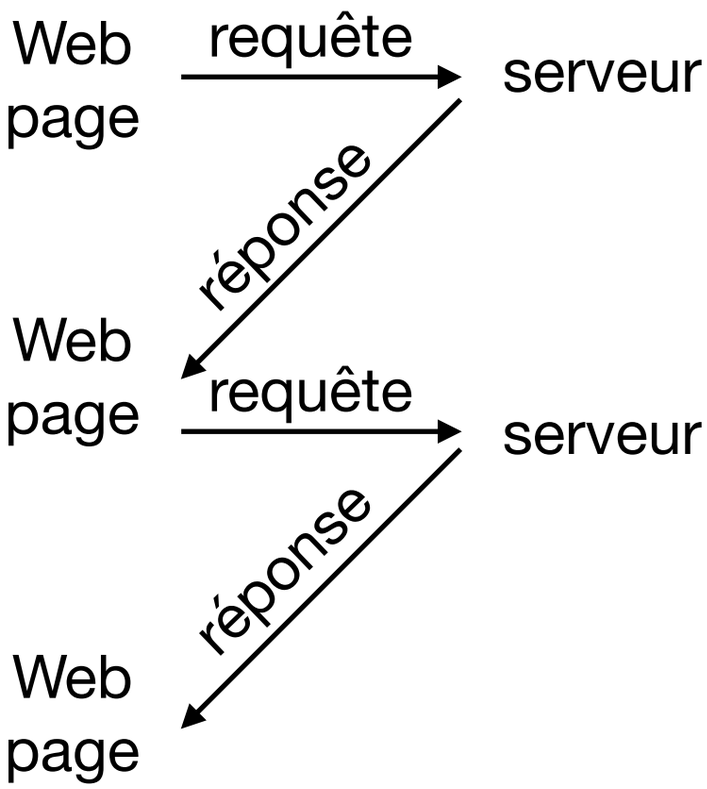
AJaX (Asynchronous Javascript XML)

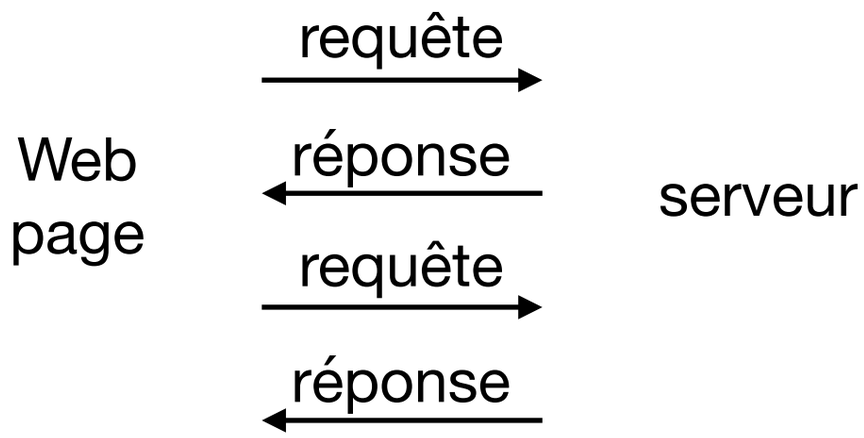
felipe@IFT3225 H2020

Principe

Jusqu’à maintenant (synchrone):



Ajax (asynchrone):



Pros:

- Plus fluide
  - pas de chargement d’une nouvelle page
- Plus naturel à l’utilisateur
- Non bloquant (asynchrone)

Faits

- Technologie développée initialement par Microsoft pour IE
- Terme inventé en 2005
- W3C en 2006
- WHATWG en 2012

XMLHttpRequest

Living Standard — Last Updated 17 February 2020

Participate:

GitHub whatwg/xhr (new issue, open issues)
IRC: #whatwg on Freenode

Commits:

GitHub whatwg/xhr/commits
Snapshot as of this commit
@xhrstandard

Tests:

web-platform-tests xhr/ (ongoing work)

Translations (non-normative):

日本語

Abstract

The XMLHttpRequest Standard defines an API that provides scripted client functionality for transferring data between a client and a server.

