

Web-technológia II.

Kliens-szerver kommunikáció, AJAX

Hatwágner F. Miklós

Széchenyi István Egyetem, Győr

2019. február 28.

Szuperglobális változók

A **szuperglobális változók** főbb tulajdonságai:

- automatikusan kapnak értéket
- minden hatókörben elérhetők, a **global** kulcsszó nélkül is
- nem használhatók pl. indirekt változóként (egyik változó értéke a másik neve)

Néhány változó és feladatuk:

- **\$_GET** URL lekérdező karakterláncának adatai
- **\$_POST** HTTP fejlécben küldött (űrlap) adatok
- **\$_COOKIE** süti adatok
- **\$_REQUEST** fenti három bármelyikétől származó adatok; egyforma kulccsal rendelkezők felülírják egymást, ld. **request_order** beállítás a php.ini fájlban
- **\$_SERVER** szerver beállítások; **PHP_SELF** kulcsú elem a futtatott script neve
- **\$_ENV** környezeti változók értékei
- **\$_FILES** feltöltött fájlok adatai

Szuperglobális változók

pelda18.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Szuperglobális változók</title>
  </head>
  <body>
    <?php
      $szgvalt = [
        '$_GET' => $_GET, '$_POST' => $_POST,
        '$_COOKIE' => $_COOKIE, '$_REQUEST' => $_REQUEST,
        '$_SERVER' => $_SERVER, '$_ENV' => $_ENV,
        '$_FILES' => $_FILES
      ];
      foreach($szgvalt as $k=>$v) {
        echo "<h1>$k</h1>\n<pre>";
        print_r($v);
        echo "</pre>\n";
      }
    ?>
  </body>
</html>
```

Szuperglobális változók

pe1da19.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Űrlapok kezelése</title>
  </head>
  <body>
    <form action="<?php echo $_SERVER["PHP_SELF"]; ?>" method="post">
      <?php $szamlalo=isset($_POST["szamlalo"])?$_POST["szamlalo"]+1:1; ?>
      <input type="hidden" name="szamlalo" value="<?php echo $szamlalo ?>">
      <div><label>Szöveg:<input type="text" name="szoveg"></label></div>
      <div><label>Jelszó:<input type="password" name="jelszo"></label></div>
      <fieldset>
        <legend>Rádiógombok</legend>
        <div><label>
          <input type="radio" name="radio" value="r1" checked>Választás1
        </label></div>
        <div><label>
          <input type="radio" name="radio" value="r2">Választás2
        </label></div>
      </fieldset>
      <fieldset>
        <legend>Jelölőnégyzetek</legend>
        <div><label>
          <!-- A név lehetne jelolo[] is, hasonlóan a select-hez! -->
          <input type="checkbox" name="jelolo1" value="jA" checked>VálasztásA
        </label></div>
```

Szuperglobális változók

pelda19.php

```
<div><label>
  <input type="checkbox" name="jelolo2" value="jB">VálasztásB
</label></div>
</fieldset>
<div><label>Legördülő lista<select name="legordulo">
  <option value="11">1</option>
  <option value="12">2</option>
</select></label></div>
<div><label>Többszörös választás
  <select name="tobbszoros[]" multiple size="2">
    <option value="t1">1</option>
    <option value="t2">2</option>
  </select></label></div>
<div><label>
  Szerkesztő<textarea name="szerkesztzo"></textarea>
</label></div>
<div><label>
  <input type="submit" name="kuldes" value="Küldés">
</label></div>
<fieldset>
  <legend>Korábban küldött adatok:</legend>
  <pre><?php print_r($_POST); ?></pre>
</fieldset>
</form>
</body>
</html>
```

Az inputot maga **az ördög küldi**, mindig **ellenőrizni** kell!

- formai megfeleléség ellenőrzése (validation)
- tiltott karakterek eltávolítása (sanitation)

Néhány hasznos **függvény**:

- **filter_has_var()** ellenőrzi, hogy a szuperglobális tömb adott eleme létezik-e
- **filter_input()** szuperglobális tömb adott elemét teszteli
- **filter_var()** tetszőleges változó tartalmát teszteli

A **szűrőkhöz** tartozó opciókkal akár intervallumba esés is ellenőrizhető.

