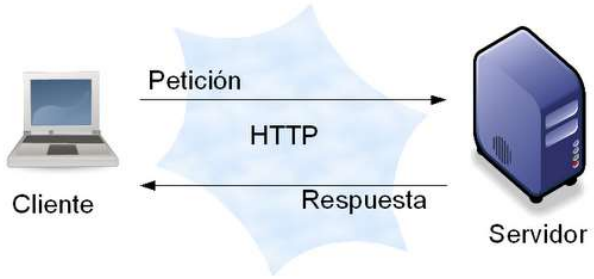
## Tabla de contenido

# UT1: Arquitecturas y lenguajes de programación en clientes web.

En esta unidad introduciremos las distintas arquitecturas y lenguajes de programación en clientes web.

## El modelo cliente-servidor

El modelo cliente-servidor es un modelo de comunicación que permite la distribución de tareas dentro de una red de ordenadores. En este modelo, los ***clientes*** son los **dispositivos que solicitan servicios**, y los ***servidores*** son los dispositivos que proporcionan esos servicios.




La interacción entre clientes y servidores se basa en peticiones y respuestas. Los clientes envían peticiones a los servidores, y los servidores responden a esas peticiones.

El modelo cliente-servidor es uno de los modelos de comunicación más utilizados en la actualidad. Se utiliza en una amplia gama de aplicaciones, como el correo electrónico, la navegación web, el intercambio de archivos y las aplicaciones empresariales.

Las características principales del modelo cliente-servidor son:

- **Centralización de recursos**: Los servidores centralizan los recursos y aplicaciones, lo que permite que sean accesibles a múltiples clientes.
- **Escalabilidad**: El modelo cliente-servidor es escalable, es decir, puede adaptarse a las necesidades cambiantes de las aplicaciones.
- **Seguridad**: El modelo cliente-servidor puede proporcionar un mayor nivel de seguridad que otros modelos de comunicación, ya que los recursos y aplicaciones se encuentran en un único servidor.

Algunos ejemplos de aplicaciones que utilizan el modelo cliente-servidor son:

- ***Correo electrónico***: Los clientes de correo electrónico envían peticiones a los servidores de correo electrónico para enviar y recibir mensajes de correo electrónico.
- ***Navegación web***: Los navegadores web envían peticiones a los servidores web para cargar páginas web.
- ***Intercambio de archivos***: Los clientes de intercambio de archivos envían peticiones a los servidores de intercambio de archivos para descargar o cargar archivos.

- ***Aplicaciones empresariales***: Las aplicaciones empresariales, como los sistemas de gestión de inventario o los sistemas de gestión de recursos humanos, suelen utilizar el modelo cliente-servidor.

El modelo cliente-servidor ofrece una serie de ventajas respecto a otros modelos de comunicación, como el modelo punto a punto. Algunas de estas ventajas son:

- **Eficiencia**: El modelo cliente-servidor permite que los recursos y aplicaciones se utilicen de forma más eficiente, ya que no es necesario que cada cliente tenga acceso a todos los recursos.
- **Seguridad**: El modelo cliente-servidor puede proporcionar un mayor nivel de seguridad que otros modelos de comunicación, ya que los recursos y aplicaciones se encuentran en un único servidor.
- **Facilidad de mantenimiento**: El modelo cliente-servidor facilita el mantenimiento de las aplicaciones, ya que los cambios se pueden realizar en un único lugar, el servidor.

Sin embargo, el modelo cliente-servidor también tiene algunas desventajas, como:

- **Coste**: El modelo cliente-servidor puede ser más caro que otros modelos de comunicación, ya que requiere la adquisición de servidores y otros equipos.
- **Complejidad**: El modelo cliente-servidor puede ser más complejo que otros modelos de comunicación, ya que requiere la coordinación entre clientes y servidores.

En general, el modelo cliente-servidor es una opción muy adecuada para aplicaciones que requieren la distribución de recursos y aplicaciones en una red de ordenadores.

## El navegador o browser

El cliente web más habitual es el navegador web (que llamaremos de aquí en adelante **browser o navegador**). Este programa permite que el usuario escriba un localizador universal de recurso (llamado URL) y se encarga de generar la petición en internet. Serán los routers y máquinas de la web los encargados de redirigir esta petición al servidor.
El servidor web por su parte contiene un repositorio de los recursos disponibles para peticiones (páginas web, hojas de estilo, imágenes, vídeos, etc.). Cuando recibe una petición envía al cliente el/los recursos solicitados).
En la actualidad los navegadores son clientes web complejos. No sólo interpretan texto sino que cuentan con intérpretes de lenguajes de programación especiales, cuyo código puede insertarse dentro de las páginas web para darles dinamismo. A estos lenguajes se les suele llamar lenguajes de scripts.

Según W3Counter (compañía que suministra un sistema gratuito de estadísticas para sitios web) la siguiente imagen muestra una estadística del uso de los navegadores durante el mes de agosto de 2023 (https://www.w3counter.com/globalstats.php?year=2023&month=8)