En pratique

1. Créer un objet `XMLHttpRequest`
2. Enregistrer un gestionnaire (*callback*)
3. Spécifier les paramètres de la connexion
4. Envoi de la requête

Grandement facilité par JQuery ou Prototype

1. Créer un objet

En principe:

```
var req = new XMLHttpRequest();
```

En pratique:

```
var XMLHttpFactories = [
    function () {return new XMLHttpRequest()},
    function () {return new ActiveXObject("Msxml3.XMLHTTP")},
    function () {return new ActiveXObject("Msxml2.XMLHTTP.6.0")},
    function () {return new ActiveXObject("Msxml2.XMLHTTP.3.0")},
    function () {return new ActiveXObject("Msxml2.XMLHTTP")},
    function () {return new ActiveXObject("Microsoft.XMLHTTP")}
];

function createXMLHTTPObject() {
    var xmlhttp = false;
    for (var i=0;i<XMLHttpFactories.length;i++) {
        try {
            xmlhttp = XMLHttpFactories[i]();
        }
        catch (e) {
            continue;
        }
        break;
    }
    return xmlhttp;
}
```

Rangé dans createXHR de ajax.js

2- Enregister un gestionnaire

```
var xhr = createXHR();

xhr.onreadystatechange  = function()
{
        alert(xhr.readyState);
        if ((xhr.readyState === 4) &&
(xhr.status === 200))
            // do something
};
```

```
// states
const unsigned short UNSENT = 0;
const unsigned short OPENED = 1;
const unsigned short HEADERS_RECEIVED = 2;
const unsigned short LOADING = 3;
const unsigned short DONE = 4;
readonly attribute unsigned short readyState;
```

event handler	event handler event type
onloadstart	loadstart
onprogress	progress
onabort	abort
onerror	error
onload	load
ontimeout	timeout
onloadend	loadend

The following is the event handler (and its corresponding even
by the XMLHttpRequest object:

event handler	event handler event type
onreadystatechange	readystatechange

3&4- lancer la requête

```
var xhr = createXHR();

xhr.onreadystatechange  = function()    {
       alert(xhr.readyState);
       if ((xhr.readyState === 4) && (xhr.status === 200))
          // do something
};

// etablir la connection
xhr.open("GET", "hello.txt");
xhr.send();
```

For web developers (non-normative)

***client* . open(*method*, *url* [, *async* = true [, *username* = null [, *password* = null]]])**

Sets the request method, request URL, and synchronous flag.

Throws a "SyntaxError" DOMException if either *method* is not a valid HTTP method or *url* cannot be parsed.

Throws a "SecurityError" DOMException if *method* is a case-insensitive match for `CONNECT`, `TRACE`, or `TRACK`.

Throws an "InvalidAccessError" DOMException if *async* is false, current global object is a Window object, and the timeout attribute is not zero or the responseType attribute is not the empty string.

Un exemple GET complet

```
function submitForm() {
   var xhr = createXHR();

   xhr.onreadystatechange  = function()    {
       alert(xhr.readyState);
       if ((xhr.readyState === 4))
          document.getElementById('dyn').value =
           (xhr.status  == 200) ?
                    xhr.responseText :
                    ("Error code " + xhr.status);

   // etablir la connection
   xhr.open("GET", "hello.txt");
   xhr.send();
}
```

ressource contenant le texte « bonjour ! »

```
<body>
    <form action="">
         <input type="submit" value="send"  onclick="submitForm()">
         <input type="text" id="dyn" size="40" value="">
    </form>
</body>
```

Exemple POST

Ajax POST Demo - Sending a Plain Text

The demo makes use of a short PHP script to receive the data and put it into a text file.

Enter a text into the field below and click on the submit button at right to send it to the server.
Then you can retrieve your text by a click on the link that will appear below the form.

For security issue, the name of the file to create must be defined and recognized on the server.

dfgdfgdg

Submit

View the content of the file you have created, ajax-post.txt.

Cette page est une version modifiée de: xul.fr

le contenu du texte saisi est envoyé via XMLHttpRequest par POST à ajax-post.php

Une enveloppe est ajoutée au contenu:
ajax-post.php?file=ajax-post.txt&content=<content>

La requête

```
function Write(url, content)
        {
                var xhr = createXHR();

                xhr.onreadystatechange = function() {
                  if (xhr.readyState == 4) {
                    Modify("View the content of the file you have created <b> " +
                                " <a href='ajax-post.txt' target='_parent'>ajax-post.txt</a></b>.");
                  }
                }

                xhr.open("POST", url, true);
                xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
                xhr.send(content);
}
```

ajax-post.php

```php
<?php if (isset($_GET['source'])) die(highlight_file(__FILE__, 1)); ?>

<?php

$fname = $_POST["file"];
if (strcmp($fname, "ajax-post.txt") != 0)
  die("You are not authorized to change this file.");


$value = $_POST["content"];

// sauvegarde dans un fichier du texte transmis url-encodé via post
$nfile = fopen($fname, "w");

if($nfile != false) {
    fwrite($nfile, $value);
    fclose($nfile);
}

?>
```

pour éviter la prolifération de ressources sur votre espace web !