Tartalom szűrése

pelda20.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Szűrők használata</title>
  </head>
  <body>
    <form action="<?php echo $_SERVER["PHP_SELF"]; ?>" method="post">
      <?php
        $validSzuro = [ FILTER_VALIDATE_EMAIL, FILTER_VALIDATE_FLOAT,
                        FILTER_VALIDATE_IP, FILTER_VALIDATE_URL];
        $tisztitoSzuro = [ FILTER_SANITIZE_EMAIL,
                          FILTER_SANITIZE_NUMBER_FLOAT,
                          FILTER_SANITIZE_STRING, FILTER_SANITIZE_URL];
        if(filter_has_var(INPUT_POST, "szoveg")) {
          if(($szurt=filter_var($_POST["szoveg"],
                                $validSzuro[$_POST["validalas"]]))===FALSE) {
            echo "<p>A tartalom nem érvényes.</p>\n";
          } else {
            echo "<p>A szűrt tartalom: $szurt</p>\n";
          }
        }
      </?php>
    </form>
  </body>
</html>
```

pelda20.php

```
if(($tiszta=filter_input(INPUT_POST, "szoveg",
    $tisztitoSzuro[$_POST["tisztitas"]]))==FALSE) {
    echo "<p>Tisztítási hiba.</p>\n";
} else {
    echo "<p>A tisztított tartalom: $tiszta</p>\n";
} } ?>

<div><label>Szöveg:<input type="text" name="szoveg"></label></div>
<div><label>Validálás: <select name="validalas">
    <option value="0">E-Mail</option>
    <option value="1">Szám</option>
    <option value="2">IP-cím</option>
    <option value="3">URL</option>
</select></label></div>
<div><label>Tisztítás: <select name="tisztitas">
    <option value="0">E-Mail</option>
    <option value="1">Szám</option>
    <option value="2">Karakterlánc</option>
    <option value="3">URL</option>
</select></label></div>
<div><input type="submit" name="kuldes" value="Küldés"></div>
</form></body></html>
```


Kis mennyiségű adat megőrzése két oldalletöltés között.

- összefoglaló
- EU-s direktíva
- sütik olvasása: `$ _COOKIE` tömbön keresztül
- `setcookie()` süti létrehozására, módosítására és törlésére alkalmas. A tárolt értéket automatikusan aláveti az `urlencode()/urldecode()`-nak. Törlés azonos adatokkal/üres/false értékkel, és múltbeli lejáráttal lehetséges.
- sütik a HTTP fejlécben utaznak, ezért kiküldetésüket más nyomtatás **nem előzheti meg!**
- ne alapozzunk az alapértelmezett pufferezésre, de `ob_start()`, `ob_end_flush()` segíthet
- `header()` tetszőleges HTTP fejléc előállításához, pl. átirányításhoz

pelda21.php

```
<?php
    $ujraTolt = false;
    if(!empty($_POST["nev"]) && // létrehozás
        !empty($_POST["ertek"])) {
        setcookie($_POST["nev"], $_POST["ertek"], time()+60*60*24);
        $ujraTolt = true;
    }
    foreach($_POST as $k => $v) { // törlés
        if(substr($k, 0, 2) === "t_") {
            $nev = substr($k, 2);
            $ertek = $_COOKIE[$nev];
            setcookie($nev, $ertek, time()-1);
            $ujraTolt = true;
        }
    }
    if($ujraTolt) { // oldal újratöltése
        header("Location: ".$_SERVER['REQUEST_URI']);
        exit;
    }
?>
```

pelda21.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sütik kezelése</title>
  </head>
  <body>
    <form action="<?php echo $_SERVER["PHP_SELF"] ?>" method="post">
      <h1>Jelenlegi sütik és tartalmuk</h1>
      <?php
        if(empty($_COOKIE)) {
          echo "<p>Nincsenek sütik.</p>";
        } else {
          echo "<p>Törléshez jelölje a sütit.</p>\n";
          echo "<ol>\n";
          foreach($_COOKIE as $k => $v) {
            echo "  <li><input type=\"checkbox\"
              name=\"t_<$k\">$k : $v</li>\n";
          }
          echo "</ol>\n";
        }
      <?>
```

pelda21.php

```
<h1>Új süti felvétele</h1>
<div><label>Név:<input type="text" name="nev"></label></div>
<div><label>Érték:<input type="text" name="ertek"></label></div>
<div><input type="submit" name="kuldes" value="Küldés"></div>
</form>
</body>
</html>
```

