

UD 1

Selección de arquitecturas y herramientas de programación

DESARROLLO WEB EN ENTORNO CLIENTE

Técnico de Grado Superior Desarrollo de Aplicaciones Web

2024-25



J. Mario Rodríguez
jrodper183e@g.educaand.es

Contenidos

- Modelos de programación en entornos cliente/servidor.
- Mecanismos de ejecución de código en un navegador Web.
- Capacidades y limitaciones de ejecución.
Compatibilidad con navegadores Web.
- Lenguajes de programación en entorno cliente.
- Características de los lenguajes de script.
Ventajas y desventajas sobre la programación tradicional.
- Tecnologías y lenguajes asociados.
- Integración del código con las etiquetas HTML.
- Herramientas de programación.

Resultados de aprendizaje

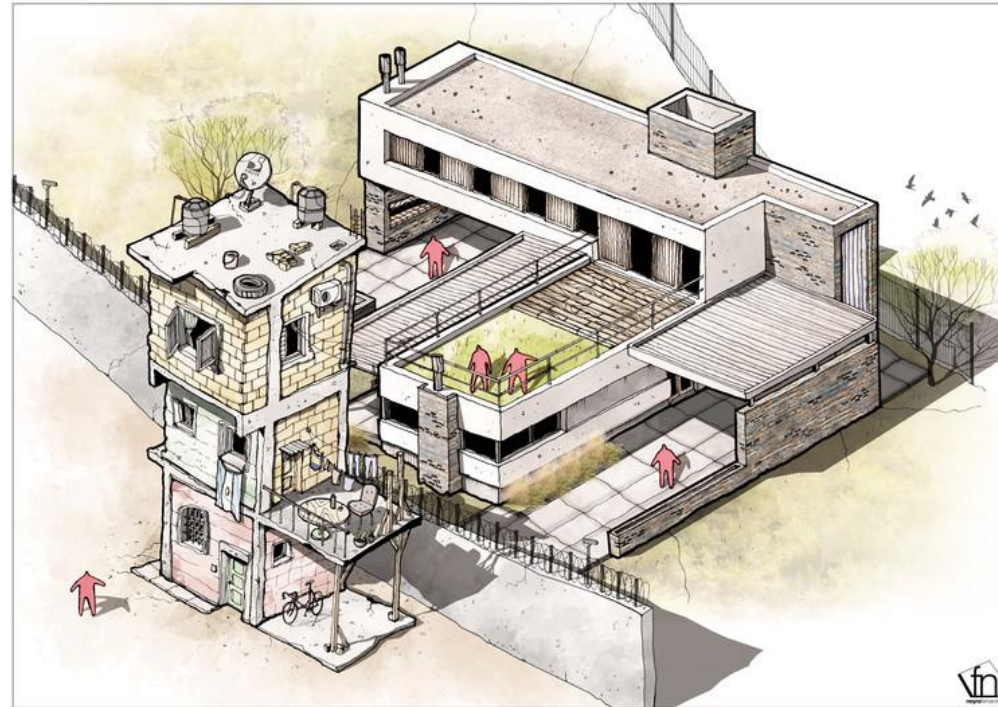
1. Selecciona las arquitecturas y tecnologías de programación sobre clientes Web, identificando y analizando las capacidades y características de cada una.
Criterios de evaluación:
 - a) Se han caracterizado y diferenciado los modelos de ejecución de código en el servidor y en el cliente Web.
 - b) Se han identificado las capacidades y mecanismos de ejecución de código de los navegadores Web.
 - c) Se han identificado y caracterizado los principales lenguajes relacionados con la programación de clientes Web.
 - d) Se han reconocido las particularidades de la programación de guiones y sus ventajas y desventajas sobre la programación tradicional.
 - e) Se han verificado los mecanismos de integración de los lenguajes de marcas con los lenguajes de programación de clientes Web.
 - f) Se han reconocido y evaluado las herramientas de programación sobre clientes Web.

Modelos de programación en entornos cliente/servidor

Se habla de **arquitectura cliente/servidor**

¿Qué es “arquitectura”?

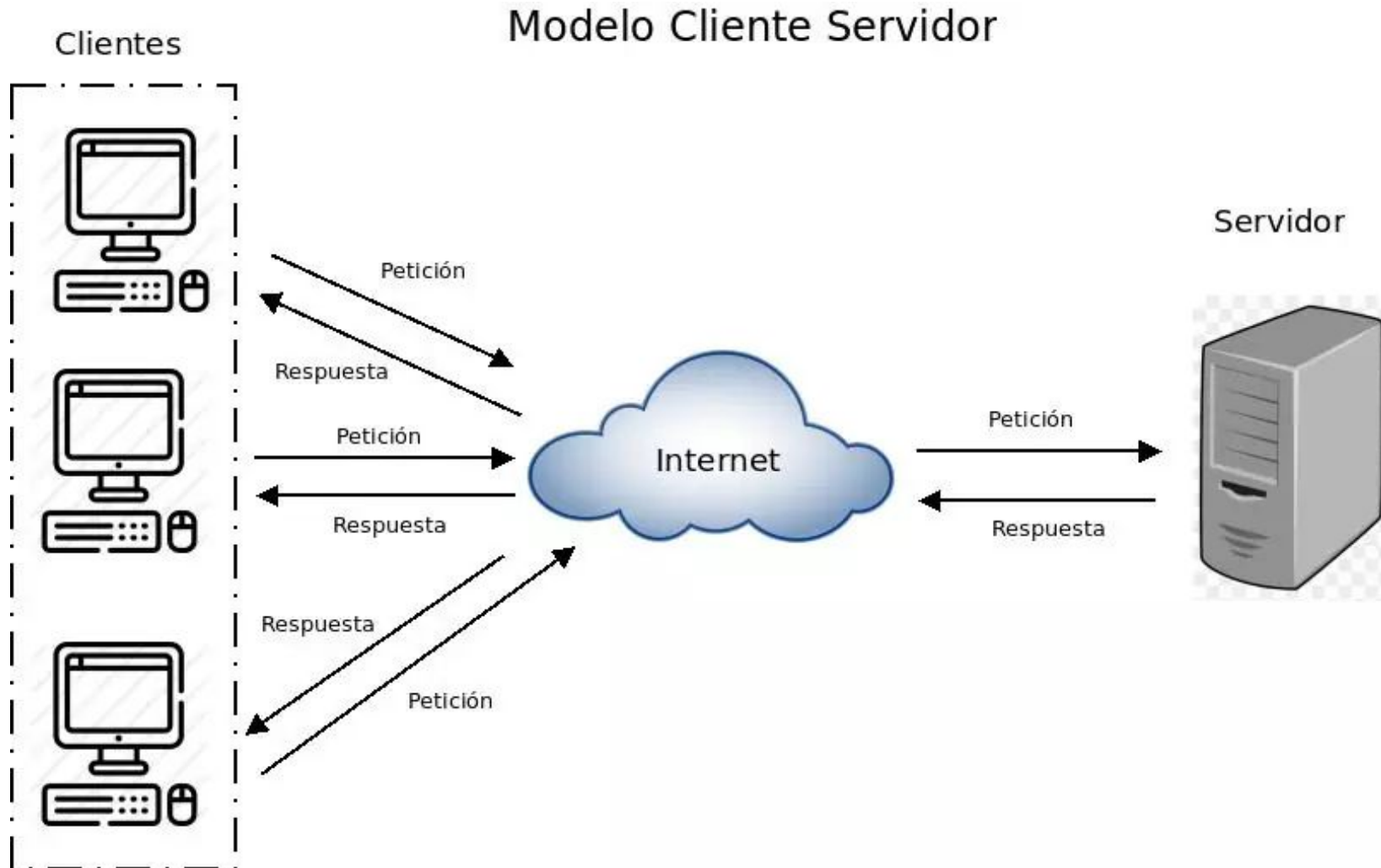
La arquitectura describe cómo se organizan y cómo se comunican los dispositivos, componentes y aplicaciones



es una de las principales usadas en muchísimos servicios y protocolos de Internet.

¿es la única?
¿cuáles más conocéis?

Modelos de programación en entornos cliente/servidor



COMPONENTES

- Red
- Cliente
- Servidor
- Protocolo
- Servicios
- Base de datos

Modelos de programación en entornos cliente/servidor

CLIENTE

- El cliente es el encargado de facilitar la interacción con el usuario, es decir, la interfaz gráfica.
- Es la puerta de entrada a todo el sistema.
- Es el sistema que solicita servicios o datos de un servidor.



SERVIDOR

- El servidor es el sistema que contiene los datos o recursos, los cuales proporcionar a otros sistemas de la red.
- Requiere del uso de lenguajes de programación complejos.
- Es la arquitectura interna y el que determina el funcionamiento global del sistema.

