UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

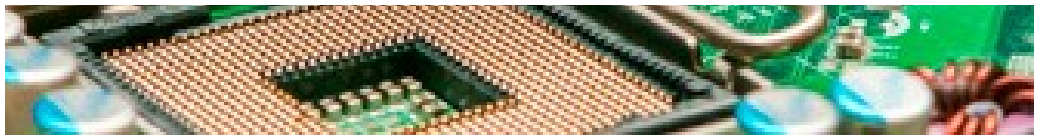
Máster Universitario en Ingeniería Informática

CyberSeguridad
Tema 4a: Tecnologías WEB

José Ismael Ripoll Ripoll

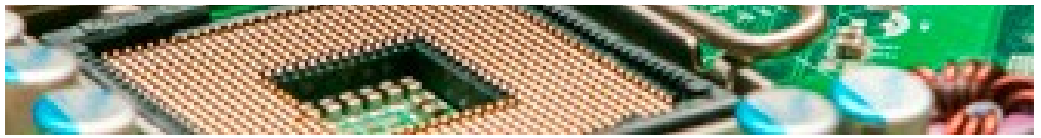
Máster Oficial Universitario en Ingeniería Informática
muiinf.webs.upv.es

MUIInf

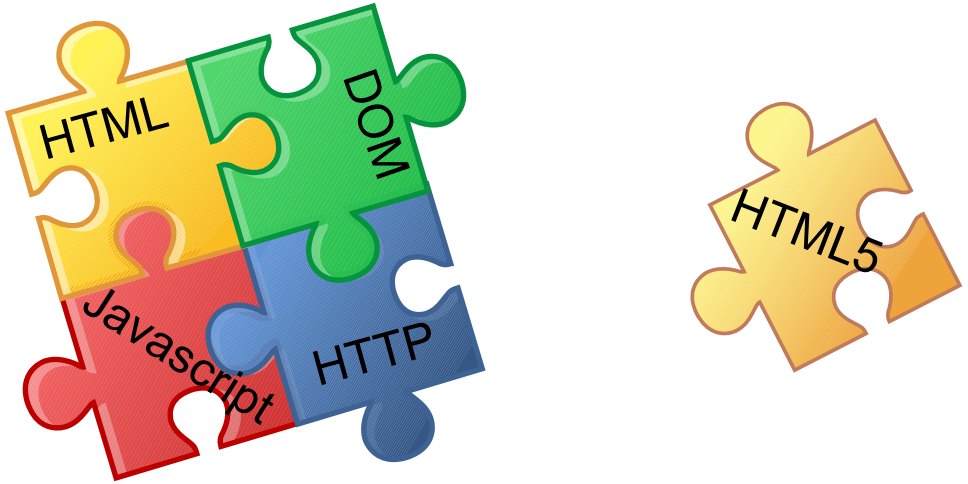


Contenidos

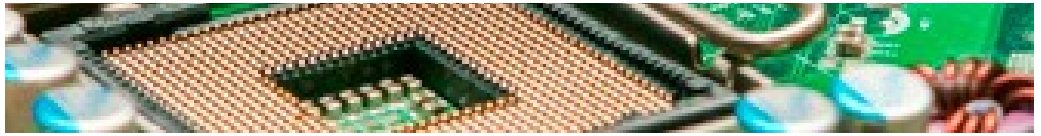
- Introducción
- HTTP: HyperText Trasport Protocol
- HTML y el DOM
  - HyperText Markup Language
  - Document Object Model
- Javascript
- SOP: Same Origin Policy



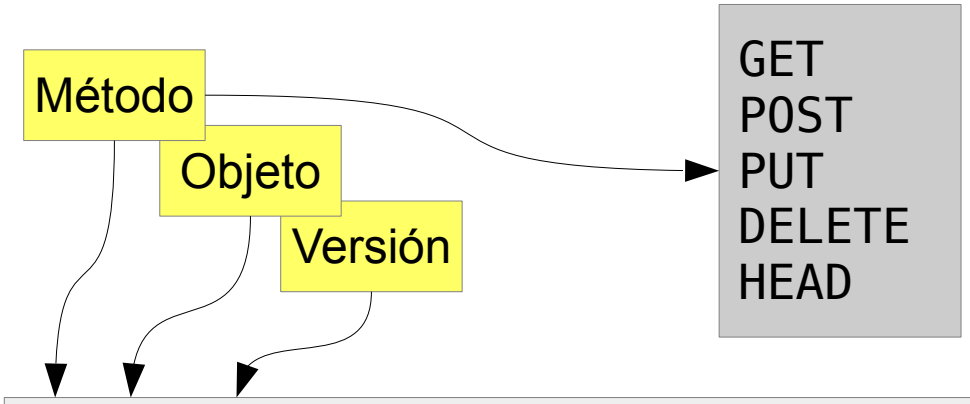
- "La WEB" es un popurri de tecnologías que se interrelacionan de formas poco claras en muchas ocasiones.
- Es un conjunto de estándares y prácticas en constante evolución.
  - En 2020 se empezó a activar por defecto el HTTP/3 en los navegadores.
- Hay que estar preparado para el caos y el lío de nombres



