

Ingeniería de Aplicaciones Web

Diego C. Martínez

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Web

1989



Tim Berners-Lee



World Wide Web

los documentos
están escritos
en hipertexto
(HTML)



El protocolo de
comunicación es HTTP



Navegadores

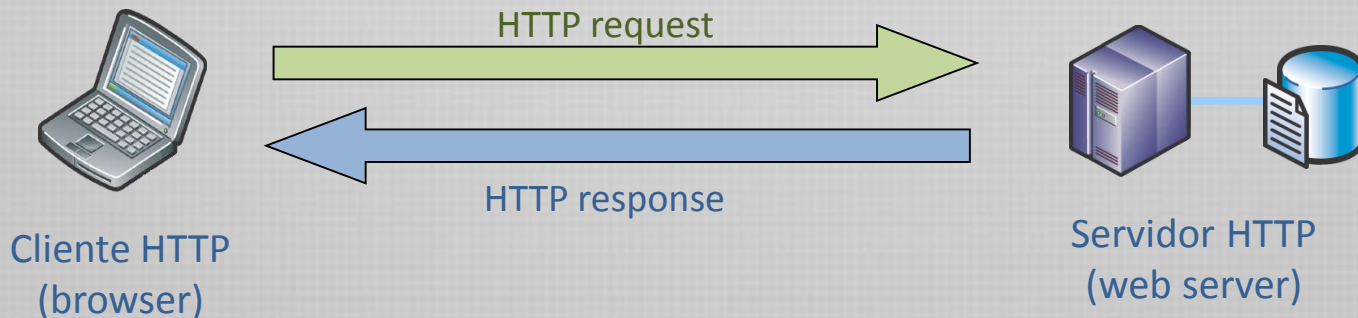
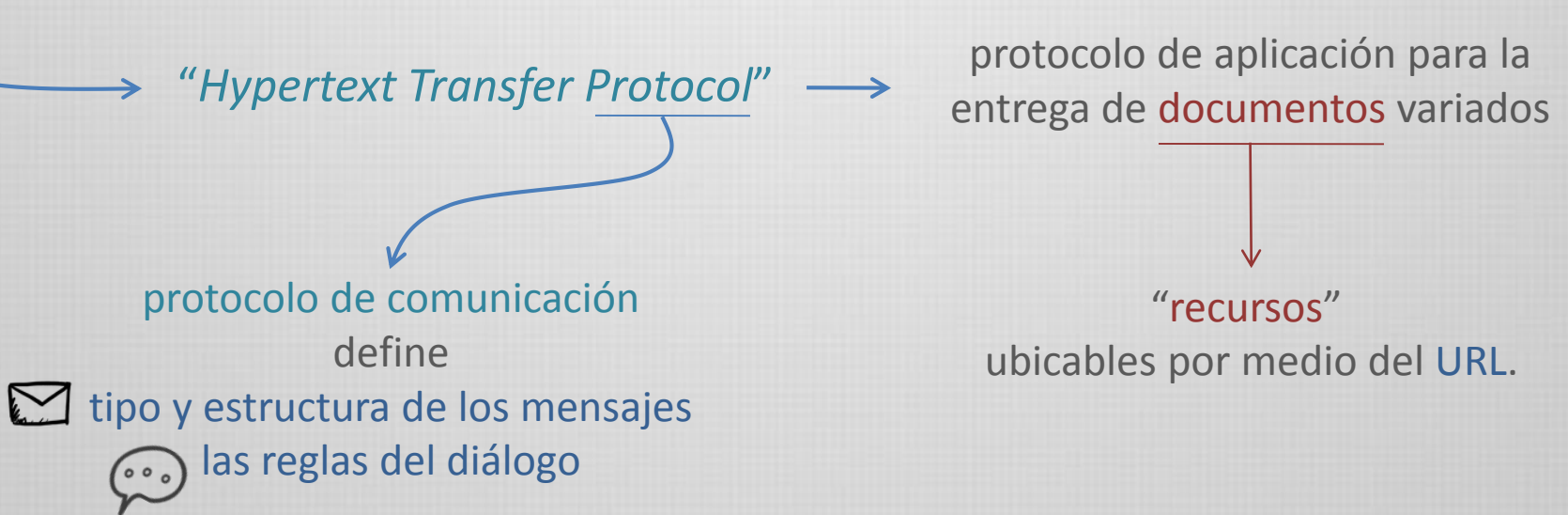


En 1994 Berners-Lee funda el *World Wide Web Consortium* (W3C) en el MIT, con apoyo de DARPA.

La idea central era asegurar la **compatibilidad** por medio de la definición de estándares, denominados *W3C Recommendations*.

HTTP

HTTP es el protocolo de red para la Web.



HTTP es un protocolo sin estado, o *stateless protocol*: no mantiene información sobre la conexión entre transacciones.

URLs - sintaxis

El nombre estandarizado de un recurso en Internet, que lo identifica unívocamente se denomina **URL - Uniform Resource Locator**.

Un subconjunto de una clase más general de identificadores (URI - Uniform Resource Identifier)

Una URL describe el recurso por su locación (dónde está) y el protocolo que se entiende para obtenerlo (HTTP, HTTPS, FTP, SMTP, etc).

