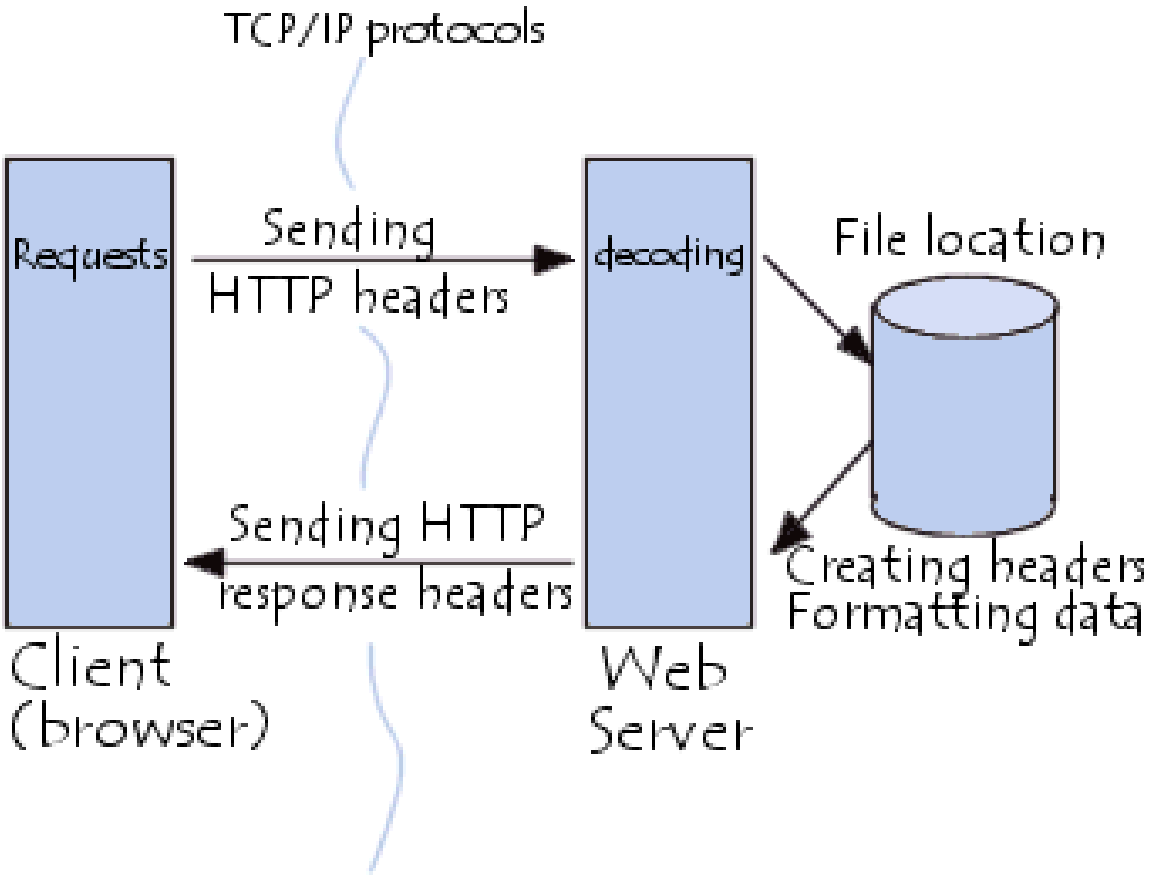
El Protocolo HTTP



Contenidos

1. El protocolo HTTP.
    1. Orígenes y evolución.
    2. Funcionamiento.
        1. Cabecera HTTP Request.
        2. Cabecera HTTP Response.
    3. Codificación de los datos.
2. HTTP y web.
    1. Aspectos básicos.
    2. Peticiones asíncronas (Ajax, API Fetch).