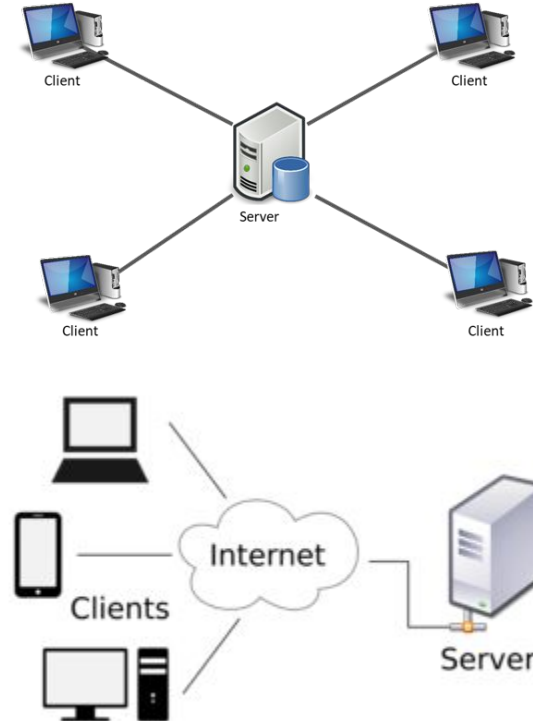
CLIENTE/SERVIDOR = FRONTEND/BACKEND

Modelos de programación en entornos cliente/servidor

Cliente y servidor comparten recursos.

Es una **arquitectura distribuida** debido a que el servidor y el cliente se encuentran en distintos equipos (aunque podrían estar en la misma máquina) y se comunican por medio de la red o Internet.


- Son desarrollados como dos aplicaciones diferentes de forma independiente, pudiendo ser construidas en distintas tecnologías, respetando el mismo **protocolo** de comunicación.



Servidor = backend
Cliente = frontend
Cliente y servidor = fullstack

Modelos de programación en entornos cliente/servidor

¿POR QUÉ SEPARAR EL CLIENTE DEL SERVIDOR?

- 
1. Centralizar la información/datos.
 2. Separación de responsabilidades.



El **servidor** es la única entidad que tendrá acceso a los datos y solo nos ofrecerá/servirá a los clientes en los que confía, de modo que protegemos la información y podremos atender a varios clientes a la vez.
Está instalado en equipos con muchos recursos.



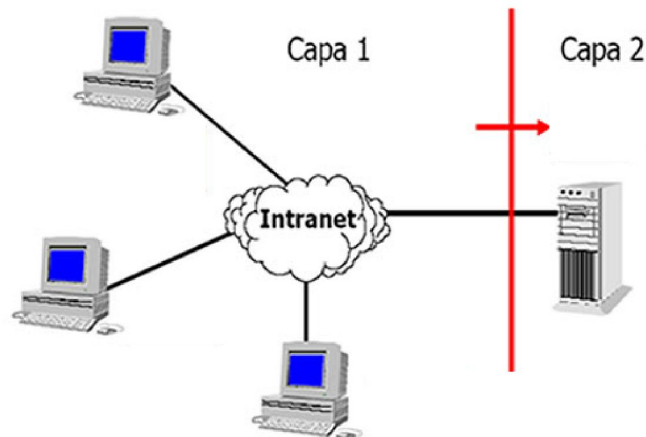
El **cliente** suele estar instalado en computadoras con bajos recursos, ya que desde allí no se procesa nada.

- ✓ Por lo general un cliente conecta a un servidor pero también hay clientes que se conectan a múltiples servidores para funcionar.

Modelos de programación en entornos cliente/servidor

Modelo de dos capas

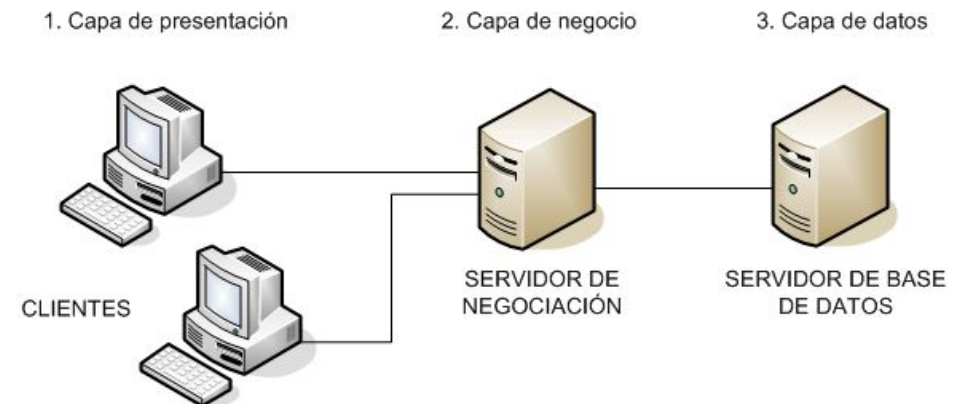
- El cliente ataca los servidores de un servidor donde se ejecuta todo lo necesario para dar servicio.
- Modelo por defecto en el que el servidor, recibe una petición y es capaz de responder a sus propios recursos.



¿Existen arquitecturas de más de 3 capas?

Modelo de tres capas

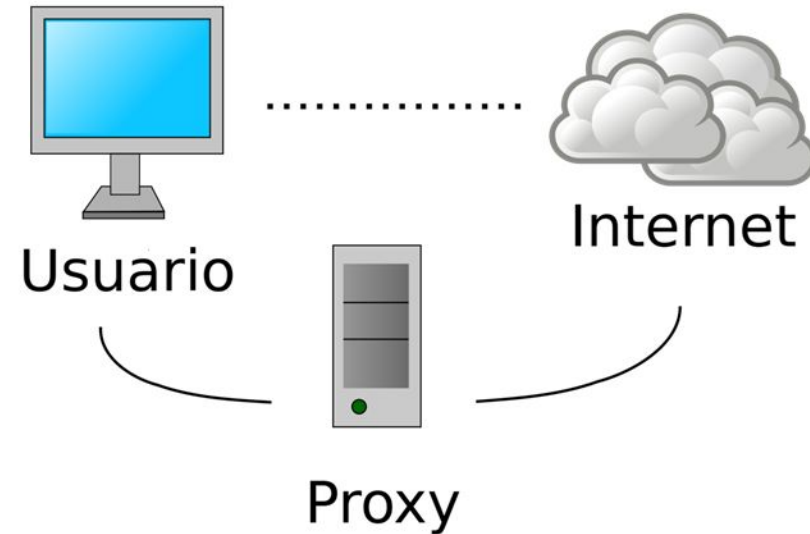
- El servidor al que atacan los clientes es el intermediario entre otro equipo que almacena datos.



¿Y los proxies?

Modelos de programación en entornos cliente/servidor

Qué es un servidor Proxy



Un equipo informático que **hace de intermediario entre las conexiones de un cliente y un servidor** de destino, filtrando todos los paquetes entre ambos.

Siendo tú el cliente, esto quiere decir que el proxy recibe tus peticiones de acceder a una u otra página, y se encarga de transmitírselas al servidor de la web para que esta no sepa que lo estás haciendo tú. De esta manera, cuando vayas a visitar una página web, en vez de establecer una conexión directa entre tu navegador y ella puedes dar un rodeo y enviar y recibir los datos a través de esta proxy. **La página que visites no sabrá tu IP** sino la del proxy, y podrás hacerte pasar por un internauta de otro país distinto al tuyo.

Modelos de programación en entornos cliente/servidor

Algunos tipos de Proxy

- 1.Proxy web:** se basa en HTTP y HTTPS actuando como intermediario para acceder a otros servicios en Internet. Toda nuestra navegación pasará por el proxy web que estemos utilizando.
- 2.Proxy caché:** registrar el contenido de la web de antemano: HTML, CSS e imágenes para acelerar el contenido de un sitio al navegar. Los datos de una web quedan almacenados en la primera visita y si hay una segunda no necesita revisarlos todos de nuevo.
- 3.Proxy inverso:** un servidor que acepta todo el tráfico y lo reenvía a un recurso específico. Este tipo de proxy aporta seguridad al servidor principal, restringiendo el acceso a rutas definidas.

Modelos de programación en entornos cliente/servidor

ALGUNAS CARACTERÍSTICAS CLIENTE-SERVIDOR

- + No es dependiente de una plataforma, los usuarios pueden usar el Sistema Operativo que deseen.
- + Existe un alto nivel de interacción en las interfaces gráficas que son ligeras.
- + El sistema puede escalarse fácilmente sin la necesidad de realizar ninguna modificación en el cliente.
- + Estructura modular para cada uno de los componentes, separación lógica del sistema.

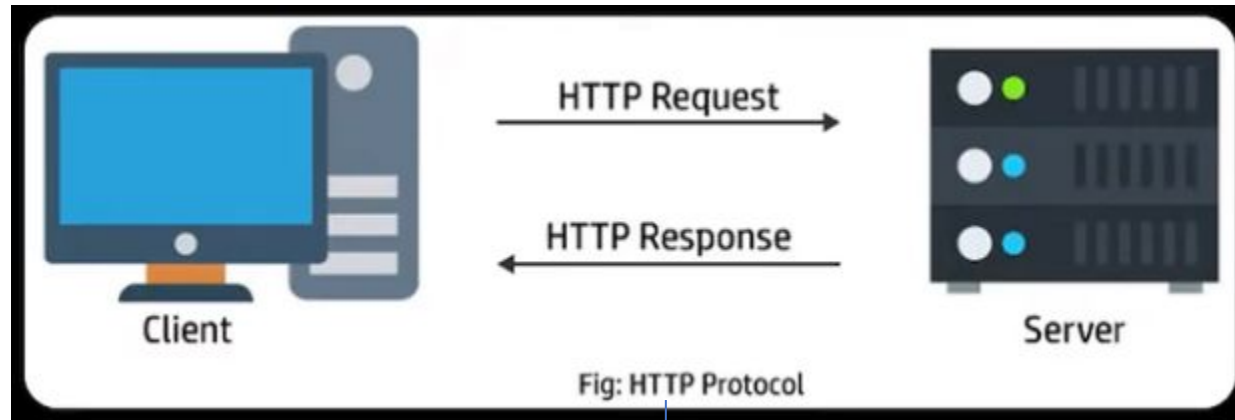
Limitaciones

- El sistema puede fallar al centralizar todo en un servidor; si este cae, ningún usuario podrá conectarse.
- A nivel económico a veces es costoso.