<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>

<scheme> Protocolo para acceder al recurso

<user>:<password> Usuario y password requerido para acceder al recurso

<host>:<port> Nombre del host o dirección IP del servidor que tiene el recurso + port

<path> Nombre local del recurso

<params> Parámetros de entrada para el recurso

<query> Parámetros de entrada para el recurso

<frag> Referencia a una porción del recurso (de uso para el cliente)

URLs - sintaxis

Ejemplos: `http://cs.uns.edu.ar`
`http://www.uns.edu.ar:80`
`http://www.fis-ski.com/uk/disciplines/alpineskiing`
`https://www.bancogalicia.com.ar`
`ftp://ftp.prep.ai.mit.edu/pub/gnu`
`ftp://anonymous:my_passwd@ftp.prep.ai.mit.edu/pub/gnu`
`http://www.youtube.com/watch?v=0edCAOkdXgU`

En HTTP el port por defecto es el 80 y no requiere usuarios ni passwords.

Una URL puede ser **absoluta** o **relativa**

- las URL **absolutas** contienen toda la información necesaria para obtener el recurso

Las anteriores son todas absolutas

- las URL **relativas** son interpretadas en relación con otra URL.

`Go!`

La URL relativa se completa con

- una URL base declarada en el recurso
- la URL base del recurso que contiene la URL relativa

(VER RFC 2396 de la W3C)

URLs - sintaxis

Con el objetivo de que las URLs sean generales y portables por protocolos diferentes de las capas inferiores, algunos caracteres requieren **codificación**:

<http://www.unhost.com.ar/~usuario/un articulo para la web.html>



<http://www.unhost.com.ar/%7usuario/un%20articulo%20para%20la%20web.html>

Otros caracteres están reservados y no pueden usarse sin codificación, entre ellos:

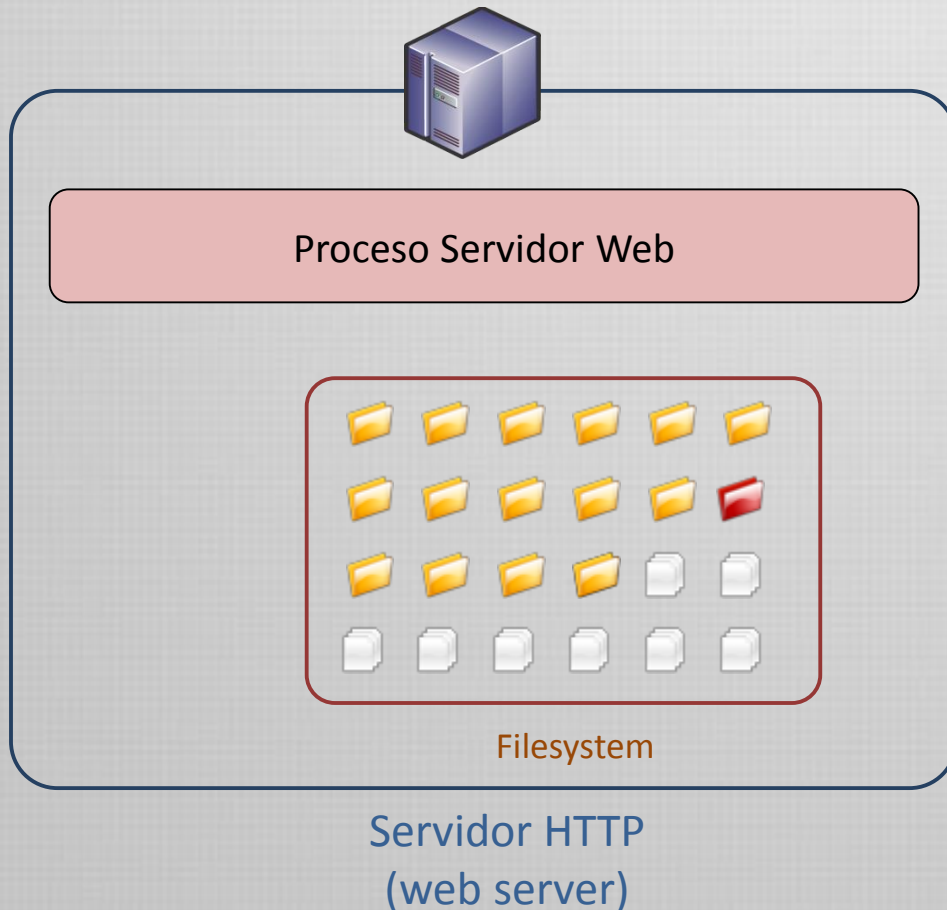
% / . . # ; : ? < > { }

Visitar: <http://www.w3.org/Addressing>

Servidores web

El servidor web actúa como una interfaz entre un conjunto de recursos y los clientes.

Típicamente, los recursos se encuentran todos ubicados en un directorio o carpeta específica, denominado *document root* o *docroot*.



Usualmente denominado también

htdocs

public_html

www

webroot

webaps

Para solicitar un recurso del servidor, es necesario mencionar

- el *host* y
- el *camino hacia el recurso*

No necesariamente el camino físico:

