UNIDAD 6

Bibliotecas de JavaScript (Frameworks)

Biblioteca Prototype

Prototype

Dojo

Mootools

Historia de Ajax

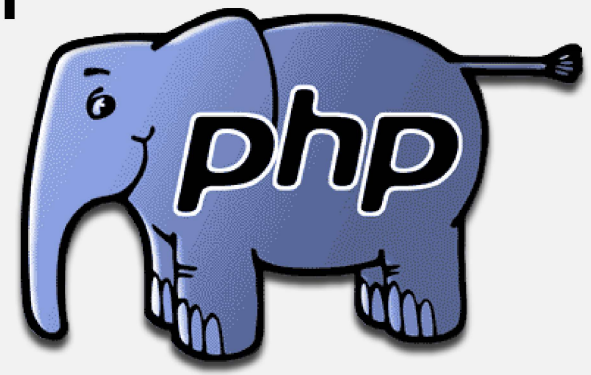
Conceptos básicos

Diferencias entre aplicaciones Web tradicionales y aplicaciones Web con Ajax

Objetos de Ajax

JAVA

Java es una de las plataformas con mayor cantidad de librerías para convertir JSON. Sólo en el sitio json.org hay más de 12 librerías distintas. Su uso es muy similar a los propuestos antes en PHP y .NET, por lo que no vale la pena ejemplificarlos de nuevo.



PROTOTYPE

Qué es?

Prototype es una librería Open Source creada para extender las funcionalidades de Ja- vaScript y reducir la tarea de codificación a los programadores. Tiene la gran ventaja de distribuirse en un solo archivo .js, es muy liviana y en los últimos tiempos se con- virtió en un estándar de facto en el mercado. Muchas otras hacen uso de Prototype, por lo que en numerosos casos se la considerará un requisito obligatorio en aplicaciones AJAX.

Instalación

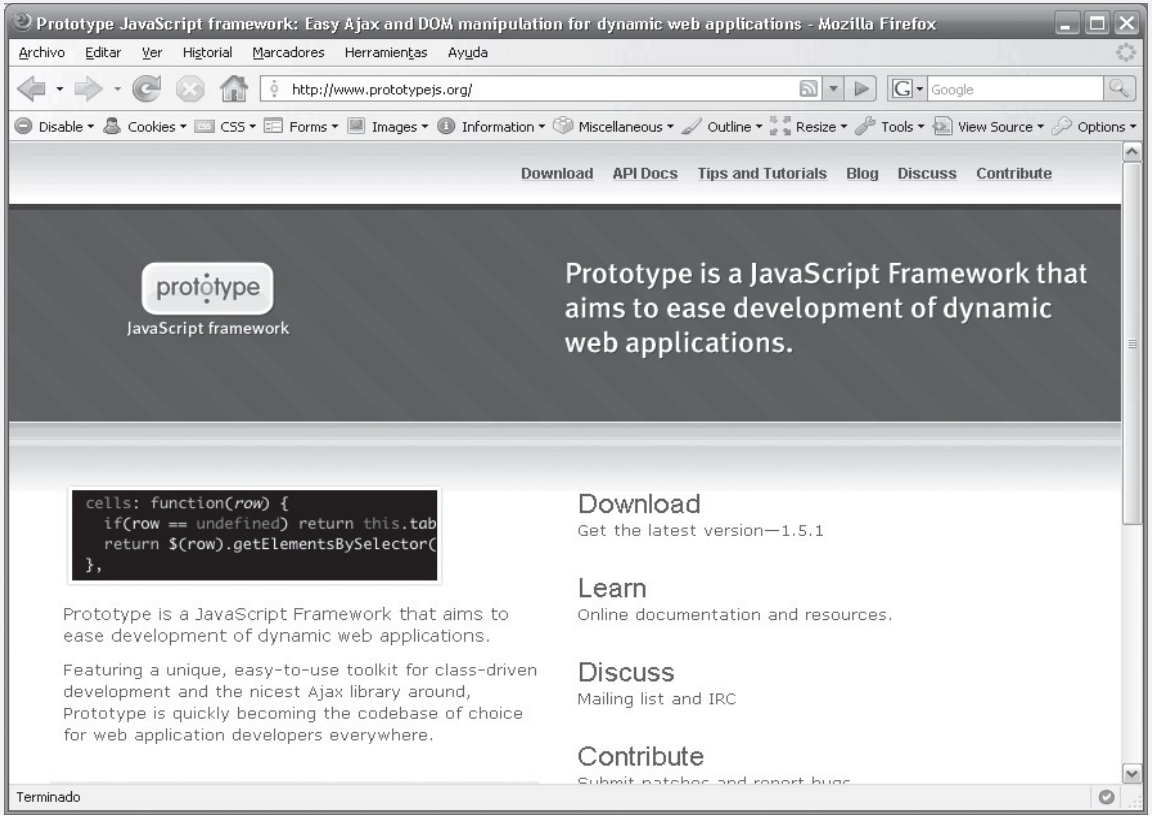
Su instalación es muy sencilla. Sólo basta con visitar el sitio web www.prototypejs.org y descargar la última versión, que no es más que un archivo prototype.js que se debe incorporar en nuestro proyecto y, luego, incluirlo en un tag script. En el sitio web se puede consultar toda la documentación de las API que incorpora la librería. A continuación, se detallarán los agregados más utilizados.

