Introducción y Herramientas De Desarrollo

Laura Isabel López Naves
IES Julián Marías

¿Qué vamos a ver?

- [x] Las aplicaciones web
- [x] ¿Qué es JavaScript?
- [x] Tipo de comunicación
- [x] Herramientas de desarrollo
- [x] Primer Script
- [x] Formas de incluir código JavaScript
- [x] Async y defer
- [x] Donde encontrar ayuda

Las aplicaciones web

De forma sencilla se puede definir una **aplicación web** como cualquier herramienta alojada en un servidor a la que se puede acceder desde un navegador web a través de una red de ordenadores.

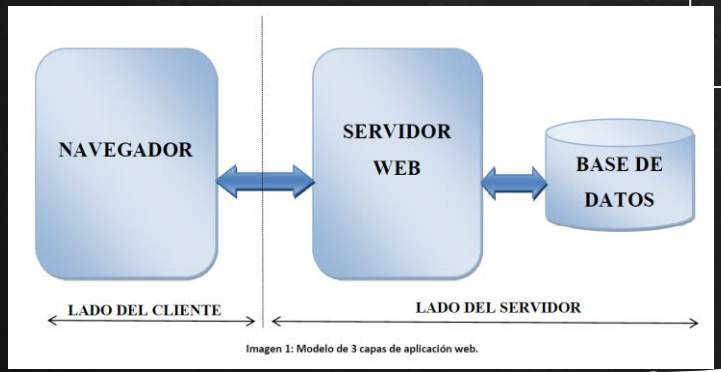
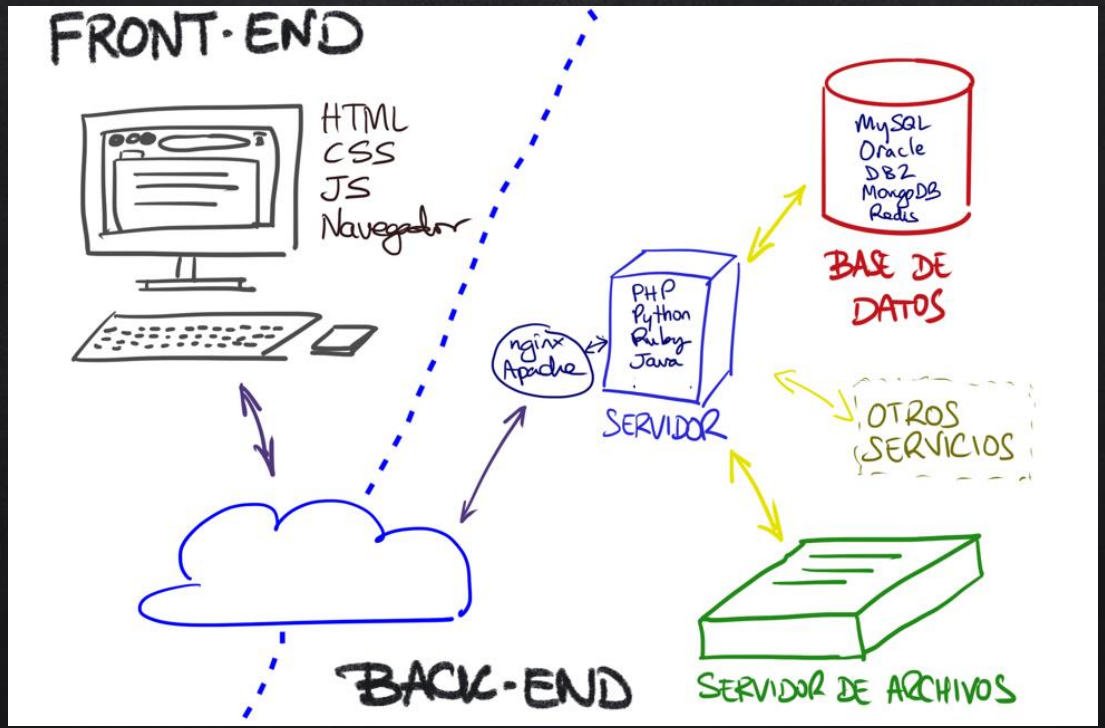
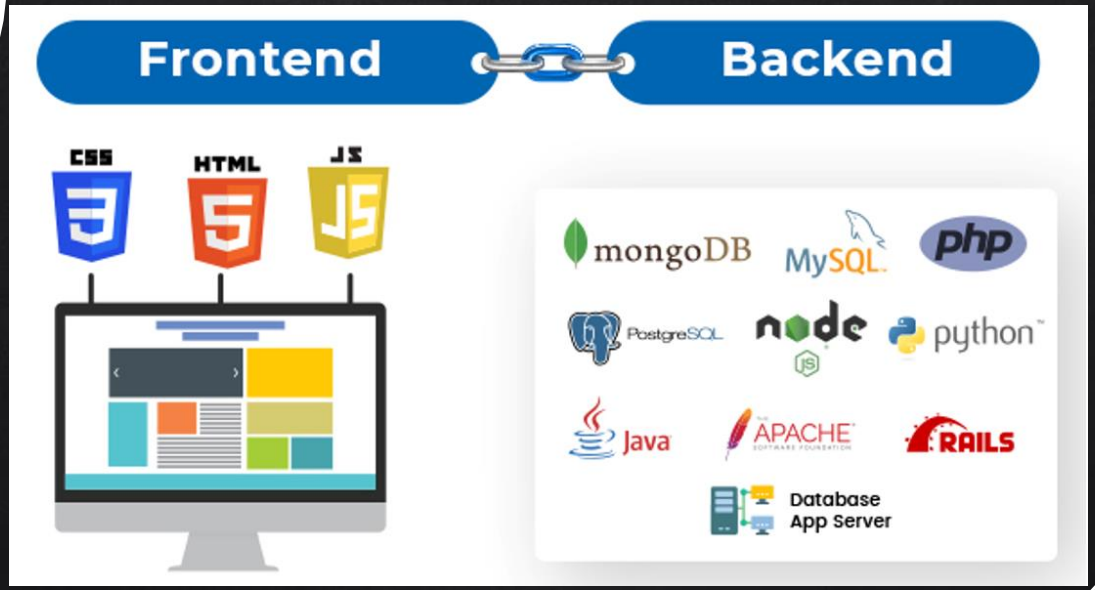


