JESUÏTES educació

DAWM06UF4
COMUNICACIÓ ASÍNCRONA CLIENT-SERVIDOR

UF4.2.1 AJAX

CFGS Desenvolupament d'Aplicacions Web
M06. Desenvolupament web en entorn client
Fundació Jesuïtes Educació - Escola del Clot
Sergi Grau sergi.grau@fje.edu

OBJECTIUS

- Aprendre les característiques de les comunicacions asíncrones client/servidor
- Comprendre com funciona la tecnologia AJAX
- Utilitzar els objectes XHR i XHR2

QUÈ ÉS AJAX ?

- AJAX són un conjunt de tecnologies que permeten desenvolupar RIAs, és a dir, aplicacions web amb característiques de les aplicacions d'escriptori.

- Alternatives a AJAX són HTML 5 o les que requereixen la instal·lació de plug-ins: Adobe Flash, Adobe Flex, Java Applets, JavaFX i/o Microsoft Silverlight.

- El creador del terme va ser Jesse James Garrett d'Adaptative path al 2005.

- Es fa fer molt popular amb Google Suggest al 2005

ARQUITECTURES D'APLICACIONS WEB BASADES EN XHR

JESUÏTES educació
DAWM06UF4 sergi.grau@fje.edu

- AJAX (acrònim de Asynchronous Javascript and XML) és un conjunt de tecnologies (XHTML/CSS, DOM, Javascript, XHR, XML/XSLT) que permet carregar i mostrar una pàgina, per després mitjançant scripts accedir al servidor en "background" demanant informació i mostrant les dades rebudes per actualitzar la pàgina de manera parcial.

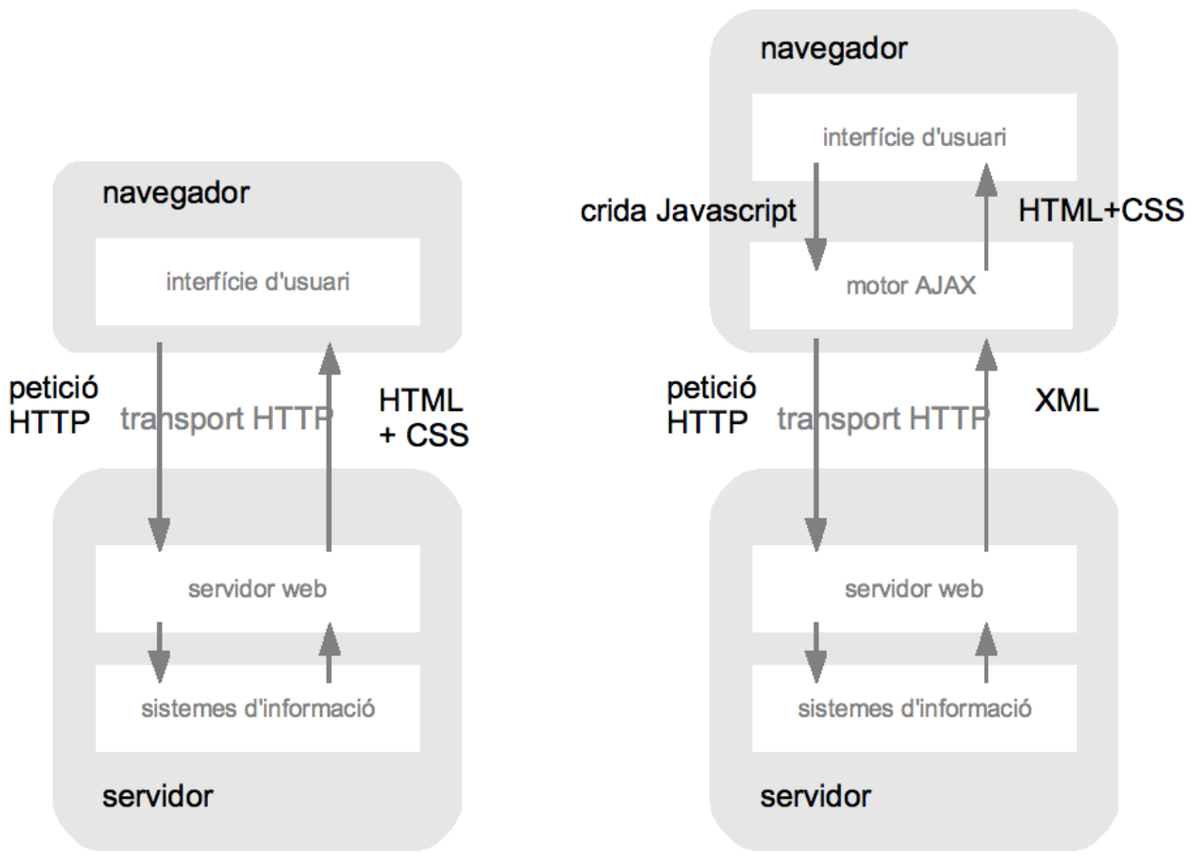
- El model clàssic d'aplicacions web consisteix en que l'usuari realitza una petició HTTP al servidor web, aquest efectua el procés requerit i retorna (habitualment) informació al client en forma de document HTML.