Fájlok feltöltése

- Form kódolása legyen **multipart/form-data**, küldés **POST** módban!
- HTML5-től több fájl is küldhető egyszerre:
`<input type="file" name="fajlok[]" multiple>`
- Méretkorlátozás:
`<input type="hidden" name="MAX_FILE_SIZE" value="2097152">`
php.ini beállításai
- A fájlok átmeneti könyvtárba kerülnek, onnan **el kell mozgatni** őket!
- A **\$_FILES** nem fájlonként tartalmaz tömböket, hanem a fájlok tulajdonságai lesznek tömbök!
- **Hiba** detektálás, pl.: `$_FILES['fajlok']['error']`, ha nincs hiba:
UPLOAD_ERR_OK
- További fájl tulajdonságok (`$_FILES` tömb kulcsai):
 - `name` eredeti fájlnev a kliensen
 - `type` MIME típus, pl. `image/gif`
 - `size` fájl mérete bájtokban
 - `tmp_name` átmeneti fájlnev a szerveren
- Hasznos segédfüggvények:
 - `move_uploaded_file()` fájl végleges helyre mozgatása
 - `scandir()` fájlbejegyzések listázása
 - `array_filter()` tömbelemek szűrése callback fv.-nyel
 - `getimagesize()` képméret lekérdezése

pelda22.php

```
<?php
// jogosultságokra figyelni!
define("FELHASZNALO", "feltoltes");
define("KEP_MAPPA", "/home/".FELHASZNALO."/www/");
if (!empty($_FILES["kepek"])) {
    for($i=0; $i<count($_FILES["kepek"]["type"]); $i++) {
        if(strpos($_FILES["kepek"]["type"][$i],"image/") === 0) {
            $nev = $_FILES["kepek"]["name"][$i];
            if(!move_uploaded_file($_FILES["kepek"]["tmp_name"][$i],
                KEP_MAPPA.$nev)) {
                echo "<p>Nem sikerült felmásolni a $nev fájlt.";
            } } } }
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Képek feltöltése</title>
    </head>
    <body>
        <h1>Eddig feltöltött képek</h1>
```

pelda22.php

```
<?php
$kepek = array_filter(scandir(KEP_MAPPA),
    function($fajl) { return is_file(KEP_MAPPA.$fajl); });
if(empty($kepek)) {
    echo "<p>Nincsenek még képeink, töltsön fel párat!</p>\n";
} else {
    foreach($kepek as $kep) {
        $meret = getImageSize(KEP_MAPPA.$kep);
        echo "<figure>\n
            <img src=\"/~\".FELHASZNALO.\"/$kep\"
                width=\"\${meret[0]}\" height=\"\${meret[1]}\">\n
            <figcaption>$kep</figcaption>\n
            </figure>\n";
    }
}
?>
<h1>Új kép feltöltése</h1>
<form action="<?php echo $_SERVER["PHP_SELF"]; ?>"
    method="post" enctype="multipart/form-data">
    <input type="hidden" name="MAX_FILE_SIZE" value="2097152">
```

pelda22.php

```
<div><label>
    Képek:<input type="file" name="kepek[]" multiple>
</label></div>
<div><input type="submit" value="Küldés"></div>
</form>
</body>
</html>
```


AJAX – Asynchronous JavaScript and XML

Szükséges technológiák:

- (X)HTML, adatok megjelenítése
- DOM (Document Object Model), az oldal részeinek módosítása teljes újratöltés nélkül
- JavaScript, DOM-kezelés, kliens-szerver kommunikáció
- valamilyen szerver-oldali programozási nyelv (pl. PHP)

Opcionális technológiák:

- CSS, a (z X)HTML oldal formázásához
- XML, adatsere formátuma (egyre ritkábban)
- JSON (JavaScript Object Notation), adatsere formátum
- XSLT (Extensible Stylesheet Language Transformations), XML-nek (X)HTML-lé alakításához
- XMLHttpRequest (XHR) közvetítő obj.

Célok:

- kliens és szerver közötti forgalom csökkentése, de
- a felhasználóknak ne okozzunk meglepetéseket (újszerű UI), és
- ne zárjunk el felhasználókat a tartalomtól (pl. JS hiánya).

Kommunikáció megvalósításának lehetőségei:

- rejtett keret
- rejtett IFrame (inline frame)
- **XHR** → ezt használjuk!

Előnyök:

- tiszta, áttekinthető JS/PHP kód
- kérdés/válasz fejlécek elérhetőek
- HTTP állapotkód ismert

Hátrányok:

- navigációs gombok használata nehézkes
- IE6-: ActiveX vezérlőt igényel, amit letilthatnak vagy eleve nincs is telepítve
- csak tartományon belüli kommunikáció lehetséges (Same Origin Policy, vagy proxy/**CORS** stb. kell)

Feladat: számok négyzetre emelése

Mire lesz szükség?

