Capa de aplicación

página web

HTML

estáticas, dinámicas

usan consultar al servidor

para actualizar su estado

- **Problema:** trabajar con páginas estáticas se torna muy ineficiente cuando la información cambia frecuentemente o cuando la información de la página varía de acuerdo a diferentes parámetros.
- Queremos evitar tener que modificar a mano las páginas estáticas a cada rato o tener que producir demasiadas páginas estáticas.
- **Solución:** usar **páginas dinámicas**:
  - Páginas HTML son generadas por medio de programas que ejecutan del lado del servidor..
  - Esos programas toman parámetros de entrada.
  - Esos parámetros de entrada suelen ser ingresados como valores de campos de formulario HTML.

Que el servidor web tenga que construir páginas dinámicas puede ser ineficiente por los siguientes motivos:

1. La página nueva a generar dinámicamente en el servidor puede tener una parte importante en común con la página que ya tiene el browser;
   - Y esa parte que se repite se genera de nuevo y tiene que enviarse de nuevo por la red.
   - Esa parte repetida va a tener que ser interpretada de nuevo por el browser.

Solución: página única del navegador. Desde ahí el usuario puede hacer consultar al servidor. y pedido de datos.

2. El cliente se queda bloqueado esperando luego de hacer un pedido HTTP al servidor web y recién puede continuar ejecutándose cuando recibe una página (estática o generada dinámicamente).
   - Estos son los llamados **pedidos sincrónicos**.
   - Si el procesamiento de un pedido del lado del servidor toma mucho tiempo, el no poder usar la aplicación web mientras tanto para otra cosa puede ser bastante desagradable para el usuario.
   - Luego de hacer pedido de datos la aplicación puede seguir haciendo otras tareas mientras se procesa ese pedido. A esto se lo llama **pedido asincrónico**.

Cuando llegan los datos se **actualiza** la interfaz del usuario de la página única en el browser.

- Solo se cambia la parte de la UI que se necesite; lo que no cambia, no se toca; y todo esto se hace dentro del browser.

URL's: es la forma de identificar un sitio web con "host".

Partes de un URL:

- Nombre del protocolo (HTTP).
- Nombre de dominio de host que contiene la página
- El nombre del archivo que contiene la página (camino al archivo).

```
http://www.someschool.edu/someDept/pic.gif
       nombre de host      camino
```

para las páginas dinámicas el URL puede contener el nombre del programa y los parámetros.

P. ej: `demo_get2.asp?fname=Henry&name=Ford`

Este ejemplo nos dice:

- Los parámetros son pares: `nombre=valor`.
- Se separa nombre de programa con '?'
- Se separan parámetros con '&'.

Incluir los parámetros en el URL es una opción, pero los URL tienen un tamaño máximo, por ello a veces no.

Solución 2: los parámetros se ingresan separados por & en un campo especial del pedido HTTP (llamado **cuerpo de la entidad**).

- Los URL tienen un tamaño máximo.
- Cuando los parámetros exceden ese tamaño máximo no se puede usar la solución 1;
- Es ahí que se torna útil la solución 2. → método POST.

Sitio web: contiene info, cont pag web relacionadas

app web: ofrece un servicio y permite realizar una tarea
ej: google docs.

Navegadores

Se conectan con HTTP a los servidores para intercambiar los archivos de una página web.

Orden seguido:

1. El cliente inicia una conexión TCP (crea socket) con el servidor web, usando el puerto 80
2. El servidor web acepta la conexión TCP del cliente.
3. Mensajes HTTP (mensajes del protocolo de capa de aplicación) intercambiados entre el browser (cliente HTTP) y el servidor Web (servidor HTTP)
4. La conexión TCP se cierra.

- La pag web no solo son HTML, puede contener JPG, GIF.