Imagen 1: Modelo de 3 capas de aplicación web.

Las aplicaciones web



Las aplicaciones web



Las aplicaciones web

En el lado del Servidor (server-side) se incluye el hardware y software necesarios para desplegar y ofrecer un servidor Web.

- **Tecnologías PHP, ASP NET, JSP, servlets java, spring etc.**

En el lado del Cliente (client side) se utilizan los navegadores como interfaz de usuario

- **Tecnologías HTML, XHTML, CSS**
- **Elementos de programación Javascript VBScript, applets de java, etc**
- **Tecnología de intercambio de datos XML o JSON**

Las aplicaciones web

El porcentaje de ejecución de cada uno ha variado con los años.

Aunque depende del diseñador de la aplicación o de la funcionalidad de la misma (negocio para el que se crea) el hecho de que cada vez la parte cliente se ejecute en ordenadores más potentes y que las líneas (web) de comunicación sean más rápidas está haciendo que la carga de trabajo sea mayor en la parte cliente.

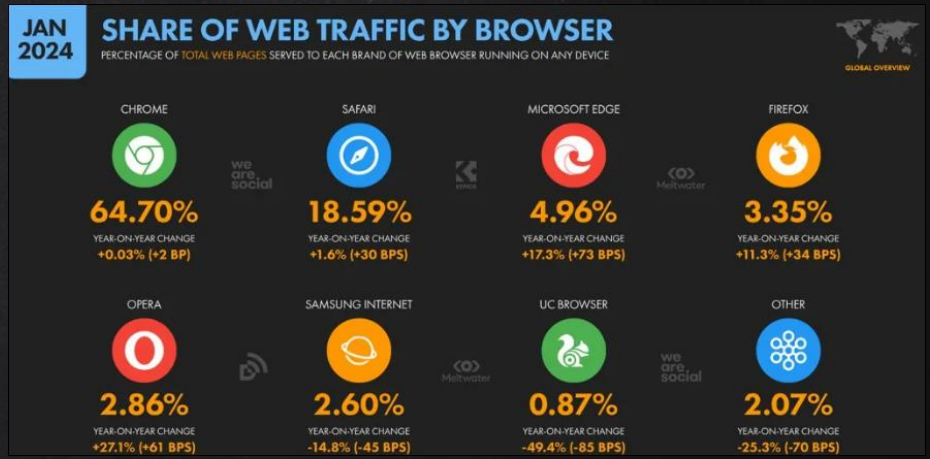
Las aplicaciones web

- Se conoce como navegador web (navegador) o también explorador web (explorador) a un programa informático que permite al usuario ingresar a las páginas Web que desee, siempre que conozca la dirección URL en donde se encuentra (por ejemplo: www.google.com) o bien que haga clic en un hipervínculo que conduzca a dicha página.

- La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados. Además, permite visitar páginas web e interactuar en ella, es decir, podemos enlazar un sitio con otro, imprimir, enviar y recibir datos...

Las aplicaciones web

¿Qué navegador es el más utilizado?



Uso de navegadores en 2024

Las aplicaciones web

Procedimiento Interacción Navegador y Usuario

1. Usuario indica la URI del recurso al que quiere acceder.

2. Navegador lanza la petición al servidor y cuando recibe respuesta la muestra al usuario

✗ Cada navegador tiene una forma diferente de comportarse puesto que cada uno tiene una arquitectura particular. Se ha hecho un esfuerzo por parte de organismos tales como W3C o ECMA International para promover estándares para la web. Aun así, no son de obligado seguimiento y aún hoy en día surgen multitud de problemas debido a las diferencias a veces notables, entre la interpretación que hacen diferentes navegadores sobre el mismo recurso web.

Las aplicaciones web

Elementos en el esquema de ejecución de un navegador

1. Interfaz de usuario → Capa que muestra al usuario las herramientas para controlar el acceso a recursos (a través de la URI)

2. Motor del buscador → Barra de direcciones donde insertar la URI

3. Motor de renderización → Una vez que se recibe el recurso solicitado, se interpreta de arriba hacia abajo. La forma de visualización dependerá cómo el navegador lo interprete todo (XML, CSS, JavaScript, etc.)

