- GET paraméterek
- POST paraméterek
- HTTP fejlécek

Válasz (output)

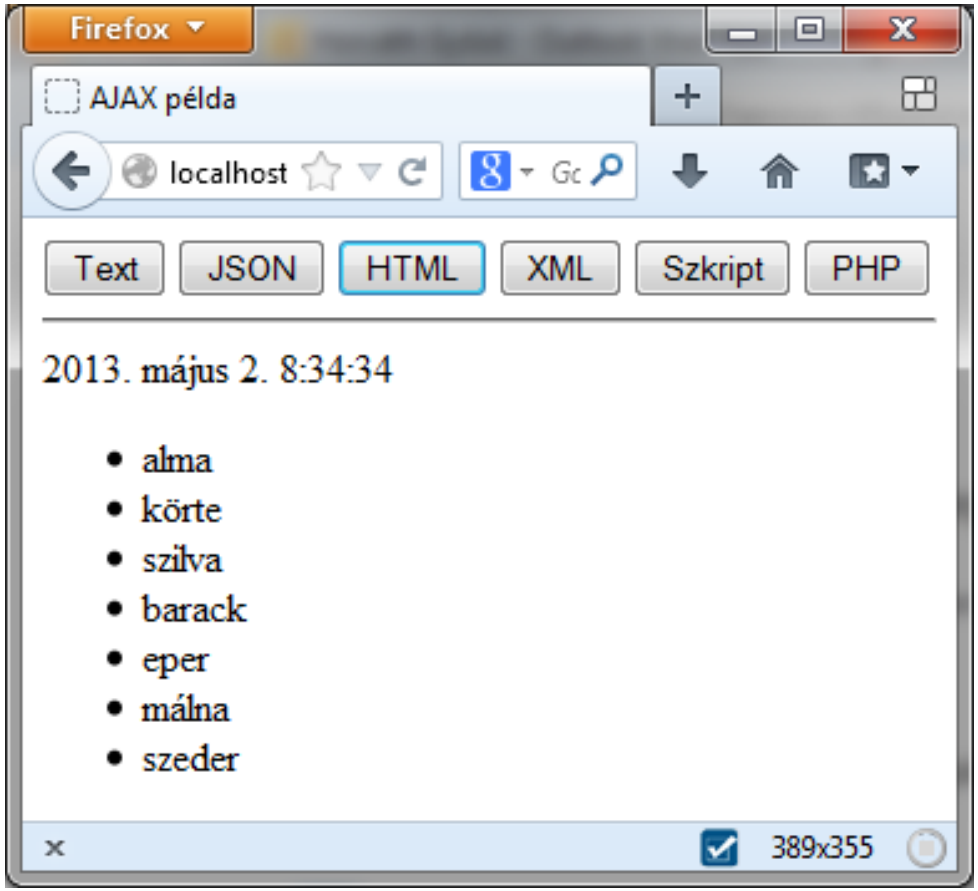
- Egyszerű szöveg
- JSON
- HTML
- XML
- JavaScript
- stb. (pl. CSS,…)