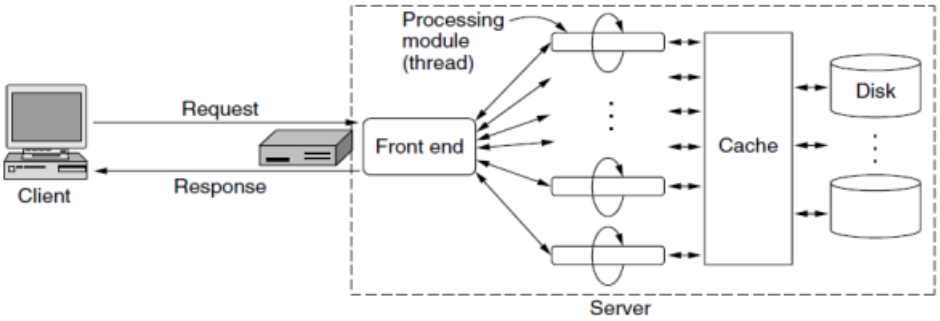
para ello el navegador reconoce los tipos MIME de cada archivo y tiene una herramienta o plugin para poder ver ese tipo de archivo.

- **Plug-in (conector):** es un módulo de código.
  - **Ejemplos**: Flash Player, Quick Time player
  - **El navegador obtiene los plug-in**
    - De un directorio especial de disco
  - Un navegador instala un plug-in como una **extensión de si mismo**.
  - Los plug-in se ejecutan **dentro** del proceso del navegador.
  - **Consecuencias**: los plug-in tienen acceso a la página actual y pueden modificar su apariencia..

- **Plug-in (cont):**
  - La **interfaz del plug-in**: conjunto de procedimientos que todos los plug-in tienen que implementar a fin de que el navegador pueda llamarlos
- La **interfaz del navegador**: conjunto de procedimientos del navegador que el plug-in puede llamar.
  - **Ejemplo**: procedimientos para asignar y liberar memoria, desplegar un mensaje en la línea de estado del navegador y consultar al navegador sobre los parámetros.
- **Una vez que el plug-in ha completado su trabajo**
  - se lo elimina de la memoria del navegador

- **Aplicaciones auxiliares.**
  - Se ejecutan en un proceso separado del browser.
  - No ofrecen interfaz al navegador ni usan servicios de este.
  - Suelen aceptar el nombre de un archivo y lo abren y despliegan.
  - **Ejemplo**: application/pdf para archivos pdf.

Los servidores utilizan caché para guardar la página más solicitada y poder responder a la conexión y solicitud más rápido.



Front-end: paso-reverso. Actúa contra otros servidores back-end y se los devuelve al cliente.

- **Solución:** Arquitectura con un **módulo front end** y k **módulos de procesamiento – MP-** (hilos).
  - Todos los MP tienen acceso al caché y a uno o más discos.

- **Pasos de un Servidor Web con múltiples hilos para manejar pedidos de páginas estáticas.**
  1. Cuando llega una solicitud el front end la acepta y construye un registro corto que la describe.
  2. Después entrega el registro a uno de los MP.
  3. Si se trata de pedido de página estática el MP primero verifica el caché para ver si el archivo está allí.
  4. Si el archivo está en caché actualiza el registro para incluir un apuntador al archivo.
  5. Si el archivo no está en caché el MP inicia una operación de disco.
     - Cuando el archivo llega del disco se coloca en la caché y se regresa al cliente.
  6. Mientras uno o más MP están bloqueados esperando a que termine una operación del disco, otros MP pueden estar trabajando en otras solicitudes.
  7. Conviene tener además múltiples discos, para que más de un disco pueda estar ocupado al mismo tiempo.

Cookies

- almacena el estado de la sesión de una app-web. Las volver a entrar a un sitio-web el navegador se fija en las cookies a ver el último estado de la sesión.

❑ **¿Qué es un cookie y qué tamaño tiene?**

- Una **cookie** es un pequeño archivo o cadena (de a lo sumo 4 KB).
- El **contenido de una cookie** toma la forma nombre = valor.
- **Ejemplo:** Carrito = 1-5011; 1-7013; 2-1372

Campos de una cookie.