- Son el marco para transportar documentos WEB:
  - HTML, XML, JPEG, PNG, SVG, OGG, Etc.....
- Suele utilizarse sobre TCP pero también se puede sobre UDP.
- Es un protocolo de TEXTO plano.
- No está orientado a conexión.
  - Cada petición/respuesta debe ser autocontenida.
- Pensado para tener miles de clientes anónimos.
  - Pero ahora se usa para todo → problemas.

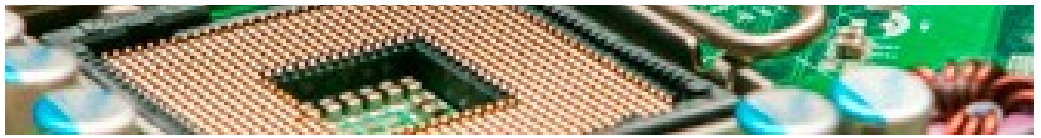


§ Aspecto de una **petición** HTTP

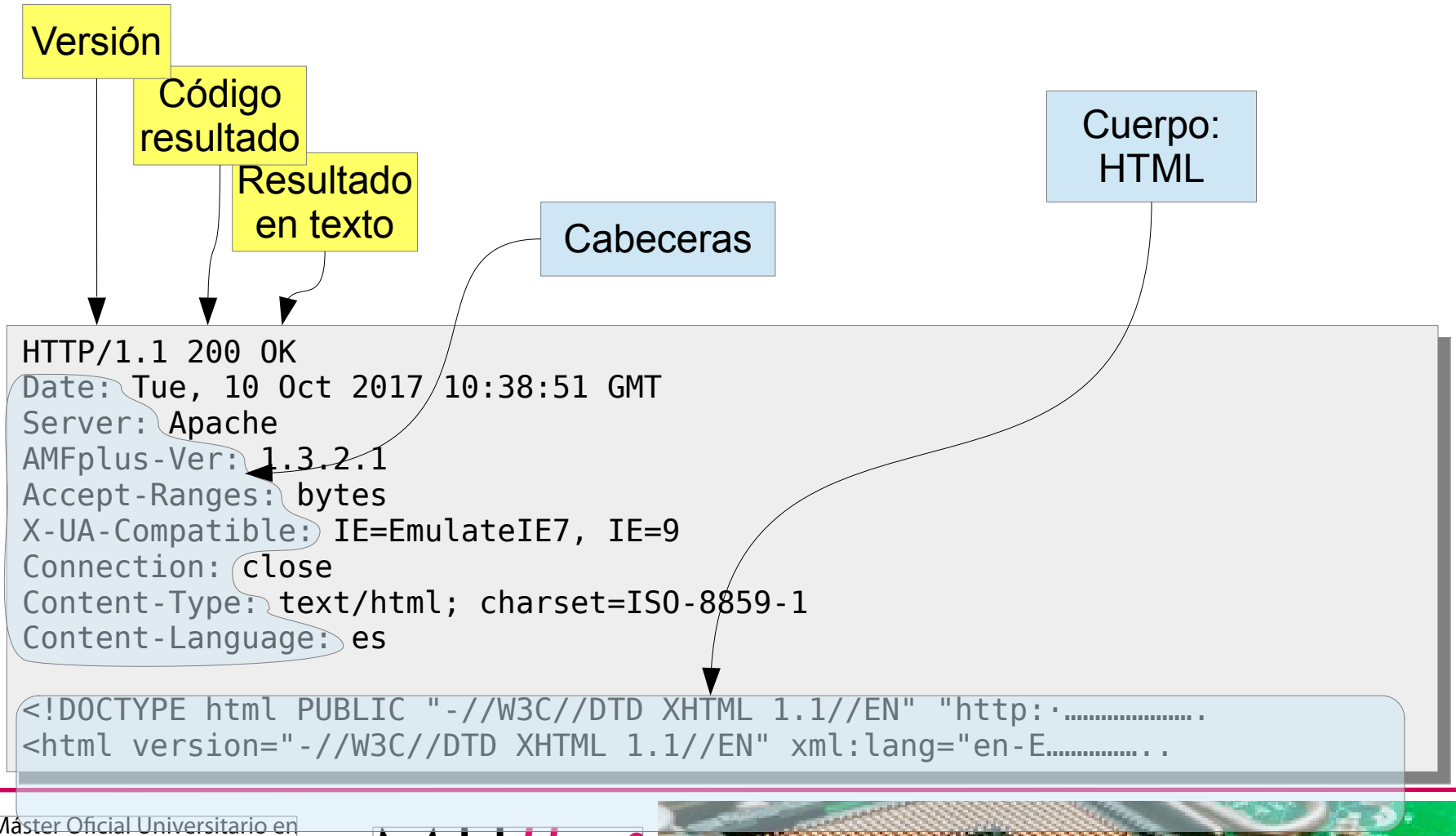



```
GET / HTTP/1.1
Host: localhost:9090
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Firefox/54.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Cabeceras

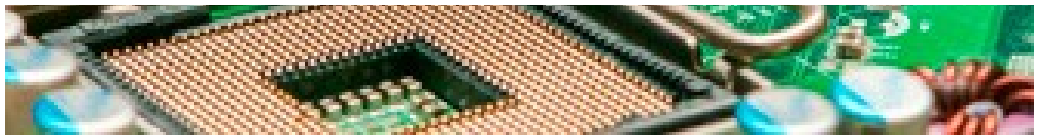


§ Aspecto de una **respuesta** HTTP



Cabeceras HTTP

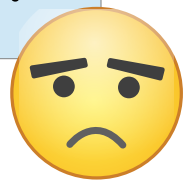
- Las cabeceras HTTP contienen variada información.
- Son un cajón de sastre para resolver problemas que se escapan al HTML.
- Por ejemplo las COOKIES.