+ El model AJAX presenta una alternativa al funcionament clàssic. En comptes de definir la interacció amb l'usuari de la forma "arrencar i parar", s'afegeix un intermediari (anomenat motor AJAX) entre el l'usuari i el servidor que permet que les crides al servidor es facin de manera que no es requereixi un enviament actiu d'informació per part de l'usuari.

ARQUITECTURES D'APLICACIONS WEB BASADES EN XHR

JESUÏTES educació
DAWM06UF4 sergi.grau@fje.edu

- Comparació del funcionament de l'arquitectura client-servidor amb el model clàssic de les aplicacions web (a l'esquerra) i el model AJAX (a la dreta).

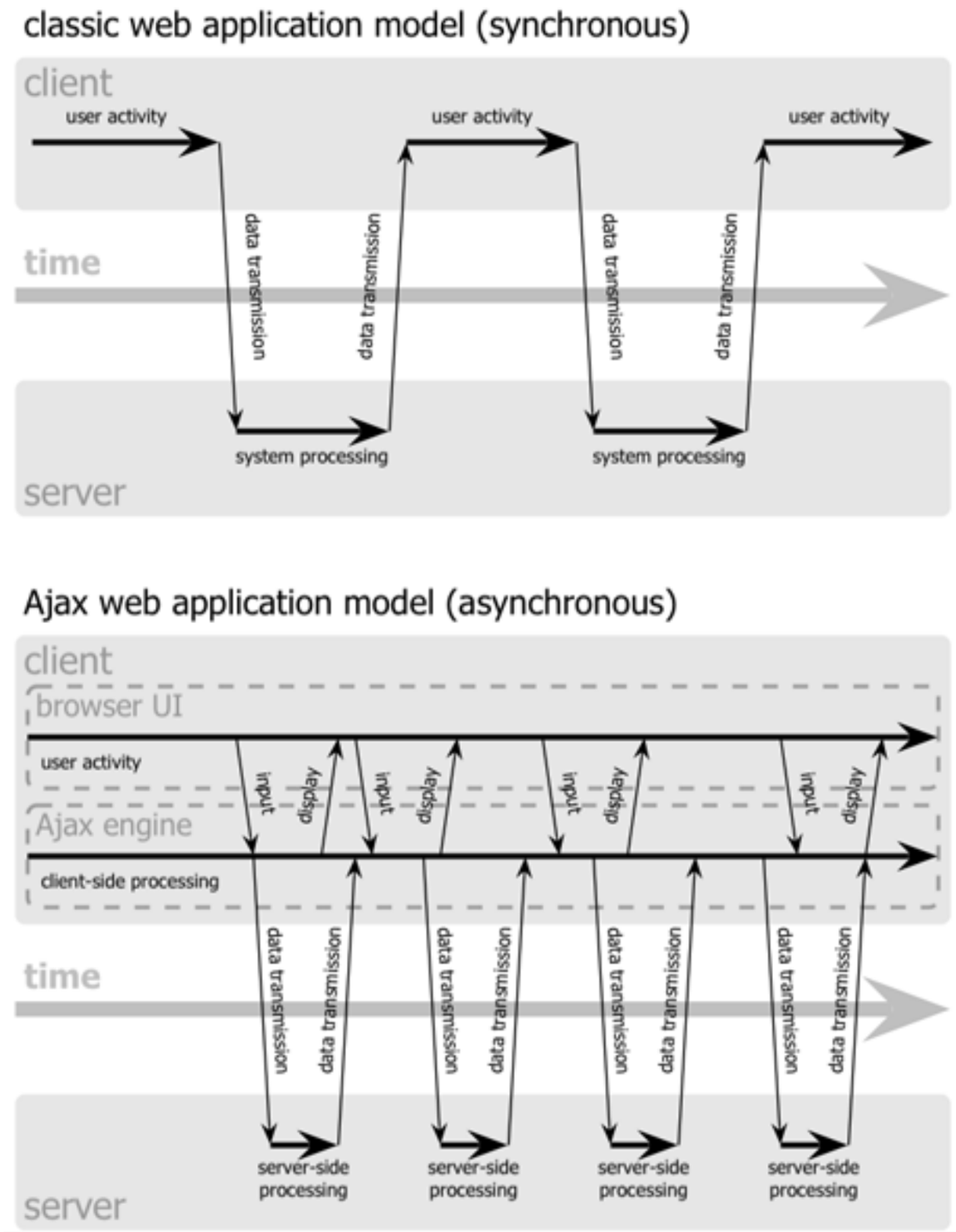


model clàssic d'aplicacions web

model AJAX d'aplicacions web

ARQUITECTURES D'APLICACIONS WEB BASADES EN XHR

JESUÏTES educació
DAWM06UF4 sergi.grau@fje.edu



EL MOTOR AJAX I L'OBJECTE XHR

- El motor AJAX està habitualment escrit en Javascript i es troba inserit en un marc ocult de la pàgina web. Aquest motor s'encarrega de la presentació de la interfície d'usuari i permet que la interacció es faci de manera asíncrona (independentment de la comunicació amb el servidor).

- Els diversos recursos del servidor són cridats mitjançant Javascript des de la pàgina web. Això permet fer peticions del contingut de manera asíncrona (també anomenades crides en background). Finalment, el contingut que XMLHttpRequest retorna pot ser tractar com a XML o com a text pla.

- **XMLHttpRequest** és el component tècnic que fa possible les comunicacions asíncrones amb el servidor. Va ser creat per Microsoft i el primer navegador en implementar-ho va ser l'Internet Explorer 5.

- Cal indicar que hi ha diverses implementacions de XMLHttpRequest. Disposem de l'objecte XMLHttpRequest per la majoria de navegadors (Firefox, Mozilla, Opera, Safari) i en el cas del Microsoft Internet Explorer disposem del component ActiveX Microsoft.XMLHTTP. Ara bé, cal tenir present que les versions més antigues dels navegadors no suporten l' XMLHttpRequest.

