Ajax – XHR2

XMLHttpRequest

- Objeto XMLHttpRequest provee las funciones necesarias para trabajar con AJAX
  - Encargado de la comunicación con el servidor
  - Se trabaja con sus propiedades antes de enviar solicitudes al servidor
  - Cuando responde el servidor, invoca las funciones que corresponden
- Permite modificar la página sin recargarla por completo

XMLHttpRequest

- Limitantes
  - Las solicitudes son solo de HTML, XML y texto
  - Las variables y valores se deben codificar en la URL
  - Sujeto a “same-origin policy”
  - No se pueden subir archivos directamente
    - Se simulaba con iframes o un plugin externo
- Se desarrolla una nueva versión de XMLHttpRequest: XHR2

XHR2

- Incluye
  - Asignar timeouts a request
  - Mejor manejo de datos con “FormData objects”
  - Transferencia de datos binarios
  - Monitoreo del progreso de transferencia de datos
  - Permite “cross-origin request” de forma segura
  - Permite sobre-escribir el “media type” y “encoding” de las respuestas

Timeout a request

```
function makeRequest() {
  var url = 'data.json';
  var onLoadHandler = function(event){
     // Parse the JSON and build a list.
  }
  var onTimeOutHandler = function(event){
    var content = document.getElementById('content'),
      p = document.createElement('p'),
      msg = document.createTextNode('Just a little bit longer!');
      p.appendChild(msg);
      content.appendChild(p);

      // Restarts the request.
      event.target.open('GET',url);

      // Optionally, set a longer timeout to override the original.
      event.target.timeout = 6000;
      event.target.send();
  }
  var xhr = new XMLHttpRequest();
  xhr.open('GET',url);
  xhr.timeout = 3000;
  xhr.onload = onLoadHandler;
  xhr.ontimeout = onTimeOutHandler;
  xhr.send();
}

window.addEventListener('DOMContentLoaded', makeRequest, false);
```

XHR2

- Solicitando datos desde otro dominio:

```
var xhr = new XMLHttpRequest();
var onLoadHandler = function(event) {
  /* do something with the response */
}
xhr.open('GET','http://other.server/and/path/to/script');
xhr.onload = onLoadHandler;
xhr.send();
```

  - La diferencia es que el que recibe la solicitud permite el acceso a quien lo solicita
  - Envía el header Access-Control-Allow-Origin en la respuesta

XHR2

- FormData Object

```
var xhr = new XMLHttpRequest();
var dataToSend = new FormData(); // create a new FormData object

xhr.open('POST','/processing_script');

dataToSend.append('name','Joseph Q. Public'); // add data to the object
dataToSend.append('age','52');
dataToSend.append('hobby','knitting');

xhr.send(dataToSend); // send the object
```

- Además permite el envío de datos binarios

XHR2

- Monitoreo de progreso de transferencia de datos

```
var onProgressHandler = function(event) {
  if(event.lengthComputable) {
    var howmuch = (event.loaded / event.total) * 100;
        document.querySelector('progress').value = Math.ceil(howmuch);
  } else {
    console.log("Can't determine the size of the file.");
  }
}

var onLoadHandler = function() {
  displayLoadedMessage();
}

var onErrorHandler = function() {
  displayErrorMesssage();
}

xhr.upload.addEventListener('progress', onProgressHandler, false);
xhr.upload.addEventListener('load', onLoadHandler, false);
xhr.upload.addEventListener('error', onErrorHandler, false);
```

XHR2

- Permite cambiar el Mime-type de la respuesta:

```
var xhr = new XMLHttpRequest();
xhr.open('GET','data.xml');
xhr.overrideMimeType('application/xml');
xhr.send();
xhr.addEventListener('load', function(event) {
  console.log( event.target.responseXML );
}, false);
```

- Se asegura de evitar problemas de mime-type

XHR2

- Obteniendo datos binarios

```
var xhr = new XMLHttpRequest();
xhr.open('GET', '/path/to/image.png', true);
xhr.responseType = 'blob';

xhr.onload = function(e) {
  if (this.status == 200) {
    // Note: .response instead of .responseText
    var blob = new Blob([this.response], {type: 'image/png'});
    ...
  }
};
xhr.send();
```

XHR2

- Datos binarios
  - La propiedad “responseType” sirve para indicar que tipo de datos se necesita
  - Puede tomar valores: text, arraybuffer, blob o document
- Los datos de la respuesta, se obtienen desde “response”
  - Pueden ser: DOMString, ArrayBuffer, Blob, Document

XHR2

```
window.URL = window.URL || window.webkitURL;  // Take care of vendor prefixes.

var xhr = new XMLHttpRequest();
xhr.open('GET', '/path/to/image.png', 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); // Clean up after yourself.
    };
    img.src = window.URL.createObjectURL(blob);
    document.body.appendChild(img);
    ...
  }
};
xhr.send();
```

XHR2

- Enviar Datos: Se usa FormData

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

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

  xhr.send(formData);
}
```

XHR2

- Subir archivos:

```
function uploadFiles(url, files) {
  var formData = new FormData();

  for (var i = 0, file; file = files[i]; ++i) {
    formData.append(file.name, file);
  }

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

  xhr.send(formData);  // multipart/form-data
}

document.querySelector('input[type="file"]').addEventListener('change', function(e) {
  uploadFiles('/server', this.files);
}, false);
```