El protocolo HTTP

- **Orígenes y evolución**.
  - La primera versión del protocolo y el primer servidor web fue creado por Tim Berners Lee en el CERN. Solo disponía del método GET (versión 0.9).
  - HTTP/1 de 1996 en el RFC 1945. Se amplía el conjunto de métodos (GET, POST, HEAD), aparecen las primeras cabeceras y se permite la transferencia de cualquier tipo de archivo, no solo HTML.
  - HTTP/1.1 de 1999 en el RFC 2616.
  - HTTP/1.1 revisado en 2014 RFCs (7230-7237).
    - RFC7230: HTTP/1.1 Message Syntax and Routing.
    - RFC7231: HTTP/1.1 Semantics and Content.
    - RFC7232: HTTP/1.1 Conditional Request.
    - RFC7233: HTTP/1.1 Range Request.
    - RFC7234: HTTP/1.1 Caching.
    - RFC7235: HTTP/1.1 Authentication.
    - RFC7236: Initial HTTP Authentication Scheme Registrations.
    - RFC7237: Initial HTTP Method Registration.

El protocolo HTTP

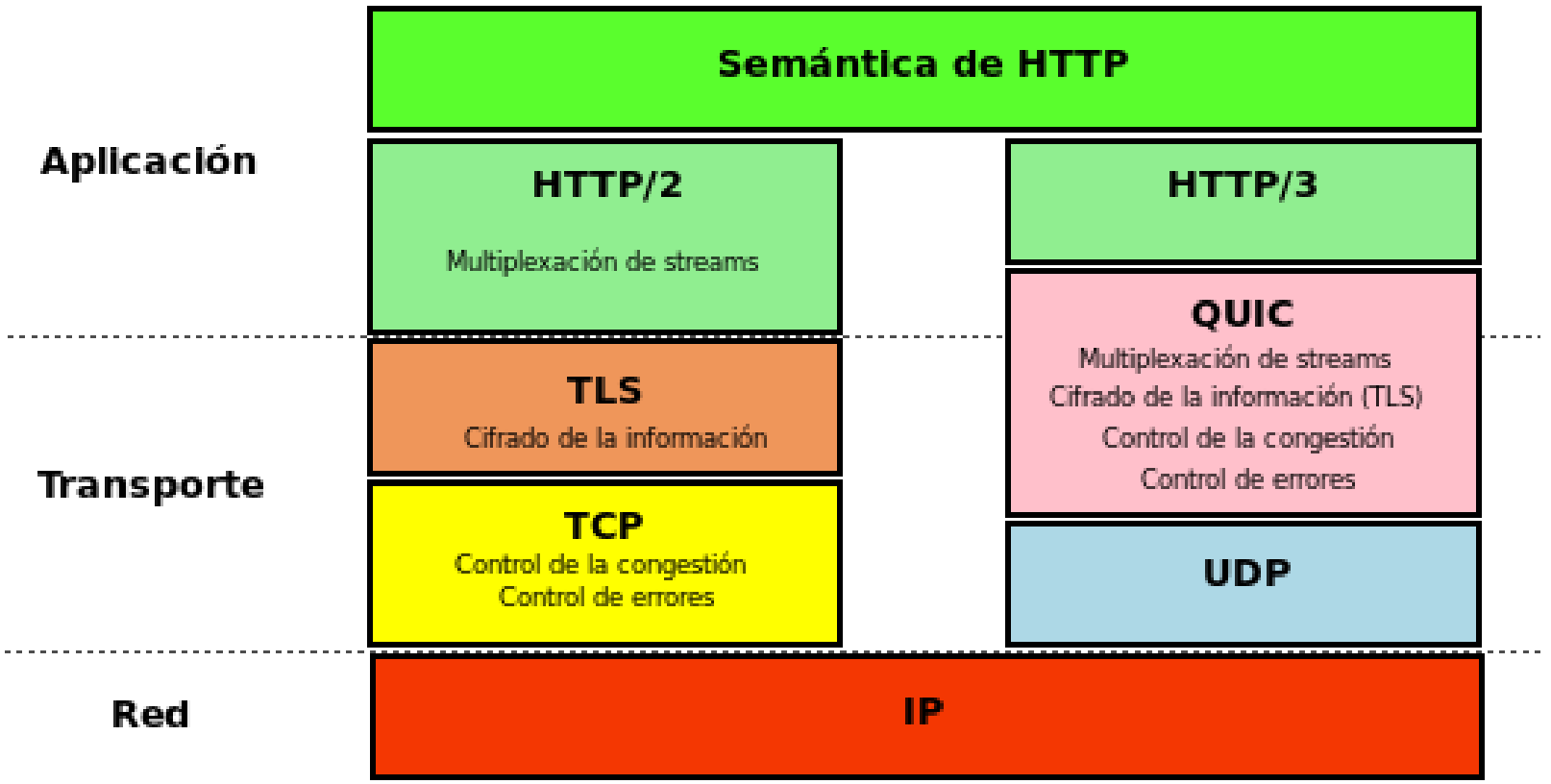
- **Orígenes y evolución**.
  - **HTTP/2** es una derivación del protocolo **SPDY** (*speedy*) desarrollado por Google para solventar los problemas de rendimiento de HTTP/1.1.
    - Definido en 2015 en el RFC 7540 y revisado en 2020 en el RFC 8740.
    - Transmisión en binario en lugar de en texto plano.
    - Compresión de las cabeceras mediante el algoritmo HPACK.
    - Multiplexación de las conexiones como sustituto del *pipelining* de HTTP/1.1 (se verá más adelante).
    - Mecanismo ***Server Push*** que permite al servidor enviar al cliente contenido relacionado con la petición sin que este lo solicite.
    - Problemas de congestión en la capa de transporte: **TCP HoL Blocking**.