<http://unhost.com/~john/index.html>

puede ser

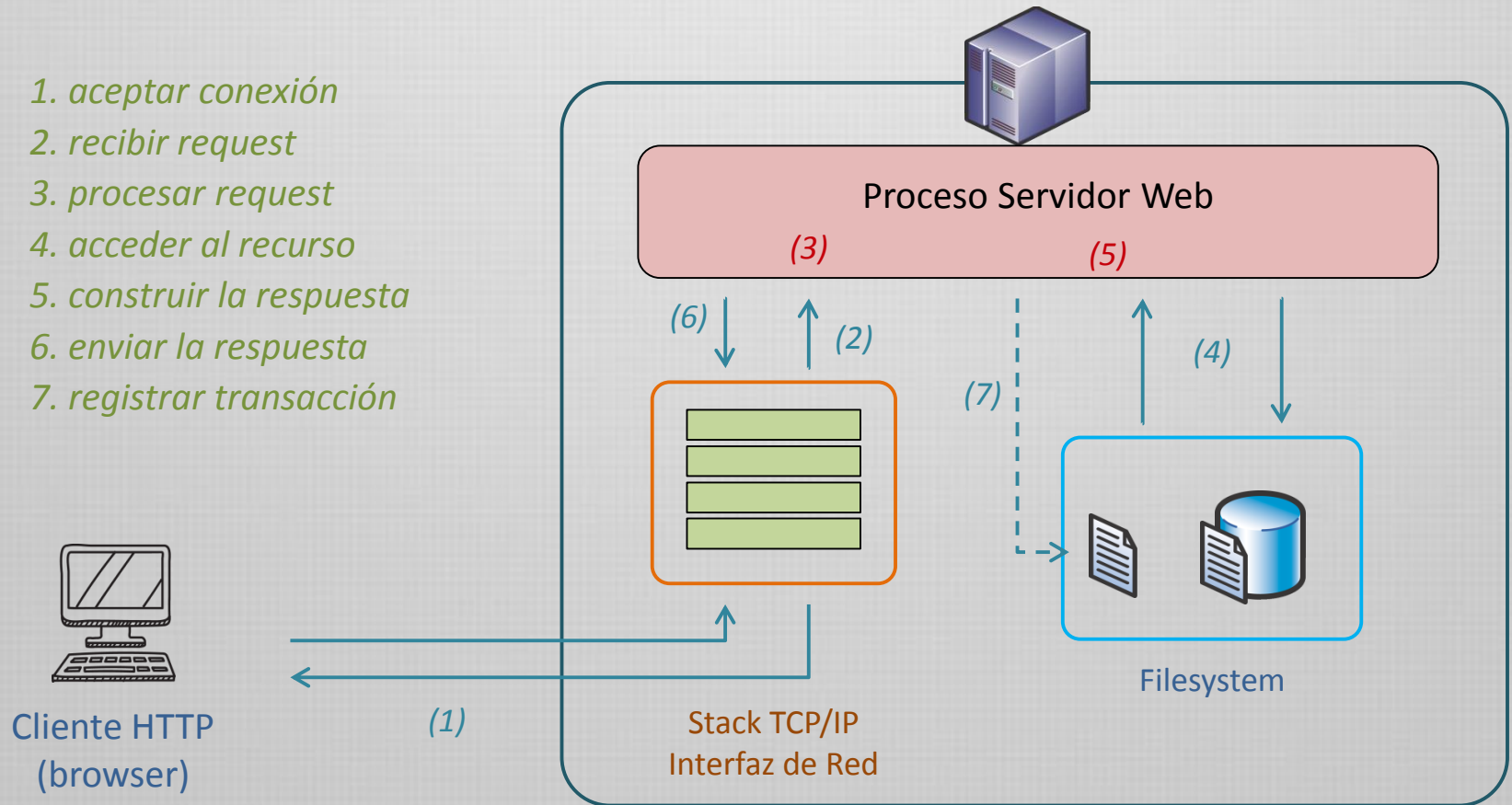
</home/users/john/web/index.html>

en el servidor *unhost.com*

Servidores web

El esquema de trabajo de un servidor web es el siguiente:

1. *aceptar conexión*
2. *recibir request*
3. *procesar request*
4. *acceder al recurso*
5. *construir la respuesta*
6. *enviar la respuesta*
7. *registrar transacción*

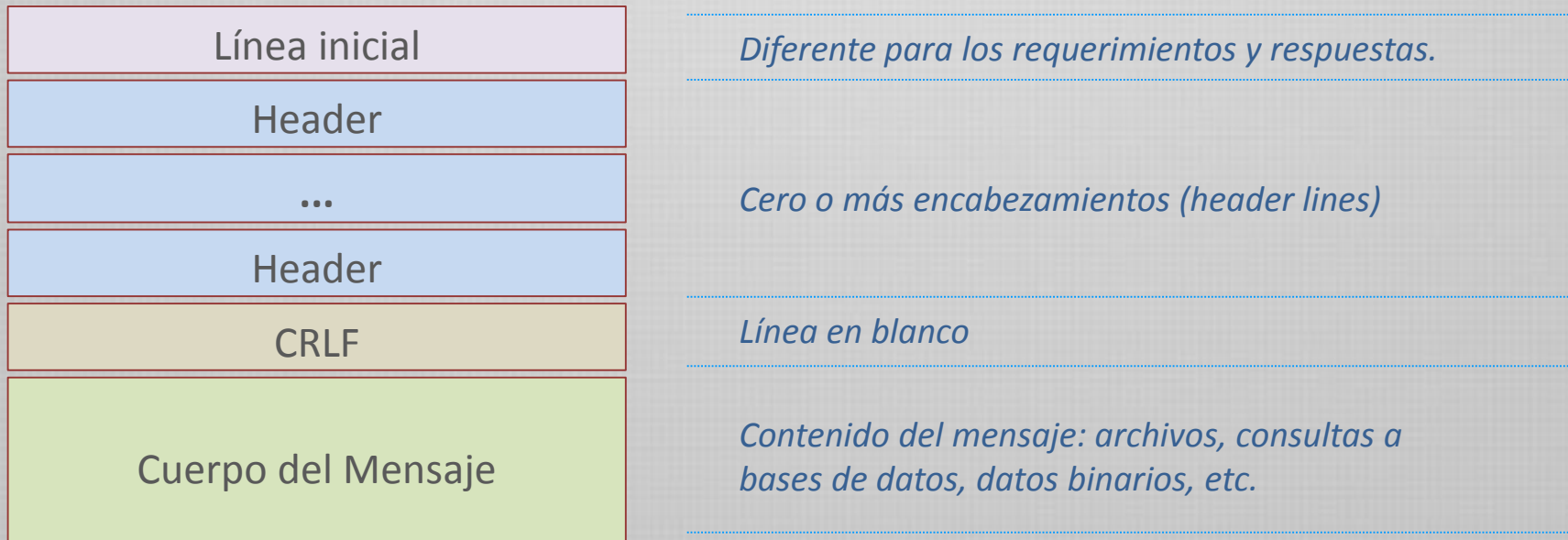




Como **protocolo de comunicación**, HTTP define el **tipo y estructura** de los mensajes que se envían y las reglas del diálogo entre los dos participantes.