AVANTATGES D'AJAX PER CREAR RIAS

- Ofereix crides asíncrones al servidor. Un resultat que es desprèn
  d'aquest fet és que l'usuari veu una continuïtat en les seves accions, és a dir, la pantalla del navegador no es queda en blanc esperant la resposta del servidor a una petició.

- Presenta una facilitat en el tractament de la informació que es sol·licita. Li passem la URL amb la informació que es vol consultar, i quan està disponible es pot tractar com a XML/ XHTML mitjançant el mètode responseXML o text amb el mètode responseText.

AVANTATGES D'AJAX PER CREAR RIAS

- Millora la usabilitat i l'experiència de l'usuari amb la interfície d'usuari, oferint interfícies riques pròximes a les aplicacions d'escriptori. Es pot utilitzar en diversos usos com: validació de formularis en temps real, autocompletament, operacions mestre/detall, sofisticats controls d'usuari, refrescament de dades en la pàgina i notificacions del costat del servidor.

- Tot el procés es realitza amb esdeveniments Javascript (com onclick, onblur, etc.)

INCONVENIENTS D'AJAX PER CREAR RIAS

- Suposa una complexitat major en el procés de desenvolupament de les aplicacions web, especialment pel canvi de mentalitat necessari per utilitzar la nova arquitectura en el procés de disseny de les aplicacions.

- Cal detectar el tipus de navegador del client per accedir a l'objecte correcte (XMLHttpRequest o Microsoft.XMLHTTP)

- Els navegador antics, els simplificats i aquells que tenen desactivada la interpretació de Javascript no el suporten. A més, XMLHttpRequest no forma part encara de l'especificació de la tecnologia Javascript, de la qual hi ha diverses implementacions.