El protocolo HTTP

- **Orígenes y evolución**.
  - **HTTP/3** desde febrero de 2021 es un Internet Draft y en 2022 el IETF lo propuso como estándar en el RFC 9114.
    - Usa **QUIC** (https://quicwg.org/), un protocolo de la capa de transporte desarrollado por Google que trabaja sobre UDP. Definido en el RFC 9000.
    - Viene a solucionar el problema de HTTP/2 (*head-of-line blocking*) que supone que la pérdida de un paquete en una comunicación cause problemas y retrasos en todas las transacciones activas.
    - TLS está integrado en el protocolo por lo que el cifrado de extremo a extremo ya no es opcional.

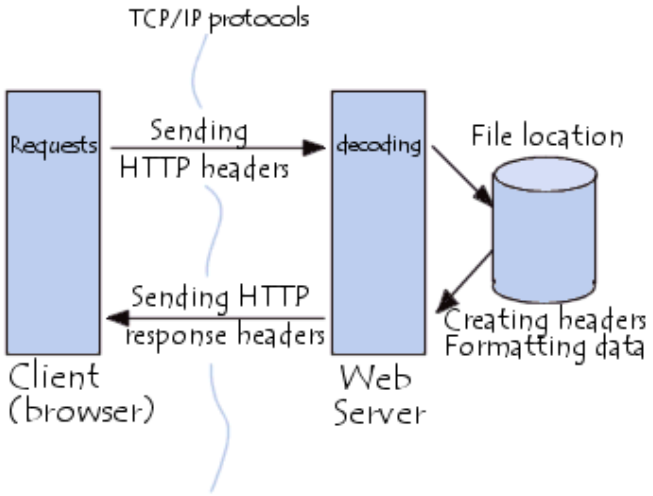
El protocolo HTTP

- **Orígenes y evolución.**
  - Comparativa HTTP/2 y HTTP/3



El protocolo HTTP

- **Funcionamiento**.
  - **HTTP Request**. Petición que envía el cliente al servidor.
  - **HTTP Response**. El servidor procesa la petición y envía la respuesta al cliente.