1. **dominio** (nombre del dominio de destino del cookie)
   - Cada dominio puede almacenar hasta 20 cookies por cliente.
2. **ruta** en la estructura del directorio del servidor.
   - Identifica qué partes del árbol de archivos del servidor podrían usar el cookie.
3. El **campo contenido** toma la forma nombre = valor.
4. El campo **expira**.
   - **Si este campo está ausente** El navegador descarta el cookie cuando sale.
   - **Si se proporciona una hora y una fecha:** Se mantiene la cookie hasta que expira ese horario.
5. El campo **seguro**.
   - Se usa para indicar que el navegador solo puede retornar la cookie a un servidor usando un transporte seguro (p.ej: SSL, TLS).
   - Esto se usa para aplicaciones seguras (p.ej. comercio electrónico, actividades bancarias, etc.)

- para eliminar la cookie el servidor la reenvía con una fecha caducada.

- El navegador tiene un directorio de cookies donde las almacena.

> antes de cada solicitud a un servidor, se fija si tiene una cookie correspondiente a ese dominio. Si la hay, la envía junto con la solicitud para que el servidor la procese.

HTTP

Protocolo request y response.

HTTP 1.0:

Se manda un objeto por conexión TCP.

Manda 5 obj ⇒ crea 5 conexiones.

↓ ineficiente

HTTP 1.1:

Muchos objetos enviados en una sola conexión.

Los pedidos son procesados en orden y los resultados se mandan en orden.

Problema: no se envían los que el navegador no pidió pero sí uno a necesita.

Pide un .html pero no el .css o .js y luego los necesitaba.

HTTP 2.0:

- **Solución: protocolo HTTP 2.0**
  - Por medio de mecanismo server push empuja archivos que sabe que van a necesitarse pero que el cliente puede no saber inicialmente.
  - Las respuestas a los pedidos pueden volver en cualquier orden.
  - HTTP 2.0 comprime los encabezados y los envía en binario para reducir uso de ancho de banda.
  - Cada respuesta lleva un identificador de su pedido.

Pedidos HTTP.

Informaciones que debería tener un mensaje de pedido:

- En caso que se quiera recibir una página:
  - El **URL** de un documento.
  - La **especificación de programa que genera página web**
- El **tipo de acción** que se quiere hacer en el sistema de archivos del servidor web (meter páginas, borrar páginas, etc.) → campo método, primera línea.
- Mandar **información sobre la máquina/software del cliente** para que servidor web pueda retornar páginas adecuadas al cliente.
- Mandar **información de estado de sesión** para que el servidor se entere.
- **Restricciones sobre el tipo de páginas** que el cliente puede aceptar.

método de pedido.

	Method	Description
→	GET	Read a Web page
	HEAD	Read a Web page's header
→	POST	Append to a Web page
	PUT	Store a Web page
	DELETE	Remove the Web page
	TRACE	Echo the incoming request
	CONNECT	Connect through a proxy
	OPTIONS	Query options for a page

- **Método PUT**: Sube un archivo en el campo **cuerpo de la entidad** en el camino especificado por el campo URL.
- **Método DELETE**: Borra el archivo especificado en el campo URL.

El **método HEAD** simplemente solicita el encabezado de la respuesta del servidor web, sin la página o datos de la respuesta – o sea, feedback sobre el resultado del pedido, tipo de contenido, etc.

El **método OPTIONS:** permite que el cliente consulte al servidor por una página y obtenga los métodos y encabezados http que pueden ser usados con esa página.

➢ También puede usarse para saber los métodos http soportados por el servidor web en general.