Tartalom előállítása

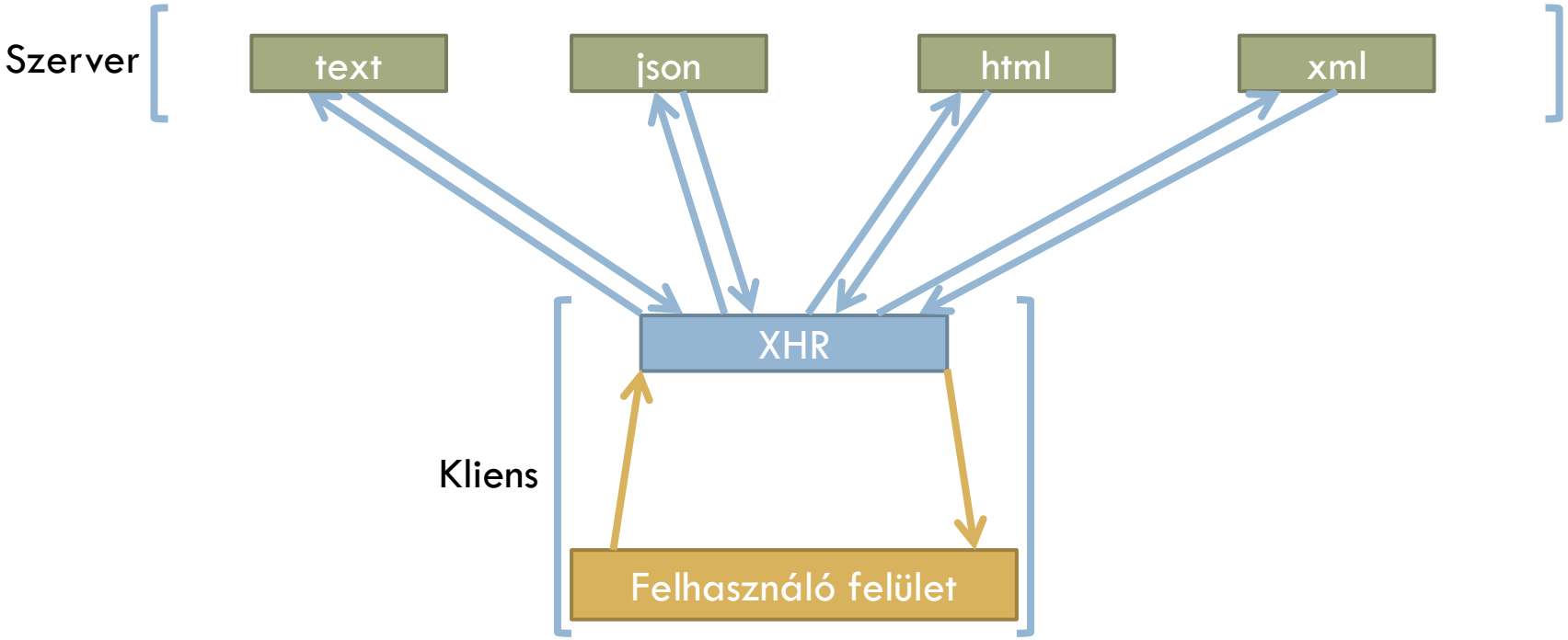
- A válasz az előállítás módjától függően lehet
  - Statikus tartalom
  - Dinamikus tartalom
- A kliens szempontjából mindegy, hogy ki állítja elő a tartalmat (nem is tud róla)
- Csak a válasz formátumára koncentrálunk

Példa

- Adott gyümölcsök listája, jelenítsük meg felsorolásként.



Architektúra



HTML kód


```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>AJAX példa</title>
    <script type="text/javascript" src="ajax.js"></script>
    <script type="text/javascript" src="format.js"></script>
  </head>
  <body>
    <input type="button" id="btnText" value="Text">
    <input type="button" id="btnJSON" value="JSON">
    <input type="button" id="btnHTML" value="HTML">
    <input type="button" id="btnXML" value="XML">
    <input type="button" id="btnScript" value="Szkript">
    <hr>
    <div id="output"></div>
  </body>
</html>
```

Szöveges válasz feldolgozása


```
alma,körte,szilva,barack,eper,málna,szeder
```

```
function lista(t) {
  return '<ul><li>' + t.join('</li><li>') + '</li></ul>';
}
function text() {
  ajax({
    url: 'gyumolcs.txt',
    siker: function (xhr, text) {
      console.log(text);
      var t = text.split(',');
      $('output').innerHTML =
        (new Date()).toLocaleString() + lista(t);
    }
  });
}
```

Szöveges válasz feldolgozása

- A szöveges válasz formátumától függ
- Egyedi feldolgozás
- Sokszor bonyolult logika
- Bármi lehet
- Ritkábban használják

JSON válasz feldolgozása


```
[
  "alma",
  "körte",
  "szilva",
  "barack",
  "eper",
  "málna",
  "szeder"
]
```

```
function json() {
  ajax({
    url: 'gyumolcs.json',
    siker: function (xhr, text) {
      var json = eval(text);
      // vagy
      var json = JSON.parse(text);
      console.log(json);
      $('output').innerHTML =
        (new Date()).toLocaleString() + lista(json);
    }
  });
}
```

JSON válasz

- JSON általános adatleíró formátum
- Nagyon elterjedt
- Gyakran használatos
- Egyszerű az értelmezése
  - eval
  - JSON.parse
- Utána már JavaScript adatszerkezetekkel kell dolgozni

HTML válasz feldolgozása


```
<ul>
  <li>alma</li>
  <li>körte</li>
  <li>szilva</li>
  <li>barack</li>
  <li>eper</li>
  <li>málna</li>
  <li>szeder</li>
</ul>
```

```
function html() {
  ajax({
    url: 'gyumolcs.html',
    siker: function (xhr, text) {
      var html = text;
      console.log(html);
      $('output').innerHTML =
        (new Date()).toLocaleString() + html;
    }
  });
}
```