https://ccm.net/contents/273-the-http-protocol

El protocolo HTTP

- **Funcionamiento**.
  - **HTTP Request**. Petición que envía el cliente (*User-Agent*) al servidor. Consta de:
    - **Petición**. Especifica el método o comando, la URL y la versión del protocolo empleada.
    - **Cabeceras**. Información adicional de la petición (navegador, sistema operativo, …).
    - **Cuerpo**. Información opcional, por ejemplo, para enviar información vía POST. Debe estar separada por una línea en blanco de la información anterior.

```
METHOD URL VERSION
HEADER: Value
HEADER: Value
Empty line
BODY OF THE REQUEST
```

```
GET / HTTP/1.1
Host: www.ieszaidinvergeles.org
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:85.0)
Gecko/20100101 Firefox/85.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,
*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

El protocolo HTTP

- **Funcionamiento**.
  - **Petición HTTP Request**. Especifica el método o comando, la URL y la versión del protocolo empleada.
    - **GET**. Solicita el recurso en la URL especificada.
    - **HEAD**. Solicita la cabecera del recurso en la URL especificada.
    - **POST**. Envía información al programa en la URL especificada.
    - **PUT**. Reemplaza el recurso objetivo con la información suministrada
    - **DELETE**. Borra el recurso en la URL especificada.
    - **PATCH**. Aplica modificaciones parciales al recurso.
    - **OPTIONS**. Describe las opciones de comunicación.
    - **TRACE**. Realiza una comunicación de prueba.
    - **CONNECT**. Establece un túnel con el servidor identificado por el recurso solicitado.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

El protocolo HTTP

- **Funcionamiento**.
  - **Cabeceras HTTP Request**. Información adicional de la petición (navegador, sistema operativo, …).
    - **Accept**. Tipo de contenido MIME aceptado (Ej. text/html).
    - **Accept-Charset**. Conjunto de caracteres esperado.
    - **Accept-Encoding**. Codificación aceptada por el navegador.
    - **Accept-Language**. Idioma esperado.
    - **Authorization**. Identificación del navegador en el servidor.
    - **Content-Encoding**. Codificación del cuerpo de la petición.
    - **Content-Language**. Idioma en el cuerpo de la petición.
    - **Content-Length**. Longitud del cuerpo de la petición.
    - **Content-Type**. Tipo de contenido MIME del cuerpo de la petición.
    - **Date**. Fecha en que comienza la transferencia.
    - **Forwarded**. Empleado por máquinas entre el navegador y el servidor.
    - **From**. Permite especificar el email del cliente.
    - **Orig-URL**. URL desde donde se origina la petición.
    - **Referer**. URL desde la cual se ha hecho la petición.
    - **User-agent**. Información sobre el cliente.

El protocolo HTTP

- **Funcionamiento**.
  - **Parámetros**. Complementa la ruta al recurso solicitado por el método GET.
    - El carácter ***?*** indica el comienzo de los parámetros.
    - Cada parámetro es una pareja ***clave=valor***. En el ejemplo hay tres: ***q=hacking***, ***va=b*** y ***t=hc***.
    - El carácter ***&*** es el separador de cada parámetro.

```
GET /?q=hacking&va=b&t=hc HTTP/2
Host: duckduckgo.com
User-Agent: Mozilla/5.0
```

El protocolo HTTP

- **Funcionamiento**.
  - **HTTP Response**. Respuesta del servidor al cliente. Consta de:
    - **Línea de estado**. Contiene tres elementos: versión del protocolo, código de estado y significado del código.
    - **Cabeceras**. Información adicional de la respuesta (similar a la petición HTTP Request).
    - **Cuerpo**. Contiene el recurso solicitado.

```
VERSION-HTTP CODE EXPLANATION
HEADER: Value
HEADER: Value
Empty line
BODY OF THE RESPONSE
```