Método Post:

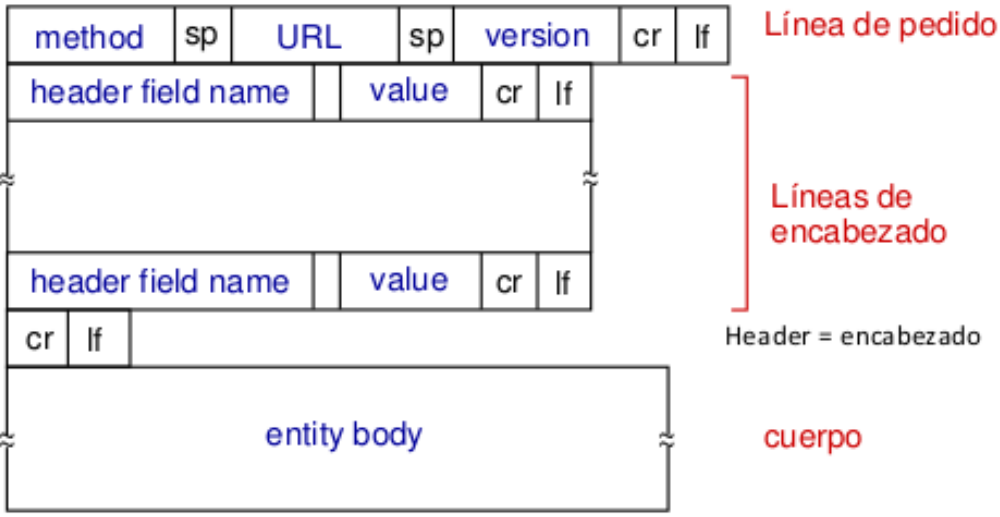
- La página web a menudo contiene input de formulario.
- El input es subido al servidor en un campo llamado el **cuerpo de la entidad**.

Método que usa URL:

- Usa el **Método Get.**
- El input es subido en el campo URL de la línea del pedido:

```
www.somesite.com/animalsearch?monkeys&banana
```

El método de solicitud va en la primera línea, la demás información va en encabezados de pedido.



- A la línea de solicitud le pueden seguir líneas adicionales llamadas **encabezados de solicitud**.

Respuesta HTTP:
- Feedback sobre el pedido → código
- documentos solicitados.
- tipo MIME del doc enviado
- estado de error.

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

toda la info adicional al documento enviado, se maneja como encabezados que tienen la forma "nombre encabezado: valor".

Partes de una respuesta HTTP

1. **Línea de estado.**
2. (opcional) **encabezados de respuesta.**
3. Luego vienen el cuerpo de la respuesta:
   - p.ej. página estática usando archivo en formato HTML.
   - P.ej. datos pedidos usando documento en formato XML.

```
Trying 4.17.168.6...
Connected to www.ietf.org.
Escape character is '^]'.
HTTP/1.1 200 OK
Date: Wed, 08 May 2002 22:54:22 GMT
Server: Apache/1.3.20 (Unix) mod_ssl/2.8.4 OpenSSL/0.9.5a
Last-Modified: Mon, 11 Sep 2000 13:56:29 GMT
ETag: "2a79d-c8b-39bce48d"
Accept-Ranges: bytes
Content-Length: 3211
Content-Type: text/html
X-Pad: avoid browser bug

<html>
<head>
<title>IETF RFC Page</title>

<script language="javascript">
function url() {
 var x = document.form1.number.value
 if (x.length == 1) {x = "000" + x }
 if (x.length == 2) {x = "00" + x }
 if (x.length == 3) {x = "0" + x }
 document.form1.action = "/rfc/rfc" + x + ".txt"
 document.form1.submit
}
</script>

</head>
```

Ej de respuesta.

Los encabezados de los mensajes HTTP pueden ser de solicitud o respuesta o ambos, se usan para especificar información o fijar restricciones. eventos o estado de sesión.

Header	Type	Contents
User-Agent	Request	Information about the browser and its platform
Accept	Request	The type of pages the client can handle
Accept-Charset	Request	The character sets that are acceptable to the client
Accept-Encoding	Request	The page encodings the client can handle
Accept-Language	Request	The natural languages the client can handle
Host	Request	The server's DNS name
Authorization	Request	A list of the client's credentials
Cookie	Request	Sends a previously set cookie back to the server
Date	Both	Date and time the message was sent
Upgrade	Both	The protocol the sender wants to switch to
Server	Response	Information about the server
Content-Encoding	Response	How the content is encoded (e.g., gzip)
Content-Language	Response	The natural language used in the page
Content-Length	Response	The page's length in bytes
Content-Type	Response	The page's MIME type
Last-Modified	Response	Time and date the page was last changed
Location	Response	A command to the client to send its request elsewhere
Accept-Ranges	Response	The server will accept byte range requests
Set-Cookie	Response	The server wants the client to save a cookie