- mensajes hacia el servidor —————> *inbound messages/ requests*
- mensajes desde el servidor —————> *outbound messages / responses*

La **estructura** de los mensajes es la misma para todos





Una línea inicial de *request* tiene tres partes:

- El nombre del método, en mayúsculas.
- El path local del recurso solicitado
- La versión de HTTP utilizada.

```
GET /path/to/file/index.html HTTP/1.0  
POST subscribir.php HTTP/1.0
```

Una línea inicial de *response* tiene tres partes:

- La versión HTTP
- El código de status de la respuesta
- Descripción del status

```
HTTP/1.0 200 OK  
HTTP/1.0 404 Not Found
```

El primer dígito del *código de status* identifica la categoría:

- **1xx** indica un mensaje de información únicamente
- **2xx** indica éxito en general
- **3xx** redirecciona el cliente a otro URL
- **4xx** indica un error en la parte del cliente
- **5xx** indica un error en la parte del servidor



- ▶ **200 OK**

El pedido tuvo éxito, y el recurso es retornado en el cuerpo del mensaje.

- ▶ **404 Not Found**

El recurso solicitado no existe

- ▶ **301 Moved Permanently**

- ▶ **302 Moved Temporarily**

- ▶ **303 See Other (HTTP 1.1 only)**

El recurso se movió a otro URL y debe ser pedido automáticamente por el cliente. Usualmente es utilizado por un script que redirecciona al visitante.

- ▶ **500 Server Error**

Error inesperado. Típicamente un error en algún script alojado en el servidor que impide que se ejecute correctamente.

El detalle completo de los códigos puede encontrarse en las *especificaciones de la W3C*:

HTTP 1.0 : RFC 1945

HTTP 1.1 : RFC 2616

El protocolo HTTP – header lines

Las **líneas de encabezamiento** proveen información sobre el pedido (request) o la respuesta (response) o el objeto que se está enviando en el cuerpo del mensaje.

Header-Name: value

HTTP 1.0 define 46 headers, pero ninguno es requerido.

HTTP 1.1 define 46 headers y requiere `Host` en los pedidos realizados.

Algunos headers importantes del *cliente*:

- ▶ **From:** es la dirección de mail de quien realiza un pedido.
- ▶ **User-agent:** es el programa que realiza el pedido, de la forma “*Nombre/x.xx*”

Ejemplo: **User-agent: Mozilla/3.0Gold**

Algunos headers importantes del *servidor*:


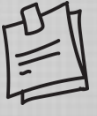
- ▶ **Server:** identifica el software del servidor, en el mismo formato “*Nombre/x.xx*”