HTML válasz

- Szabványos formátum
- Elterjedt
- Egyszerű a feldolgozása
  - tipikusan a választ egy másik elembe kell helyezni

XML válasz feldolgozása

- gyumolcs.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<gyumolcsok>
  <gyumolcs>alma</gyumolcs>
  <gyumolcs>körte</gyumolcs>
  <gyumolcs>szilva</gyumolcs>
  <gyumolcs>barack</gyumolcs>
  <gyumolcs>eper</gyumolcs>
  <gyumolcs>málna</gyumolcs>
  <gyumolcs>szeder</gyumolcs>
</gyumolcsok>
```

XML válasz feldolgozása


```
function xml() {
  ajax({
    url: 'gyumolcs.xml',
    siker: function (xhr, text) {
      var xmldom = xhr.responseXML;
      console.log(xmldom);

      var gyumolcsok = xmldom.getElementsByTagName('gyumolcs');
      var t = [];
      for (var i = 0; i < gyumolcsok.length; i++) {
        t.push(gyumolcsok[i].firstChild.nodeValue);
      };

      $('output').innerHTML =
        (new Date()).toLocaleString() + lista(t);
    }
  });
}
```

XML válasz

- Szabványos adatleírási formátum
- Nagyon elterjedt
  - főleg vállalati alkalmazásokban
  - webes alkalmazásokban kevésbé
- Eredetileg ezt várták fő formátumnak
  - xhr.responseXML
- responseXML az XML dokumentum értelmezett DOM fáját tartalmazza (ld. HTML DOM)

JavaScript kód feldolgozása


```
function getGyumolcsok() {
  return [
    "alma",
    "körte",
    "szilva",
    "barack",
    "eper",
    "málna",
    "szeder"
  ];
}
```

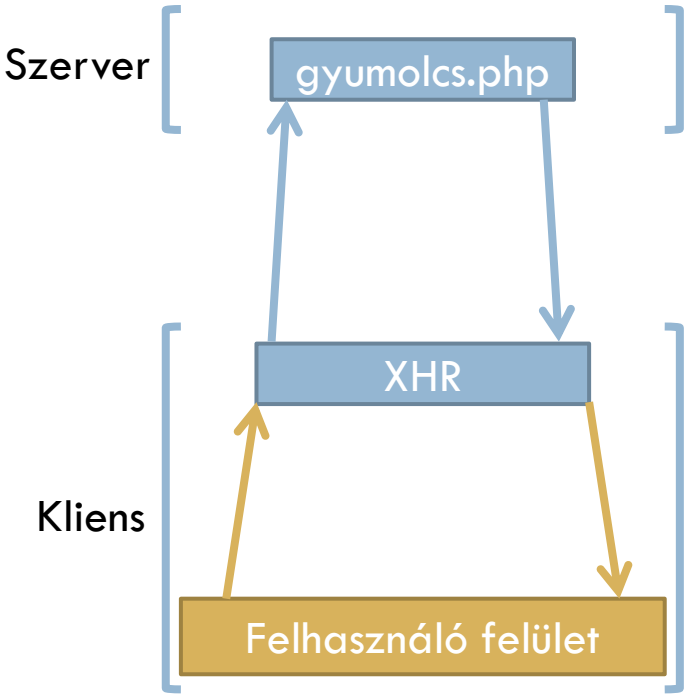
```
function script() {
  ajax({
    url:  'gyumolcs.js',
    siker:  function (xhr, text) {
      console.log(text);
      eval(text);
      var t = getGyumolcsok();
      $('output').innerHTML =
        (new Date()).toLocaleString() + lista(t);
    }
  });
}
```

JavaScript kód feldolgozása

- Jól illeszkedik a kliensoldali környezetbe
- Feldolgozása egyszerű

Dinamikus válasz

- A választ egy program (PHP) állítja elő
- Válasz formátuma bármi lehet
  - text
  - json
  - html
  - xml
  - stb.



gyumolcs.php

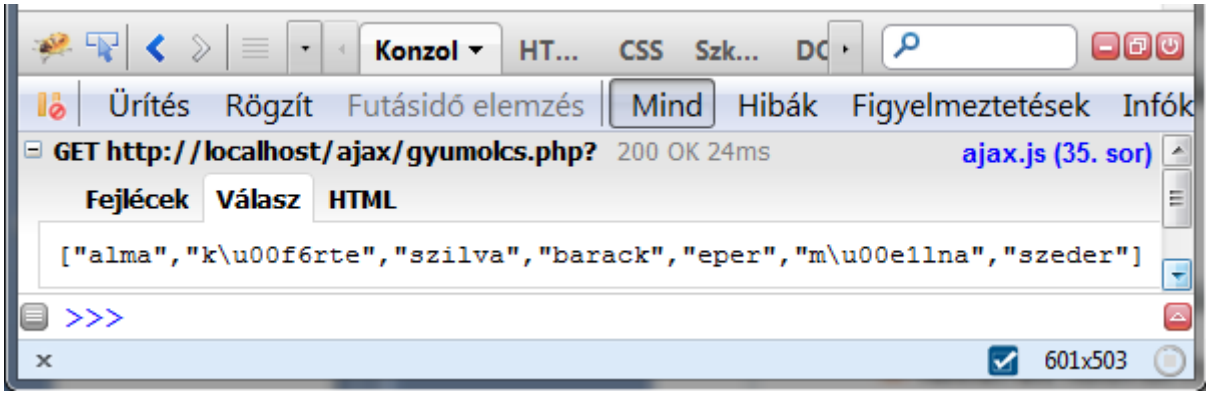
- JSON válasz generálása egyszerű