Exemple avec XML

ressource XML côté serveur

%more data/vote.xml

```
<vote>
<navigateur name="MSIE">127</navigateur>
<navigateur name="Safari">539</navigateur>
<navigateur name="FX2">355</navigateur>
<navigateur name="Konqueror">15</navigateur>
<navigateur name="Chrome">425</navigateur>
<navigateur name="Opera">19</navigateur>
</vote>
```

Mise à jour par poll.php qui reçoit (via Ajax/GET) le choix du navigateur sélectionné dans le formulaire de la page ajax-poll.html

Exemple avec XML

```php
<?php

if (isset($_GET['source'])) die(highlight_file(__FILE__, 1));

  /*
   *  code: poll.php
   */

header('Content-Type: text/xml; charset: UTF-8');
echo '<?xml version="1.0" encoding="UTF-8" ?>';

$option = $_GET['vote'];

$filename = "data/vote.xml";
$xmlDoc = new DOMDocument();
$xmlDoc->load($filename);


$navigators = $xmlDoc->getElementsByTagName('navigateur');
foreach ($navigators as $navigator) {
  $name = $navigator->getAttribute('name');
  if (!strcmp($name,$option)) {
    $navigator->nodeValue ++;
    break;
  }
}

$xmlDoc->saveHTMLFile($filename);
echo $xmlDoc->saveHTML();
?>
```

permet au client de savoir que la réponse est du XML

une façon de parser du XML en PhP

que l'on peut ensuite scripter DOM

Exemple avec XML

Dans poll.js: envoi via Ajax du vote

```
function getVote() {

    // this l'input sur lequel un click a ete detecte
    var cle = this.getAttribute('value');

    // modification du noeud ou est affiche le vote
    document.getElementById("vote").innerHTML = cle;

    // communication ajax
    xhr = createXHR();

    var url="poll.php" + "?vote=" + cle + "&nocache="+Math.random();

    xhr.onreadystatechange=stateChanged;

    xhr.open("GET",url,true);
    xhr.send();
}
```

Exemple avec XML

Dans poll.js: traitement une fois le XML reçu (par le client)

```
function stateChanged() {
  if ((xhr.readyState==4) && (xhr.status == 200))  {

    // a) recuperation de la reponse sous forme XML
    var xml = xhr.responseXML;
    var navigators = xml.getElementsByTagName('navigateur');

    // b) somme des votes
    var total = 0;
    for (var i=0; i< navigators.length; i++)
      total +=  parseInt(navigators[i].firstChild.nodeValue,10);

    // c) creation d'une table avec tous les votes
    var str = "<table>\n<tr cols=\"3\"><th>" + total + " reponses</th></tr>\n" ;
    …

    // d) modification du champ ou est affiche le sondage
    document.getElementById("poll").innerHTML= str;

  }
}
```

déjà passé et scriptable DOM

JQuery & Ajax

$.load()

.load(url [, data] [, complete]) *Returns: jQuery*

Description: *Load data from the server and place the returned HTML into the matched elements.*

.load(url [, data] [, complete]) version added: 1.0

url
Type: String
A string containing the URL to which the request is sent.

data
Type: PlainObject or String
A plain object or string that is sent to the server with the request.

complete
Type: Function(String responseText, String textStatus, jqXHR jqXHR)
A callback function that is executed when the request completes.

```
$( "#result" ).load( "ajax/test.html", function() {
  alert( "Load was performed." );
});
```

note: injection directe dans la DOM

load: exemple

complete_load.html

idée:
lancer des requêtes Ajax depuis `complete_load` pour injecter le contenu HTML correspondant au lien cliqué