Ejemplo: **Server: Apache/1.2b3-dev**

- ▶ **Last-modified:** es la fecha de modificación del recurso otorgado

Ejemplo: **Last-Modified: Fri, 31 Dec 1999 23:59:59 GMT**

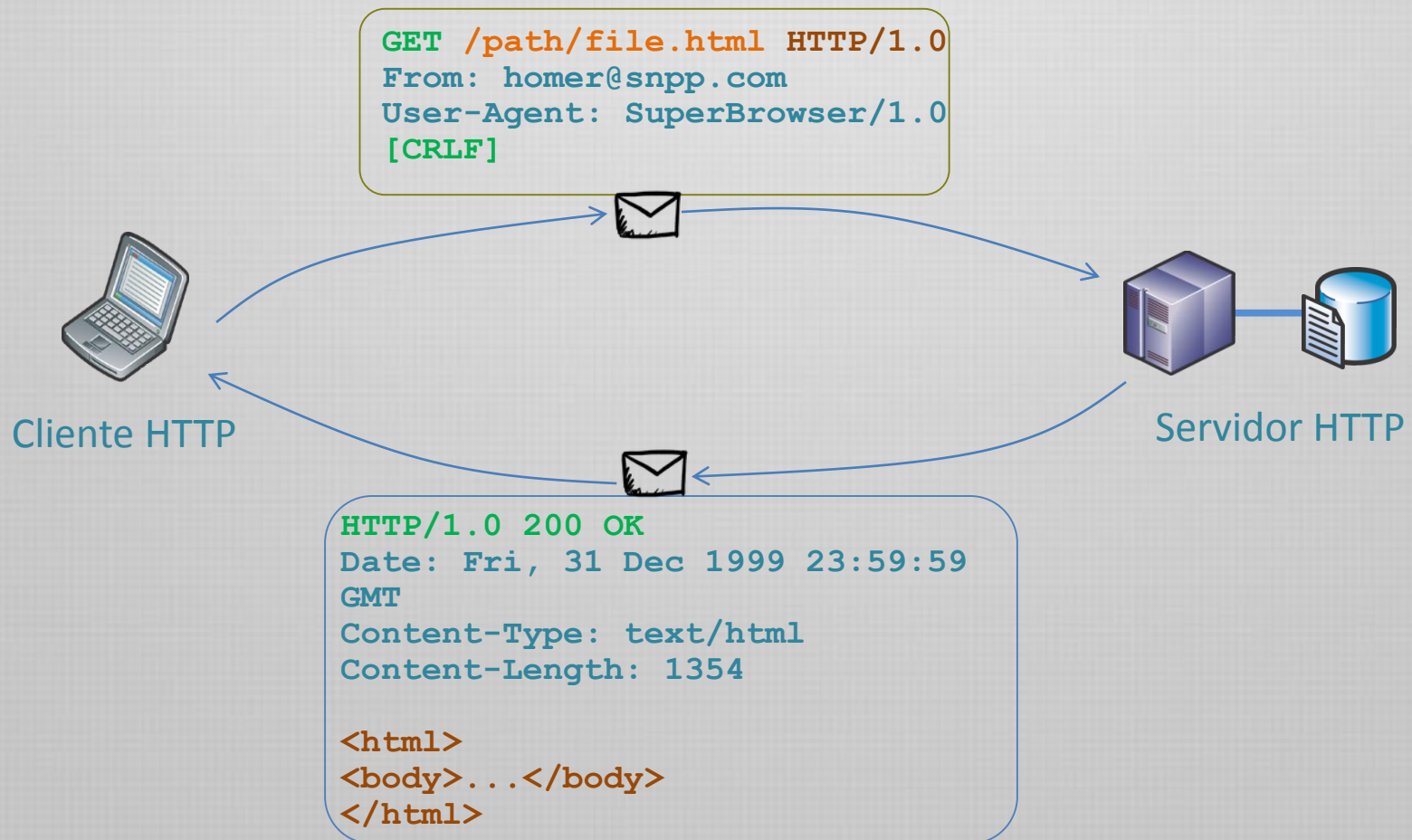
Métodos HTTP

- GET** —→ *Obtiene un documento del servidor.
El mensaje no tiene cuerpo.*
- POST** —→ *Envía datos al servidor para el procesamiento.
Los datos se especifican en el cuerpo del mensaje.* 
- HEAD** —→ *Solicita solo los encabezados de un documento.
El mensaje no tiene cuerpo.*
- PUT** —→ *Almacena el cuerpo del request en el servidor, bajo el nombre
indicado como URL.* 
- TRACE** —→ *Hace un rastreo del mensaje al servidor. El mensaje no tiene cuerpo*
- OPTIONS** —→ *Determina qué métodos pueden operar en un servidor.
El mensaje no tiene cuerpo*
- DELETE** —→ *Remueve un documento del servidor. El mensaje no tiene cuerpo*

Requests - responses

Si el mensaje incluye un cuerpo, entonces habrá dos *headers* que describen ese cuerpo:

- **Content-Type**: *MIME-type* del cuerpo, como *text/html* or *image/gif*.
- **Content-Length**: número de bytes en el cuerpo.



El protocolo HTTP - métodos

Además de **GET**, los métodos más usados son **HEAD** y **POST**.

El método **HEAD** solicita al servidor únicamente los headers de la respuesta, sin cuerpo.
Permite ver si existe algún recurso, ver si ha sido modificado, etc.

El método **POST** es esencial para la interacción web.

Este método envía información al servidor para ser procesada por quien corresponda, como un script CGI o PHP.

Procesará los datos enviados

```
POST /empleados/despedir.cgi HTTP/1.0
From: burns@snppl.com
User-Agent: HTTPTool/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 23
```

```
nombre=Homero&sector=7G
```

Cuerpo del mensaje

El protocolo HTTP – POST vs GET

```
POST /empleados/despedir.cgi HTTP/1.0
From: burns@snpp1.com
User-Agent: HTTPTool/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 23
```

nombre=Homeroysector=7G

GET /empleados/despedir.cgi?nombre=Homeroysector=7G HTTP/1.0

→ Sin cuerpo de mensaje

GET debe usarse *preferentemente* cuando los datos son pocos.
En otro caso debe usarse el método **POST**.

En ambos casos, el URL será codificado (*URL-encoded*) para transmitir los caracteres especiales (espacio, &, %, etc).

Documentos

Recordemos:

la Web surge como un servicio de Internet para estudiar y explorar documentos multimediales.



*documentos de texto, con imágenes y/o sonido.
facilidades de exploración de información relacionada.*



*"The frontiers of a book are never clear-cut,
because it is caught up in a system of references to
other books, other texts, other sentences: it is a
node within a network of references"*

*The archeology of knowledge
Michel Foucault, 1969*

Esto requiere una forma especial de estructurar la información, para interrelacionarla y visualizarla adecuadamente.

Hipertexto

Sabemos que la idea de *hipertexto* es anterior al surgimiento de la web, e incluso de las redes globales...

*"Hypertext is a **non-sequential writing**, a text that branches and allows choices to the reader, best read at an interactive screen. As popularly conceived, this is a series of text chunks connected by links which offer the reader different pathways"*

Theodor Nelson (1960)



HTML

Propuesta de Berners-Lee:

Enriquecer un documento con marcas especiales para estructurar, vincular y componer la información, de manera tal de “convertirlo” en hipertexto.



+

<marcado>

=

*HyperText
Markup
Language.*

El aporte de Berners-Lee no se limita al HTML.

*De hecho, ya existían lenguajes de marcado
(SGML- Standard generalized markup language).*

Su propuesta es una **red global de documentos**, unidos por

“global hypertext links”

*Berners-Lee implementa el primer browser de documentos HTML
y el primer servidor HTML de la historia, en una computadora NeXT.*

Estructura de HTML

Las marcas (*tags*) se encierran entre **<** y **>**

La versión original (de Berners-Lee) era simple, pero con posterioridad diferentes navegadores comenzaron a agregar su propias modificaciones (“browsers-war”).