Kliens oldal (JS):

- AJAX objektum létrehozása
- kérés összeállítása (GET / POST)
- válasz feldolgozása

Szerver oldal (PHP):

- válasz előállítás (text, XML, JSON)

AJAX: XHR objektum létrehozása

Az objektum létrehozása böngészőfüggő:

- **IE5,6:** ActiveX objektumon keresztül
 - **Vista:** MSXML2.XMLHttp.6.0
 - **ajánlott:** MSXML2.XMLHttp.3.0
- **IE7+, többi:** natív JS megvalósítás

ajax.js

```
function getXHR() {  
    if(window.XMLHttpRequest) {  
        return new XMLHttpRequest();  
    } else if(window.ActiveXObject) {  
        return new ActiveXObject('MSXML2.XMLHTTP.3.0');  
    } else {  
        return null;  
    }  
}
```

AJAX: Kérés indítása

Kérés indítása előtt, aszinkron lekérés esetén **eseménykezelő** callback fv.-t kell készíteni, beállítani. A válasz megérkezteivel (is) le fog futni.

Eseménykezelő beállítása

```
var ajax = getXHR();  
if(ajax) {  
    ajax.onreadystatechange = esemenyKezeloFv;  
    ...  
}
```

A kérést összeállítani az **open** metódussal lehet:

Kérés összeállítása

```
ajax.open(típus, url, mod);
```

típus lekérés típusa, jellemzően GET vagy POST

url szerver oldali program url-je

mod aszinkron (**true**) vagy szinkron (**false**) működés

AJAX: Kérés indítása

Lekérés típusa:

- **GET**: jellemzően adatok lekéréséhez, amit gyakran megismételnek, elmenthetik a könyvjelzők közé is
- **POST**: adatok küldéséhez, feltöltéséhez, ami a szerveren (pl. adatbázis) változást eredményez. Az adatokat jellemzően egyszer küldik csak el.

URL:

- lehet abszolút vagy relatív
- JS **csak a saját tartományába** (pl. `xenia.sze.hu`) intézhet kéréseket

Lekérés módja:

- alapértelmezés: aszinkron
- szinkron esetben nincs szükség eseménykezelő fv.-re,
- de a válasz megérkezéséig a JS értelmező várakozik

Opcionális felhasználónév, jelszó paraméterek: HTTP azonosításhoz → a felhasználótól kell ezeket bekérni, a **JS kódba ne írjuk bele!**

Végül a kérést el kell küldeni a **send** metódussal. Paramétere: a szervernek elküldendő adatok.

Alap küldése a szervernek (negyzet.js)

```
document.getElementById("urlap").onsubmit = function() {  
    var szerver = "http://xenia.sze.hu/";  
    var szam = encodeURIComponent(  
        document.getElementById("szam").value);  
    ajax.open("GET", szerver + "~wajzy/ajax/negyzet.php?szam=" +  
        szam, true);  
    ajax.send(null);  
    return false;  
};
```

Megjegyzések:

- A válaszra várakozás félbeszakítható az **abort** metódus hívásával (ld. `setTimeout` fv.)
- A `send` metódusnak mindenképp kell paramétert átadni, legalább egy **null**-t.
- Elküldés előtt a lekérdező karakterláncot át kell alakítani **encodeURIComponent** segítségével

AJAX: Válasz feldolgozása

Az objektum (valamennyi) állapotváltozásáról az `onreadystatechange` eseménykezelővel értesülhetünk. A `readyState` adattag lehetséges értékei:

- | | |
|---------------------------|------------|
| 0 a kérést nem küldték el | 3 letöltés |
| 1 kapcsolat megnyitva | 4 kész |
| 2 fejlécek megérkeztek | |

A válasz felhasználható? Ld. `status`, `statusText` adattagok!

- | | |
|----------------------------|-----------------------|
| 200 OK | 401 Nincs hitelesítve |
| 301 Véglegesen áthelyezve | 403 Tiltva |
| 304 Nincs módosítva | 404 Nem található |
| 307 Ideiglenes átirányítás | 500 Belső szerverhiba |

A magyar nyelvű állapot üzenetek a [Google](#) oldaláról származnak.