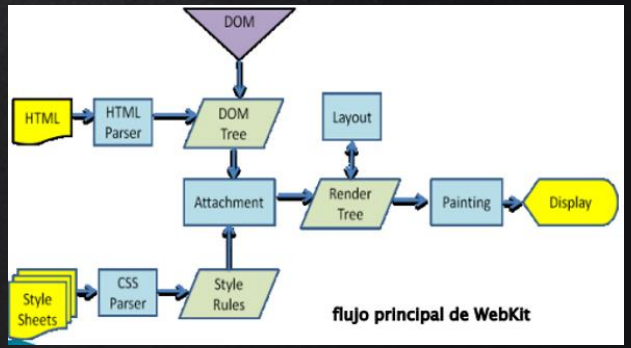
- ✗ Si el recurso tiene código de un lenguaje que no reconoce no podrá mostrar el recurso correctamente.
- ✗ Dos navegadores con diferentes intérpretes pueden mostrar salidas distintas.
- ✗ Motores más conocidos: Gecko (Mozilla) Trident (IE), Webkit (Safari), Blink (Chrome y Opera).

Las aplicaciones web

3. Motor de renderización → El motor de renderización empieza a analizar el documento HTML y convierte las etiquetas en nodos DOM en un árbol denominado "árbol de contenido".

Analiza los datos de estilo, tanto en los archivos CSS externos como en los elementos de estilo. Los datos de estilo, junto con las instrucciones visuales del código HTML, se utilizan para crear otro árbol: el árbol de renderización.

El árbol de renderización contiene rectángulos con atributos visuales, como el color y las dimensiones. Los rectángulos están organizados en el orden en el que aparecerán en la pantalla.



¿Qué es el renderizado de una web?
El renderizado de una página web se refiere al proceso en el cual el navegador interpreta el código HTML, CSS y JavaScript de una página web y lo convierte en la representación visual que vemos en la pantalla.

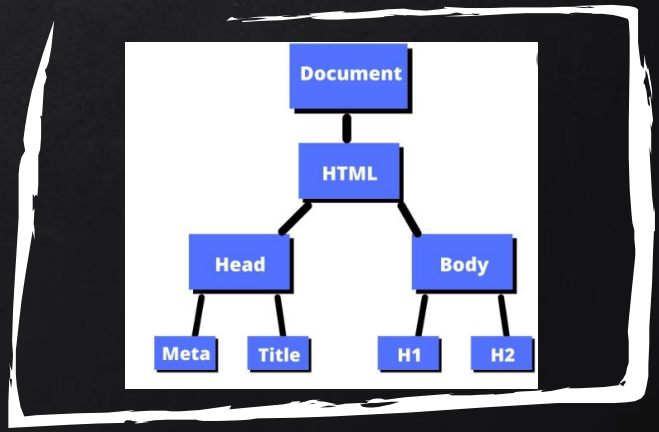
Las aplicaciones web

4. Subsistema de comunicaciones → Implementar diferentes protocolos de transmisión de información e identificación de formato de datos recibidos.

5. Intérpretes de lenguajes de scripting → Analizan y ejecutan el código incrustado en el recurso recibido

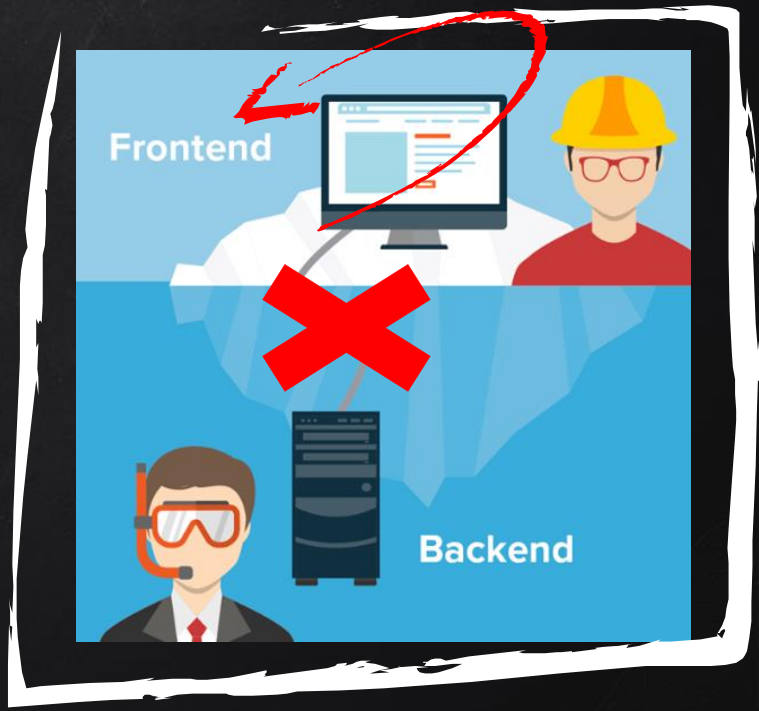
6. Modelo DOM → Conjunto de objetos jerárquicos en forma de árbol que representa cada uno de los elementos de la web.

Permite añadir el comportamiento que tendrán al interactuar usuario-página web



Las aplicaciones web

Combinando funcionalidades que proporcionan los lenguajes de scripting para interactuar con el modelo DOM, es posible programar fragmentos de código que serán ejecutados desde la parte cliente sin necesidad de acudir al servidor.