```
HTTP/1.0 200 OK
Date: Sat, 15 Jan 2000 14:37:12 GMT
Server: Microsoft-IIS/2.0
Content-Type: text/HTML
Content-Length: 1245
Last-Modified: Fri, 14 Jan 2000 08:25:13 GMT
```

El protocolo HTTP

- **Funcionamiento**.
  - **Códigos de Estado HTTP Response**.
    - **1XX**. Mensajes de Información.
      - *100 Continue*. El servidor ha recibido las cabeceras de la petición y el cliente debe proceder a enviar el cuerpo de la petición.
      - 101 Switching Protocols. El cliente ha solicitado al servidor cambiar protocolos.
    - **2XX**. Mensajes de éxito.
      - 200 OK. La petición fue existosa.
      - 201 Created. La petición se ha completado y se ha creado un nuevo recurso.
      - 202 Accepted. La petición se ha aceptado para procesarla, pero no se ha completado aún.
    - **3XX**. Mensajes de Redirección.
      - 300 Multiple Choices. Una lista de posibles recursos que coinciden con el solicitado.
    - **4XX**. Mensajes de error del cliente.
      - 400 Bad Request. Hay errores de sintaxis en la petición.
      - 403 Forbidden. El servidor se niega a responder a la petición.
      - 404 Not Found. No se encuentra el recurso solicitado en la petición.
    - **5XX**. Mensajes de error del servidor.
      - 500 Internal Server Error. Un mensaje de error genérico por fallo del servidor.

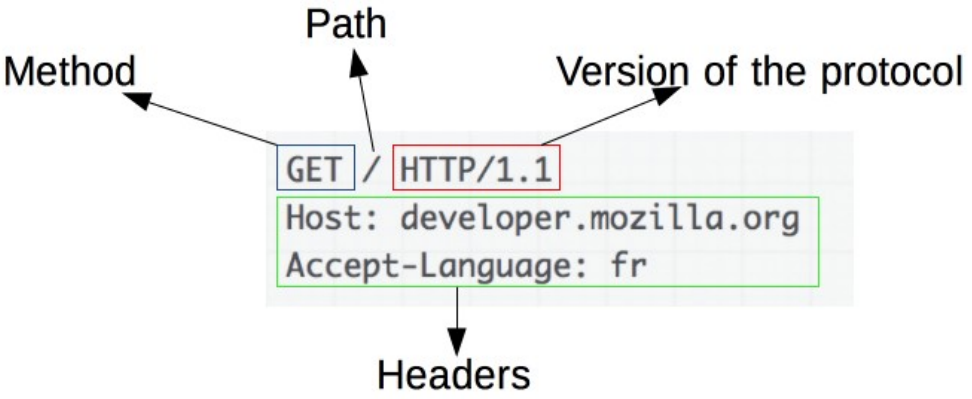
https://www.w3schools.com/tags/ref_httpmessages.asp

El protocolo HTTP