Actualmente la definición de HTML de la W3C es estricta, pero los navegadores siguen procesando documentos sin la estructura apropiada.

Cada tag tiene un nombre y posiblemente varios atributos (**attr=valor**)

The diagram illustrates three different HTML tag structures. Each structure is shown with an opening tag and a closing tag, with arrows pointing to them from the labels 'opening tag' and 'closing tag'.

- Structure 1: `datos<tag>contenido</tag>datos`
 - The opening tag is `<tag>`.
 - The closing tag is `</tag>`.
- Structure 2: `datos<tag attr1=valor1 attr2=valor2>contenido</tag>datos`
 - The opening tag is `<tag attr1=valor1 attr2=valor2>`.
 - The closing tag is `</tag>`.
- Structure 3: `datos<tag_unario>datos`
 - This is a self-closing tag, so there is no separate closing tag.

Primera versión de HTML

Tags de la primera versión de HTML:

`<TITLE> ... </TITLE>` *Título del documento*

` ... ` *Hipervínculo.*

`<ISINDEX>` *Indicador de una página índice*

`<PLAINTEXT>` *Texto a ser tomado en forma literal*

`<LISTING> ... </LISTING>` *Texto de font de tamaño fijo, tomado literalmente*

`<P>`, `<H1>`, `<H2>`, `<H3>`, `<H4>`, `<H5>`, `<H6>` *Párrafo y encabezados*

`<ADDRESS> text ... </ADDRESS>` *Direcciones, contacto, footers*

`<HP1>...</HP1>` `<HP2>... </HP2>` *Texto resaltado. Ya no se usa.*

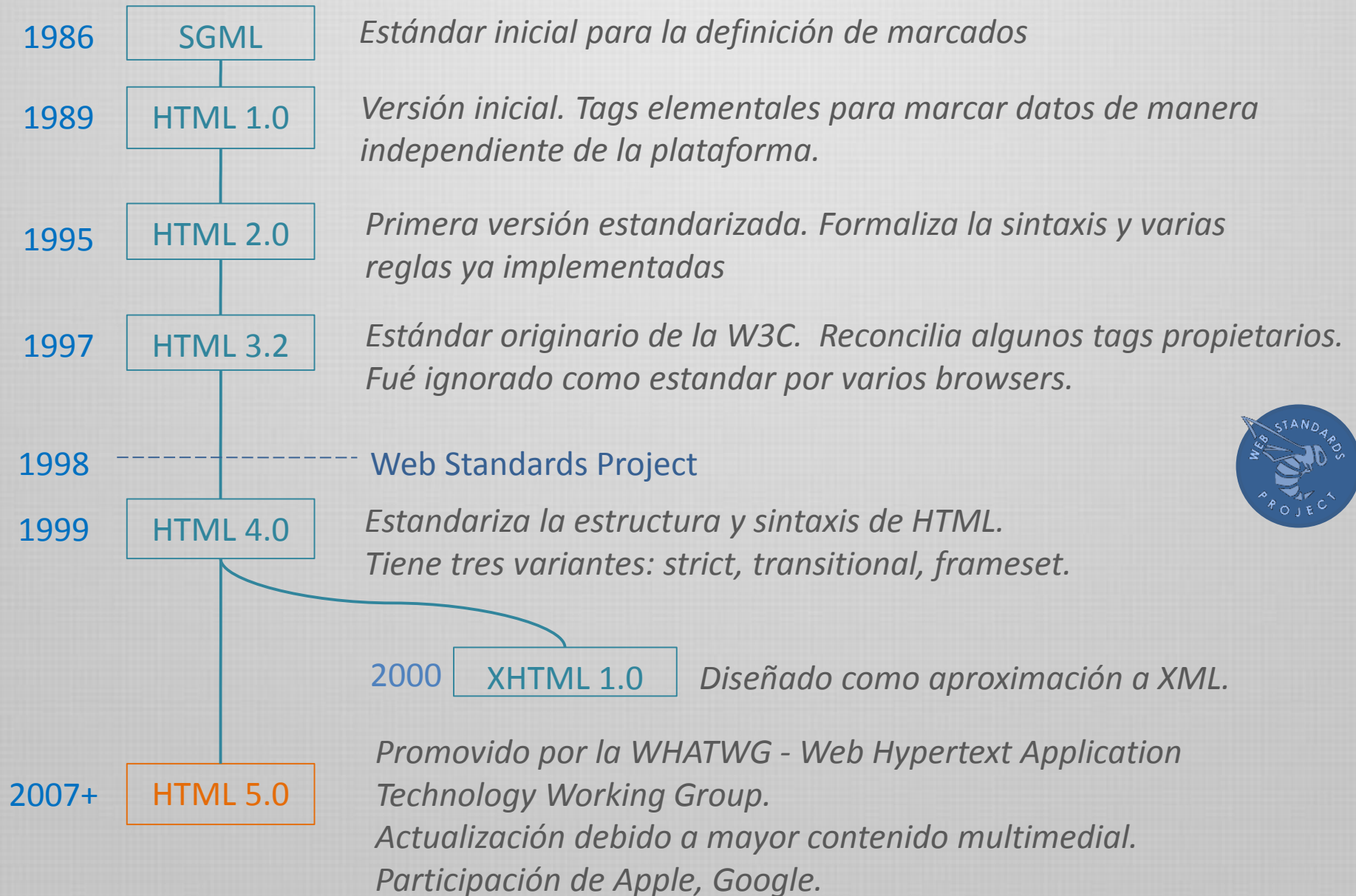
Listas descriptivas

```
<DL>
<DT>Term<DD>definition paragraph
<DT>Term2<DD>Definition of term2
</DL>
```