Modelos de programación en entornos cliente/servidor

Flujo de interacción

El cliente envía solicitudes HTTP hacia un servidor que se encuentra en algún lugar.
El servidor procesa la solicitud y envía una respuesta (HTML, JSON, XML, etc.)



¡Protocolo HTTP!

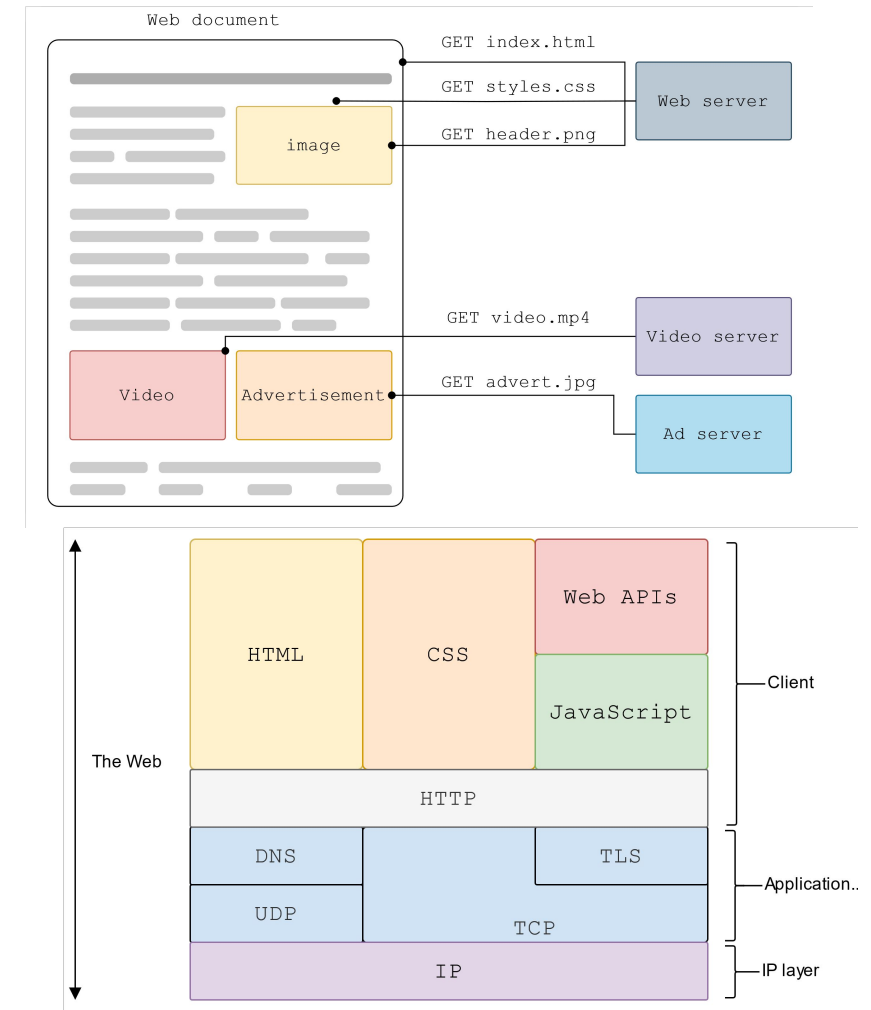
¿Solo se transmite
"texto"?

¿Qué significa HTTP?

Modelos de programación en entornos cliente/servidor

HTTP

Hypertext Transfer Protocol (HTTP) (o *Protocolo de Transferencia de Hipertexto en español*) es un protocolo de la [capa de aplicación](#) para la transmisión de documentos hipermedia, como HTML. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque se puede utilizar para otros propósitos también. Sigue el clásico [modelo cliente-servidor](#), en el que un cliente establece una conexión con el servidor, realiza una petición y espera hasta que recibe una respuesta del mismo. Se trata de un [protocolo sin estado](#), lo cual significa que el servidor no guarda ningún dato (estado) entre dos peticiones. Aunque en la mayoría de casos se basa en una conexión del tipo TCP/IP, se puede usar sobre cualquier [capa de transporte](#) segura o de confianza, es decir, sobre cualquier protocolo que no pierda mensajes silenciosamente, tal como UDP.



¡Protocolo HTTP!

¿Qué significa HTTP?

<https://developer.mozilla.org/es/docs/Web/HTTP>

Modelos de programación en entornos cliente/servidor

Tipos MIME

El tipo **Extensiones multipropósito de Correo de Internet (MIME)** es una forma estandarizada de indicar la naturaleza y el formato de un documento, archivo o conjunto de datos. Está definido y estandarizado en [IETF RFC 6838](#). La [Autoridad de Números Asignados de Internet \(IANA\)](#) es el organismo oficial responsable de realizar un seguimiento de todos los tipos MIME oficiales, y puede encontrar la lista más actualizada y completa en la página de [tipos de medios \(Media Types\)](#).

Los navegadores a menudo usan el tipo MIME (y no la extensión de archivo) para determinar cómo procesará un documento; por lo tanto, es importante que los servidores estén configurados correctamente para adjuntar el tipo MIME correcto al encabezado del objeto de respuesta.

Sintaxis

Estructura general

tipo/subtipo



Media Types

Last Updated

2024-09-16

Registration Procedure(s)

Expert Review for Vendor and Personal Trees. For Standards Tree, see [\[RFC6838\]](#), Section 3.1.

Expert(s)

Alexey Melnikov, Darrel Miller, Murray Kucherawy (backup)

Reference

[\[RFC6838\]](#)[\[RFC4855\]](#)

<https://www.iana.org/assignments/media-types/media-types.xhtml>

¿Solo se transmite
"texto"?

¡Protocolo HTTP!

¿Qué significa HTTP?

Modelos de programación en entornos cliente/servidor

Algo más sobre HTTP...

Caché

La caché HTTP almacena una respuesta asociada con una solicitud y reutiliza la respuesta almacenada para solicitudes posteriores.

Cookies

Datos que un servidor envía al navegador web del usuario. El navegador guarda estos datos y los envía de regreso junto con la nueva petición al mismo servidor.

Cookies de sesión

```
GET /sample_page.html HTTP/1.1
Host: www.example.org
Cookie: yummy_cookie=choco; tasty_cookie=strawberry
```

Cookies Permanentes

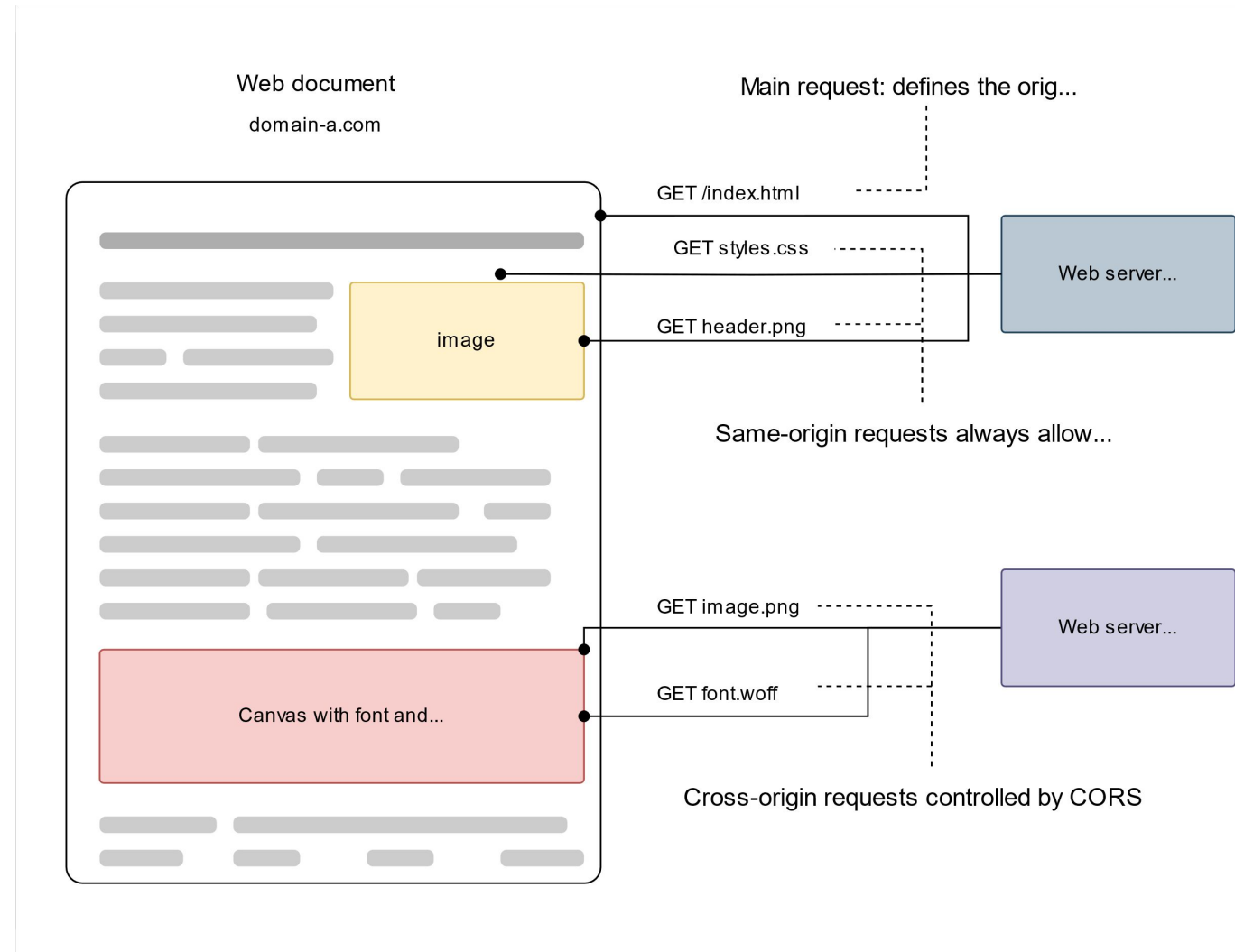
```
Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT;
```