```php
$gyumolcsok = array(
  "alma",
  "körte",
  "szilva",
  "barack",
  "eper",
  "málna",
  "szeder",
);
echo json_encode($gyumolcsok);
```

További tudnivalók

Eszközök

- JavaScript konzolok
  - Nyomon követik az AJAX-os hívásokat
    - kérés
    - válasz
    - értelmezett válasz
    - lekérés időtartama
  - console.log



Korlátozások

- Biztonsági okokból nem engedélyezett különböző domainek között ajax kommunikáció
  - Kivéve JSONP (JSON with Padding): script elem dinamikus beszúrásával idegen tartalom futtatása
- Nincsen szerver push itt sem: ajax kéréseket mindig a kliens kezdeményezi

API JSON válasz

http://www.omdbapi.com/?s=hobbit&callback=feldolgoz


```
{
  "Search":[
    {
      "Title":"The Hobbit: An Unexpected Journey",
      "Year":"2012",
      "imdbID":"tt0903624",
      "Type":"movie"
    },
    {
      "Title":"The Hobbit: The Desolation of Smaug",
      "Year":"2013",
      "imdbID":"tt1170358",
      "Type":"movie"
    },
    {
      "Title":"The Hobbit",
      "Year":"1977",
      "imdbID":"tt0077687",
      "Type":"movie"
    }
  ]
}
```



```
feldolgoz({"Search":[{"Title":"The Hobbit: An Unexpected Journey","Year":"2012","imdbID":"tt0903624","Type":"movie"},{"Title":"The Hobbit: The Desolation of Smaug","Year":"2013","imdbID":"tt1170358","Type":"movie"},{"Title":"The Hobbit","Year":"1977","imdbID":"tt0077687","Type":"movie"}]});
```

JSONP


```
function omdbHivas(keresett) {
  var script = document.createElement('script');
  script.type = 'text/javascript';
  script.src =
'http://www.omdbapi.com/?s='+keresett+'&callback=feldolgoz';
  document.body.appendChild(script);
}

function feldolgoz(json) {
  console.log(json);
}
```

```
feldolgoz({"Search":[{"Title":"The
Hobbit: An Unexpected
Journey","Year":"2012","imdbID":"t
t0903624","Type":"movie"},{"Title"
:"The Hobbit: The Desolation of
Smaug","Year":"2013","imdbID":"tt1
170358","Type":"movie"},{"Title":"
The
Hobbit","Year":"1977","imdbID":"tt
0077687","Type":"movie"}]});
```

Ergonómiai szempontok

- Más oldalszervezés
- HASZNÁLHATÓSÁG
- Jelezni, hogy a háttérben művelet zajlik
- A felhasználó adatait ne írjuk át
- Jelezni, ha hiba van
- Jelezni, ha jó

További kérdések

- Mi van, ha nincs JavaScript?
- URL cache-elése GET kérésnél
- Állapottartás
- Frissít gomb
- Vissza gomb

Előnyök és hátrányok

Előnyök

- Nincs szaggatás
- Kényelmes felület
- Gyors
- Jóval kisebb adatforgalom
- Párhuzamos kérések

Hátrányok

- JavaScript kell
- Eltérő támogatottság
- Vissza gomb
- Frissít gomb
- Más oldalszervezési logika

Összefoglalás

- AJAX elvek
- XMLHttpRequest objektum
- Magas szintű ajax() segédfüggvény
- Válasz feldolgozások
- JSONP