INCONVENIENTS D'AJAX PER CREAR RIAS

- Presenta problemes en el procés de depuració i seguretat de les aplicacions web pel costat del client.

- No són possibles (per seguretat de JavaScript) les crides de recursos fora del domini de la pàgina sol·licitada.

ESTÀNDARDS IMPLICATS EN LA TECNOLOGIA AJAX

- Pel que fa a la presentació dels documents (les pàgines web) s'utilitza XHTML (o HTML) i CSS. La utilització conjunta d'XHTML i CSS permet separar el contingut del format de presentació de la informació.

- DOM (Document Object Model) ofereix la exhibició i interacció dinàmica amb les pàgines web. DOM permet accedir als diversos elements que conté un document HTML i a determinats elements dels navegadors. Per tant, millora la interacció de l'usuari amb les pàgines web.

- Per accedir al DOM i a XHR s'utilitza JavaScript /ECMAScript.

ESTÀNDARDS IMPLICATS EN LA TECNOLOGIA AJAX

- L'intercanvi i manipulació dels documents es pot fer mitjançant XML i XSL (llenguatge per a expressar fulls d'estil que descriu com mostrar un document XML).

- També es pot utilitzar JSON com a format d'intercanvi.

- HTTP és el protocol de nivell d'aplicació que s'utilitza per accedir a la informació que està allotjada en un servidor web. Segueix el model de peticions i respostes entre un client i un servidor i es basa en un servei de transport fiable (TCP). AJAX utilitza HTTP com a protocol de nivell de d'aplicació per intercanviar informació entre client i servidor.

- Serveis web, REST i SOAP (Simple Object Access Protocol).

EXEMPLE COMPATIBILITAT IE 6


```
var xhr;
try {
    // Firefox, Opera 8.0+, Safari, Chrome
    xhr = new XMLHttpRequest();
}
catch (e) {
    // Internet Explorer
    try {
        xhr = new ActiveXObject("Msxml2.XMLHTTP"); //ie6+
    }
    catch (e) {
        try {
            xhr = new ActiveXObject("Microsoft.XMLHTTP");//ie5
        }
        catch (e) {
            alert("El teu navegador no suporta AJAX!");
            return false;
        }
    }
}
```

EXEMPLE IE 7+

```
xhr = new XMLHttpRequest();

if (!xhr) {
  alert(‘problemes amb AJAX');
  return false;
}
```

```
 //callback
     xhr.onreadystatechange = function () {
          if (xhr.readyState == 4) {

document.getElementById("cadenaInvertida").innerHTML =
xhr.responseText;
          }
     }


     xhr.open("GET", "Exercici3.php?cadena=" + cadena, true);
     xhr.send(null);
}
```

EXEMPLE

- La propietat onreadystatechange té assignat un callback, que és codi executable que es passa com un paràmetre a una altra funció. En el nostre cas passem una funció a l'objecte XHR, el qual cridarà a la funció quan estigui disponible.

- Aquesta propietat és un controlador d'esdeveniments que s'activa cada vegada que l'estat de la sol.licitud de canvis canvia. Els estats van des de zero (sense inicialitzar) fins a quatre (completa). Això és important perquè el programa no ha d'esperar la resposta abans de continuar.

EINES I PLATAFORMES PER A DESENVOLUPAR AMB AJAX.

+ Existeix un nombre molt elevat d'eines i plataformes que permeten desenvolupar aplicacions Web que utilitzen AJAX. La manera més senzilla consisteix a utilitzar directament l'objecte XMLHttpRequest i JavaScript. Això és recomanable quan es vol desenvolupar una petita aplicació però no és adient en el cas d'aplicacions de mida mitjana o gran. Existeixen diversos bastiments que permeten utilitzar AJAX i que simplifiquen en alguna mesura el procés de desenvolupament i de manteniment d'aquestes aplicacions.

EINES I PLATAFORMES PER A DESENVOLUPAR AMB AJAX.

- Aquests frameworks es podem classificar en funció del llenguatge que s'utilitza per programar-los. Els més habituals són els que ajuden al desenvolupament de la capa client i que es basen en XHTML, CSS i Javascript, essent neutrals envers la tecnologia emprada en el servidor. Els més coneguts són Jquery, Angular I React. El major inconvenient que presenta la utilització d'aquests bastiments és que el programador ha de tenir bons coneixements de Javascript, i que aquest no és el llenguatge que s'utilitzarà normalment el costat del servidor.