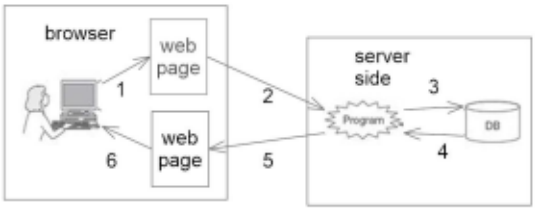
Algunos Encabezados HTTP

HTML

Una pag HTML es una serie de elementos, cada elemento para tener atributos. (ej href)

Pasos para generar páginas dinámicas del lado del servidor:

1. Un usuario llena un formulario y hace click en el botón de envío.
2. Se envía un mensaje al servidor web con el contenido del formulario.
   - Se proporciona el mensaje a un programa o una secuencia de comandos.
   - El programa procesa el mensaje.
3. El programa solicita información a un servidor de bases de datos.



4. El servidor de bases de datos responde con la información requerida.
5. El programa genera una página HTML personalizada y la envía al cliente.
6. El browser muestra la página recibida al usuario.

Formularios html.

Etiquetas para definir **formularios HTML**

Tag	Description
<form action="", method =""> ... <\form>	Declara un formulario. Action es URL de la página ejecutable que procesa formulario. Method especifica cómo los datos se mandan al servidor (p.ej. GET, POST)
<select> ... <\select>	Para especificar una lista de la que usuario elige un elemento.
<option value = ""> ... <\option>	Para indicar opción de <select>
<textarea rows = "" cols=""> ... </textarea>	Control de ingreso de texto de varias líneas
<input name="" type="" value="">	Permite definir campo de input donde type puede ser: button, radio, password, text, submit, checkbox, hidden, etc.

Paginas dinamicas:

Son aquellas que programas server side.

Tareas que suelen hacer las páginas dinámicas:

- Procesar parámetros de formularios
- Procesar encabezados de pedido HTTP
- Pedir datos a fuentes de datos (e.g. bases de datos)
- Generar página web con los datos recibidos.
- Generar encabezados de respuesta HTTP

php, java server node, etc.

Php se usa para procesar los datos enviados por un formulario.

Acceder a ellos mediante $_POST $_GET, también para acceder a los encabezados de los HTTPrequest.

- $_SERVER: contiene información de encabezados, caminos y localización de scripts.
- Para acceder a encabezados poner como argumento alguna de las siguientes:
  - HTTP_USER_AGENT, SERVER_ADDR, SERVER_NAME, SERVER_SOFTWARE, SERVER_PROTOCOL, REQUEST_METHOD, REQUEST_TIME, QUERY_STRING, HTTP_ACCEPT, HTTP_ACCEPT_CHARSET, HTTP_HOST, etc.

PHP puede crear nuevos Headers y cookies y acceder a valores de los mismos.

Implementación de una web

- **Problema**: ¿Cómo diseñar aplicaciones de página única?
- **Solución**: El servidor web en respuesta a un pedido HTTP solo enviará datos (**no genera** página web con esos datos) para actualizar parte de una página web en el navegador (sin tener que recargar una página completa) y el browser se ocupa de actualizar la interfaz del usuario (IU).
  - Cuando llegan datos del servidor web se genera e inserta la parte nueva de la IU y se deja igual la parte de la IU que debe preservarse.
  - **Pedidos asincrónicos**: El cliente hace un pedido al servidor web y el cliente puede seguirse ejecutando para otra cosa, mientras el pedido es procesado del lado del servidor.

En cada pedido asincrónico el cliente usa los datos para actualizar la IU.

IU de pag web una → con 2 partes.

parte → con elementos IU.