Modelos de programación en entornos cliente/servidor

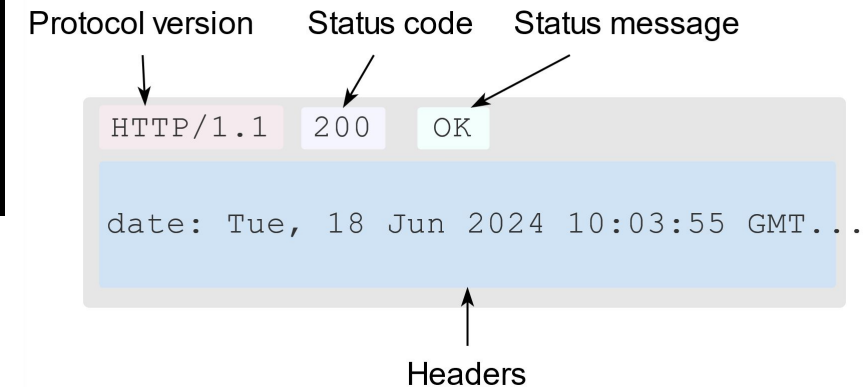
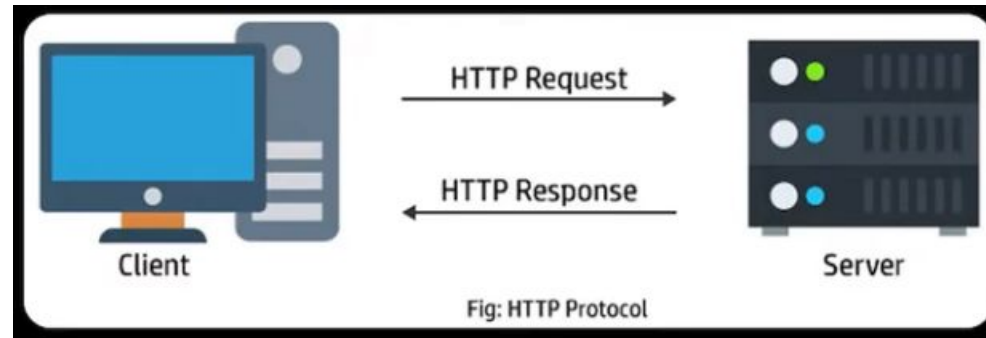
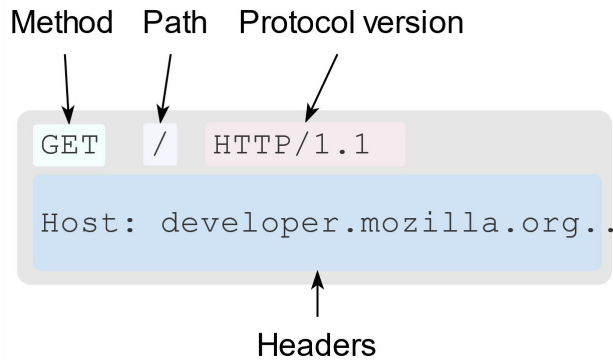
Algo más sobre HTTP...

CORS

El Cross-Origin Resource Sharing (CORS) es un mecanismo de seguridad utilizado por los navegadores web para permitir o restringir las solicitudes de recursos (como archivos JavaScript, CSS, imágenes, etc.) desde un origen (dominio, protocolo y puerto) a otro origen diferente al de la página web actualmente visualizada.



Modelos de programación en entornos cliente/servidor



Una típica sesión HTTP

- 1.El cliente establece típicamente una conexión TCP.
- 2.El cliente manda su petición, y espera por la respuesta.
- 3.El servidor procesa la petición, y responde con un código de estado y los datos correspondientes.

¿Cuál es el puerto por defecto?

Modelos de programación en entornos cliente/servidor

Peticiones HTTP Respuestas

```
GET /index.html HTTP/1.1
Host: wikipedia.org
Accept: text/html
```

```
HTTP/1.1 200 OK
Server: wikipedia.org
Content-Type: text/html
Content-Length: 2026
```

```
<html>
...
</html>
```

```
POST /api/autores HTTP/1.1
Host: miWebApi.com
Content-Type: application/json
Cache-Control: no-cache

{
  "Nombre": "Felipe Gavilán",
  "Edad": 999
}
```

```
HTTP/1.1 200 OK
Date: Thu, 03 Jan 2019 23:26:07 GMT
Server: gws
Accept-Ranges: bytes
Content-Length: 68894
Content-Type: text/html; charset=UTF-8

<!doctype html><html ...
```

- Una línea de petición, con un método
- Un conjunto de campos cabecera
- Un cuerpo, el cual es opcional



MÉTODOS



→ GET
← PUT
← POST
✗ DELETE



¿Estos son todos?

Modelos de programación en entornos cliente/servidor

Peticiones HTTP Respuestas

```
GET /index.html HTTP/1.1
Host: wikipedia.org
Accept: text/html
```

```
HTTP/1.1 200 OK
Server: wikipedia.org
Content-Type: text/html
Content-Lenght: 2026
```

```
<html>
...
</html>
```

```
POST /api/autores HTTP/1.1
Host: miWebApi.com
Content-Type: application/json
Cache-Control: no-cache

{
  "Nombre": "Felipe Gavilán",
  "Edad": 999
}
```

```
HTTP/1.1 200 OK
Date: Thu, 03 Jan 2019 23:26:07 GMT
Server: gws
Accept-Ranges: bytes
Content-Length: 68894
Content-Type: text/html; charset=UTF-8

<!doctype html><html ...
```

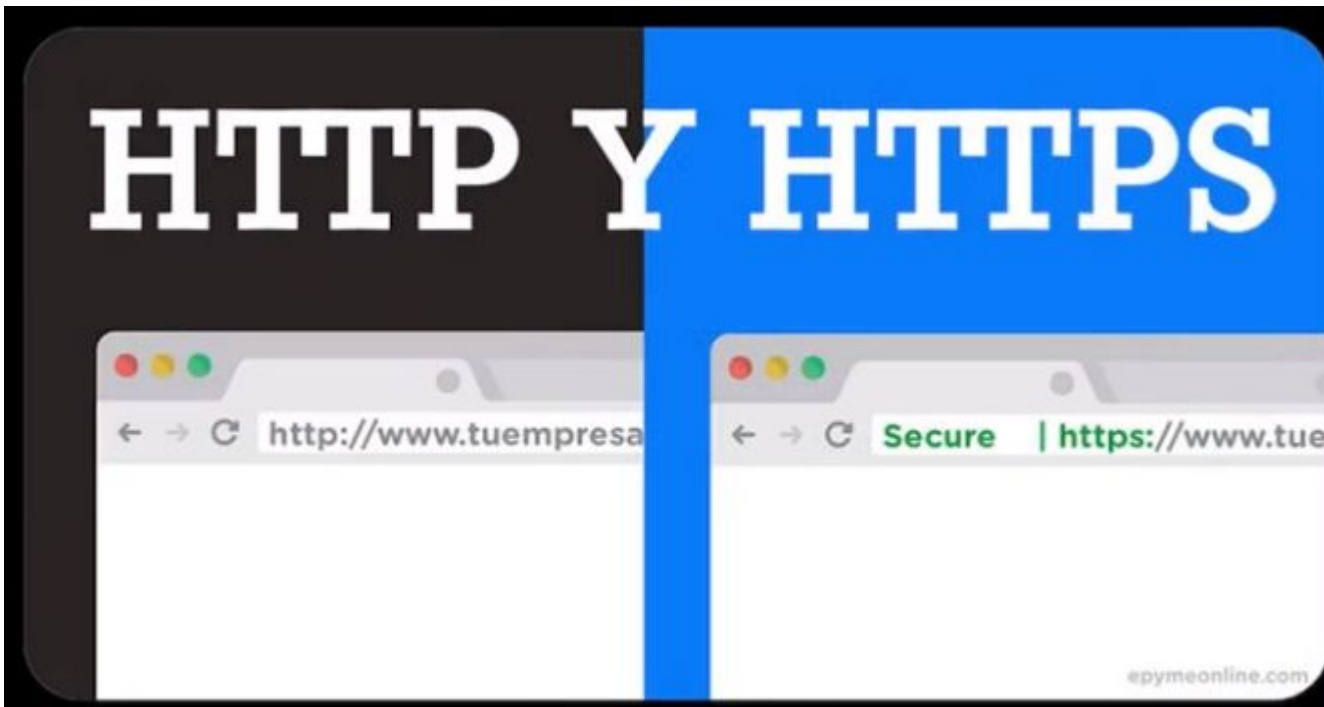
- Una línea de petición, con un método
- Un conjunto de campos cabecera
- Un cuerpo, el cual es opcional

CÓDIGOS

- **1xx:** Mensaje informativo.
- **2xx:** Exito
 - 200 OK
 - 201 Created
 - 202 Accepted
 - 204 No Content
- **3xx:** Redirección
 - 300 Multiple Choice
 - 301 Moved Permanently
 - 302 Found
 - 304 Not Modified
- **4xx:** Error del cliente
 - 400 Bad Request
 - 401 Unauthorized
 - 403 Forbidden
 - 404 Not Found
- **5xx:** Error del servidor
 - 500 Internal Server Error
 - 501 Not Implemented
 - 502 Bad Gateway
 - 503 Service Unavailable

¿Hay más?