Utilitarios

- **$**

Dentro de la gama de utilitarios, hay muchos y muy útiles. El primero que se va a ana- lizar es el signo $ presente no se usaba para mucho, es posible que una variable (y, por lo tanto, también una funci comience con el signo $, incluso que su nombre sea sólo ese símbolo. Es por eso que Prototype ha cr llamada $ que reemplaza a una de las funciones más utilizadas en una aplicación AJAX: document.getE que con hacer:

$("divContenido")



Se estaría recibiendo un objeto que represente al elemento XHTML que tenga id div- Contenido. Qu
sencillo de leer, además de la obvia reducción de có- digo a escribir y a transferir desde el servidor a
forma, si se quiere cambiar el color de fondo de una celda que posee id, se podría hacer:

$("celda1").bgColor = "red";

o si se quiere definir la función onclick de un botón:

$("btnEnviar").onclick = function() { alert("Gracias por presio- narme") };

no obstante, la función $ no es sólo una mera traducción de document.getElement ById. La función n
dinámico de parámetros y, cuando se envía más de uno, lo que se obtiene es un vector cuyas posiciones
los elementos pedidos.

Por ejemplo:

$("btnEnviar", "btnCancelar", "btnSalir")

Este código devolvería un vector de 3 posiciones con tres objetos de tipo botón, con los ids solicitados

- **$$**

La función $$ recibe uno o más selectores de tipo CSS (de clase, de etiqueta o de id) y devuelve un v los elementos que cumplen con el o los selectores dados. Hay combinaciones completas que se pu encontrar con facili- dad y rapidez elementos HTML que cumplan con alguna condición.

Ejemplos:

Devuelve todos los divs

```
$$("div");
```

Devuelve todos los span de clase titulo

```
$$("span.titulo");
```

Devuelve todos los links que abran en ventana nueva

```
$$("a[target='_blank']");
```

- **$F**

Devuelve el valor (la propiedad value) del elemento HTML dado, por lo general una eti- queta de formulario (un textbox, o un select).

Var nombre = $F("txtNombre");

Var pais =$F("lstPaises");

- **$w**

Copiada del lenguaje Ruby, esta función recibe un string con palabras separadas por espacio y devuelve un vector con cada palabra.

Var arraPaises = $w(“Argentina Brasil Chile Uruguay”);