AJAX: Válasz feldolgozása

Szerver válasza:

- **responseXML** adattagban XML formátum esetén,
- **responseText** minden egyéb esetben

Válasz feldolgozása (negyzet.js)

```
ajax.onreadystatechange = function() {  
    if (ajax.readyState == 4) {  
        var eredmény = document.getElementById("eredmeny");  
        if ((ajax.status >= 200 && ajax.status < 300)  
            || (ajax.status == 304) ) {  
            eredmény.innerHTML = ajax.responseText;  
        } else {  
            eredmény.innerHTML = ajax.statusText;  
        }  
    }  
};
```

Erőforrások felszabadítása

```
ajax = null;
```

AJAX: További fájlok

negyzet.html

```
<!doctype html>
<html lang="hu">
<head>
  <meta charset="utf-8">
  <title>Négyzetre emelés</title>
</head>
<body>
  <h1>Számok négyzetre emelése AJAX segítségével</h1>
  <form id="urlap" action="#">
    <fieldset>
      <legend>Négyzetre emelés</legend>
      <div><label for="szam">Szám:</label>
        <input type="number" id="szam" required></div>
      <div><p>Eredmény: <span id="eredmeny"></span></p></div>
      <div><input type="submit" value="Számol" id="szamol"></div>
    </fieldset>
  </form>
  <script src="ajax.js"></script>
  <script src="negyzet.js"></script>
</body>
</html>
```

negyzet.php

```
<?php
header("Content-type: text/plain; charset=UTF-8");
if(isset($_GET["szam"]) && is_numeric($_GET["szam"])) {
    $szam = $_GET["szam"];
    echo $szam * $szam;
} else {
    echo "Hibás adat.";
}
?>
```

Változtatások:

- az open metódus első paramétere **POST** lesz
- az adatok nem az URL-ben utaznak,
- hanem **HTTP adatként** → send metódus paramétere
- az open **után** űrlapfejléc küldése (setRequestHeader fv.)
- módosítani kell a PHP fájlt is (vagy eleve \$_REQUEST-et kellett volna használni)

AJAX: POST típusú kérés

Módosított eseménykezelő (negyzet_post.js)

```
document.getElementById("urlap").onsubmit = function() {  
    var szerver = "http://xenia.sze.hu/";  
    var szam = "szam=" + encodeURIComponent(  
        document.getElementById("szam").value);  
    ajax.open("POST", szerver + "~wajzy/ajax/negyzet_post.php", true);  
    ajax.setRequestHeader("Content-Type",  
        "application/x-www-form-urlencoded");  
    ajax.send(szam);  
    return false;  
};
```

Módosított szerver oldali szkript (negyzet_post.php)

```
<?php  
header("Content-type: text/plain; charset=UTF-8");  
if(isset($_POST["szam"]) && is_numeric($_POST["szam"])) {  
    $szam = $_POST["szam"];  
    echo $szam * $szam;  
} else {  
    echo "Hibás adat."  
}  
?>
```

AJAX: Adatcsomag összeállítása FormData segítségével

- nem minden böngészőben érhető el!
- nincs szükség a paramétereket **elválasztó karakterek** (&, ?), a **tartalom típusának** megadására és az előzetes **kódolásra** (encodeURIComponent)

Módosított eseménykezelő (negyzet_fd.js)

```
document.getElementById("urlap").onsubmit = function() {
    var szerver = "http://xenia.sze.hu/";
    ajax.open("POST", szerver + "~wajzy/ajax/negyzet_post.php", true);
    if(typeof FormData == "function") {
        var szam = FormData();
        szam.append("szam", document.getElementById("szam").value);
    } else {
        var szam = "szam=" + encodeURIComponent(
            document.getElementById("szam").value);
        ajax.setRequestHeader("Content-Type",
            "application/x-www-form-urlencoded");
    }
    ajax.send(szam);
    return false;
};
```

Első lépés: szerver oldal működik?

Egyszerű nyomkövetés:

- Firefox → FireBug
- Opera → Dragonfly
- Safari → Web Inspector
- de a beépített fejlesztőeszközök is elég jók már

Válasz adatok ellenőrzése:

- XML → [XML Validator](#)
- JSON → [JSONLint](#)

Több AJAX lekérés → több AJAX objektum

Ugyanaz a lekérdezés változó eredményt ad (pl. részvényindex) →
cache kikapcsolás

- `<?php header("Cache-Control: no-cache"); ?>`
- `var url = "http://ajax.hu/ajax.php?ib=" +
new Date.getTime();`