- Altres opcions consisteixen a utilitzar eines de desenvolupament que permeten utilitzar el llenguatge emprat en el costat servidor per a generar la capa client. Els més coneguts són ECHO, GWT i DWR, Microsoft .NET o SAJAX si el llenguatge servidor és PHP.

LLISTAT DE FRAMEWORKS QUE PERMETEN AJAX

- http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks

- http://en.wikipedia.org/wiki/List_of_Ajax_frameworks

ATRIBUTS DE XHR

+ onreadystatechange
+ readyState
+ responseText
+ responseXML
+ status
+ statusText

METODES DE XHR

+ abort
+ getAllResponseHeaders
+ getResponseHeader
+ open
+ send
+ setRequestHeader

XMLHTTPREQUEST

- Hi ha dos mètodes per instanciar un objecte XMLHttpRequest object (XHR).
- Microsoft Internet Explorer utilitza un component ActiveX
- var xhr = new ActiveXObject("Microsoft.XMLHTTP");
- Mentre que Mozilla i Safari, és un objecte nadiu.
- var xhr = new XMLHttpRequest();
- Caldrà determinar quin navegador executa el codi i suportar l'objecte adequat.

ONREADYSTATECHANGE

- L'atribut onreadystatechange assigna la funció que se executarà cada vegada que readyState canvii de valor.

- xhr.onreadystatechange = callback;

- xhr – Objecte XMLHttpRequest

- callback – funció a executar

- Normalment utilitzem onreadystatechange per a definir una funció per a llegir les dades rebudes del costat servidor (quan **readyState** tingui valor 4 i llegim les dades amb responseXML, responseText..)

READYSTATE

- Per a rebre la petició realitzada, elprimer que farem serà comprovar l'estat de la petició i ho farem amb el mètode readyState que ens pot tornar qualsevol dels següents valors:

- 0 (No inicialitzat) - Les dades de la petició no s'han definit.
  1 (Obert) - La recepció de dades està en curs.
  2 (Carregat) - La recepció de dades ha finalitzat però les dades no estan disponibles.
  3 (Interactuant) - L'objecte encara no és llest per a cap altra petició però ha rebut ja les dades.
  4 (Completat) - L'objecte és llest per a una altra petició

- Una vegada estem en estat carregat, ja podem llegir el text rebut usant el responseText o responseXML.

RESPONSETEXT

- Una alternativa a onreadystatechange és la utilització d'un timeout amb setTimeout abans de la lectura.

- Sempre serà necessari comprovar l'estat de la propietat readyState de XHR, només amb el valor 3 o 4 es pot accedir als atributs responseXML i responseText.

- L'atribut responseText retorna el text del document obtingut del costat servidor en una petició amb XMLHttpRequest.

- cadena = xhr.responseText;

- La propietat responseText s'utilitza per a tractar les dades rebudes des del servidor que no tenen format XML

RESPONSEXML

- L' atribut responseXML retorna una referencia al cos del document obtingut del servidor en una petició amb XHR en format XML.

- docXML = xhr.responseXML;

- La propietat responseXML s'utilitza per a tractar les dades rebudes en format XML

- Aquesta propietat retorna null sempre que la resposta XML del servidor no tingui l'encapçalament text/xml, application/xml o acabi en +xml.

- Per extreure les dades caldrà utilitzar DOM amb JavaScript.

STATUS

- La propietat status s'utilitza per comprovar que no hi ha hagut problemes en la comunicació amb el servidor, podrem accedir a les dades sempre i quan l'estat de la connexió tornat amb readyStatechange sigui igual a 3 (rebent) o 4 (a punt).

- enter = xhr.status;

- El codi d'estat HTTP per a una transmissió correcta és el 200, serà convenient comprovar aquesta dada abans d'accedir a les dades amb responseXML o responseText, (per a més informació veure apunts d'HTTP)

STATUSTEXT

- L’atribut statusText torna el text de l’estat HTTP de la transmissió tornat pel servidor web.

- cadena = xhr.statusText;

- La propietat statusText no és d’ús comú, normalment al seu lloc usarem status, podrem accedir a les dades sempre i quan l'estat de la connexió tornat amb readyStatechange sigui igual a 3 (rebent) o 4 (a punt).