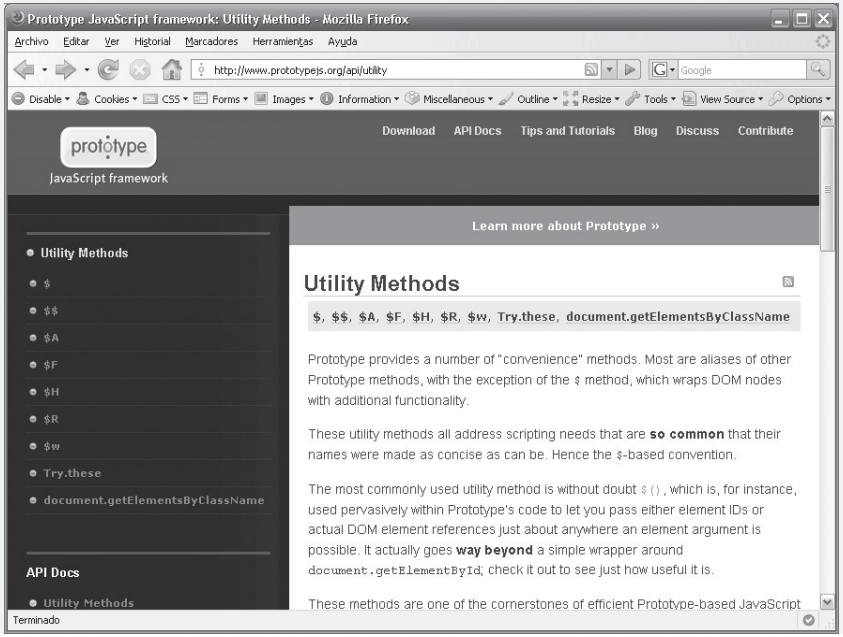
TRY.THESE

Esta función recibe una lista de funciones por parámetro y devuelve el resultado de la primera de ellas que no genere una excepción (error). Por ejemplo, se podría mejorar nuestra librería de AJAX cuando se instancia el objeto XMLHttpRequest evitando ifs:

```
Función obtnerXHR(){
Return Tr these(
Función(){ return new XMMLHttpRequest()},
Function (){return new ActiveXObject('Msxml2.XMLHTTP')},
Function(){return ne ActiveXObject('Microsoft.XMLHTTP')}
)||false;
}
```

OTROS

- Hay otros utilitarios menos usados, como $R, $A y $H. El primero permite generar un objeto ObjectRange, el segundo, convertir cualquier objeto enumerable a un Array y el último, cualquier objeto a un HashTable (array asociativo). Dejamos para revisar en la documentación oficial de Prototype su uso y ejemplos. $R permite obtener una secuencia de objetos de tipo e numerables, por ejemplo, para recibir una lista con los números 1 a 1000, se puede hacer $R(1, 1000) y las letras del alfabeto $R('a', 'z'). Si se quieren convertir esos objetos a Array, sólo resta aplicarle también $A.



AGREGADOS A OBJETOS

Cuando importamos prototype.js en nuestra página web, de manera instantánea muchos objetos adqu
características nuevas, como propiedades y métodos agregados por la librería.

- **Strings**

Todos los strings ahora poseen, entre otros, los métodos siguientes:

- blank Devuelve true si el string está vacío o sólo contiene espacios
- camelize Devuelve una versión en formato camello del string en cuestión. Por ejemplo: nom
  devuelve nombreCliente
- capitalize Pone la primera letra en mayúscula
- desherize Reemplaza todos los guiones bajos (_) por medios (-)
- empty Devuelve true si el string está vacío por completo (no tiene ni siquiera espacios)
- endsWith Recibe una cadena e indica si el string finaliza con ella. Por ejemplo, es útil para co
  extensiones de nombre de archivo
- escapeHTML Convierte caracteres especiales a HTML

- evalJSON Evalúa el string como un JSON y devuelve un objeto. Recibe un parámetro lógico o
  si queremos comprobar que no sea código malicioso
- evalScripts Ejecuta todos los tag SCRIPT que hubiera en el string
- startsWith Recibe una cadena e indica si el string comienza con ella
- strip Elimina todos los espacios antes y después del texto (equivalente a un Trim en otros lenguaje
- stripScripts En caso que el string fuera HTML, elimina todos los tags script que hubiera
- stripTags En caso que el string fuera HTML elimina todos los tags (aunque no el texto que tu
- times Recibe un parámetro numérico y devuelve una versión con el string repetido n veces.
- toArray Convierte el string a un array de caracteres.
- toJSON Convierte el string a un JSON compatible.
- toQueryParams En caso que el string tuviera un formato estilo QueryString (clave=valor&clave2=va
  un objeto con las propiedades clave y clave2.
- truncate Recibe una cantidad y un sufijo opcional. Trunca el string a la cantidad dada e incorp
  final. Por defecto el sufijo es "...". Permite mostrar textos largos en espacios reducidos.
- underscore Convierte un string en notación camello a separado por guiones bajos.
- unescapeHTML Devuelve una versión en texto normal de un string con caracteres en formato Esca