AJAX: XML válasz

negyzet_xml.php

```
<?php
header("Content-type: application/xml");
echo "<?xml version=\"1.0\" encoding=\"utf-8\"
    standalone=\"yes\"?>\n<negyzet>\n";
if(isset($_GET["szam"]) && is_numeric($_GET["szam"])) {
    $szam = $_GET["szam"];
    echo "\t<alap>$szam</alap>\n";
    echo "\t<kitevo>2</kitevo>\n";
    echo "\t<eredmeny>".$szam*$szam."</eredmeny>\n";
} else {
    echo "\t<hiba>Hibás adat.</hiba>\n";
}
echo "</negyzet>\n";
?>
```


AJAX: XML válasz

negyzet_xml.js

```
ajax.onreadystatechange = function() {  
    if (ajax.readyState == 4) {  
        var eredmeny = document.getElementById("eredmeny");  
        if ((ajax.status>=200 && ajax.status<300)  
            || (ajax.status==304)) {  
            var valasz = ajax.responseXML;  
            var hiba = valasz.getElementsByTagName("hiba");  
            if(hiba.length) {  
                eredmeny.innerHTML = hiba[0].firstChild.nodeValue;  
            } else {  
                var alap = valasz.documentElement.children[0].firstChild.nodeValue;  
                var kitevo = valasz.documentElement.childNodes[3].  
                    firstChild.nodeValue;  
                var hatvany = valasz.documentElement.lastChild.  
                    previousSibling.firstChild.nodeValue;  
                eredmeny.innerHTML = hatvany +  
                    " (" + alap + "<sup>" + kitevo + "</sup>";  
            }  
        } else {  
            eredmeny.innerHTML = ajax.statusText;  
        }  
    }  
};
```

Egy XML csomópont attribútumának értékét a `getAttribute` függvénnyel lehet lekérdezni.

Problémák:

- Az XML terjengős, pl.: `<etkor>18</etkor>`
- A DOM bejárása körülményes és lassú

JSON jellemzői:

- Douglas Crockford: a JS szintaktikája önmagában is alkalmas adatok rögzítésére
- Tömörebb és gyorsabban feldolgozható
- Feldolgozás régebben `eval` függvénnyel, ma JSON objektummal (**JSON.parse** metódus)
- IE7- nem támogatja, de a kód [letölthető](#) a szerző oldaláról (json.org)
- Nyugodtan hivatkozható a HTML-ből, mert ha nincs szükség rá, nem jön létre új JSON objektum

AJAX: JSON válasz

negyzet_json.html

```
...  
<script src="json2.js"></script>  
...
```

negyzet_json.php

```
<?php  
header("Content-type: application/json; charset=UTF-8");  
if(isset($_GET["szam"]) && is_numeric($_GET["szam"])) {  
    $szam = (int)$_GET["szam"];  
    echo json_encode(array("alap" => $szam, "kitevo" => 2,  
        "eredmeny" => $szam*$szam));  
} else {  
    echo json_encode(array("hiba" => "Hibás adat."));  
}  
?>
```

PHP segédfüggvények: `json_encode()` és `json_decode()`.

negyzet_json.php

```
ajax.onreadystatechange = function() {  
    if (ajax.readyState == 4) {  
        var eredmeny = document.getElementById("eredmeny");  
        if ((ajax.status>=200 && ajax.status<300)  
            || (ajax.status==304) ) {  
            var valasz = JSON.parse(ajax.responseText);  
            if(typeof valasz.hiba == "string") {  
                eredmeny.innerHTML = valasz.hiba;  
            } else {  
                var alap = valasz.alap;  
                var kitevo = valasz.kitevo;  
                var hatvany = valasz.eredmeny;  
                eredmeny.innerHTML = hatvany + " (" + alap +  
                    "<sup>" + kitevo + "</sup>";  
            }  
        } else {  
            eredmeny.innerHTML = ajax.statusText;  
        }  
    }  
};
```

- Kérés fejléc beállítása

```
ajax.setRequestHeader("Content-Type", "text/xml");
```

```
application/xml (ajánlott)
```

```
application/json
```

- `JSON.stringify`

Fájlok küldése:

- `FormData` objektummal
- rejtett keretekkel

Készítsünk fórumot, amelyben mindenki hozzászólhat egy fontos témához! Lehesen beküldeni

- A hozzászóló nevét,
- a hozzászólást,
- és a program automatikusan állapítsa meg és tárolja a hozzászólás idejét is!

Próbáljuk folyamatosan aktualizálni az oldal tartalmát, de minimális adatforgalom mellett! (Igazából a legjobb megoldás erre a [WebSockets](#) használata lenne, de most elégedjünk meg az új hozzászólások másodpercenkénti [pollozásával](#).)

Megoldás: [forum.html](#), [forum.js](#), [forum.php](#).