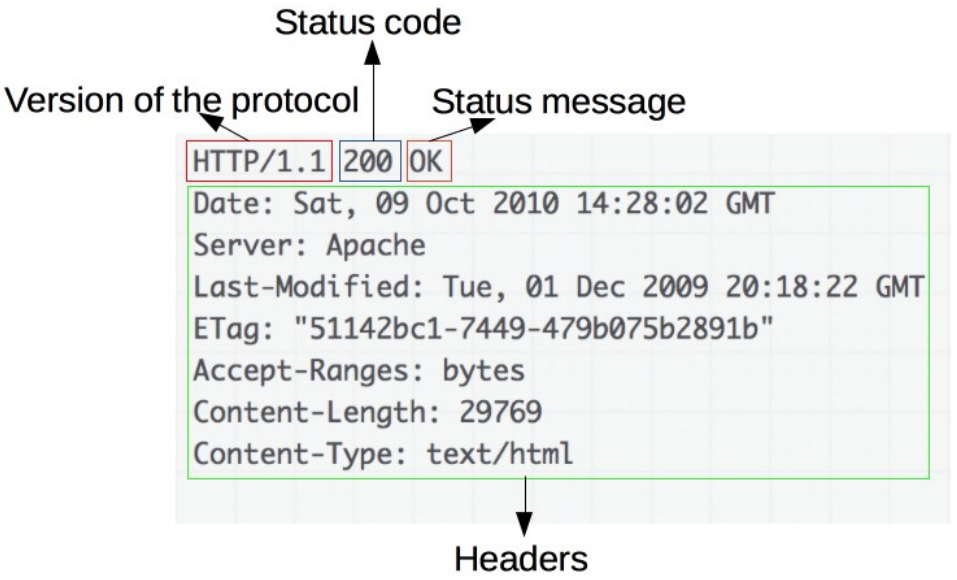
- **Funcionamiento**.
  - **Cabeceras HTTP Response**. Información adicional de la respuesta.
    - **Content-Encoding**. Codificación del cuerpo de la petición.
    - **Content-Language**. Idioma en el cuerpo de la petición.
    - **Content-Length**. Longitud del cuerpo de la petición.
    - **Content-Type**. Tipo de contenido MIME del cuerpo de la petición.
    - **Date**. Fecha en que comienza la transferencia.
    - **Expires**. Fecha de validez de los datos de la respuesta.
    - **Forwarded**. Empleado por máquinas entre el navegador y el servidor.
    - **Location**. Redirige a una nueva URL asociada con el documento.
    - **Server**. Características del servidor que ha realizado la respuesta.

El protocolo HTTP

HTTP REQUEST



HTTP RESPONSE



https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

El protocolo HTTP

- **Codificación de los datos**.
  - Para transmitir caracteres especiales en las peticiones es necesario codificarlos.
  - **URL Encoding**. La codificación de cualquier carácter se realiza con el carácter ***%*** seguido del valor ASCII hexadecimal del carácter a codificar. Ej: Búsqueda en DuckDuckGo de ***M&M's***.

https://duckduckgo.com/?q=**m%26m%27s**&t=ffab&ia=web

Carácter	ASCII Decimal	ASCII HEX	Valor codificado
espacio	32	20	**%20**
#	35	23	**%23**
%	37	25	**%25**
&	38	26	**%26**
?	63	3F	**%3F**

El protocolo HTTP