También aparece una nueva clase en JavaScript llamada Template, que permite definir cierta plantilla y un JSON, así como reemplazar cada propiedad del objeto en la plantilla en las ocurrencias que se indiquen.

```
// ésta es la plantilla

var plantilla = new Template('El curso de #{curso} es dictado por #{profesor}.');


// los datos a reemplazar var cursoAjax = {

curso: 'AJAX y JS Avanzado', profesor: 'Maximiliano R.
Firtman'

};

// aplica la plantilla
alert(plantilla.evaluate(cursoAjax));
```

ARRAYS

Todos los arrays reciben los métodos siguientes. Cuando el objeto que tenemos no es estrictamente un array, sino una colec
document.getElementsByTagName) podemos convertirlo a array en forma explícita para tener estos métodos utilizando $A

- **clear:** Limpia todo el array.
- **clone:** Devuelve una copia exacta del array.
- **compact:** Devuelve una versión comprimida del array, eliminando todas las posiciones nulas o indefinidas.
- **each:** Ejecuta una función recibida por parámetro en cada posición del array. Equivale a un for que recorra todo el ar
  función a cada elemento.
- **first:** Devuelve el primer elemento.
- **flatten:** En el caso de un vector multidimensional (p. ej., una matriz), lo convierte a una sola dimensión.
- **indexOf:** Devuelve la posición de la primera ocurrencia de un elemento recibido por parámetro, o -1 si no existe.
- **last:** Devuelve el último elemento.
- **reduce:** Si el array contiene una sola posición, devuelve el único elemento.
- **reverse:** Invierte la posición de los elementos.
- **toJSON:** Convierte el vector a un string estilo JSON.
- **without:** Recibe una lista de valores y devuelve el vector, pero sin los valores recibidos.

Si trabajamos con Prototype, debemos dejar de utilizar for in para recorrer los vectores. Sin entrar e
detalles técnicos, los métodos que Prototype tienen problemas con el for in. La solución es cambiar e
siguiente:

```
For (var i int vector)          {alert(vector[i]);              }
```

por el que sigue:

```
vector each (función(item){   alert(item);          });
```

También aparecen métodos nuevos que se pueden aplicar a todos los elementos que sean enumerabl
podemos ver en la documentación. Algunos de estos métodos son en realidad muy útiles.

ELEMENTOS DOM

Cuando se acceda a un elemento DOM por medio de las funciones propias de Prototype, como $ o $$, enton
accediendo a un elemento nodo extendido, que posee más propiedades y métodos. Más adelante en este capít
tema de DOM y se agregan estos elementos de Prototype.

- **Event**

Prototype incorpora a JavaScript un manejo de eventos algo superior al clásico que trae el estándar del lengua
se puede hacer uso del patrón de diseño Ob- server para agregar o eliminar un observador a un evento de un
Esto permite una forma más sencilla de agregar y eliminar funciones que escuchan a que cierto evento ocurra.

Para ello, hay una clase Event que posee los métodos observe y stopObserving.

Veamos un ejemplo:

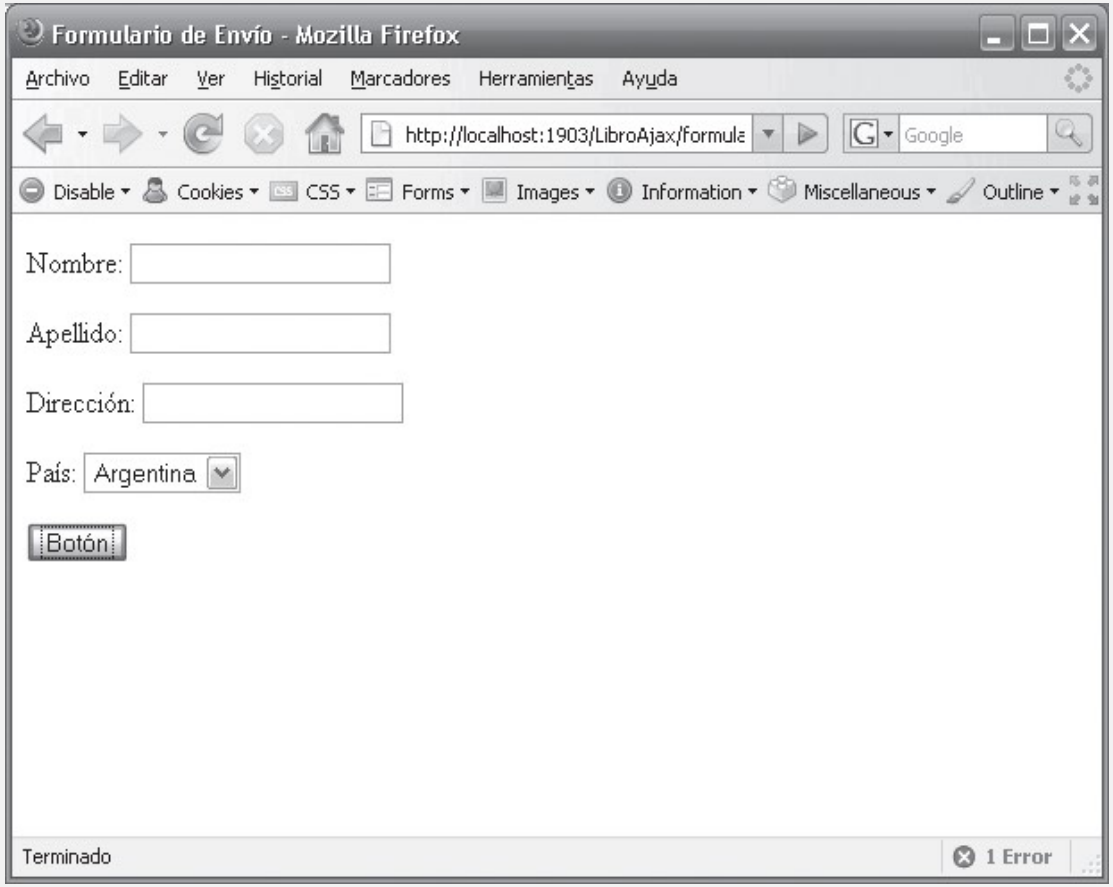
```
Event observe(window, `load`, function(){
Event observes(`btnEnviar`, `click`, mostrarMensaje); });
Función mostrarMensaje(){
alert("Gracias");
Event.stopObserving(`btnEnviar`, `click`, mostrarMensaje);    }
```

FORM

Si a un formulario HTML le incluimos un ID (p. ej., "form1") y lo accedemos a través de $("form1"), te
métodos siguientes:

- **disable:** Desactiva de manera funcional todos los elementos que están dentro del for
- **enable:** Vuelve a activar en forma funcional todos los elementos.
- **findFirstElement:** Devuelve el primer control que esté activado y no sea de tipo hidden.
- **focusFirstElement:** Pone el foco en el primer control del formulario.
- **getElements:** Devuelve una colección con todos los controles de ingreso del formulario.
- **getInputs:** Devuelve una colección con todos los controles INPUT. Es posible restringi queremos mediante los parámetros.
- **reset:** Limpia todo el formulario.
- **serialize:** Devuelve todos los controles en formato string URL para enviarlos en una vía GET o POST.
- **serializeElements:** Devuelve una serie de controles enviados por parámetro en formato string

Así, si se desea convertir un clásico formulario a un formulario AJAX, sólo se debe realizar la petición al
enviando los parámetros recibidos del método serialize. Además, se debe cambiar el botón de Enviar de
uno de tipo button, si no lo estaremos enviando al viejo estilo (fig. 3-9). Veamos un ejemplo:



DOJO TOOLKIT

Dojo es un framework muy completo para realizar aplicaciones ricas de Internet con JavaScript. Una de sus desventajas es que es uno de los más pesados a la hora de cargar código JavaScript, pero a su favor tiene su gran funcionalidad, que incorpora como un nuevo modelo de eventos y facilidades de escritura y utilitarios para las tareas más comunes en un desarrollo de este tipo. Su sitio oficial es dojotoolkit.org, donde se encontrarán ejemplos y la documentación oficial, además de la librería en realidad, Dojo Toolkit surgió como la unión de varios frameworks y librerías que se unieron bajo un mismo paraguas de trabajo.