  - El servidor puede añadir, en una respuesta, la cabecera "Set-Cookie:" con el valor que quiera.
  - En las siguientes peticiones, el cliente debe devolver en la cabecera "Cookie:" en valor de la última cookie recibida.
  - El navegador debe guardar todas las cookies para la próxima vez que se comunique con ese servidor.
  - La gestión de las cookies suele ser automática. Las recoge el navegador y las vuelve a insertar en la cabecera HTTP de forma transparente.

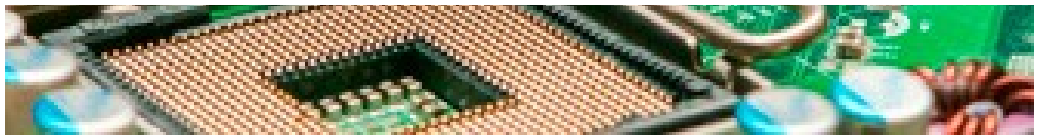


Cabeceras HTTP

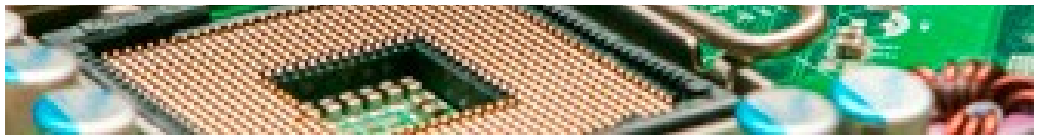
- Accept:, Accept-Charset:, Accept-Encoding:, Accept-Language:, User-Agent:, Server:, Allow:
- Content-Type:, Content-Length:, Content-Range, Content-Encoding, Content-Language, Content-Location:, Location:, Referer:
- Date:, If-Modified-Since:, If-Unmodified-Since:, If-Match:, If-None-Match:, If-Range:, Expires:, Last-Modified:, Cache-Control:, Via:, Pragma:, Etag:, Age:, Retry-After:
- Set-Cookie:, Cookie:
- Authorization:, WW-Authenticate:
- Host:, Connection: etc…

Faltan las más interesantes!



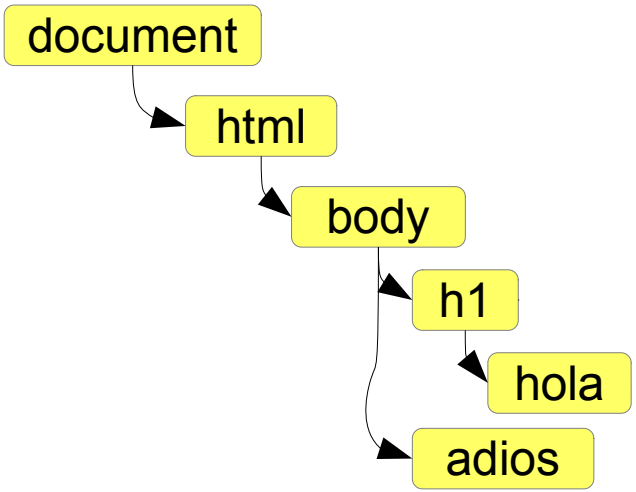


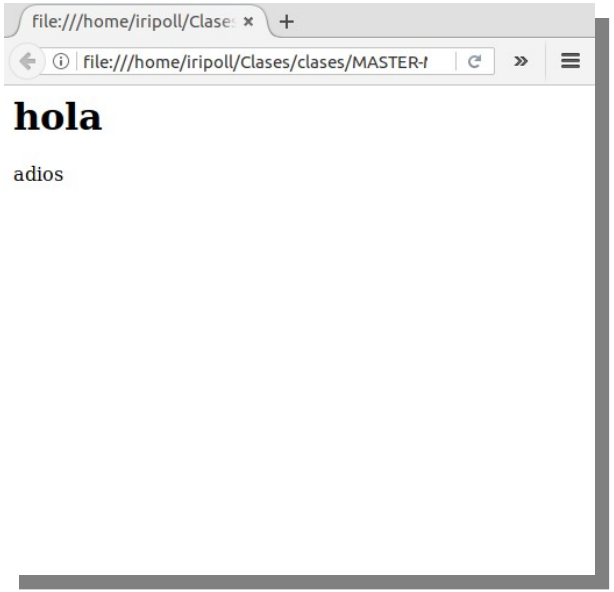
- El navegador descarga e "interpreta" una página web.
- La página está escrita en HTML y puede incluir otros tipos de ficheros: SVG, figuras, vídeos, etc., y código que se ejecutará (Javascript).
- JavaScript != JAVA
- **Javascript puede interactuar** con el contenido de la página.
- El API que posibilita la interacción javascript HTML es el DOM.
- DOM es una forma de representar y nombrar la información del HTML de forma que se puede usar desde Javascript.

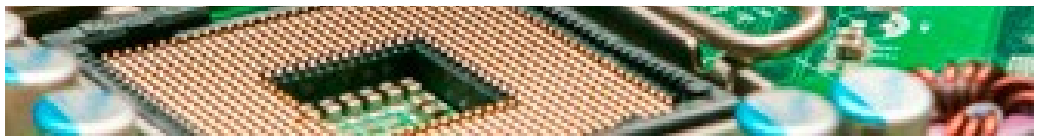


- El navegador crear un "**árbol**" con los elementos de la paǵina cuanto esta se carga.
- Una vez cargada, el javascript puede leer y modificar el árbol DOM.
- Un cambio en la estructura del DOM se ve reflejada en cómo se visualiza la página.