Modelos de programación en entornos cliente/servidor



Explicar qué es HTTP, qué es HTTPS y sus diferencias

Modelos de programación en entornos cliente/servidor

¿Os acordáis de mi?

Pues... falta algo

HTTP

HTML



Modelos de programación en entornos cliente/servidor

URI...

...o URL?



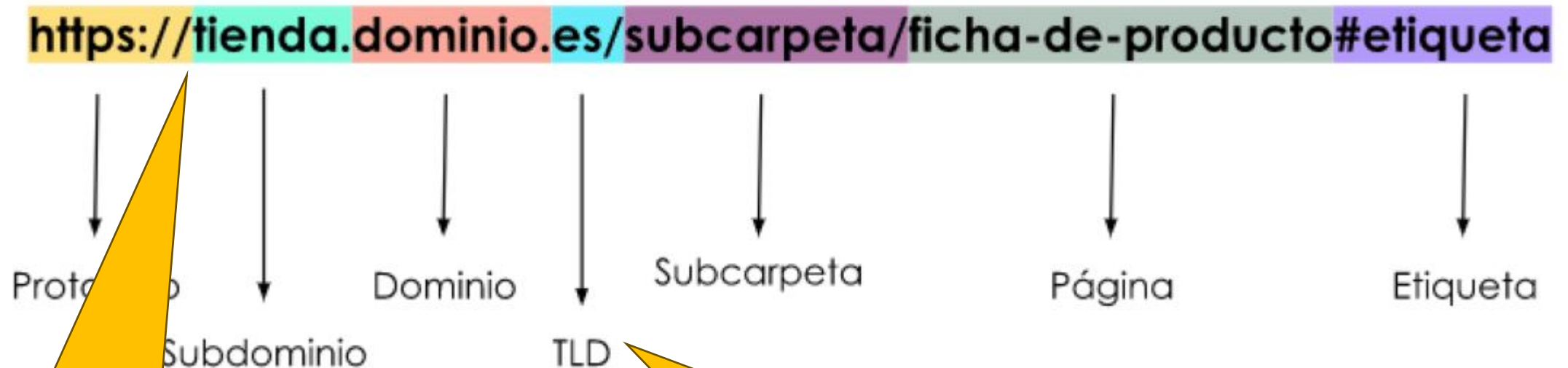
¿Qué es una URI?

Una URI o identificador uniforme de recursos es una cadena de caracteres que generalmente identifica cualquier recurso web mediante un nombre, una ubicación o ambos. Un localizador uniforme de recursos (URL) y un nombre uniforme de recursos (URN) son los dos tipos de URI.

Explicar qué es URI, URL y sus diferencias

Modelos de programación en entornos cliente/servidor

URLs

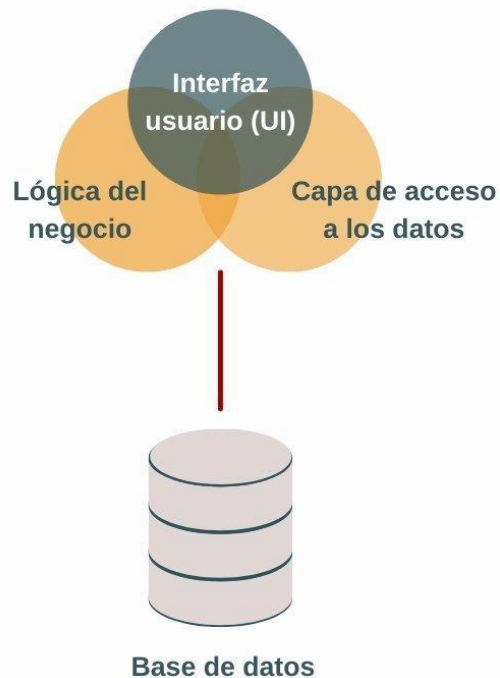


¿Dónde está el www?
¿Es lo mismo ponerlo y no ponerlo?

¿Qué significan estas siglas?
¿Cuáles son? ¿Afectan al SEO?

Modelos de programación en entornos cliente/servidor

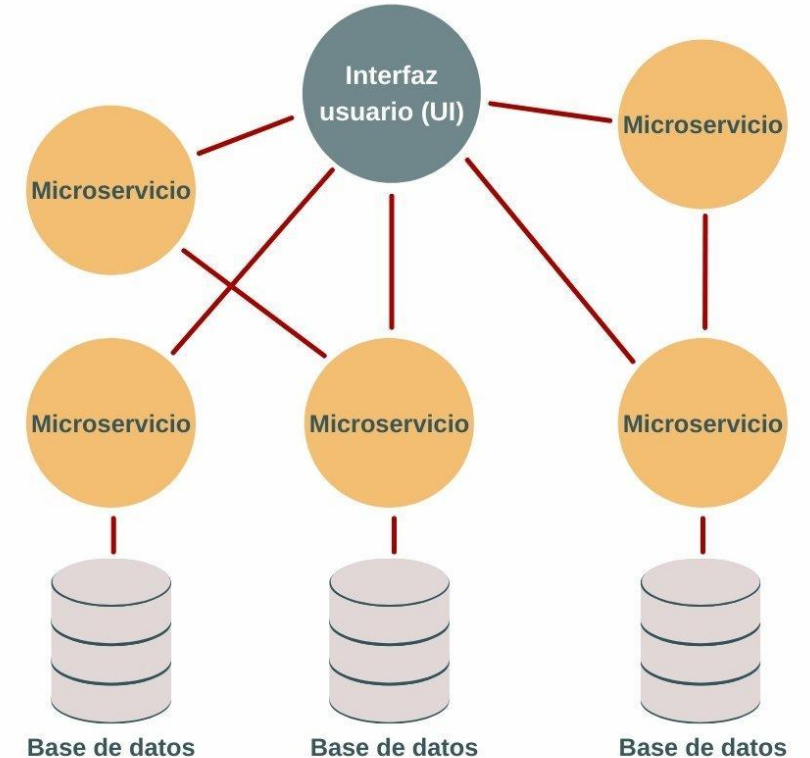
Arquitectura monolítica



Definiciones

Ventajas y desventajas de cada enfoque

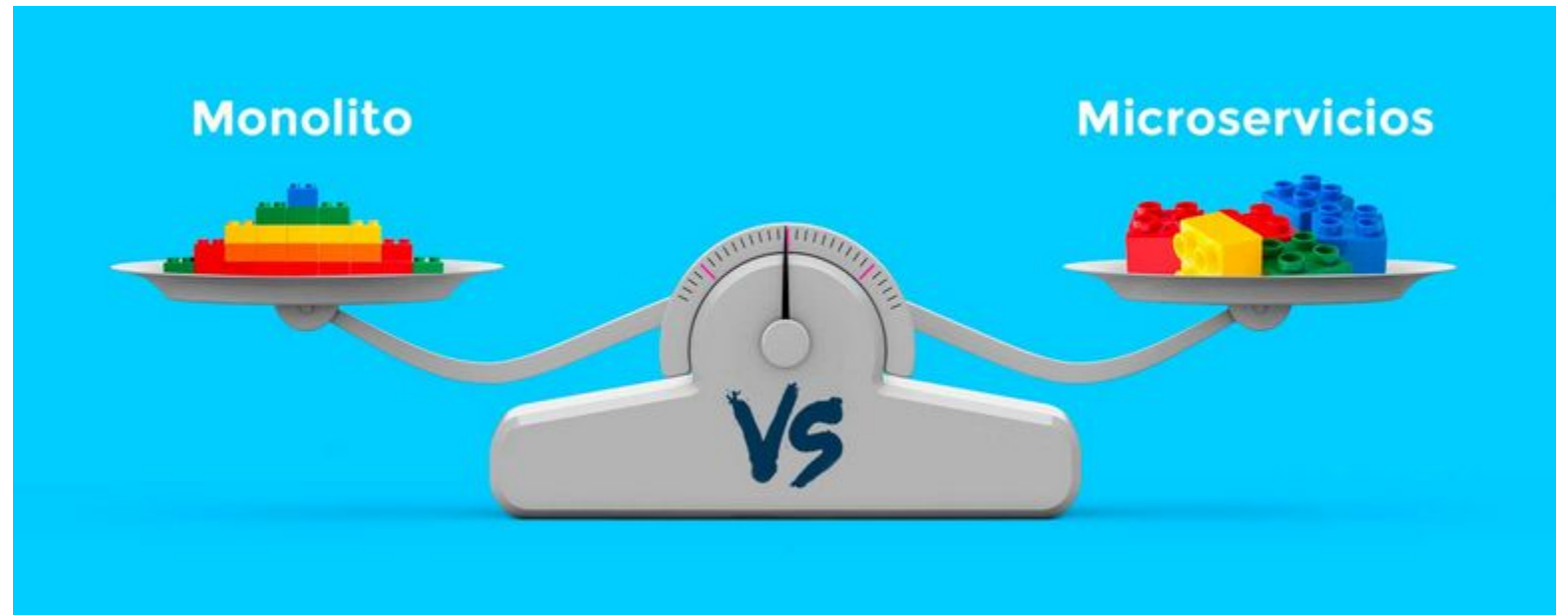
Arquitectura microservicios



Modelos de programación en entornos cliente/servidor



¿Cuál usar?
¿En qué casos?