- **Dijit**

Sobre Dojo hay una librería de widgets o controles ricos conocida como Dijit. Esta li- brería incluye cont
en categorías:

- Controles de formulario.
- Auto Completer.
- Inline Edit Box.
- TextBox (con validación y formato).
- Buttons (de varios tipos).
- Dropdown Calendar.
- Number Spinner.
- Resizeable Text Area.
- Select.
- Slider.

Controles de disposición

- Content Pane.
- Title Pane.
- Layout Container.
- Tab Container.
- Page Container.
- Split Container.
- Accordion Container.
- Dialog.

Controles de comando

- ToolBar.
- Menúes Popup.

Controles de asistencia al

- Progress Bar.
- Tooltip.

Controles avanzados

- Color Palette.
- Tree.
- Rich Text Editor.
- Grid.
- Toggler.

Todos los widgets soportan temas para combinar estilos y colores, y contienen una lista de atributos que
comparten en común y hacen más sencilla la tarea de programarlos.

MÓDULOS

Entre los módulos de funcionamiento que incorpora, se pueden mencionar (fig. 10-9):

- Charts: permite crear gráficos estadísticos.
- Offline: permite crear aplicaciones desconectadas junto a Google Gears.
- Storage: permite almacenar información en el cliente.
- Drag and Drop.
- Animación.
- Botón de Atrás.
- Internacionalización y múltiples idiomas.

INTRODUCCIÓN A MOOTOOLS

MooTools es un framework JavaScript muy potente que nos facilitará e] desarrollo de interfaces visuales, manipulación del DOM, manejo de AJAX, etc.

Su filosofía es similar a la de jQuery, vista en el capítulo anterior y nuevamente , al ser una librería escrit muy senci_lla de utilizar y no requiere ningún componente extraño adicional para funcionar, simplement Ja página en que la vamos a usar y la usamos. Así de sencillo.

MooTools aporta al desarrollador muchas ventajas. Veamos algunas:

- Es un framework modular y el desarrollador puede elegir qué componentes usar y cuáles no.
- Es ligero. El framework no pesa demasiado y su procesamiento no carga al navegador.
- MooTools está orientado a objetos y sigue los principios DRY (Don't Repeat Yourself. Repetirse supon
- Permite crear aplicaciones web muy dinámicas con un diseño único, atractivo e interactivo.
- Es un framework desarrollado por programadores para programadores. En la versión 1.3 se ha reduci variables globales y se ha mejorado el rendimiento global.
- MooTools es gratuito y de código abierto bajo licencia MIT, lo que significa que permite modificar y ad nuestras necesidades.

PRIMEROS PASOS

El primer paso consiste en descargar la librería de la web oficial de MooTools: http://mootools.ne/t

Descargamos la opción MooTools Core 1.3 with compatibility para trabajar con los ejemplos de este capítulo.

Para aprender a usar la librería, es mejor descargar el Core completo. Después, una vez controlado el tema, pod
descargar los componentes que realmente necesitamos para la web en producción, desde la zona de descargas

Al Terminar de seleccionar los componentes deberemos elegir la compresión deseada. Es recomendable dejar la
(sin comprensión) marcando la casilla de compatibilidad De esta forma la librería será compatible con la versió

Los archivos que descargamos para las versiones normal y comprimidas se Jlaman mootools-core-1.3-ful/-com
core-1.3-.ful/-compat-yc.js respectivamente. Para enlazarlos con nuestro documento es recomendable renombra
esta forma, las futuras actualizacionesd la librería sólo nos implicará cambiar el archivo fuente sin tocar el códig

Por lo tanto, el primer paso para emplear MooTools en un documento será enlazar desde el head del d
librería.