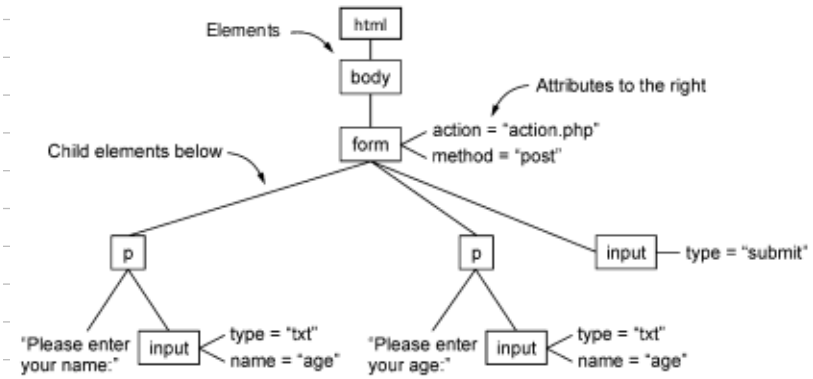
↳ arbol DOM de elementos iu.

DOM es una representación de una página HTML que es accesible a programas.

Esta representación es estructurada como un **árbol DOM** que refleja la estructura (jerárquica) de los elementos HTML.

- ❑ En la raíz está el elemento *HTML* que representa la página.
- ❑ El elemento *HTML* es el padre de elementos *head* y *body*.
- ❑ El elemento *body* a su vez puede ser padre de elementos *form*, *ol*, *ul*, etc.
- ❑ Además al lado de un elemento puede uno dibujar sus **atributos**. (p.ej. para *form* los atributos *action* y *method*).

```
<html>
<body>
  <form action="action.php" method="post">
    <p> Please enter your name: <input type="text" name="name"> </p>
    <p> Please enter your age: <input type="text" name="age"> </p>
        <input type="submit">
  </form>
</body>
</html>
```



los programas utilizan el DOM para observar/obtener partes de la página. usan terminología entera.

por ej en JS document.forms[0].elements[3].value = "..."

es acceder al primer formulario del DOM, mirar el valor del 4to elemento del form.

- **Problema**: ¿Cómo pasar de una pantalla de IU a otra?
- **Solución**: Usamos mecanismos como los siguientes:
  - Cambian algunos elementos de la IU
    - Algunos se eliminan, otros se agregan, otros se modifican.
    - Se modifica el árbol de la primera pantalla para pasar a la segunda pantalla.
    - Para esto se usa una secuencia de comandos.
  - Cuando ocurren eventos de usuario en ciertos elementos de IU, entonces se generan cambios en IU.
    - Estas son reglas ECA (evento-condición-acción)
    - La acción es una secuencia de comandos que cambia el árbol de elementos de IU por otro árbol.

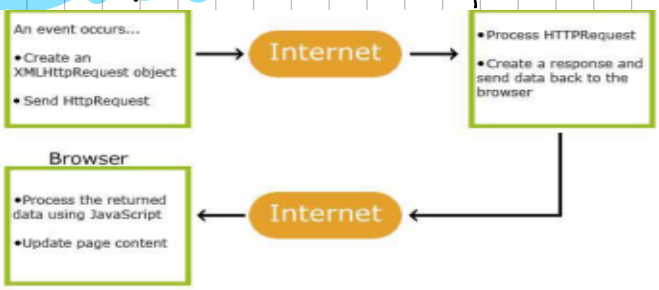
Java Script se utiliza para pasar de una pantalla a la otra.

- Algunas sentencias y métodos que permiten modificar una página HTML siendo visualizada.

Construcción	Description
e.innerHTML = c;	Cambia contenido HTML de *e* por *c*
e.a = v;	Cambia valor de atributo *a* de elemento *e* por *v*.
e.setAttribute(a,v)	Crea en elemento *e* el atributo *a* con valor *v*. Si *a* ya estaba , solo cambia su valor.
document.createElement(*N*)	Crea elemento HTML de etiqueta *N*
document.createTextNode(*T*)	Crea nodo de texto *T*
e.appendChild(n)	Se agrega elemento *n* como último hijo del elemento *e*
e.removeChild(n)	De elemento *e* se remueve subelemento *n*.
e.replaceChild(n1,n2)	De elemento *e* se reemplaza subelemento *n1* por elemento *n2*.