- El text d’estat HTTP per a una transmissió correcta és 'OK'.

ABORT

- El mètode abort atura totes les connexions asíncrones obertes per l'objecte XMLHttpRequest i el reinicialitza posant a zero el seu estat (readyState).

- xhr.abort ();

- Habitualment s’utilitza abort abans de realitzar una nova petició al servidor a través d’un objecte que està realitzant o rebent una altra petició anterior.

GETALLRESPONSEHEADERS

- El mètode getAllResponseHeaders torna en una sola cadena de caràcters els encapçalaments HTTP que s'han rebut del servidor en una connexió usant l'objecte XMLHttpRequest.

- cadena = xhr.getAllResponseHeaders ();

- La cadena contindrà tots els encapçalaments enviats pel servidor excepte el d'estat (per exemple HTTP/1.1 200 OK), els encapçalaments seran tornats separats per la combinació 'salt de línia + retorn de carro'.

- Podrem accedir als encapçalaments sempre i quan l'estat de la connexió tornat amb readyStatechange sigui igual a 3 (rebent) o 4 (a punt).

GETRESPONSEHEADER

- El mètode getResponseHeader torna en una sola cadena de caràcters un dels encapçalaments HTTP que s'han rebut del servidor en una connexió usant el objecte XMLHttpRequest.

- cadena = xhr.getAllResponseHeaders (NomCap);

- La cadena contindrà tots els encapçalaments amb nom igual a NomCap enviats pel servidor separats per la combinació 'coma' + 'espai'.

- Podrem accedir als encapçalaments sempre i quan l'estat de la connexió tornat amb readyStatechange sigui igual a 3 (rebent) o 4 (a punt).

OPEN

- El mètode open prepara una connexió HTTP a través del objecte XMLHttpRequest (amb un mètode i una URL especificats) i inicialitza tots els atributs de l'objecte.

- xhr.open (mètode, URL [, Sincronia [, Usuari [Pwd]]]);

- Mètode = String amb el mètode de connexió (GET o POST)
  URL=URL per a la petició HTTP
  Sincronia= Booleà opcional en true per usar manera asíncrona i en false per a síncron.
  Usuari = Cadena de caràcters opcional amb el nom d'usuari per a l'autenticació, Pwd = Cadena de caràcters opcional amb la contrasenya de l'usuari Usuai per l'autenticació.

- Al cridar a open l'atribut readyState a 1, reseteja els headers de tramesa i els torna atributs responseText, responseXML, status i statusText als seus valors inicials

SEND

- El mètode send envia la petició amb les dades passades per paràmetre com a cos de la petició a través del objecte XMLHttpRequest.

- xhr.send (data);

- Data = Cos de la petició HTTP.

- El paràmetre data pot ser una referència al DOM d'un document o una cadena de caràcters.

- Es recomana passar sempre el paràmetre data encara que no sigui requerit en alguns navegadors.

- No es permeten les crides a dominis, ports o protocols diferents del de la pàgina que crida la funció. L'URL pot contenir un usuari i una contrasenya que s'usaran en cada connexió i tindran preferència davant dels passats per paràmetre a la funció. Els paràmetres Usuari i Pwd solen s'usaran per enviar-los si es rep una resposta 401 - Access Denied mentre que per URL s'usaran sempre.

- Per realitzar la connexió haurem d'usar send després de open

SETREQUESTHEADER

- El mètode setRequestHeader afegeix un encapçalament HTTP a la petició HTTP a través de l'objecte XMLHttpRequest.

- xhr.setRequestHeader (nom, valor);

- nom = Nom de l'encapçalament HTTP.
  valor = Valor de l'encapçalament HTTP.

- El paràmetre nom no podrà ser Accept-Charset, Accept-Encoding, Content-Length, Expect, Date, Host, Keep-Alive, Referer, TE, Trailer, Transfer-Encoding ni Upgrade, tampoc no podrà contenir espais, punts o salts de línia.

- El paràmetre valor no podrà contenir salts de línia. Cal que setRequestHeader quan el valor de readyState sigui 1.