- **Codificación de los datos**.
  - **HTML Encoding**. Los datos mostrados en HTML deben ser correctamente codificados para que no sean interpretados por el navegador.
    - Usando **entidades HTML**. ***<lt*** (<), ***>gt*** (>), ***&amp*** (&), ***&ntilde*** (ñ).
    - Usando su **valor ASCII decimal**. ***&#D*** donde ***D*** es el código ASCII decimal del carácter a codificar.
    - Usando su **valor ASCII hexadecimal**. ***&#xH*** donde ***H*** es el código ASCII hexadecimal del carácter a codificar.
    - El carácter ***;*** se usa para indicar la finalización del código.

La etiqueta
 se usa en HTML para insertar un salto de línea.

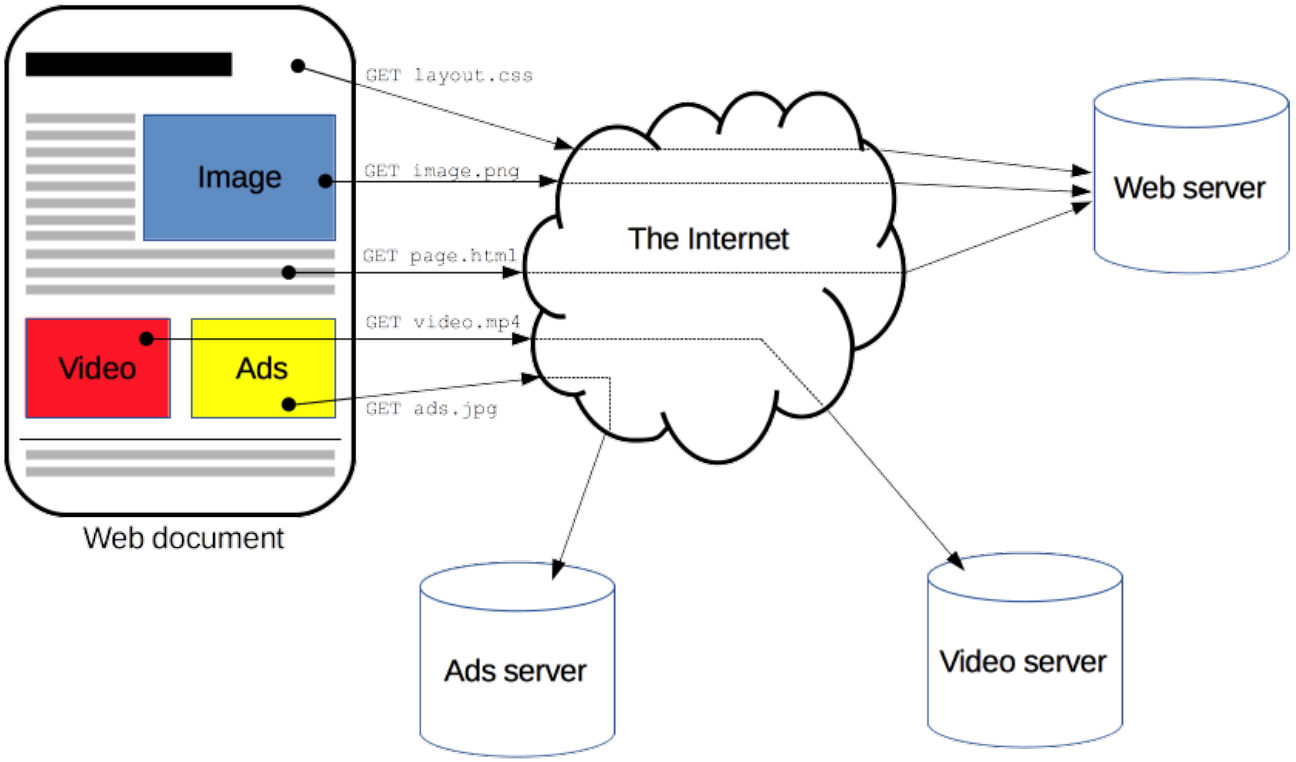
La etiqueta
 se usa en HTML para insertar un salto de línea.

La etiqueta
 se usa en HTML para insertar un salto de línea.

La etiqueta
 se usa en HTML para insertar un salto de línea.

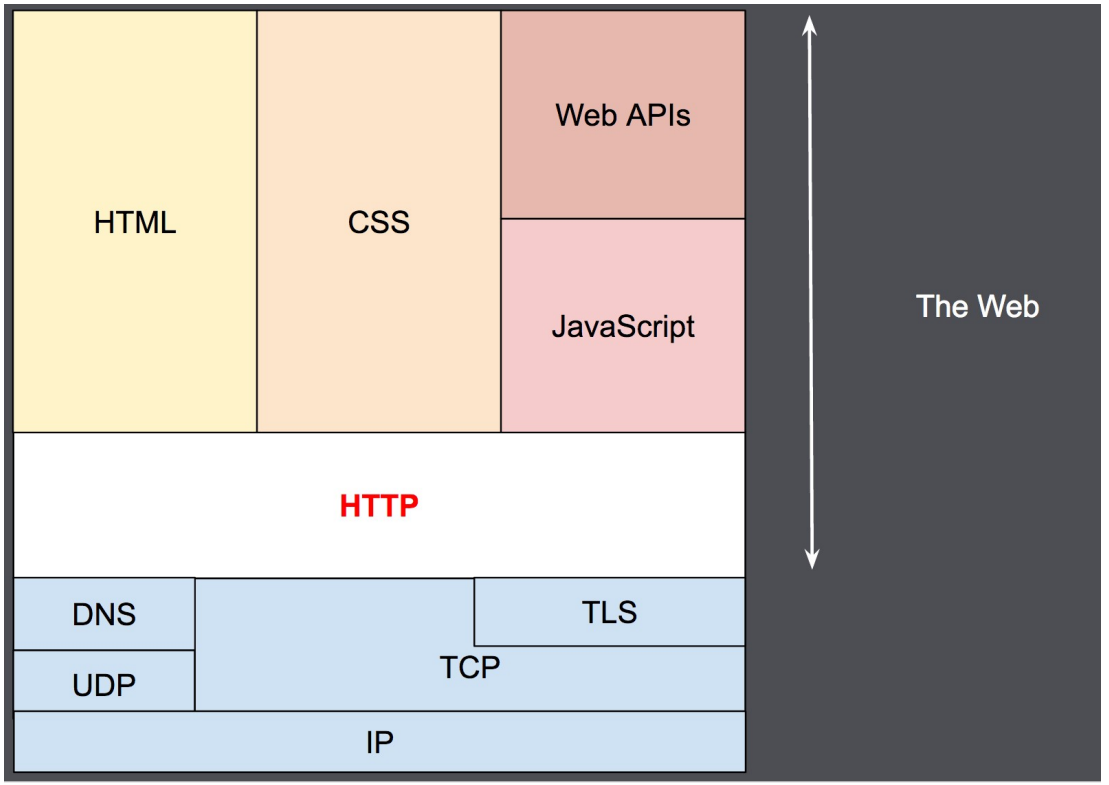
HTTP/1.1 y Web

- El protocolo HTTP se utiliza para construir los documentos web a base de peticiones individuales a los documentos HTML y los recursos incluidos en los mismos.



HTTP/1.1 y Web

- Se emplea en conjunción con el protocolo de la capa de transporte TCP o en conjunción con algún método de encriptado como TLS lo que da lugar a HTTPS.



HTTP/1.1 y Web

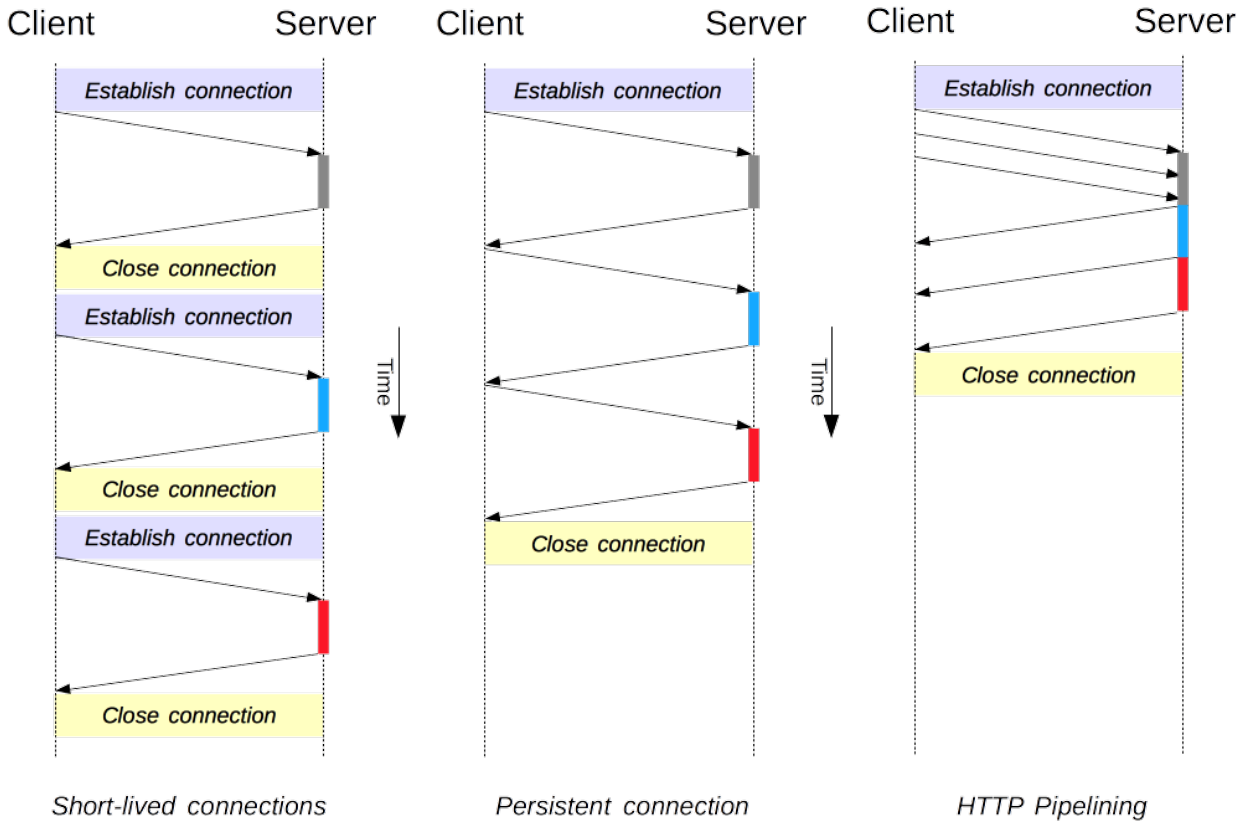
- **Aspectos básicos**:
  - **Simple**. Está diseñado en un estilo sencillo y legible por los humanos lo que facilita la depuración y las pruebas.
