UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Programador Web Avanzado

Clase N° 2: Ajax

Profesor: Ing. Leandro Rodolfo Gil Carrano
Email: leangilutn@gmail.com

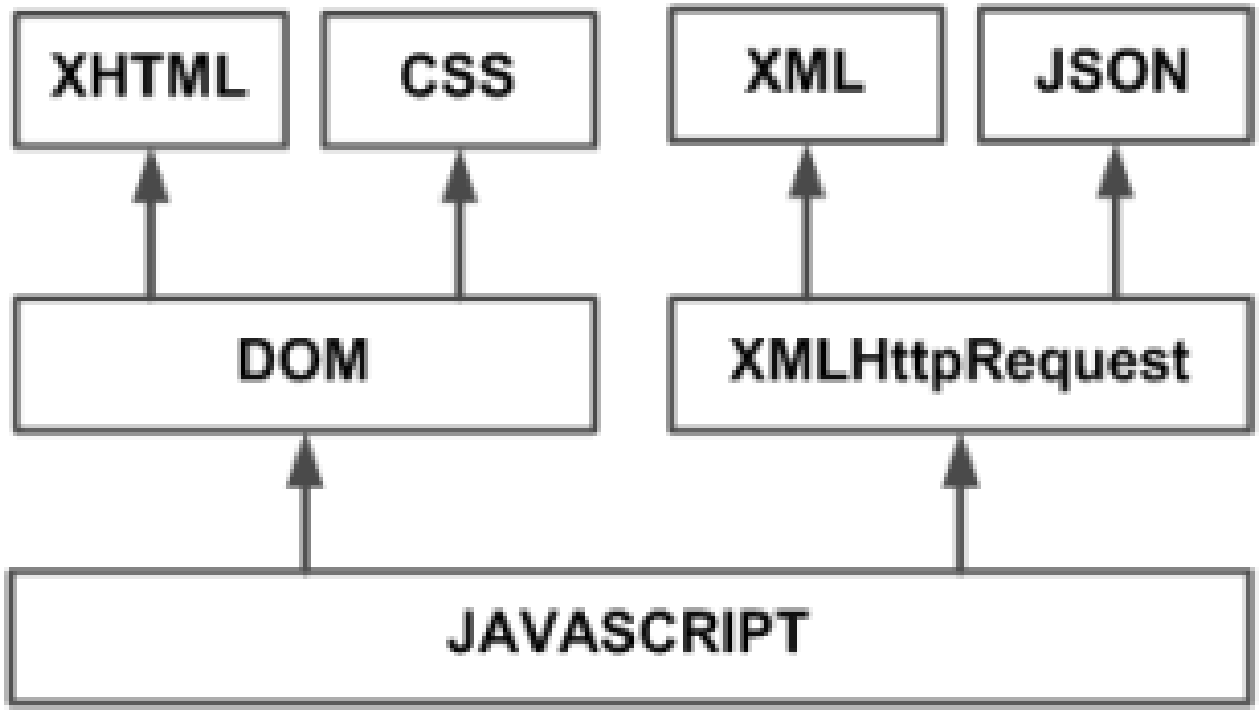
Ajax - Definición

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o **RIA** (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax - Tecnologías

