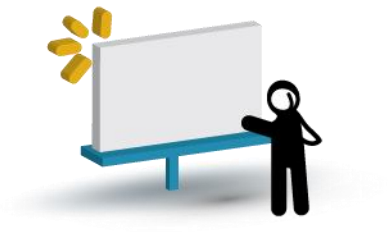
UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES
Centro de e-Learning

Proffessional backend developer

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Modulo 2: Javascript

Unidad 4: Ajax



Presentación:

En la presente unidad veremos que es ajax, como funciona y haremos ejemplos prácticos utilizando javascript .

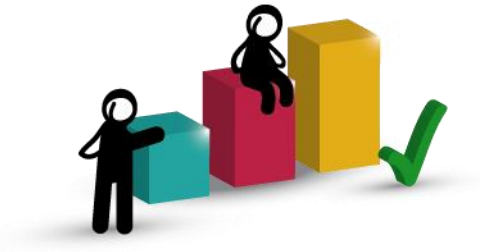
Objetivos:

Que los participantes*:

- Aprendan conceptualmente el funcionamiento de ajax
- Sepan aplicarlo utilizando javascript
- Conozcan como utilizarlo sintácticamente.

Bloques temáticos*:

- Conceptos básicos
- Ajax
- Ajax – Implementar en javascript



Consignas para el aprendizaje colaborativo

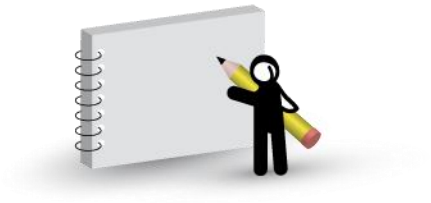
En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.



Tomen nota*

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

Conceptos básicos

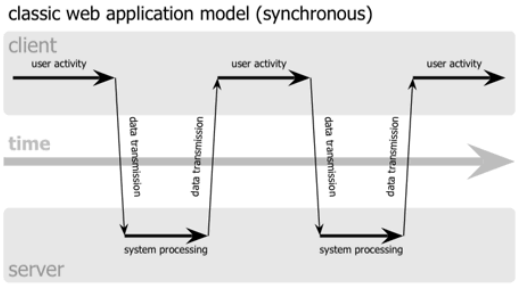
Asincrónico

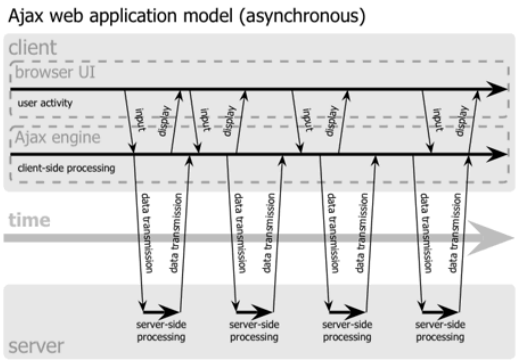
Asincronía hace referencia al suceso que no tiene lugar en total correspondencia temporal con otro suceso.

En educación a distancia (EaD), la asincronía se refiere a actividades que no requieren la conexión simultánea del facilitador y los participantes, sino que cada quien participa en su propio tiempo. Un ejemplo son los foros asíncronos, en los cuales todos participan en la conversación, pero en diferentes momentos.

Nosotros en este curso tenemos un tipo de educación asincrónica, en ¿qué momento esa educación se convierte en sincrónica?

El asincronismo en la programación significa que yo puedo realizar un request al servidor, continuar trabajando el lo que sigue del código y cuando el servidor me responda atender dicho response.





Request

Son las peticiones que le realizamos al servidor, cada vez que apretamos F5 en nuestro navegador web estamos haciendo un request.

Response

Es la respuesta a un determinado request dado por un servidor.