Lenguajes Interpretados y compilados

- Los lenguajes interpretados se ejecutan mediante un intérprete, que procesa las órdenes que incluye el programa una a una.
- Esto hace que sean menos eficientes en tiempo que los compilados (típicamente 10 veces más lentos), ya que la interpretación de la orden debe hacerse en tiempo de ejecución.
- Como ventajas podemos decir que son más fáciles de depurar y son más flexibles para conseguir una independencia entre plataformas (portabilidad).

Ejemplos de lenguajes interpretados: JavaScript, Python, Perl, PHP, Bash

Lenguajes compilados

✗ Por otra parte, los lenguajes compilados necesitan del compilador como paso previo a la ejecución. Como resultado de compilar el código fuente se obtiene el código máquina, y esto hace que su ejecución sea eficiente en tiempo.

Ejemplos de lenguajes compilados: C, C ++...

Para pensar...

¿Java es compilado o interpretado?

Lenguajes interpretados y compilados

Java, un caso especial

No todos los lenguajes de programación entran en la división entre compilados e interpretados.
En el caso del lenguaje Java, el código fuente es independiente de la plataforma en que se programa.

El compilador de Java genera un código máquina especial (el bytecode), y la Máquina Virtual de Java (JVM), existente para cada plataforma (Linux, Windows, Solaris), sabe interpretar este bytecode, ejecutándolo.
De este modo, escribiendo una sola vez el código, se obtienen aplicaciones multiplataforma.

¿Qué es JavaScript?

- JavaScript es un leguaje interpretado creado en 1995 por NetScape para extender las capacidades del lenguaje HTML.
- Inicialmente se popularizó como un lenguaje de navegador con sintaxis similar a Java. Actualmente se utiliza en múltiples entornos y frameworks
- En 1997 se estandarizó por ECMA (European Computer Manufacturer's Associaton) en su versión ECMA 262 o ECMAScript.
- Actualmente la mayor parte de los navegadores cumplen con el ECMAScript 2015, también llamado ECMAScript versión 6 o ES6.

  Para ver la compatibilidad de los navegadores con ES6:

  http://kangax.github.io/compat table/es6/

  Para ver las características soportadas por los navegadores

  https://caniuse.com/

¿Qué es JavaScript?

A tener en cuenta!

- ✗ Versión de ECMAScript en el navegador:
  - No se puede configurar un navegador para que trabaje bajo una determinada versión de ECMAScript. Los nuevos navegadores van incorporando características nuevas a medida que van apareciendo. Se puede decir que un navegador soporta ES6 o ES7 pero no que "trabaje" en modo concreto. JavaScript es compatible hacia atrás.
- ✗ Versión de ECMAScript en el servidor:
  - El principal entorno utilizado es Node.js (hay otros como Deno).

Funcionalidades de JavaScript

A tener en cuenta!

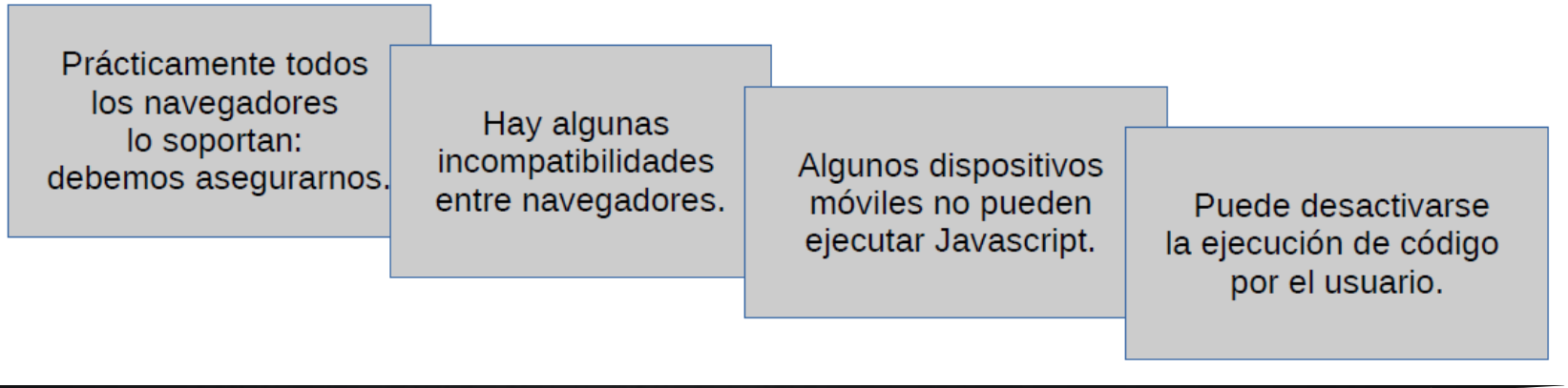
¿Qué podemos y qué no podemos hacer con él?

- Lenguaje interpretado en el navegador: puede estar deshabilitado.
- No puede escribir ficheros en el servidor.
- Reacciona a la interacción del usuario.
- Controla múltiples ventanas, marcos, plugins, applets…
- Pre-procesa datos en el cliente.
- Modifica estilos y contenido de navegadores.
- Puede solicitar ficheros al servidor.

Compatibilidad de JavaScript

A tener en cuenta!