Cuando la página carga todo su contenido, excepto imágenes está en un estado llamado domready y cua
cargado las imágenes se llama load.

En este caso usaremos el domready para poder ejecutar nuestros ejemplos, ya que mucho antes de que
de elementos, la página puede estar lista para ejecutar acciones Javascript que alteren el DOM.

- **Plugins**
- Una de las grandes ventajas de MooTools es la posibilidad de utilizar plugins de terceros. Podríamos d
  es un módulo o complemento que añade algún tipo de función a un sistema. Por lo tanto, estos eleme
  van a permitir potenciar aún más las posibilidades de MooTools.

HISTORIA AJAX

La historia de AJAX está íntimamente relacionada con un objeto de programación llamado *XMLHttpReq* este objeto se remonta al año 2000, con productos como Exchange 2000, Internet Explorer 5 y Outloo

Todo comenzó en 1998, cuando **Alex Hopmann** y su equipo se encontraban desarrollando la entonce de Exchange 2000. El punto débil del servidor de correo electrónico era su cliente vía web, llamado OV *Access*). Durante el desarrollo de OWA, se evaluaron dos opciones: un cliente formado sólo por páginas que se recargaban constantemente y un cliente realizado completamente con HTML dinámico o DHTM pudo ver las dos opciones y se decantó por la basada en DHTML. Sin embargo, para ser realmente útil a faltaba un componente esencial: «algo» que evitara tener que enviar continuamente los formularios con

Motivado por las posibilidades futuras de OWA, Alex creó en un solo fin de semana la primera versión denominó XMLHTTP. La primera demostración de las posibilidades de la nueva tecnología fue un éxito, más difícil: incluir esa tecnología en el navegador Internet Explorer. Si el navegador no incluía XMLHTTP el éxito del OWA se habría reducido enormemente. El mayor problema es que faltaban pocas semanas lanzara la última beta de Internet Explorer 5 previa a su lanzamiento final. Gracias a sus contactos en la consiguió que su tecnología se incluyera en la librería MSXML que incluye Internet Explorer.

El nombre del objeto (XMLHTTP) se eligió para tener una buena excusa que justificara su inclusión en Internet Explorer, ya que este objeto está mucho más relacionado con HTTP que con XML.

CONCEPTOS BÁSICOS

AJAX(por sus iniciales, JavaScript asíncrono y XML). AJAX no es un lenguaje de programación, es interp
utiliza una combinación de: Un objeto XMLHttpRequest integrado en el navegador (para solicitar datos
servidor web) JavaScript y HTML DOM (para visualizar o utilizar los datos)

Las aplicaciones AJAX pueden usar XML para transportar datos, pero es igualmente común transportar
texto plano o texto JSON. **AJAX permite que las páginas web se actualicen asincrónicamente**
intercambio de datos con un servidor web entre bastidores. Esto significa que es posible actualizar part
web, sin recargar toda la página.

La palabra reservada de AJAX para crear un objeto *XMLHttpRequest*. se puede utilizar para intercambiar
servidor entre bastidores. Esto significa que es posible actualizar partes de una página web, sin recargar

Sintaxis para crear un objeto XMLHttpRequest:

```
<!DOCTYPE html>
<html>
<body>

<h1>The XMLHttpRequest Object</h1>

<p id="demo">Let AJAX change this text.</p>

<button type="button" onclick="loadDoc()">Change Content</button>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

XMLHTTPREQUEST OBJECT METHODS (MÉTODOS DE OBJETO DE SOLICITUD)

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(method,url,async,user,psw)	Specifies the requestmethod: the request type GET or POSTurl: the file location async: true (asynchronous) or false (synchronous) user: optional user name psw: optional password
send()	Sends the request to the server Used for GET requests
send(string)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

PROPIEDADES DE LOS OBJETOS XMLHTTPREQUEST

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

ENVIAR UNA SOLICITUD A UN SERVIDOR

Method	Description
open(method, url, async)	Specifies the type of request method: the type of request: GET or POST url: the server (file) location async: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(string)	Sends the request to the server (used for POST)

GET Y POST

GET es más simple y rápido que POST, y puede ser utilizado en la mayoría de los casos. Sin embargo, u
peticiones POST cuando:Un archivo en caché no es una opción (actualizar un archivo o base de datos
Envío de una gran cantidad de datos al servidor (POST no tiene limitaciones de tamaño). Enviando la e
usuario (que puede contener caracteres desconocidos), POST es más robusto y seguro que GET.

Respuesta GET