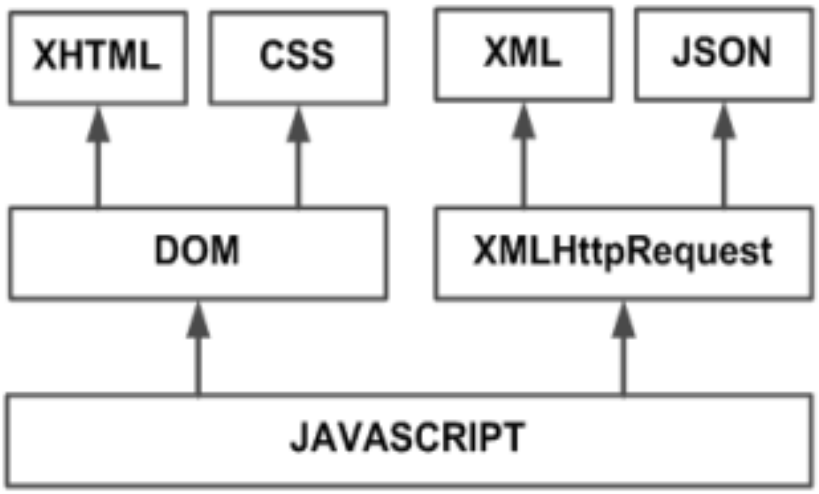
Ajax

Definición

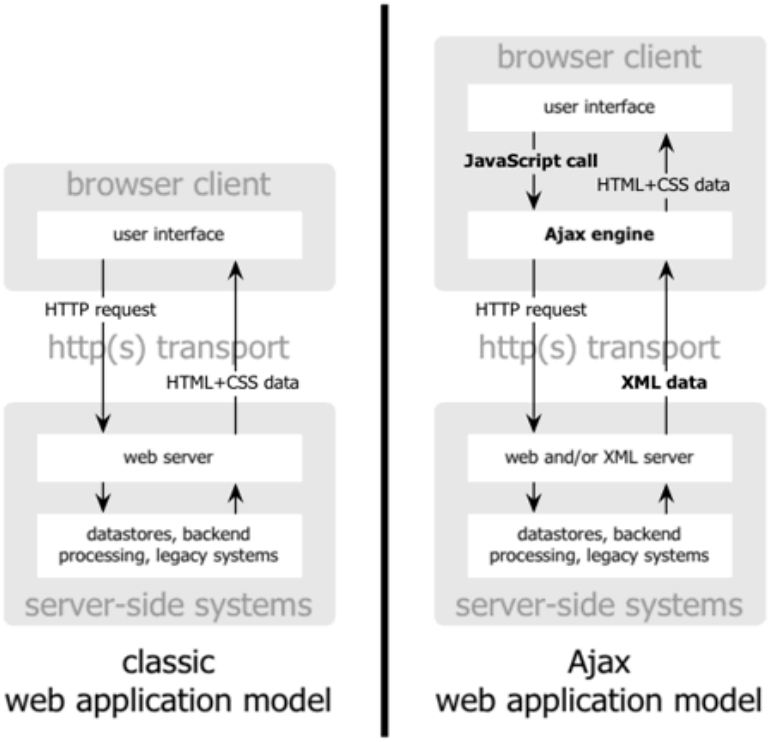
AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación 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.

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.



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 – Implementar en javascript

Ejemplo 1

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>
```

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

```
 peticion_http = new XMLHttpRequest();
```

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 = muestraContenido;

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

Como podemos ver **onreadystatechange** llama a la función muestra contenido, que evalúa el estado de la petición ajax y en caso de la que misma haya sido completada de manera exitosa hace un alert con el contenido captura en el response.

Open – Send

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);
```

XMLHttpRequest

- **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 – métodos

- **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

- **open("método", "url“, asíncrono):** 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()

Ejercicio propuesto

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_server.php**
- El contenido devuelto por el response del ajax debe ser mostrado en el div vacio

Bibliografía utilizada y sugerida

https://es.wikipedia.org/wiki/AJAX

https://developer.mozilla.org/es/docs/Web/Guide/AJAX

Lo que vimos:

En esta unidad hemos aprendido los conceptos teóricos y prácticos esenciales para poder trabajar con ajax. Hemos visto como aplicar el mismo en javascript nativo.

Lo que viene:

En la próxima unidad veremos la introducción a uno de las bibliotecas UI mas utilizadas … React Js