Algun Java Script puede procesar eventos asociados a etiquetas Html.

Pedidos/respuestas HTTP.



1. Se especifica pedido HTTP usando *open*().
2. Un pedido HTTP se envía con *send*() y va a ser un pedido de datos o texto.
3. El servidor va a obtener los datos pedidos (p.ej. de una base de datos), con ellos va a crear una respuesta HTTP y enviarla al navegador.
4. El cliente recibe la respuesta al pedido, procesa los datos recibidos y modifica la página actual (usando las primitivas para modificar la IU que acabamos de ver.)

Primitivas para hacer y procesar pedidos/respuestas HTTP

Construcción	Description
Objeto XMLHttpRequest	Objeto usado para intercambiar datos con servidor web.
Método Open(method, url, async, user, psw)	Para especificar el pedido HTTP
Método send()	Para enviar el pedido HTTP
Método setRequestHeader()	Para fijar encabezados del pedido HTTP
Propiedad responseText de objeto XMLHttpRequest	Se refiere a la respuesta del servidor como un string
Propiedad readyState	Estatus del XMLHttpRequest
getAllResponseHeaders()	Retorna toda la información de encabezados de respuesta

Respuestas:

- la propiedad responseText retorna la respuesta del servidor como un string.

- **Manejo de respuestas HTTP del servidor:**
  - La propiedad **readyState** mantiene el **estatus del XMLHttpRequest.**
  - La **propiedad onreadystatechange** define una función a ser ejecutada cuando el **readyState** cambia.
  - **Cambios en estado** pueden ser por ejemplo:
    - 1: Conexión con el servidor establecida
    - 2: Pedido recibido
    - 3: Procesando pedido

- P.ej:

```
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML =
        this.responseText;
  }
};
```

- Si el pedido terminó, la respuesta está lista y el pedido fue exitoso se pone como contenido HTML del elemento de identificador "demo" el texto de this.responseText.

hay metodos para obtener toda la info de todos los headers y metodos para obtener de un header en especifico.

XML (Extensible Markup Language)

XML se puede usar para definir datos así:

- Los *nombres* de los elementos de datos son expresados mediante etiquetas del tipo: <nombre>.
- Un elemento de datos de nombre *n* consiste de información que se encierra entre etiquetas: <n> y </n>.
- Esa información puede construirse usando otros elementos de datos anidados entre <n> y </n> o solo texto.
- P.ej: en <TITLE>Greatest Hits</TITLE> el elemento <TITLE> tiene solo información de texto.
- P.ej: elementos de nombre <ARTIST> anidados en elementos de nombre <CD>.

- La propiedad **responseXML** retorna la respuesta del servidor como un documento XML.
- La función **d.getElementsByTagName(*N*)** retorna los elementos XML dentro del documento *d*, que tienen nombre de etiqueta *N*.
- Se puede **recorrer un elemento XML**:
  - Al igual que cuando hablábamos del DOM, se puede pensar el documento XML como un árbol y se puede recorrer ese árbol usando **expresiones de camino** donde cada sección de una expresión de camino se separa con '/'.
  - **e.childNodes[i]** retorna el nodo hijo *i* del elemento *e*. (el *i*-ésimo subelemento dentro de *e*).
  - Si *e* es un elemento con contenido de solo texto, ***e*.nodeValue** retorna el texto del contenido de *e*.
  - Recordar que un documento HTML5 respeta XML. O sea, que nada impide generalizar la idea del DOM de HTML5 a XML.
  - P.ej: x[i].childNodes[0].nodeValue
    - es una expresión de camino donde primero se toma elemento i-esimo dentro de x, luego para ese elemento se toma primer subelemento anidado que es de solo texto y luego se toma ese texto.