Mecanismos de ejecución de código en un navegador Web



¿Entendemos la diferencia entre una página web estática y dinámica?

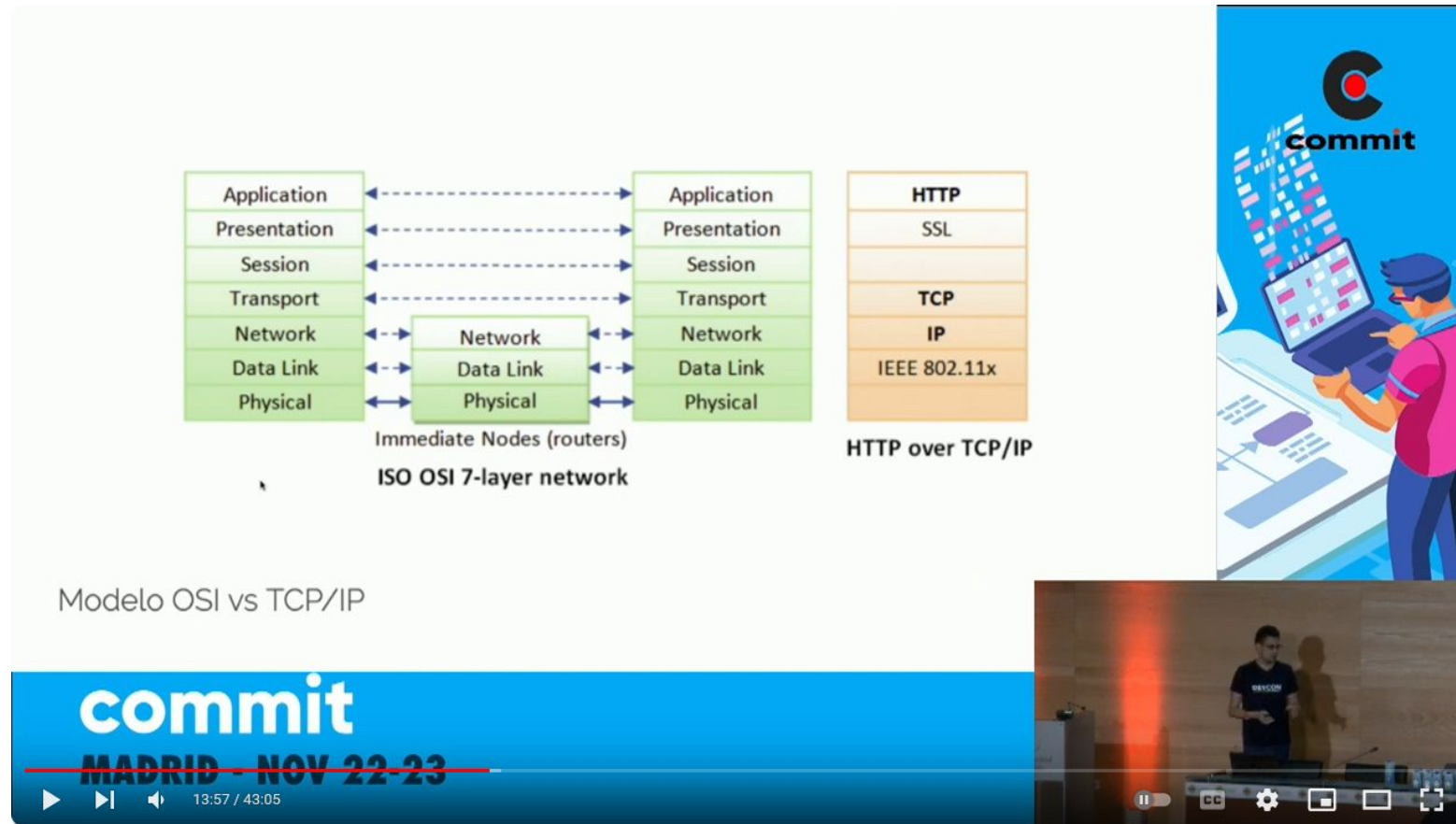
Intentemos **definirlo**...

¿Hay **ventajas y desventajas** en unas y otras?

¿Y la diferencia entre página y sitio web?

.... ¿y aplicación **aplicación web**? ... ¿qué es **RIA**?

Mecanismos de ejecución de código en un navegador Web



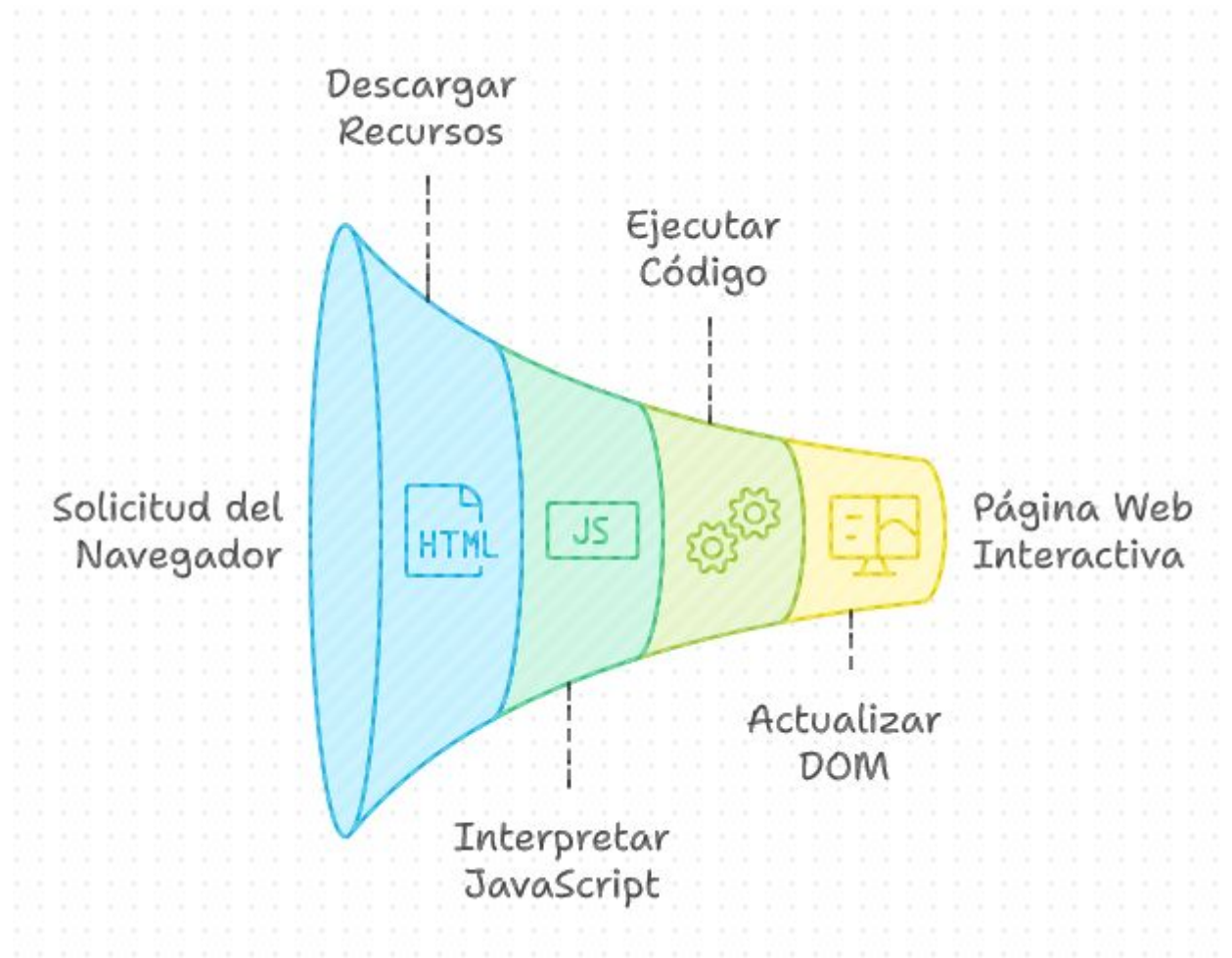
Mecanismos de ejecución de código en un navegador Web

Modelo de ejecución en el cliente:

1. El navegador descarga el HTML, CSS, JavaScript.
2. El motor de JavaScript interpreta y ejecuta el código.
3. El DOM (Document Object Model) se actualiza en tiempo real según la interacción del usuario.

Etapas del ciclo de vida del código en un navegador:

1. Carga del documento HTML y ejecución de scripts.
2. Interacciones del usuario y eventos gestionados por JavaScript.
3. Redibujado del DOM por cambios dinámicos.