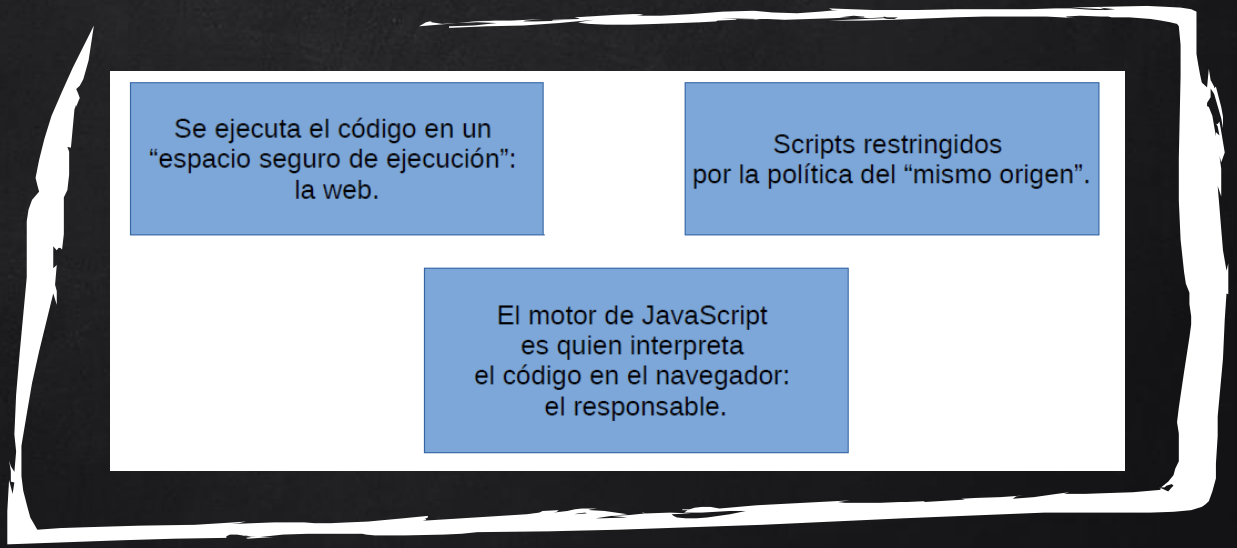
¿Es compatible JavaScript en todos los dispositivos?
¿Es soportado por todos los navegadores?
¿Se puede habilitar y deshabilitar?



Seguridad en JavaScript

A tener en cuenta!

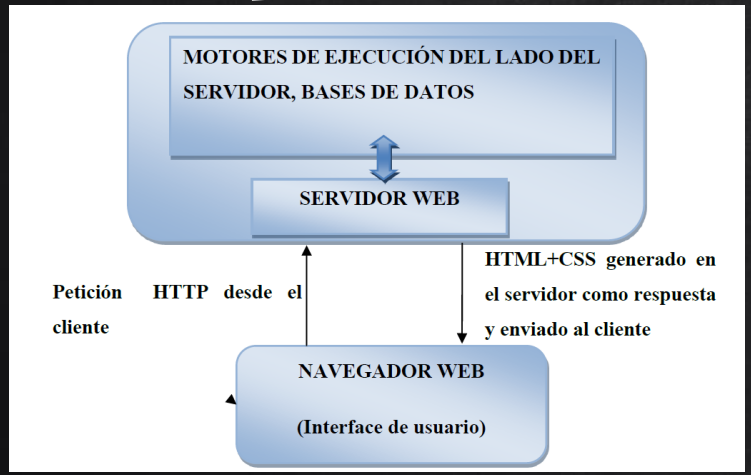
¿Podemos, mediante JavaScript, vulnerar la seguridad de un sitio web?
¿Podemos atacar un servidor mediante JavaScript?



https://developer.mozilla.org/es/docs/Web/Security/Same-origin_policy

Tipo de comunicación

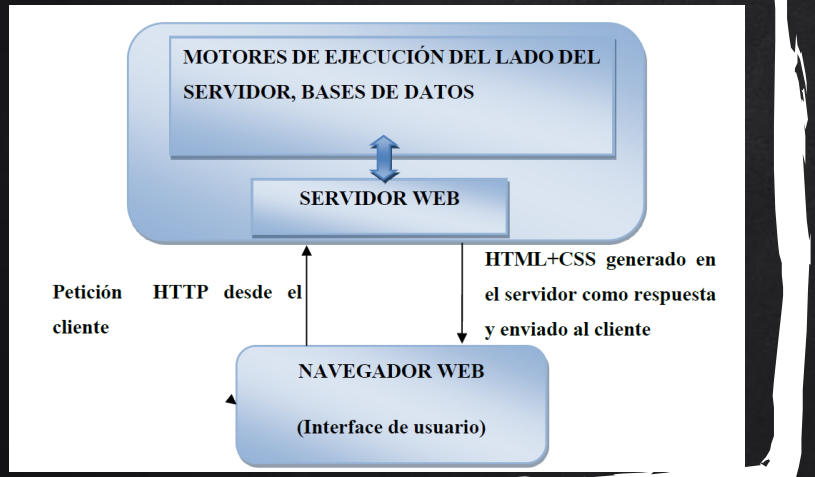
En el modelo de comunicación síncrona, cada vez que se lanza una petición desde el cliente al servidor este devuelve el recurso solicitado y el navegador lo carga y renderiza por completo.



El cliente lanza una petición al servidor. Al resolver esta petición es posible que sea necesario acceder a alguna base de datos o ejecutar código del lado del servidor.

Como resultado se generará una salida entendible por el navegador (HTML+CSS por lo general) y esta salida se envía para que sea abierta por el mismo.