CAPÇALERES

- Els encapçalaments: El primer pas perquè el navegador interpreti el contingut rebut és que tingui l'encapçalament de contingut XML (text/xml), això ho aconseguim amb el següent encapçalament HTTP:

- A més, com la nostra resposta XML serà habitualment generada de manera dinàmica, és recomanable enviar també encapçalaments de control de memòria cau per assegurar-nos que l'aplicació sempre estarà treballant amb els continguts que sol·licita i no amb una memòria cau emmagatzemada al seu navegador.

```
<?php
header("Content-Type: text/xml");
header("Cache-Control: no-cache, must-revalidate");
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
?>
```

Per realitzar la petició al servidor, utilitzarem els mètodes open, onreadystatechange i send, que serveixen respectivament per preparar la petició, seleccionar la funció de recepció i iniciar la petició.
Al mètode open, cal passar-li el mètode de petició (GET) i l'URL que s'enviarà al servidor i mitjançant la qual, el servidor, crearà la resposta que posteriorment llegirem.
Exemple 1:

```
<script>
// Creem l'objecte ...
// Preparem la petició
oXML.open('GET', 'fitxer.txt');
// Preparem la recepció
oXML.onreadystatechange = llegirDades;
// Realitzem la petició
oXML.send('');
</script>
```

Perquè això funcioni, haurem d'haver declarat la funció llegirDades per tractar les dades rebudes del servidor i mostrar-los a l'usuari, però això ho veurem més endavant.

Pas de paràmetres: En la petició AJAX podem passar paràmetres tant per POST com per GET al nostre servidor.
Per a passar paràmetres per GET (per URL) , usarem una URL amb paràmetres en la funció open independentment d'usar el mètode GET o POST, per exemple:

EXEMPLE

```
<script>
// Creem la variable paràmetre
parametre = 'Dades passades per GET';
// Creem l'objecte
...
// Preparem la petició amb paràmetres
xhr.open('GET', 'pagina.php?parametre=' + escape(parametre));
// Preparem la recepció
xhr.onreadystatechange = llegirDades;
// Realitzem la petició
xhr.send('');
</script>
```

Per passar-los per POST, haurem d'usar el mètode POST en la funció open i passar els paràmetres des de la funció send.

XHR ADVANCE LEVEL 2 (XHR2)

- XMLHttpRequest de nivell 2 introdueix una gran quantitat de noves funcions, com peticiós d'origen creuat, esdeveniments de progrés de pujades i compatibilitat amb pujada / baixada de dades binàries.

- Això permet a AJAX treballar en coordinació amb moltes de les API HTML5 més punteres, com API de filesystem, l'API de Web Audio i WebGL.

XHR ADVANCE LEVEL 2 (XHR2)

- les noves propietats de XMLHttpRequest (responseType i response) permeten indicar al navegador el format en què volem que s'ens torni les dades.

- **xhr.responseType**, abans d'enviar una petició, estableix xhr.responseType a "text", "arraybuffer", "blob" o "document", en funció de les dades que es necessitin. Si s'estableix xhr.responseType = '' (o si s'omet), s'utilitzarà la resposta predeterminada "text".

- **xhr.response**, Després d'una petició correcta, la propietat response de XHR contindrà les dades sol·licitades com DOMString, ArrayBuffer, Blob o Document (en funció del valor establert per responseType).

```
var xhr = new XMLHttpRequest();
        xhr.open('GET', 'imatge', true);
        xhr.responseType = 'blob';
        xhr.onload = function(e) {
           if (this.status == 200) {
              var blob = this.response;
              var img = document.createElement('img');
              img.onload = function(e) {
                 window.URL.revokeObjectURL(img.src);
              img.src = window.URL.createObjectURL(blob);
              document.body.appendChild(img);
           }
        };

        xhr.send();
```

XHR2 RESPOSTES DE ARRAYBUFFER

- ArrayBuffer és un contenidor genèric de longitud fixa per dades binàries. És molt pràctic si es necessita un memòria intermèdia per a dades sense processar, però el veritable potencial radica que pots crear "vistes" de les dades subjacents mitjançant matrius JavaScript.

- De fet, es poden crear diverses vistes des d'un únic ArrayBuffer. Per exemple, es pot crear una matriu d'enters de 8 bits que comparteixi el mateix ArrayBuffer amb una matriu d'enters de 32 bits a partir de les mateixes dades.