Mecanismos de ejecución de código en un navegador Web

MOTORES E INTÉRPRENTES EN LOS NAVEGADORES ACTUALES

- **Google Chrome**
 - El motor de renderizado de Google Chrome es [Blink](#) y su intérprete de JavaScript es [V8](#).
- **Mozilla Firefox**
 - El motor de renderizado de Mozilla Firefox es [Gecko](#) y su intérprete de JavaScript es [SpiderMonkey](#).
- **Safari**
 - El motor de renderizado de Safari es [WebKit](#) y su intérprete de JavaScript es [Nitro](#).
- **Microsoft Edge**
 - El motor de renderizado de Edge es ~~EdgeHTML~~ [Blink](#) y su intérprete de JavaScript es [ChakraCore](#).
- **Internet Explorer**
 - El motor de renderizado de Internet Explorer era [Trident](#) y su intérprete de JavaScript es [Chakra](#).

Capacidades y limitaciones de ejecución

- Dependencia del hardware del cliente

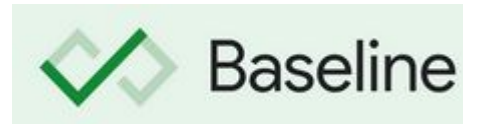
- PC, teléfono, tablet...

EJEMPLO: funcionamiento de un juego 3D en un dispositivo potente vs. gama baja

- Diferencias entre motores de los navegadores

- Cada motor puede implementar características de manera ligeramente diferente, lo que puede generar problemas de compatibilidad.

EJEMPLO: css flexible layout? [caniuse](#)



- *Sandboxing*

- Técnica de seguridad utilizada por los navegadores para limitar el acceso de las aplicaciones web al entorno del sistema operativo del usuario.

EJEMPLO: ver [File API](#)

Compatibilidad con navegadores Web

The screenshot shows the CanIUse website interface. At the top, there's a navigation bar with links for Home, News, April 7, 2024 - 8 new features, Compare browsers, and About. Below this is a large orange header with the text "Can I use" and a search icon, followed by a "Settings" link. A "Filter features" button is also present. The main content area is divided into several sections: "Latest features" with a list of recent updates like WebAssembly BigInt to i64 conversion; "Most searched features" with a list of popular queries like WebP image format; "Test a feature" with a description of the partnership with BrowserStack and a "Test on:" section showing supported devices like IE 11, Safari 17, Safari on iPhone 14, and Chrome on Galaxy S23; "Did you know?" with a note about voting for new features; "Third party tools" with a list of external resources like CanIUse Embed and CanIUse Component; "Browser scores" with a bar chart showing scores for Chrome 129 (433), Firefox 130 (414), and Safari 18.0 (406); and "Other support sites" at the bottom.

Home News April 7, 2024 - 8 new features Compare browsers About

Can I use ? Settings

Index of features Filter features

Latest features

- WebAssembly BigInt to i64 conversion in JS API
- WebAssembly Threads and Atomics
- WebAssembly Multi-Value
- WebAssembly Import/Export of Mutable Globals
- WebAssembly Non-trapping float-to-int Conversion

Most searched features

- WebP image format
- Flexbox
- gap property for Flexbox
- CSS Grid
- WebGL

Test a feature

Our partnership with BrowserStack now lets you test your website for compatibility across 2,000+ real browsers and devices.

Test on:

- IE 11
- Safari 17
- Safari on iPhone 14
- Chrome on Galaxy S23

Did you know?

If a feature you're looking for is not available on the site, you can [vote to have it included](#). Better yet, if you've done the research you can even [submit it yourself](#)!

Next

Third party tools

- The CanIUse Embed — Add support tables to your site
- CanIUse Component — Add support tables to your presentations
- CanIUse command line tool
- DoIUse...? — Lint your CSS to check what features work
- I want to use — Select multiple features and see what % of users can use them

See full list

Browser scores

Current version Dev version

Chrome 129: 433

Firefox 130: 414

Safari 18.0: 406

About these scores

Other support sites

Compatibilidad con navegadores Web

Filter

Extensiones del navegador

► Comenzar

► Conceptos

► Interfaz de usuario

► Cómo hacer

▼ API de JavaScript

Browser support for JavaScript APIs

i18n

storage

► Claves de manifiesto

► Taller de Extensión

Contact us (inglés)

► Canales de discusión

Compatibilidad de navegadores con las API de JavaScript

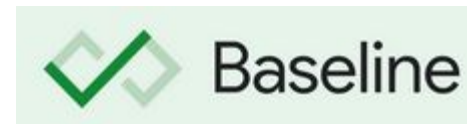
action

[Report problems with this compatibility data on GitHub](#)

	Desktop					Mobile	
	Chrome	Edge	Firefox	Opera	Safari	Firefox for Android	Safari on iOS
action	✓ 88	✓ 88	✓ 109	✓ 74	✓ 15.4	✓ 109	✓ 15.4
ColorArray	✓ 88	✓ 88	✓ 109	✓ 74	✓ 15.4	✓ 109	✓ 15.4
ImageDataType	✓ 88	✓ 88	✓ 109	✓ 74	✓ 15.4	✓ 109	✓ 15.4
disable	✓ 88	✓ 88	✓ 109	✓ 74	✓ 15.4	✓ 109	✓ 15.4
enable	✓ --	✓ --	✓ --	✓ --	✓ --	✓ --	✓ --

En este artículo

action
alarms
bookmarks
browserAction
browserSettings
browsingData
captivePortal
clipboard
commands
contentScriptGlobalScope
contentScripts
contextualIdentities
cookies
declarativeNetRequest
devtools
dns
dom
downloads
events



Compatibilidad con navegadores Web

Announcing BCD Watch

PUBLISHED 1 WEEK PAST

Support increases (14)		
API		
api → PublicKeyCredential → parseCreationOptionsFromJSON_static	Added to Chrome	2 of 3 engines
api → PublicKeyCredential → parseRequestOptionsFromJSON_static	Added to Chrome	2 of 3 engines
api → PublicKeyCredential → toJSON	Added to Chrome	2 of 3 engines
api → RTCDataChannel → binaryType → blob_value	Added to Chrome	3 of 3 engines
CSS		
css → properties → interpolate-size	Added to Chrome	1 of 3 engines
css → properties → interpolate-size → allow-keywords	Added to Chrome	1 of 3 engines
css → properties → interpolate-size → numeric-only	Added to Chrome	1 of 3 engines

BCD Watch

Keeping an eye on changes to MDN's Browser Compatibility Data

This site automatically collects, indexes, and makes available for viewing (and as subscribe-able feeds), information about updates to [Browser Compatibility Data \(BCD\)](#). Currently there are two different views (and feeds) on this data, which are compiled each Tuesday (US/Eastern time).

Perhaps you're interested in [which changes were reported last week](#), or more specifically [which features became universally implemented last week](#) (marked as available in all 3 engines).

If you encounter any problems with BCD Watch or have suggestions for improvements, please feel free to open an issue in our [Github repository](#), or submit suggested changes via pull request if you like.

Compatibilidad con navegadores Web

Cuando la compatibilidad con navegadores antiguos / no soportados es necesaria, se utilizan técnicas como:

- Progressive enhancement

Ejemplo: una página web que funcione sin JavaScript en su forma más básica, pero ofrezca una experiencia más rica con interactividad avanzada cuando el navegador soporte JavaScript.

- Graceful degradation

Ejemplo: un sitio con animaciones avanzadas en CSS para navegadores modernos que sigue siendo completamente funcional en un navegador antiguo, aunque sin animaciones o con un diseño más básico.

- Uso de bibliotecas de compatibilidad / Scripts / conversión de código

Ejemplo: “polyfill” [Babel](#) 

Lenguajes de programación en entorno cliente



Manejo de interactividad
Validación de formularios
Manipulación del DOM



HTML estructura el contenido
CSS define la apariencia
No son lenguajes de programación



Lenguajes de programación en entorno cliente

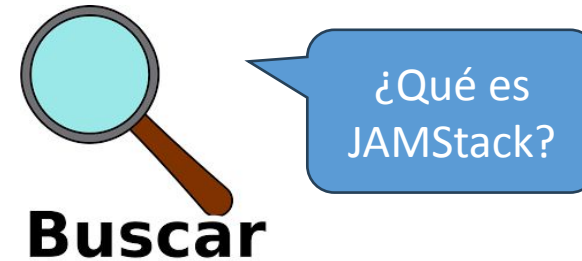
¿Y qué hay de, por ejemplo..  ?



CMS: Content Management Systems



SSG: Static Site Generators



<https://jamstack.org/generators/>

Características de los lenguajes de script.

Ventajas y desventajas sobre la programación tradicional.

- Interpretado, no requiere compilación.
- Basado en eventos.
- Flexible con tipado dinámico.

VENTAJAS

- Ejecución inmediata sin compilación
- Facilita el desarrollo y las actualizaciones dinámicas
- Permite manejar eventos de usuario

DESVENTAJAS

- Menor rendimiento comparado con lenguajes compilados
- Más difícil de depurar debido al tipado dinámico

Tecnologías y lenguajes asociados

HTML5

Nuevas etiquetas semánticas, geolocalización, almacenamiento, etc.

CSS3

Animaciones, transiciones, *media queries*, etc.