today.html

yesterday.html

lastweek.html

load: exemple

load: lastweek.html

```
<body>
<div class="wrapper">
     <div class="header">
          <p class="logo">JavaScript <i>&</i> jQuery <i class="mm">The<br>Missing<br>Manual</i></p>
     </div>
     <div class="content">
          <div class="main">
          <div id="newsItem">
               <h2 class="shadowLine">Last week&#8217;s News</h2>
               <h3>Ad minim veniam</h3>
               <p>Quis nostrud exercitation ut labore et dolore magna aliqua. Sed do eiusmod tempor incididunt cupidatat non
proident, duis aute irure dolor. Consectetur adipisicing elit, velit esse cillum dolore ut aliquip ex ea commodo consequat.
Ullamco laboris nisi ut enim ad minim veniam. Ut enim ad minim veniam, lorem ipsum dolor sit amet, ut labore et dolore magna
aliqua.</p>
               <h3> Duis aute irure dolor</h3>
               <p> ut labore et dolore magna aliqua. Sed do eiusmod tempor incididunt cupidatat non proident, duis aute irure
dolor. Consectetur adipisicing elit, velit esse cillum dolore ut aliquip ex ea commodo consequat. Ullamco laboris nisi ut eni
minim veniam. Ut enim ad minim veniam, lorem ipsum dolor sit amet, ut labore et dolore magna aliqua. ut labore et dolore magn
aliqua. Sed do eiusmod tempor incididunt cupidatat non proident, duis aute irure dolor. Consectetur adipisicing elit, velit e
cillum dolore ut aliquip ex ea commodo consequat. Ullamco laboris nisi ut enim ad minim veniam. Ut enim ad minim veniam, lore
ipsum dolor sit amet, ut labore et dolore magna aliqua.</p>
               <p> Labore et dolore magna aliqua. Sed do eiusmod tempor incididunt cupidatat non proident, duis aute irure dol
Consectetur adipisicing elit, velit esse cillum dolore ut aliquip ex ea commodo consequat. Ullamco laboris nisi ut enim ad mi
veniam. Ut enim ad minim veniam, lorem ipsum dolor sit amet, ut labore et dolore magna aliqua.</p>
               </div>
          </div>
     </div>
     <div class="footer">
          <p>JavaScript &amp;amp; jQuery: The Missing Manual, by <a href="http://sawmac.com/">David McFarland</a>. Published by <a
href="http://oreilly.com/">O'Reilly Media, Inc</a>.</p>
     </div>
</div>
</body>
```

load: complete_load.html

```
<script>
$(document).ready(function() {
  $('#newslinks a').click(function() {
     var url=$(this).attr('href');
     $('#headlines').load(url + ' #newsItem');
     return false;
  }); //end click

}); // end ready
</script>
</head>
<body>
<div class="wrapper">
    <div class="header">
        <p class="logo">JavaScript <i>&</i> jQuery <i class="mm">The<br>Missing<br>Manual</i></p>
    </div>
    <div class="content">
        <div class="main">
            <h1>News Headlines</h1>
            <ul id="newslinks">
               <li><a href="today.html">Today&#8217;s News</a></li>
               <li><a href="yesterday.html">Yesterday&#8217;s News</a></li>
               <li><a href="lastweek.html">Last Week&#8217;s News</a></li>
            </ul>
        <div id="headlines"></div>
        </div>
    </div>
    <div class="footer">
        <p>JavaScript &amp; jQuery: The Missing Manual, by <a href="http://sawmac.com/">David McFarland</
a>.
            Published by <a href="http://oreilly.com/">O'Reilly Media, Inc</a>.
         </p>
    </div>
</div>
```

Hijack

`.load` reconnait la syntaxe XPath
et permet de spécifier une sélection de la ressource récupérée

Amélioration progressive

$.get()

jQuery.get(url [, data] [, success] [, dataType]) *Returns: jqXHR*

Description: *Load data from the server using a HTTP GET request.*

jQuery.get(url [, data] [, success] [, dataType]) version added: 1.0

url
Type: String
A string containing the URL to which the request is sent.

data
Type: PlainObject or String
A plain object or string that is sent to the server with the request.

success
Type: Function(PlainObject data, String textStatus, jqXHR jqXHR)
A callback function that is executed if the request succeeds. Required if `dataType` is provided, but you can use `null` or `jQuery.noop` as a placeholder.

dataType
Type: String
The type of data expected from the server. Default: Intelligent Guess (xml, json, script, text, html).

jQuery.get([settings]) version added: 1.12/2.2

complete_login.html

```
$(document).ready(function() {
    $('#login').submit(function() {
            var formData = $(this).serialize();

            $.get('login.php',formData, processData).error('ouch');

            function processData(data) {
                if (data=='pass') {
                    $('#content').html('<p>You have successfully logged in!</p>');
                } else {
                    if (! $('#fail').length) {
                        $('#formwrapper').prepend('<p id="fail">Incorrect login
information. Please try again</p>');
                    }
                }
            } // end processData

        return false;
    }); // end submit

}); // end ready
```

login.php

```php
<?php if (isset($_GET['source'])) die(highlight_file(__FILE__, 1)); ?>
<?php

$password="secret";
$username="007";

$user=isset($_GET['username']) ? $_GET['username'] : $_POST['username'];
$pass=isset($_GET['password']) ? $_GET['password'] : $_POST['password'];

if ($user==$username && $pass==$password) {
  echo 'pass';
} else {
  echo 'fail';
}

?>
```