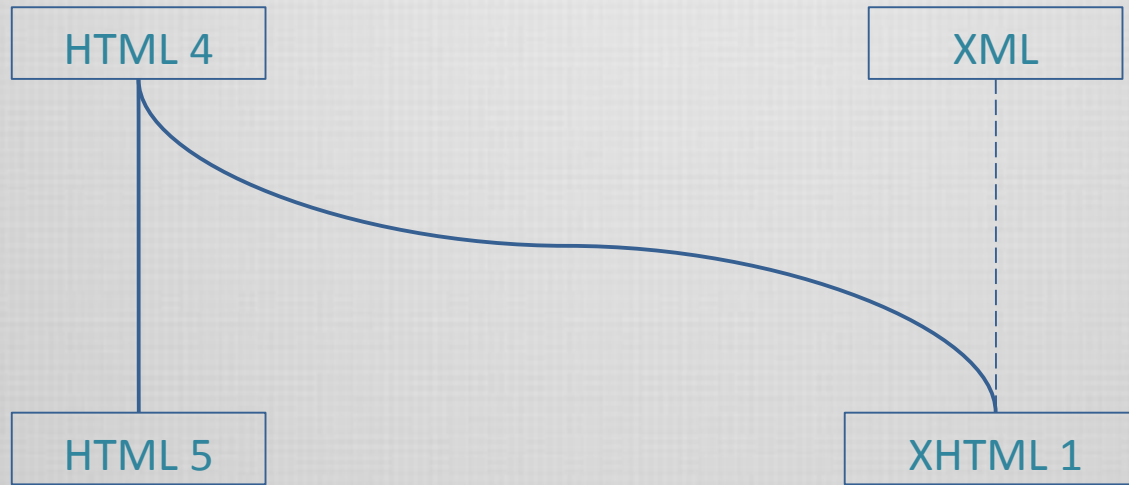
Listas no numeradas

```
<UL>
  <LI> list element
  <LI> another list element
</UL>
```

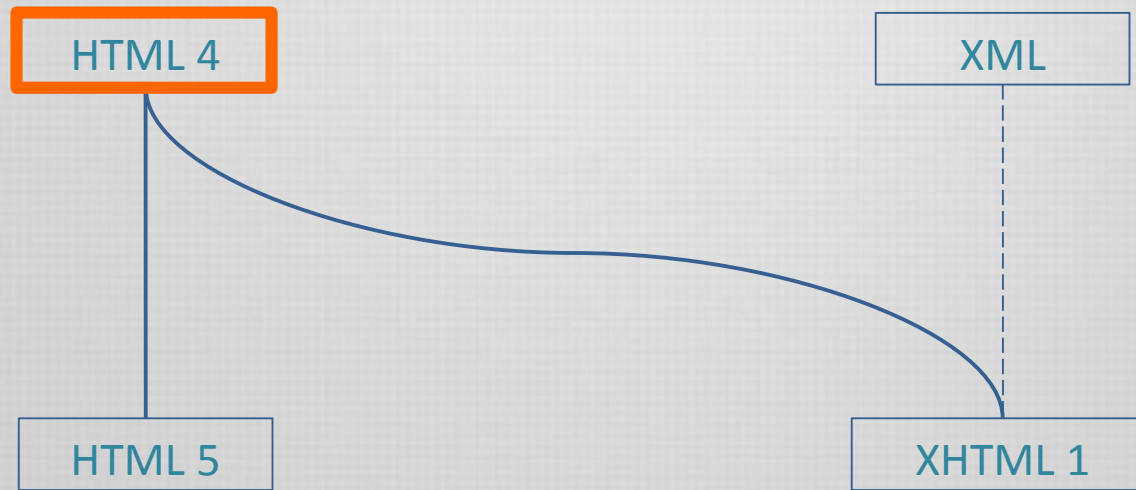
Evolución de HTML



Últimos estándares



Últimos estándares

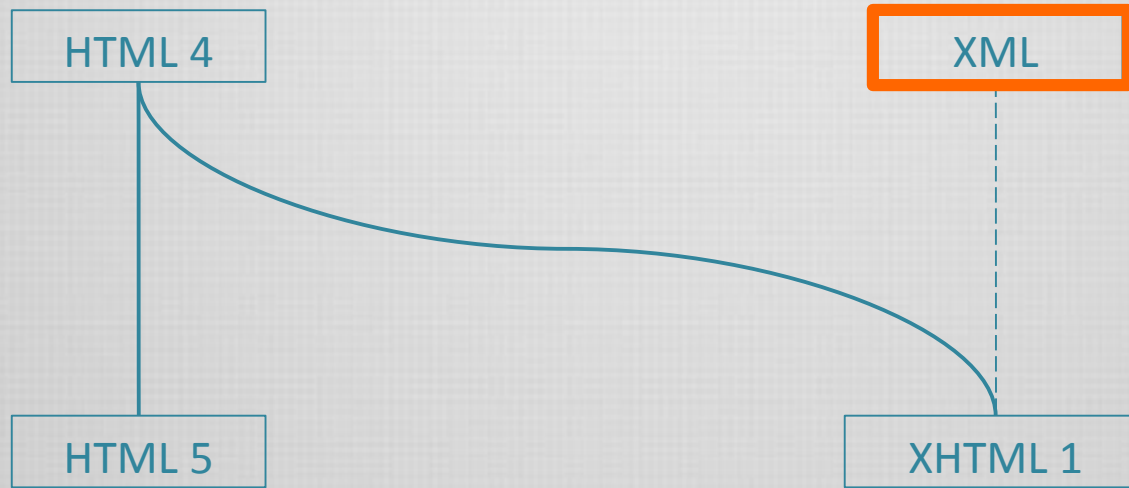


Introdujo mecanismos para
*hojas de estilo, scripts, objetos embebidos,
mejoras en accesibilidad, internacionalización.*

*Enfocado en una mejor distinción entre la **estructura** del documento y su **presentación**.*

Antiguo y paulatinamente en desuso.