Tipo de comunicación

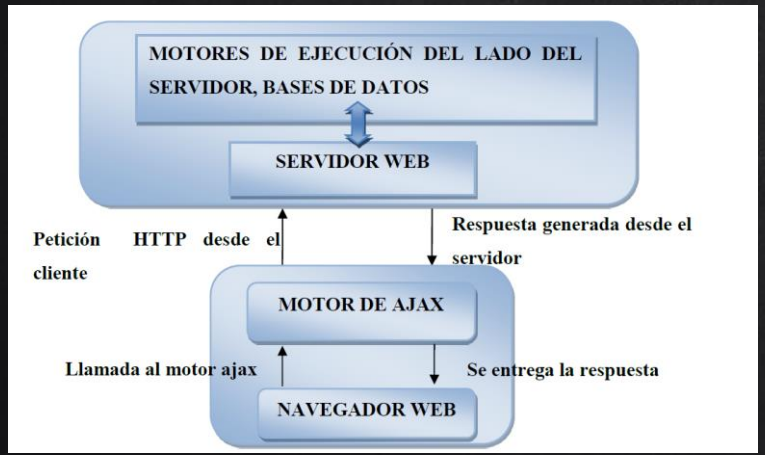


En este esquema síncrono la página que tuviese cargada el servidor se sustituye por la nueva generada, que debe ser cargada completa desde el servidor.

Para evitar recargas completas de las páginas en los casos en los que es una pequeña parte la que ha cambiado se utiliza la **comunicación asíncrona**.

Tipo de comunicación

En la comunicación asíncrona tan solo se recarga aquella parte de la página que ha cambiado y que debe ser generada de nuevo por el servidor.



En este esquema hay un módulo en el cliente encargado de detectar cuando se realiza alguna interacción sobre la página web que provoque algún cambio en la misma.

En el caso de que suceda, es el propio módulo Ajax el que se encargará de realizar la petición al servidor.

Y será el mismo módulo Ajax el que entregue y modifique en la página web únicamente la parte que ha sido modificada.

Herramientas de Desarrollo

Editor de texto

- Edición de código en diferentes lenguajes.
- Sintaxis de colores.
- Verificación de la sintaxis.
- Diferencia comentarios del resto de código.
- Genera partes de código automáticas.
- Utilidades adicionales.

Ejemplos de editores:

- Windows: Notepad++, Visual Studio Code, Eclipse, Netbeans...
- MacOs: Sublime Text, Aptana Studio, Eclipse, Netbeans...
- Linux: KompoZer, Amaya, Quanta Plus, codetech...:

Herramientas de Desarrollo

Editor de texto

Aunque JavaScript se puede escribir utilizando el bloc de notas, con el paso de los años han ido surgiendo distintos editores que facilitan el trabajo.

¿Mejores editores 2024 ?

- ✗ Visual Studio Code
- ✗ Sublime Text
- ✗ Atom
- ✗ PyCharm de JetBrains
- ✗ Brackets

- https://codepen.io/VincentGarreau/pen/bGxvQd

Herramientas de Desarrollo

Navegador web:

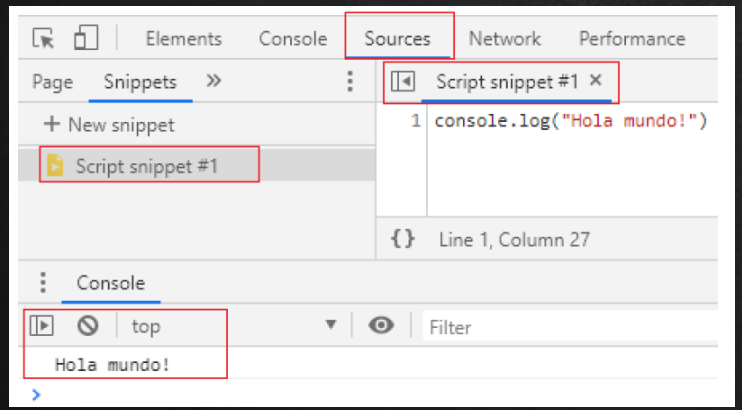
- No necesita Internet para probar los scripts de Javascript.
- Inspector de elementos HTML.
- Depurador.
- Editor de estilos.
- Otras funciones.

Ejemplos de navegadores:

- Mozilla Firefox, Google Chrome, Safari, Opera, Internet Explorer, etc.

Herramientas de Desarrollo

Las DevTools del navegador permiten escribir y probar código:



Dentro de la pestaña "Sources" hay opciones para escribir código y guardar en snippets (fragmentos) y ejecutar o

escribir directamente en la consola.

https://www.youtube.com/playlist?list=PLf8XMtbjh0dUkSTouz5cyvI1xEM0gLJ7h

Herramientas de Desarrollo

Usar el debug en Chrome

- Depurar desde tu servidor local o hosting
- Marcar un breakpoint antes de comenzar depuración
- Para recorrer línea por línea de código presionamos F11
- Para ir de breakpoint a breakpoint presionamos F8

Dentro de la pestaña “Sources” tras presionar F12 o Inspeccionar.

Herramientas de Desarrollo

¿Qué editor vamos a utilizar?