$.post()

jQuery.post(url [, data] [, success] [, dataType]) *Returns: jqXHR*

Description: *Load data from the server using a HTTP POST request.*

jQuery.post(url [, data] [, success] [, dataType]) version added: 1.0

url
Type: String
A string containing the URL to which the request is sent.

data
Type: PlainObject or String
A plain object or string that is sent to the server with the request.

success
Type: Function(PlainObject data, String textStatus, jqXHR jqXHR)
A callback function that is executed if the request succeeds. Required if `dataType` is provided, but can be `null` in that case.

dataType
Type: String
The type of data expected from the server. Default: Intelligent Guess (xml, json, script, text, html).

```
$.post( "test.php", { name: "John", time: "2pm" } );
$.post( "test.php", $( "#testform" ).serialize() );
```

$.ajax()

jQuery.ajax(url [, settings]) *Returns: jqXHR*

Description: *Perform an asynchronous HTTP (Ajax) request.*

jQuery.ajax(url [, settings]) version added: 1.5

url
Type: String
A string containing the URL to which the request is sent.

settings
Type: PlainObject
A set of key/value pairs that configure the Ajax request. All settings are optional. A default can be set for any option with $.ajaxSetup(). See jQuery.ajax(settings) below for a complete list of all settings.

jQuery.ajax([settings]) version added: 1.0

settings
Type: PlainObject
A set of key/value pairs that configure the Ajax request. All settings are optional. A default can be set for any option with $.ajaxSetup().

accepts (default: `depends on dataType`)
Type: PlainObject
A set of key/value pairs that map a given `dataType` to its MIME type, which gets sent in the `Accept` request header. This header tells the server what kind of response it will accept in return. For example, the following defines a custom type `mycustomtype` to be sent with the request:

$.ajax()

```
$.ajax({
  method: "POST",
  url: "some.php",
  data: { name: "John", location: "Boston" }
})
  .done(function( msg ) {
    alert( "Data Saved: " + msg );
  });
```

```
var menuId = $( "ul.nav" ).first().attr( "id" );
var request = $.ajax({
  url: "script.php",
  method: "POST",
  data: { id : menuId },
  dataType: "html"
});

request.done(function( msg ) {
  $( "#log" ).html( msg );
});

request.fail(function( jqXHR, textStatus ) {
  alert( "Request failed: " + textStatus );
});
```

exemple

```
$(document).ready(
   function() {

   $("input").click(

    function(event) {  // eq de la fonction getVote
      var vote = $(this).val();

      $.ajax({
          type: "GET",  // using $.get would be even easier
          url: "poll.php?vote="+vote,
          cache:false,
          dataType: "xml",
          success: function(xml) {

           var str = "<div><ul>";
           $(xml).find("navigateur").each( function() {
             var nb = $(this).text();
             str += ("<li>" + $(this).attr('name') + " : " + nb + "</li>");
           }); // pour chaque item navigateur
           $("#vote").text(vote);
           $("#poll").empty().append(str + "</ul></div>");

          } // sucess
    }); // ajax

  } // function onclick
); // chaque input - event click
}); // lorsque la page est prete
```

AJaX & JSON

ajax-json.php

```php
<?php
if (isset($_GET['source'])) die(highlight_file(__FILE__, 1));

/*
 * code: ajax-json.php
 * retourbe: un element json
 */

$person = array('nom' => 'Tremblay',
                'prenom' => 'Jean',
                'info' => array('age' => 40,
                                'sexe' => 'masculin'
                               )
               );

echo json_encode($person); // life is beautiful
?>
```

AJaX & JSON

```
function doIt() {

    var xhr = createXHR();

    xhr.onreadystatechange= function () {
      if ((xhr.readyState==4) && (xhr.status == 200))  {
       var o = JSON.parse(xhr.responseText);
        o.info.toJSON = function() {
           return "age " + this['age'] + " -- sexe: " + this['sexe'];
        }

        document.getElementById("tofill").innerHTML =
       o.nom + " " + o.prenom + "<br />" + JSON.stringify(o.info);
   }
    };

    xhr.open("GET","ajax-json.php",true);
    xhr.send(null);
};

window.addEventListener("load",doIt, false);
```

AJaX & JSON & JQuery

```
$(document).ready(function() {

    $.getJSON("ajax-json.php",null,function(o) {

      o.info.toJSON = function() {
         return "age " + this['age'] + " -- sexe: " + this['sexe'];
      };

      $('#tofill').html(o.nom + " " + o.prenom + "<br />" + JSON.stringify(o.info));

    }); // .getJSON

}); // ready
```