```
<html>
  <body>
    <h1>hola</h1>
   adios
  </body>
</html>
```

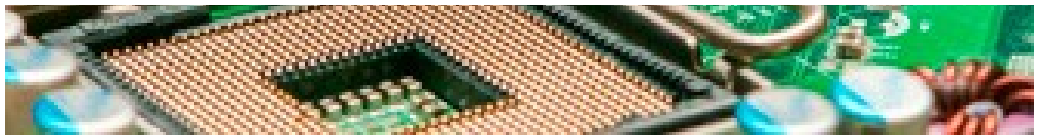




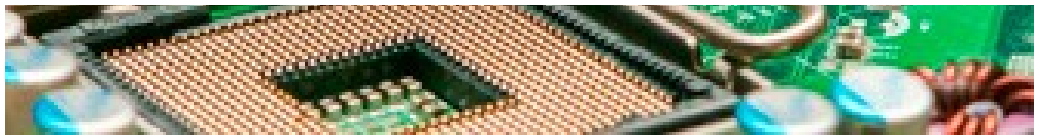


- Para poder interpretar un HTML como un árbol es necesario que los documentos HTML estén "**bien formados**".
- Debe ser un documento XML válido (o casi)
  - Los elementos deben estar cerrados y balanceados: `<p><em>hola</em></p>`
  - O elementos especiales: `<br/>`
- ¿Qué hace el parser de HTML con los documentos incorrectos?
  - Lo que buenamente puede.

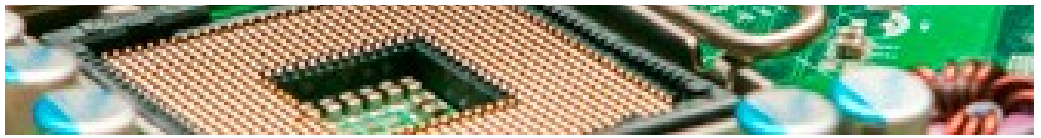
```
<div> <b>hola</div></b>
<img src="><b></im></b>
```



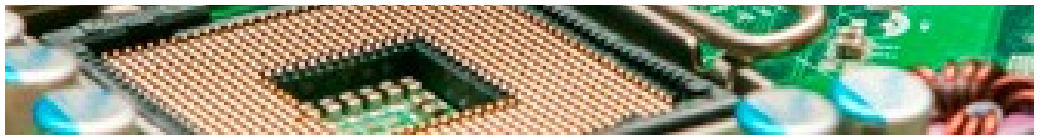
- Desde Javascript podemos:
  - Añadir elementos al DOM.
  - Consultar valores del DOM.
  - Reaccionar a eventos: clics, mover el ratón, pulsar una tecla,…
  - También podemos realizar conexiones a otras máquinas.
  - O simplemente, ejecutar un programa (minar bitcoins)
- Entonces ¿alguien que ejecute javascript en nuestro nevegador puede leer lo que estamos viendo en nuestra pantalla?
  - Depende.



- ¿Quién define qué es el DOM?
- Pues todos y nadie:
  - Miscrosoft, Firefox, Google Chrome, Safary, Opera, Fundación Apache….
- Cada vez más los DOMs tienden a converger al haber menos motores de html (Edge se ha pasado al motor desarrollado por google para Chrome)
- El desarrollo de los parsers HTML no sigue un ciclo de desarrollo muy sano.