AJAX

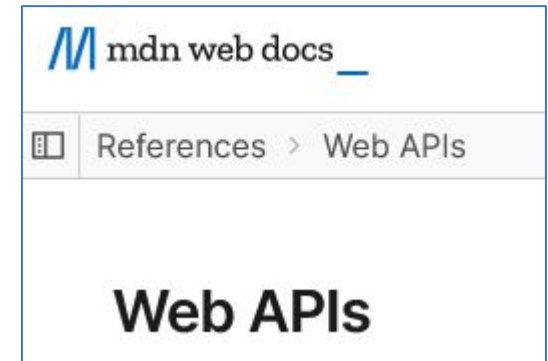
Comunicación asíncrona con el servidor sin recargar la página.

Web APIs

Web Storage, Canvas API, Fetch API para trabajar con recursos del navegador.

Librerías y frameworks

jQuery (librería), React, Vue, Angular (frameworks), etc.



Integración del código con las etiquetas HTML

Inserción de JavaScript en HTML

Uso de la etiqueta `<script>` para incluir JavaScript *en línea* o referenciar *archivos externos*.

- Ejemplo de script en línea:

```
<script>
  alert("Hola mundo!");
</script>
```

- Ejemplo de script externo:

mejora la organización
permite la reutilización del código
facilita su mantenimiento.

```
<script src="miArchivo.js"></script>
```

Es la forma habitual,
¿por qué?

Integración del código con las etiquetas HTML

Cuando se colocan etiquetas `<script>` en un documento HTML, el navegador por defecto **detiene el renderizado de la página** para descargar y ejecutar el archivo JavaScript, lo que puede hacer que el contenido de la página tarde en mostrarse.

async

el script se **descarga en paralelo** con el resto del documento y se ejecutará tan pronto como se descargue
(lo que puede ocurrir antes de que el resto de la página haya terminado de cargarse)

```
<script src="mi-script.js" async></script>
```

¿Cuándo usar uno u otro?

defer

el script se **descarga en paralelo** con el resto del documento y se garantiza que el script se ejecutará solo después de que todo el HTML se haya procesado completamente
(así los scripts no interfieran con el renderizado de la página)

```
<script src="mi-script.js" defer></script>
```

Integración del código con las etiquetas HTML

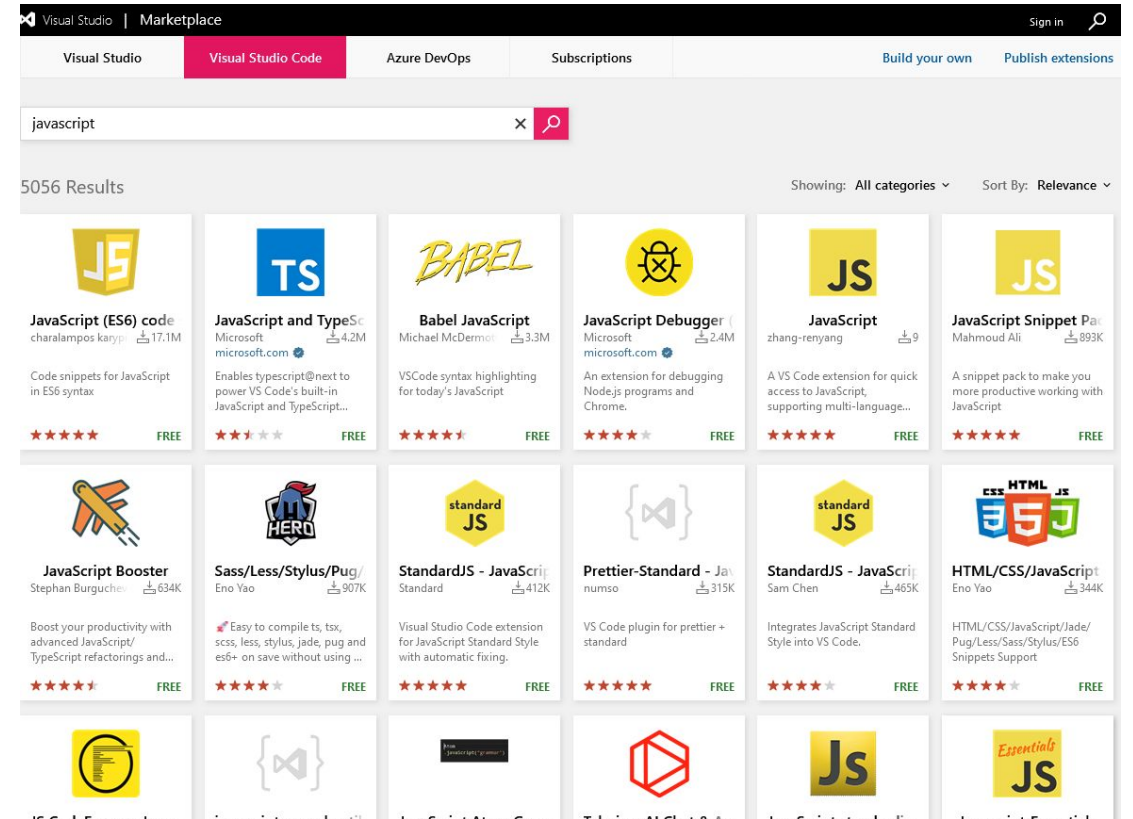
El **DOM** (la representación estructurada del documento HTML) permite a JavaScript interactuar y modificar su contenido dinámicamente:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Manipulación del DOM</title>
</head>
<body>
  <h1 id="titulo">Hola, mundo</h1>
  <button onclick="cambiarTexto()">Cambiar Título</button>

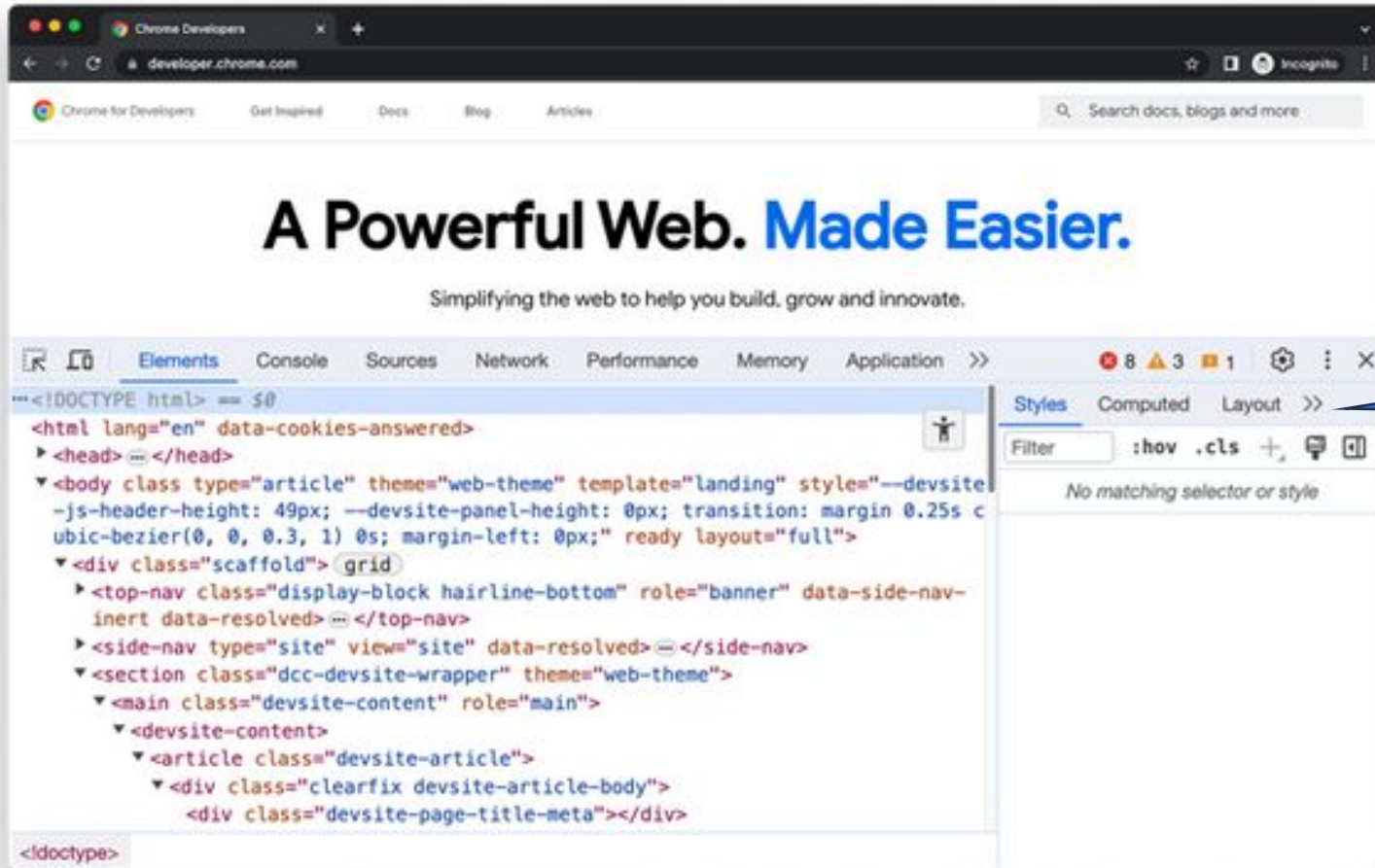
  <script>
    function cambiarTexto() {
      document.getElementById("titulo").innerHTML = "Título Cambiado";
    }
  </script>
</body>
</html>
```

- Crear, eliminar y modificar elementos
- Cambiar los estilos de los elementos
- Manejar eventos

Herramientas de programación



Herramientas de programación



¿Formas de acceder?

Herramientas de programación



PARCEL