Instalar node.js desde:

https://nodejs.org/es/es/

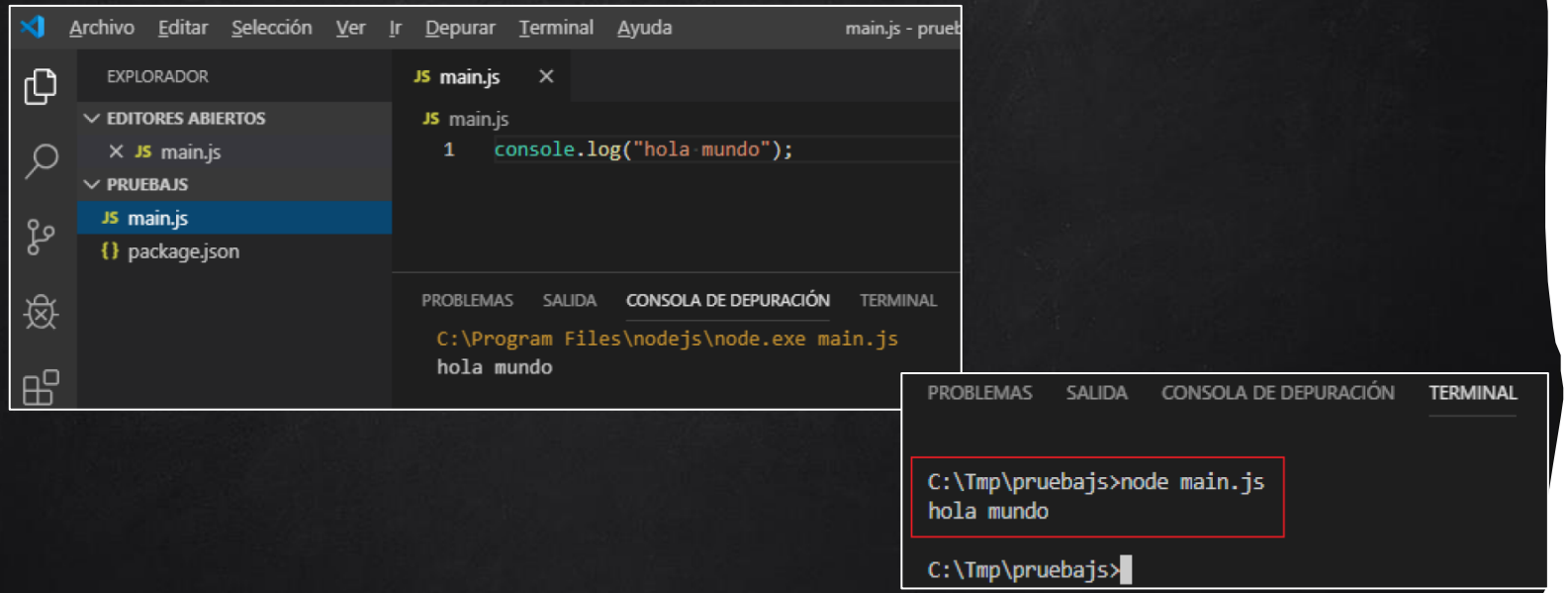
(mejor la versión LST)

Instalar VSCode descargándolo desde:

https://code.visualstudio.com/download

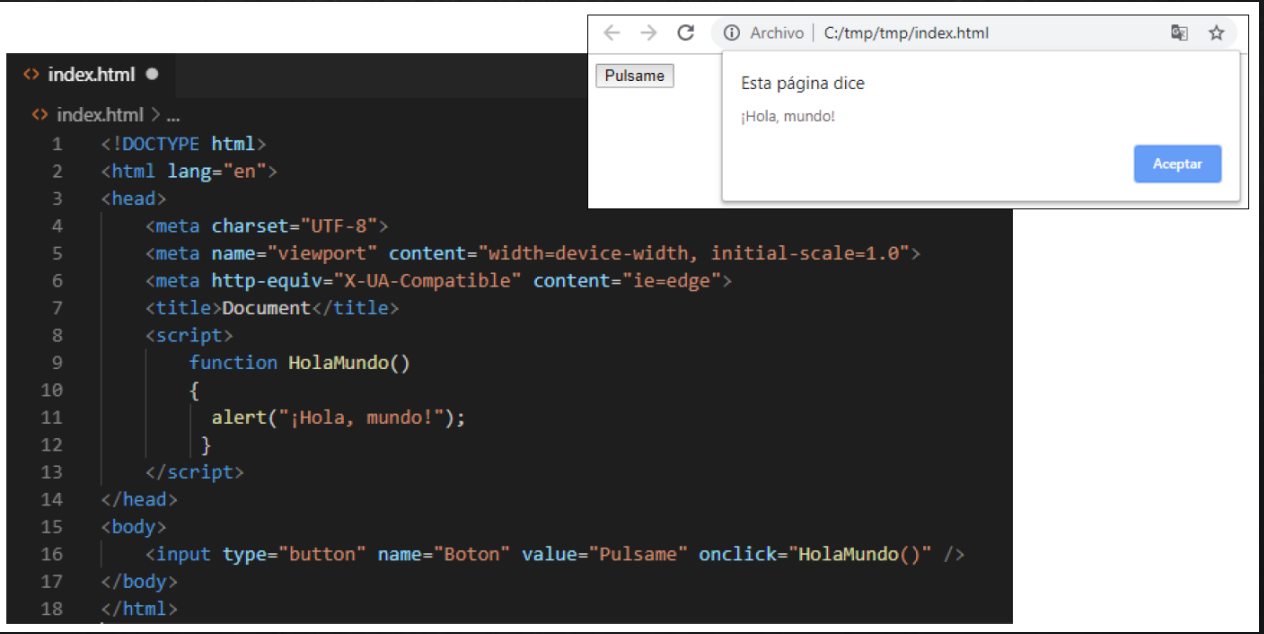
Primer Script

Primer Script independiente



Primer Script

Primer Script en un HTML

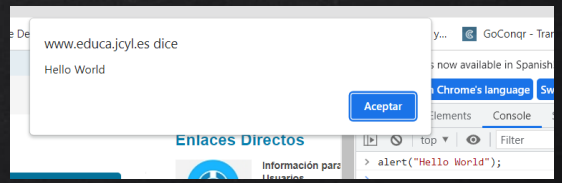


Primer Script

Donde se puede escribir un script (en el cliente)

- En cualquier lugar de la página HTML (en la cabecera o en el cuerpo). Ej <script> … </script>

- En un fichero . js utilizando el atributo " src ".
  Ej :<script src ="foo.js"></script>



- En la consola del navegador:
- Ej: alert ("Hello World");

- En una etiqueta utilizando el atributo " href " o un atributo de evento.
  Ej: <a href javascript:alert Hello Press />
  <input type="button" value="Press" onclick="alert ('Hello') " />

Primer Script

Donde se puede escribir un script (en el cliente)

- En un fichero . js utilizando el atributo " src ".
  Ej :<script src ="foo.js"></script>

VENTAJAS

- Carga más rápida de páginas.
- Separación entre estructura y comportamiento.
- Compartición de código entre páginas.
- Facilidad para depuración de errores.
- Modularidad.
- Seguridad

Primer Script

A tener en cuenta!

El navegador procesa (o parsea) cada página de arriba a abajo y cuando encuentra una etiqueta "script" se detiene a descargar y procesar su código.

Si se incluye el código en la cabecera, no se visualiza la página hasta finalizar la ejecución del script. Si el script es lento la página se bloquea.

Si se incluye el código al final de la etiqueta "body", no se descargan los ficheros JavaScript ni se procesará su código hasta que no finalice el parseo.

La recomendación actual es incluir las etiquetas de script en la cabecera y añadir los atributos " async " o defer " para mejorar la experiencia de usuario.

Primer Script

A tener en cuenta!

1. <script> (normal): el análisis HTML se detiene, se descarga el archivo (si es un script externo), se ejecuta el script y después se reanuda el análisis HTML.
2. <script async>: el script se descarga de forma asíncrona, es decir, sin detener el análisis HTML, pero una vez descargado, si se detiene para ejecutar el script. Tras la ejecución se reanuda el análisis HTML. Sigue existiendo un bloqueo en el renderizado pero menor que con el comportamiento normal. No se garantiza la ejecución de los scripts asíncronos en el mismo orden en el aparecen en el documento.
3. <script defer>: el script se descarga de forma asíncrona, en paralelo con el análisis HTML, y además su ejecución es diferida hasta que termine el análisis HTML. No hay bloqueo en el renderizado HTML. La ejecución de todos los scripts diferidos se realiza en el mismo orden en el que aparecen en el documento.

Primer Script

A tener en cuenta!

1. defer parece la mejor opción de forma general. Salvo que el script manipule o interaccione con el DOM antes de DOMContentLoaded ($(document).ready en jQuery). También sería la mejor opción si el script tiene dependencias con otros scripts y es importante el orden en el que se ejecuta cada uno.
2. async sería ideal para scripts que manipulan o interaccionan con el DOM antes de DOMContentLoaded y/o que no tienen dependencias con otros scripts.
3. Seguir utilizando JS en su forma predeterminada sería la última opción. Si el script es pequeño, preferible inline, ya que el análisis HTML se detiene pero sería una interferencia muy pequeña en comparación con la solicitud y descarga del archivo.

Primer Script

A tener en cuenta!

Los atributos “ async ” y defer ” permiten incluir los scripts en la cabecera y descargar el código tan pronto como sea posible sin bloquear el parser .

Ejemplo:

<script async src=“path/to/script1.js" ></

O

<script defer src=“path/to/script1.js" ></script> ← Preferentemente

Primer Script

¿Qué podemos hacer para proteger el código JavaScript?

- ✗ **El código en JavaScript no se puede proteger: está accesible y visible a través de un navegador.**
- ✗ ¿Qué podemos hacer para protegerlo o demostrar que ha sido elaborado por nosotros?
  - Incluir mensaje de Copyright
  - Ofuscar el código
    - www.javascriptobfuscator.com
    - https://www.welivesecurity.com/es/recursos-herramientas/ofuscacion-de-codigo-arte-ciberseguridad/

Bibliografía

- Donde encontrar ayuda y documentación
  - https://developer.mozilla.org/en-US/
  - https://javascript.info/
  - https://www.w3schools.com/js/
- Otros recursos:
  - https://www.freecodecamp.org/
  - https://flaviocopes.com/
  - https://lenguajejs.com/
  - https://www.youtube.com/playlist?list=PLlrxD0HtieHhW0NCG7M536uHGOtJ95Ut2

Gracias!

Preguntas?

Laurai.lopnav@educa.jcyl.es

Laura.lopez@iesjulianmarias.es