Más tartományokban lévő webszolgáltatásokat nem lehet JS kódból közvetlenül elérni (same origin policy). Lehetséges megoldások:

- **web proxy** – a webservert bármelyik tartományból tölthet, és az adatokat visszaküldheti a JS-nek
- Apache `mod_rewrite`, `mod_proxy` moduljai: kérések más szerverhez továbbítása és válasz visszajuttatása átlátszó módon
- rejtett belső keret – ennek forrása lehet másik tartományban
- **JSON-P** (JSON-with-padding) – más tartományból lehet JS kódot letölteni a `<SCRIPT>` elemmel!

JSON-P példa

```
function handle_data(data) {  
    // 'data' is now the object representation of the JSON data  
}  
---  
http://some.tld/web/service?callback=handle_data:  
---  
handle_data({"data_1": "hello world",  
    "data_2": ["the", "sun", "is", "shining"]}]);
```

- **CORS** (Cross-Origin Resource Sharing) – egyre több böngésző [támogatja](#)

A Yahoo! web proxyja (részlet)

```
<?php
// Allowed hostname (api.local and api.travel are also possible here)
define ('HOSTNAME', 'http://search.yahooapis.com/');
// Get the REST call path from the AJAX application
// Is it a POST or a GET?
$path = ($_POST['yws_path']) ? $_POST['yws_path'] : $_GET['yws_path'];
$url = HOSTNAME.$path;
// Open the Curl session
$session = curl_init($url);
// If it's a POST, put the POST data in the body
if ($_POST['yws_path']) {
    $postvars = '';
    while ($element = current($_POST)) {
        $postvars .= urlencode(key($_POST)).'=' . urlencode($element).'&';
        next($_POST);
    }
    curl_setopt ($session, CURLOPT_POST, true);
    curl_setopt ($session, CURLOPT_POSTFIELDS, $postvars);
}
// Don't return HTTP headers. Do return the contents of the call
curl_setopt($session, CURLOPT_HEADER, false);
curl_setopt($session, CURLOPT_RETURNTRANSFER, true);
// Make the call
$xml = curl_exec($session);
// The web service returns XML. Set the Content-Type appropriately
header("Content-Type: text/xml");
echo $xml;
curl_close($session);
?>
```


- Cél: adatok megőrzése két oldalletöltés között
- Csak a `munkamenet` azonosítóját kell megőrizni a kliensnek (alapesetben `sütivel` vagy `csak azzal`), a szerver `fájl`ban tárolja a munkamenet adatait, ha `másképp` nem rendelkezünk
- Indítás: `session_start()`, sütik esetén **ne nyomtassunk** semmit a hívása előtt!
- Azonosító lekérdezése és beállítása: `session_id()`
- Tárolt adatok elérése: `$_SESSION` tömbön keresztül
- A következő mintaprogram **nem biztonságos**, csak a munkamenetek szemléltetésére szolgál. Használjuk a különféle **keretrendszerek** kiforrott megoldásait, vagy akár **OAuth 2.0**-t!

fooldal.php

```
<?php
function fej() {
    ?>
    <!DOCTYPE html>
    <html>
        <head>
            <meta charset="utf-8">
            <title>Vállalati weboldal</title>
        </head>
        <body>
            <h1>Vállalati weboldal</h1>
        </body>
    </html>
    <?php
    $hiv = [
        "bejelentkezes" => "Bejelentkezés",
        "nyilvanos" => "Nyilvános oldal",
        "titkos" => "Titkos oldal",
        "kijelentkezes" => "Kijelentkezés"
    ];
```

fooldal.php

```
$menu = [];  
foreach($hiv as $k => $e) {  
    $menu[] = "<a href=\"".$_SERVER['PHP_SELF'].  
        "?o=".urlencode($k).\"\">$e</a>";  
}  
echo "<p>", implode(" | ", $menu), "</p>\n";  
}  
  
function lab() {  
    ?>  
    </body>  
    </html>  
    <?php  
}
```

fooldal.php

```
function munkamenet() {
    if(!empty($_POST["nev"]) && !empty($_POST["jelszo"])) {
        if($_POST["nev"] === "a" && $_POST["jelszo"] === "b") {
            $_SESSION["bejelentkezett"] = true;
            echo "<p>Sikeres bejelentkezés.</p>\n";
        } else {
            echo "<p>A felhasználónév vagy a jelszó hibás.</p>\n";
        }
    }
}

session_start();
fej();
munkamenet();
if(!empty($_GET["o"])) {
    include_once($_GET["o"]. ".php");
}
lab();
```