Últimos estándares



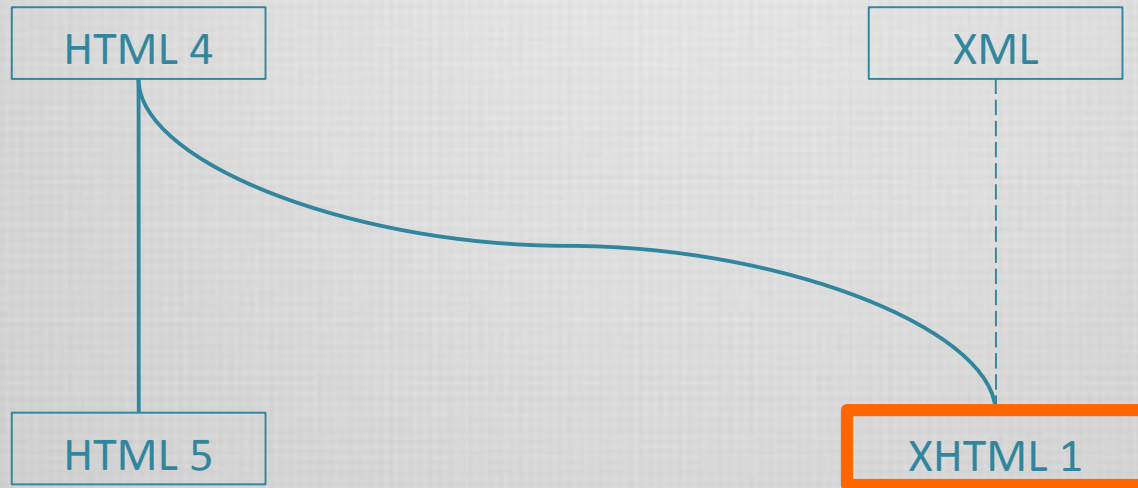
XML es el lenguaje de marcado **extensible**.
Es extensible porque los tags son definibles por el usuario.

Orientado exclusivamente a la estructuración y descripción de datos.

Es la base de la **interoperabilidad** de muchos sistemas.
Es el estándar para muchas tecnologías web (Ajax, Servicios web, etc)

Es una recomendación de la W3C desde 1998.

Últimos estándares



XHTML es una implementación de HTML en XML.

NO es un reemplazo de HTML.

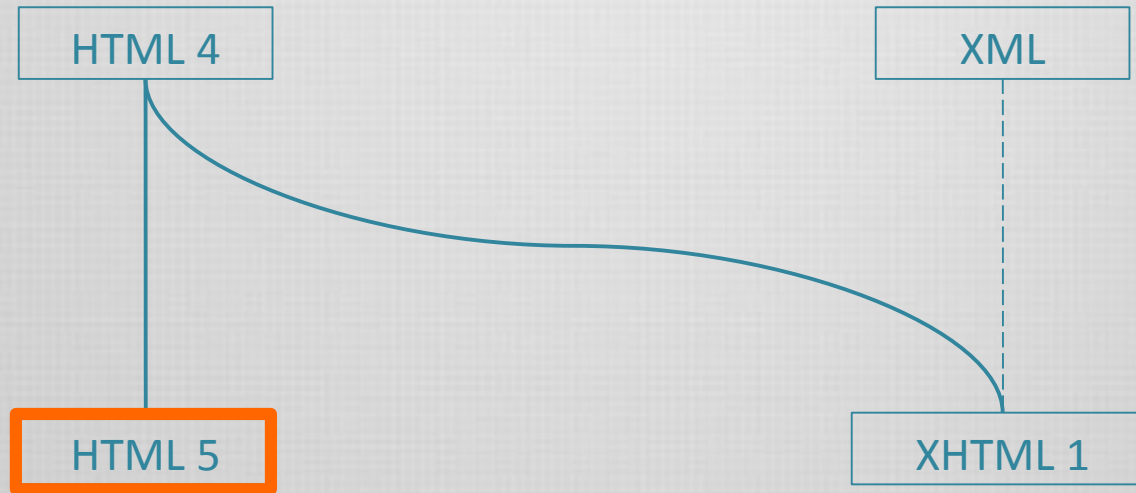
Facilita la integración con otros lenguajes tipo SGML, tales como MathML o SVG.

Al seguir reglas XML, es más estricto en su estructura que HTML.

Como es XML, requiere una definición adicional de los tags usados (DTD).

Es un estándar ampliamente usado.

Últimos estándares



HTML5 es una **evolución y actualización del HTML**.

*Han surgido muchas tecnologías desde la aparición de HTML4:
la web es mas multimedial e interactiva (Flash, Ajax, Web 2.0).*

Inicialmente no fue concebido por la W3C

Incluye algoritmos detallados para el manejo de errores (antes librado a los browsers)

Incluye nuevos elementos y atributos semánticos (como `<section>`).

Agrega soporte para audio y video.

Reduce la necesidad de plug-ins.

Tecnologías web