- Están en constante actualización y crecimiento.



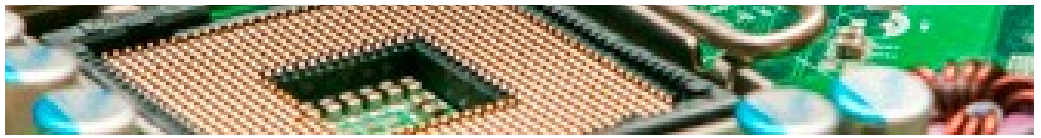
No se puede vivir sin conocer Javascript y HTML



Limitaciones de Javascript

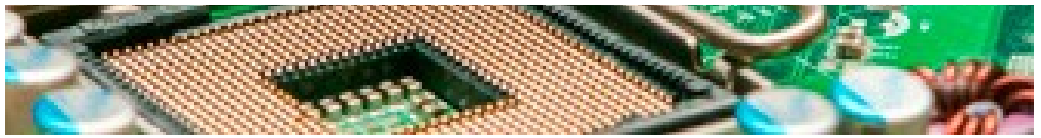
- ¿Cualquiera (anuncios embebidos en páginas web) puede ejecutar código en nuestro ordenador?
  - **SI** (no se se puede decir más alto, se sale).

- Entonces ¿puedo navegar seguro?
  - Durante mucho tiempo, hubo gente que desactivaba java y javascript (yo, por ejemplo). Hoy ya no es posible.
  - Otra solución fueron los AD-BLOCKERS.
  - Se han creado mecanismos de protección para evitar los abusos.
  - En cualquier caso: NO, no podemos estar tranquilos.
- O se limita la potencia de Javascript, o los beneficios no compensarán los riesgos.

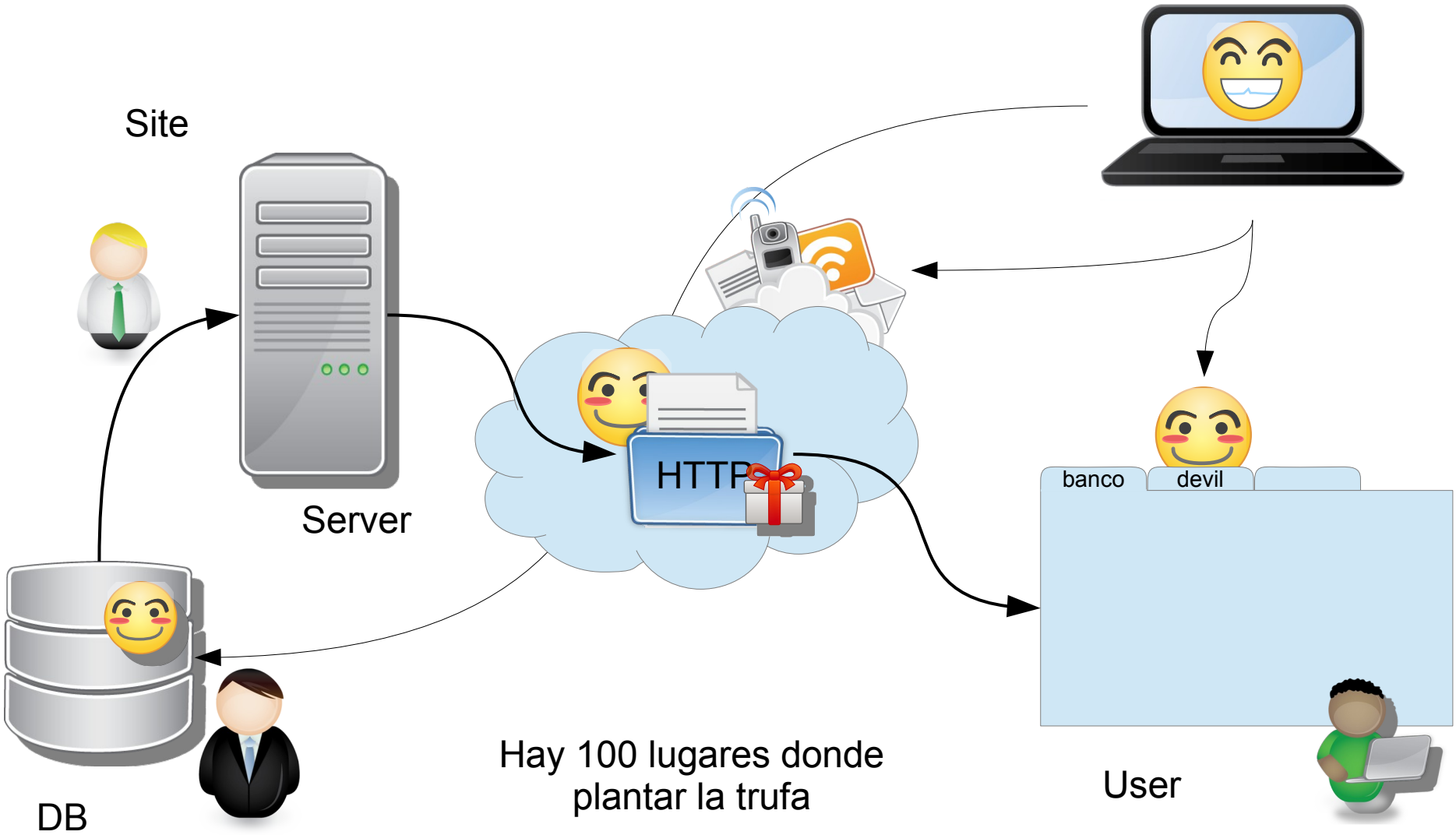


Usos maliciosos de Javascript

- Escenario de ataque:
  - El usuario está "logeado" en su servicio de webmail para consultar el correo (en una de la pestañas del navegador).
  - En otra pestaña visita un servidor malicioso.
  - Desde el sitio malicioso se le envía un javascript que:
    - lee el DOM de las otras pestañas,
    - lee lo que el usuario pulsa,
    - lee las cookies,
    - envía esta información a su CC (Command and Control).
- Por suerte, desde 1995 estos ataques ya no son posibles.
  - Same Origin Policy (SOP).



Actores


Site
Server
HTTP
banco
devil
DB
User
Hay 100 lugares donde
plantar la trufa