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

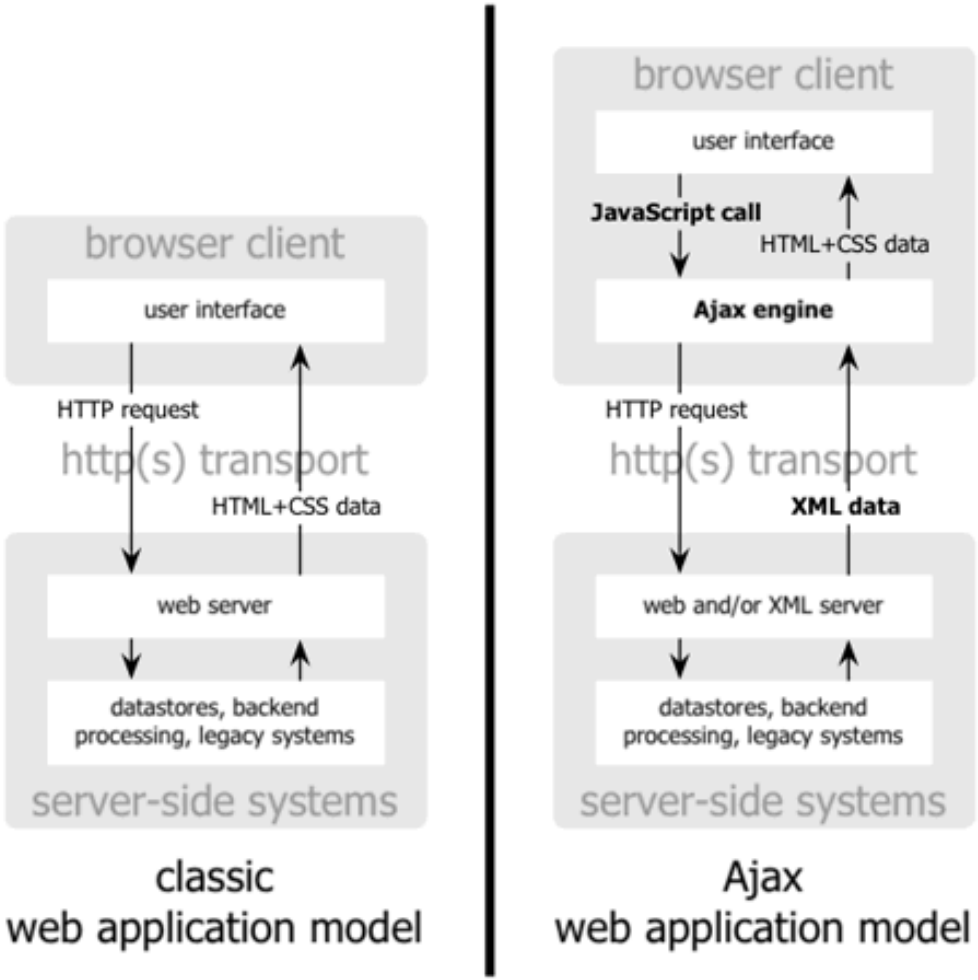
Ajax - Tecnologías



Ajax - Esquema

En el siguiente esquema, la imagen de la izquierda muestra el modelo tradicional de las aplicaciones web. La imagen de la derecha muestra el nuevo modelo propuesto por AJAX:

Ajax - Esquema



UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Javascript

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ajax – Ejemplo de aplicación

Ver **ejemplos/ejemplo_1.php**

```
<script type="text/javascript" language="javascript">

 var READY_STATE_UNINITIALIZED=0;
 var READY_STATE_LOADING=1;
 var READY_STATE_LOADED=2;
 var READY_STATE_INTERACTIVE=3;
 var READY_STATE_COMPLETE=4;
 var peticion_http;
 function muestraContenido() {
   if(peticion_http.readyState == READY_STATE_COMPLETE) {
     if(peticion_http.status == 200) {
       alert(peticion_http.responseText);
     }
   }
 }

 function descargaArchivo() {
     peticion_http = new XMLHttpRequest();
     peticion_http.onreadystatechange = muestraContenido;
     peticion_http.open("GET", "http://localhost/ajax/holamundo.txt", true);
     peticion_http.send(null);
 }

 window.onload = descargaArchivo;

 </script>
```

Ajax – Ejemplo 1

XMLHttpRquest: es el objeto clave que permite realizar comunicaciones con el servidor en segundo plano, sin necesidad de recargar las páginas.

```
function inicializa_xhr() {
    return new XMLHttpRequest();
}
```

Ajax – Ejemplo 1

Onreadystatechange: función que se encarga de procesar la respuesta del servidor. La propiedad onreadystatechange del objeto XMLHttpRequest permite indicar esta función directamente incluyendo su código mediante una función anónima o indicando una referencia a una función independiente.

```
peticion_http.onreadystatechange = funcion;

function muestraContenido() {
  if(peticion_http.readyState == READY_STATE_COMPLETE) {
    if(peticion_http.status == 200) {
      alert(peticion_http.responseText);
    }
  }
}
```