  - **Extensible**. Se puede añadir nueva funcionalidad gracias a la introducción de ***HTTP Headers*** en la versión 1.0.
  - **Sin estado**. Es un protocolo sin estado por lo que cliente y servidor no guardan estados de comunicaciones previas. La introducción de las ***cookies*** y las cabeceras permiten el uso de **sesiones**.
  - **Confiable**. HTTP requiere de un protocolo confiable en la entrega de los mensajes, por ello se emplea TCP en lugar de UDP.

HTTP/1.1 y Web

- **Aspectos básicos**:
  - El comportamiento por defecto en la versión 1.0 es abrir una conexión TCP separada para cada pareja *HTTP request/response,* lo que es menos eficiente que compartir una única conexión.
  - La versión 1.1 introdujo las ***peticiones pipelining*** (difíciles de implementar) y las ***conexiones persistentes*** (a través de la cabecera '***Connection: Keep-Alive'***). Problema: HTTP HoL Blocking (*Head of Line Blocking*).
  - La versión 2 introdujo la multiplexación de mensajes sobre la misma conexión mejorando la eficiencia. Problema: TCP HoL Blocking.
  - En la versión 3, se utiliza un nuevo protocolo en la capa de transporte que funciona sobre UDP: **QUIC**.

HTTP/1.1 y Web

- Este modelo no se cumple en conexiones tipo *Ajax* (*XMLHTTPRequest*), *WebSockets* o similares.



https://developer.mozilla.org/en-US/docs/Web/HTTP/Connection_management_in_HTTP_1.x

FIN