```
<!DOCTYPE html>
<html>
<body>

<h1>The XMLHttpRequest Object</h1>

<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>


<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "demo_get.asp", true);
  xhttp.send();
}
</script>

</body>
</html>
```

Respuesta Post

```
<!DOCTYPE html>
<html>
<body>

<h1>The XMLHttpRequest Object</h1>

<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("POST", "demo_post2.asp", true);
  xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  xhttp.send("fname=Henry&lname=Ford");
}
</script>

</body>
</html>
```

Method	Description
setRequestHeader(header, value)	Adds HTTP headers to the request header: specifies the header name value: specifies the header value

UN ARCHIVO EN UN SERVIDOR; LA URL

El parámetro url del método open(), es una dirección a un archivo en un servidor:

xhttp.open("GET", "ajax_test.asp", true)

El archivo puede ser cualquier tipo de archivo, como.txt y.xml, o archivos de scripting del servidor con
(que pueden realizar acciones en el servidor antes de enviar la respuesta de vuelta).

Las solicitudes de servidor deben enviarse asincrónicamente. El parámetro async del método open() d
ajustado a true:

xhttp.open("GET", "ajax_test.asp", true);

Al enviar asincrónicamente, el JavaScript no tiene que esperar la respuesta del servidor, sino que puede
scripts mientras se espera la respuesta del servidor.

- ejecutar otros scripts mientras se espera la respuesta del servidor
- ocuparse de la respuesta después de que la respuesta esté lista

DIFERENCIAS ENTRE APLICACIONES WEB TRADICIONALES Y APLICACIONES WEB CON AJAX

Implica hacer uso de nuevos conceptos en la web, como controles ricos de ingreso (selectores de fecha, ingreso de texto con formato), servicios de drag and drop y evitar demoras al usuario en el uso del sitio

Capacidad offline. Permite que una aplicación web siga funcionando aunque se haya perdido conectividad con Internet. Por supuesto, esto será posible en algunos casos; asimismo, si la conexión se retoma.

Productividad alta del desarrollador. Se acordaron de nosotros, y los entornos de trabajo y las herramien desarrollar aplicaciones web evolucionaron hasta encontrarse en la actualidad cercanas a la productivida de escritorio. Tendremos capacidades drag and drop, desarrollo rápido de aplicaciones, capacidad de debu otras soluciones que permitirán el desarrollo de sitios web de manera más eficiente.

- Respuesta. Se acabaron las esperas para el usuario, las aplicaciones web responden con rapidez y es po con la aplicación aun cuando se espera una respuesta del servidor.
- Flexibilidad. Los nuevos sitios web permiten una interfaz flexible con la posibilidad de modificar la apar contenido y los servicios disponibles de una manera sencilla y rápida.

- Fácil de distribuir y actualizar. Actualizar una aplicación RIA es tan simple como publicar los nuevos servidor. Incluso, hasta se podría actualizar con cientos de usuarios conectados.
- Fácil de administrar. No hay metodologías de instalación complejas, DLL ni instaladores; asimismo, la las metodologías de instalación no es mucho mayor que la de cualquier aplicación web normal.

Hay dos tipos de aplicaciones ricas de Internet: las conocidas como full RIA y las RIA embebidas. Las p aplicaciones en las que se utiliza por completo el nuevo modelo RIA. Manejan una o dos direcciones (U resource location) para todo el sitio web. Escapan al clásico concepto de página web, para convertirse web. Las RIA embebidas en realidad son una mezcla entre las aplicaciones clásicas y las web 2.0. Siguen comportándose como páginas web normales, con hipervínculos interconectados hasta llegar a un punt funcionalidad, se convierten en una RIA.