Ajax – Ejemplo 1

La petición HTTP se crea mediante el método **open()**, en el que se incluye el tipo de petición (GET), la URL solicitada (http://localhost/ajax/holamundo.txt) y un tercer parámetro que vale true.
Una vez creada la petición HTTP, se envía al servidor mediante el método **send().** Este método incluye un parámetro que en el ejemplo anterior vale null.

```
peticion_http.open(metodo, url, true);
peticion_http.send(null);
```

UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

XMLHttpRequest

XMLHttpRequest– Propiedades

- **readyState:** Valor numérico (entero) que almacena el estado de la petición
- **responseText:** El contenido de la respuesta del servidor en forma de cadena de texto
- **responseXML:** El contenido de la respuesta del servidor en formato XML. El objeto devuelto se puede procesar como un objeto DOM
- **status:** El código de estado HTTP devuelto por el servidor (200 para una respuesta correcta, 404 para "No encontrado", 500 para un error de servidor, etc.)
- **statusText** El código de estado HTTP devuelto por el servidor en forma de cadena de texto: "OK", "Not Found", "Internal Server Error", etc.

XMLHttpRequest– readyState

- **0:** No inicializado (objeto creado, pero no se ha invocado el método open)
- **1:** Cargando (objeto creado, pero no se ha invocado el método send)
- **2:** Cargado (se ha invocado el método send, pero el servidor aún no ha respondido)
- **3:** Interactivo (se han recibido algunos datos, aunque no se puede emplear la propiedad responseText)
- **4:** Completo (se han recibido todos los datos de la respuesta del servidor)

XMLHttpRequest– metodos

- **abort():** Detiene la petición actual
- **getAllResponseHeaders():** Devuelve una cadena de texto con todas las cabeceras de la respuesta del servidor
- **getResponseHeader("cabecera"):** Devuelve una cadena de texto con el contenido de la cabecera solicitada
- **onreadystatechange:** Responsable de manejar los eventos que se producen. Se invoca cada vez que se produce un cambio en el estado de la petición HTTP. Normalmente es una referencia a una función JavaScript

XMLHttpRequest– metodos

• **open("metodo", "url“, asincrono):** Establece los parámetros de la petición que se realiza al servidor. Los parámetros necesarios son el método HTTP empleado y la URL destino (puede indicarse de forma absoluta o relativa)
Por defecto, las peticiones realizadas son asíncronas. Si se indica un valor false al tercer parámetro, la petición se realiza de forma síncrona, esto es, se detiene la ejecución de la aplicación hasta que se recibe de forma completa la respuesta del servidor.
• **send(contenido):** Realiza la petición HTTP al servidor
• **setRequestHeader("cabecera", "valor"):** Permite establecer cabeceras personalizadas en la petición HTTP. Se debe invocar el método open() antes que setRequestHeader()

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ejercicio 1

Realizar el siguiente desarrollo:

- Un html que contenga un botón y un div vacio
- Al hacer click en el botón se deberá ejecutar una petición ajax al archivo **ejercicio_1.txt (Colocar el contenido que se desee)**
- El contenido devuelto por el response del ajax debe ser mostrado en el div vacio

UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

jQuery

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

jQuery– $.ajax

```
<script type="text/javascript" language="javascript">

    $.ajax({
   url: 'http://localhost/ajax/holamundo.txt',
   type: 'GET',
   async: true,
   data: 'parametro1=valor1&parametro2=valor2',
   success: function(data){
     console.log(data);
   },
   error: function(error){
     console.log(data);
   }
});

</script>
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

jQuery– $.ajax, opciones

Opción	Descripción
`async`	Indica si la petición es asíncrona. Su valor por defecto es `true`, el habitual para las peticiones AJAX
`beforeSend`	Permite indicar una función que modifique el objeto `XMLHttpRequest` antes de realizar la petición. El propio objeto `XMLHttpRequest` se pasa como único argumento de la función
`complete`	Permite establecer la función que se ejecuta cuando una petición se ha completado (y después de ejecutar, si se han establecido, las funciones de `success` o `error`). La función recibe el objeto `XMLHttpRequest` como primer parámetro y el resultado de la petición como segundo argumento
`contentType`	Indica el valor de la cabecera `Content-Type` utilizada para realizar la petición. Su valor por defecto es `application/x-www-form-urlencoded`

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

jQuery– $.ajax, opciones

`data`	Información que se incluye en la petición. Se utiliza para enviar parámetros al servidor. Si es una cadena de texto, se envía tal cual, por lo que su formato debería ser `parametro1=valor1&parametro2=valor2`. También se puede indicar un array asociativo de pares clave/valor que se convierten automáticamente en una cadena tipo *query string*
`dataType`	El tipo de dato que se espera como respuesta. Si no se indica ningún valor, jQuery lo deduce a partir de las cabeceras de la respuesta. Los posibles valores son: `xml` (se devuelve un documento XML correspondiente al valor `responseXML`), `html` (devuelve directamente la respuesta del servidor mediante el valor `responseText`), `script` (se evalúa la respuesta como si fuera JavaScript y se devuelve el resultado) y `json` (se evalúa la respuesta como si fuera JSON y se devuelve el objeto JavaScript generado)
`error`	Indica la función que se ejecuta cuando se produce un error durante la petición. Esta función recibe el objeto `XMLHttpRequest` como primer parámetro, una cadena de texto indicando el error como segundo parámetro y un objeto con la excepción producida como tercer parámetro

jQuery– $.ajax, opciones

`ifModified`	Permite considerar como correcta la petición solamente si la respuesta recibida es diferente de la anterior respuesta. Por defecto su valor es `false`
`processData`	Indica si se transforman los datos de la opción `data` para convertirlos en una cadena de texto. Si se indica un valor de `false`, no se realiza esta transformación automática
`success`	Permite establecer la función que se ejecuta cuando una petición se ha completado de forma correcta. La función recibe como primer parámetro los datos recibidos del servidor, previamente formateados según se especifique en la opción `dataType`
`timeout`	Indica el tiempo máximo, en milisegundos, que la petición espera la respuesta del servidor antes de anular la petición
`type`	El tipo de petición que se realiza. Su valor por defecto es `GET`, aunque también se puede utilizar el método `POST`
`url`	La URL del servidor a la que se realiza la petición

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

jQuery– $.get()

```
<script type="text/javascript" language="javascript">

    $.get('http://localhost/ajax/holamundo.txt',
        { articulo: '34' },
        function(datos) {
          console.log(datos);
        }
    );

</script>
```

Url: Indica la url donde se hará el request ajax
Parámetros: Enviados a través de un json
Función de success: Función de callback indicada en caso de succsess

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

jQuery– $.getJSON()

```
<script type="text/javascript" language="javascript">

    $.get('http://localhost/ajax/holamundo.txt',
        { articulo: '34' },
        function(datos) {
          console.log(datos);
        }
    );

</script>
```

La respuesta esperada (**dataType**) es un JSON
Url: Indica la url donde se hará el request ajax
Parámetros: Enviados a través de un json
Función de success: Función de callback indicada en caso de succsess

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

jQuery– $.post()

```
<script type="text/javascript" language="javascript">

    $.post('http://localhost/ajax/holamundo.txt',
        { articulo: '34' },
        function(datos) {
          console.log(datos);
        }
    );

</script>
```

Url: Indica la url donde se hará el request ajax
Parámetros: Enviados a través de un json
Función de success: Función de callback indicada en caso de succsess

JSON

JSON (acrónimo de JavaScript Object Notation, «notación de objeto de JavaScript») es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente.

JSON

```
{
    "menu": {
        "id": "file",
        "value": "File",
        "popup": {
            "menuitem": [
                {
                    "value": "New", "onclick": "CreateNewDoc()"
                },{
                    "value": "Open", "onclick": "OpenDoc()"
                },{
                    "value": "Close", "onclick": "CloseDoc()"
                }
            ]
        }
    }
}
```

Ejercicio 2

Realizar el siguiente desarrollo:

- Ejecutar una petición ajax al archivo **ejercicio_2.json**, de acuerdo a los datos leídos :
  - Al modificarse el valor seleccionado se deberá mostrar un combo heredado de este
    - Si se selecciona “1” se deberá mostrar un combo con las siguientes opciones:
      - Valor 1.1
      - Valor 1.2
    - Si se selecciona “2” se deberá mostrar un combo con las siguientes opciones:
      - Valor 2.1
      - Valor 2.2

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ejercicio 3

Armar un mock up de datos en un archivo **productos.json**
En una peticion ajax leer ese mock up de datos y renderizar los resultados en pantalla.