- Les dades subjacents segueixen sent les mateixes, simplement vam crear representacions diferents d'elles.

XHR2 RESPOSTES DE BLOB

- Si es vol treballar directament amb un Blob o no es necessita manipular ni un sol byte de l'arxiu, cal utilitzar xhr.responseType = ‘blob’;

- Un Blob té diferents usos. Per exemple, es pot guardar en IndexedDB, es pot escriure en el sistema d'arxius de HTML5 o es pot utilitzar per crear un URL del tipus Blob.

XHR2 ENVIAMENT DE DADES

- Poder descarregar dades en diferents formats està molt bé, però no ens porta enlloc si no podem enviar aquests formats enriquits una altra vegada al servidor.

- Durant algun temps, XMLHttpRequest ens ha limitat a enviar dades DOMString o Document (XML). Però això s'ha acabat.

- Amb XHR2 el mètode send () s'ha redissenyat per acceptar tots aquests tipus: DOMString, Document, FormData, Blob, File i ArrayBuffer.

XHR2 XHR.SEND(FORMDATA)

- Estem acostumats a utilitzar complements jQuery o altres biblioteques per enviar formularis AJAX. En el seu lloc, podem utilitzar FormData, un altre nou tipus de dades creat per XHR2.

- FormData permet crear còmodament un objecte <form> HTML al vol en JavaScript. A continuació, aquest formulari es pot enviar mitjançant AJAX.

- Bàsicament, estem creant de forma dinàmica un objecte <form> i realitzant un seguiment dels seus valors <input> mitjançant el mètode append.

- Per descomptat, no cal crear un objecte <form> des de zero. Els objectes FormData es poden inicialitzar a partir d'un element HTMLFormElement de la pàgina

```
function sendForm() {
  var formData = new FormData();
  formData.append('nom', 'sergi');
  formData.append('id', 123456);

  var xhr = new XMLHttpRequest();
  xhr.open('POST', '/servidor', true);
  xhr.onload = function(e) { ... };

  xhr.send(formData);
}
```

XHR2 XHR.SEND (BLOB)

- També podem enviar dades de File o Blob amb XHR, la qual cosa permet fer pujades de fitxers.

- Tingues en compte que tots els elements File són Blob

EXEMPLE

```
<progress min="0" max="100" value="0">0% completat</progress>

function upload(blobOrFile) {
  var xhr = new XMLHttpRequest();
  xhr.open('POST', '/servidor', true);
  xhr.onload = function(e) { ... };

  // listener al procés de càrrega
  var progressBar = document.querySelector('progress');
  xhr.upload.onprogress = function(e) {
    if (e.lengthComputable) {
      progressBar.value = (e.loaded / e.total) * 100;
      progressBar.textContent = progressBar.value;
    }
  };

  xhr.send(blobOrFile);
}
upload(new Blob([‘hola DAW2'], {type: 'text/plain'}));
```

ÚS COMPARTIT DE RECURSOS D'ORIGEN CREUAT (CORS)

- CORS permet a les aplicacions web d'un domini realitzar peticiós AJAX de domini creuat a un altre domini.

- És molt fàcil d'habilitar. Només cal que el servidor enviï un sol encapçalament de resposta.

- Suposem que l'aplicació resideix en exemple.com i vols extreure dades de www.exemple2.com. Normalment, si intentes executar aquesta crida AJAX, la petició fallarà i el navegador llançarà un error per la manca de correspondència dels orígens.

ÚS COMPARTIT DE RECURSOS D'ORIGEN CREUAT (CORS)

- Amb CORS, www.example2.com només ha d'afegir una capçalera per a permetre peticiós de example.com:
  - `Access-Control-Allow-Origin: http://exemple.com`
- Es pot afegir Access-Control-Allow-Origin a un sol recurs d'un lloc o a tot el domini. Per permetre que qualsevol domini pugui fer una petició, s'estableix:
  - `Access-Control-Allow-Origin: *`
- Si el punt final del servidor ha habilitat CORS, realitzar una petició d'origen creuat serà igual que realitzar una petició XMLHttpRequest normal.

BIBLIOGRAFIA

+ https://developer.mozilla.org/en-US/docs/AJAX
+ https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest
+ https://www.html5rocks.com/en/tutorials/file/xhr2/
+ http://www.w3schools.com/ajax/