bejelentkezes.php

```
<form action="<?php echo $_SERVER["PHP_SELF"] ?>" method="post">
  <fieldset>
    <legend>Adja meg bejelentkezési adatait!</legend>
    <div><label>Felhasználónév:
    <input type="text" name="nev"></label></div>
    <div><label>Jelszó:
    <input type="password" name="jelszo"></label></div>
    <div><input type="submit" value="Mehet"></div>
  </fieldset>
</form>
```

kijelentkezes.php

```
<?php
  unset($_SESSION["bejelentkezett"]);
?>
<p>Kijelentkezett az oldalról.</p>
```

nyilvanos.php

```
<p>Ezt a tartalmat mindenki láthatja.</p>
```

titkos.php

```
<?php
    if(empty($_SESSION["bejelentkezett"]) ||
        $_SESSION["bejelentkezett"]===false) {
        echo "<p>Előbb <a href=\"", $_SERVER["PHP_SELF"],
            "?o=bejelentkezes\">jelentkezzen be</a>,
            hogy elolvashassa.</p>\n";
        return;
    }
?>
<p>Ez a tartalom szupertitkos, nagy szerencse, hogy bejelentkezett.</p>
```

- Készítsen egyszerű naptár programot az egy napon belüli, személyes elfoglaltságaink kezeléséhez! Az oldal tetején álljon egy, az elfoglaltságok kezdő időpontja szerint rendezett lista, mely tartalmazza az elfoglaltság kezdetét, végét, és annak szöveges megnevezését. Lentebb lehessen megadni HTML5 time beviteli mezőkkel egy új elfoglaltság kezdeti és befejező időpontját, megnevezését. Ha az új elfoglaltság nem ütközik másikkal, vegye fel a listába, különben jelenítsen meg hibaüzenetet!
- A feladat fokozható úgy, hogy amennyiben a befejező időpont korábbi a kezdőnél, akkor automatikusan felcseréli ezeket.
- Rajzoljon táblázatot, melynek 24 sora van (az óráknak megfelelően) és 60 oszlopa (a perceknek). Amelyik percben foglaltak vagyunk, a hozzá tartozó mezőt színezzé a program pirosra, a többi zöldre! A táblázatot lássa el fejlécekkel is!

- Egy mozi nézőterén egy sorban 10 szék van, a széksorok száma pedig 20. A nézőtér székeit jelenítse meg HTML táblázatban! Minden sort jelöljön meg az abc nagybetűível, és minden oszlopot számokkal! A már foglalt székeknél jelenítsen meg egy **X** betűt, a szabad székeknél pedig egy jelölőnégyzetet helyezzen el a szék kijelöléséhez! Amikor a táblázat alatti **Lefoglal** (submit) gombra kattint a felhasználó, foglaltra kell állítani a kijelölt székeket és újra kell tölteni, frissíteni kell az oldalt.
- A programot úgy kell elkészíteni, hogy bármelyik gépen is töltsék le a weboldalt, látszódjanak a korábbi foglalások.
- A feladat továbbfejleszthető úgy, hogy nem csak a foglalás tényét, hanem az egyszerre megvásárolt jegyek tulajdonosának nevét is meg lehessen adni. Ha valaki olyan helyre (is) kívánna jegyet venni, amit más előzőleg (egy másik böngészőből) már megvásárolt, írja ki a program, hogy a kérés nem teljesíthető, és jelezze ki a gyorsabb vevő(k) nevét, és az „ütköző” székek azonosítóit is!

Fejlessze tovább az előző heti, munkatársak megjelenítésével kapcsolatos feladat megoldását úgy, hogy

- egy keresőmezőbe begépelve valamelyik munkatárs nevét vagy annak egy részét, csak az adott karakterláncot tartalmazó nevű kollégákat listázza!
- A munkatársak adatait tárolja CSV vagy JSON formátumban!
- Tegye lehetővé új kolléga adatainak rögzítését egy adatfelviteli oldalon, amire a főoldalról lehet elnavigálni! Lehesse fényképeket is feltölteni!
- Alakítsa át úgy az oldalt, hogy teljes tartalom előállítások helyett csak AJAX hívásokat intéz a szerverhez!

Larry Ullman

[Modern JavaScript: Develop and Design](#)

Peachpit Press, Berkeley, CA, USA, 2012.

N. C. Zakas, J. McPeak, J. Fawcett

[Professzionális AJAX](#)

Szak Kiadó, Bp. 2007.

[Hivatalos PHP referencia](#)