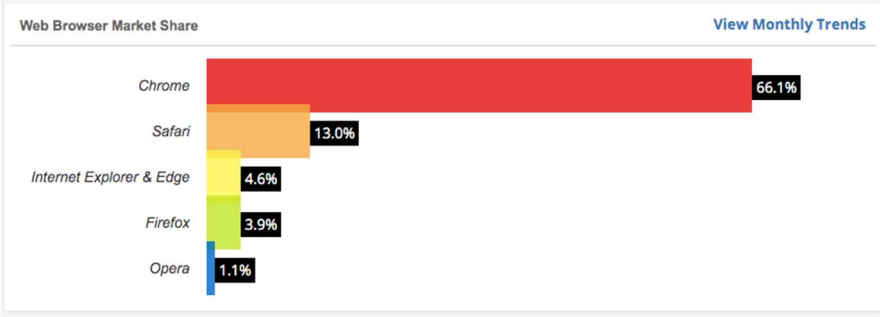



*Imagen 1: Uso de los navegadores durante el mes de agosto de 2023*

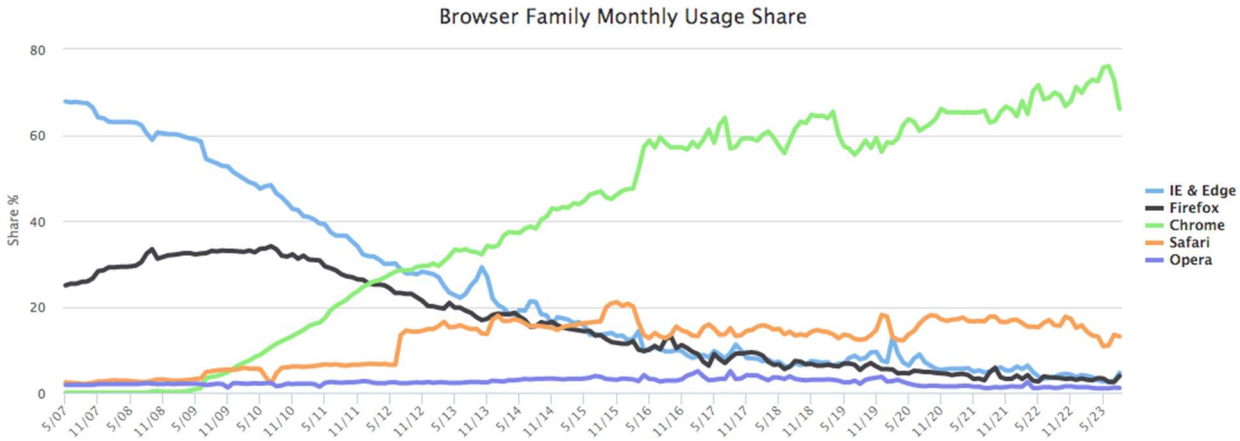



*Imagen 2: Evolución de la popularidad de los navegadores desde 2007 (Fuente:)*

Otra página de interés es https://caniuse.com/ (Can I use) que permite verificar la compatibilidad de las tecnologías web (HTML5, CSS3, JavaScript) con los diferentes navegadores. Esta información es importante para asegurar que los sitios web se vean y funcionen correctamente para todos los usuarios, independientemente del navegador que utilicen.

Can I use proporciona una lista de las tecnologías web más populares, junto con su nivel de compatibilidad con cada navegador. También proporciona información sobre las últimas versiones de las tecnologías web, así como información sobre las versiones obsoletas.

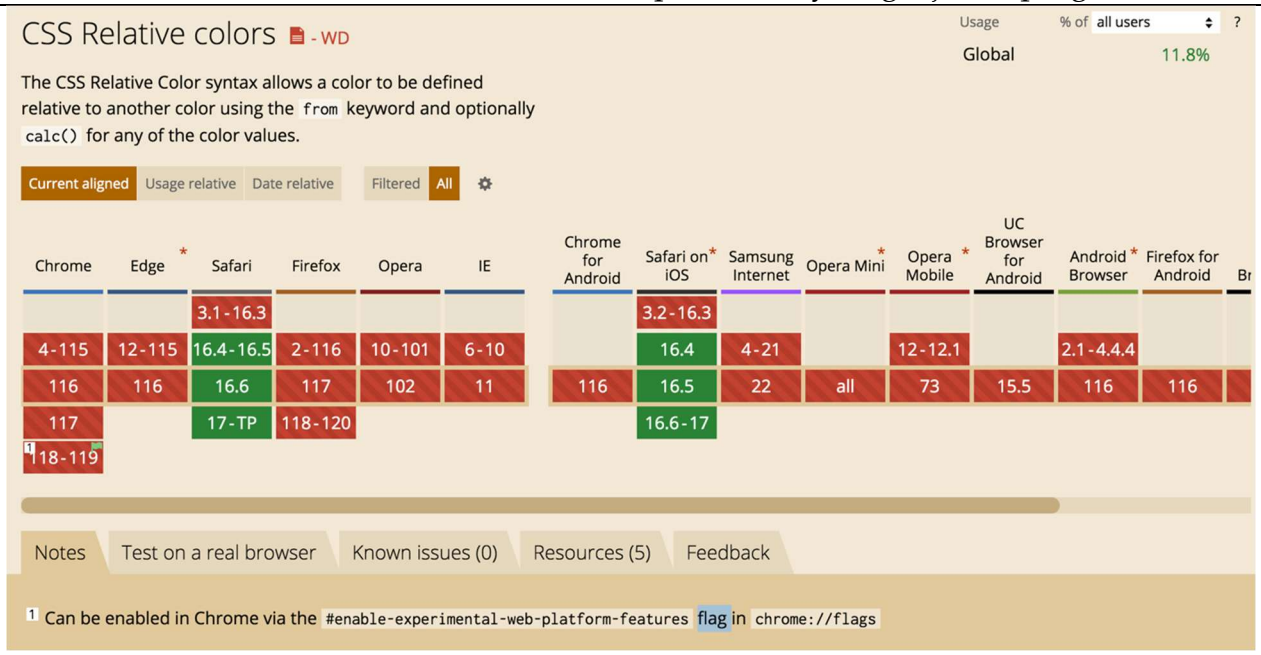



*Imagen 3: Compatibilidad para los colores relativos de CSS (Septiembre 2023)*

Por ejemplo, el siguiente código establecería el color del texto del encabezado h1 en un negro 50% más claro:

```
h1 {color: lighten(#000, 50%);}
```

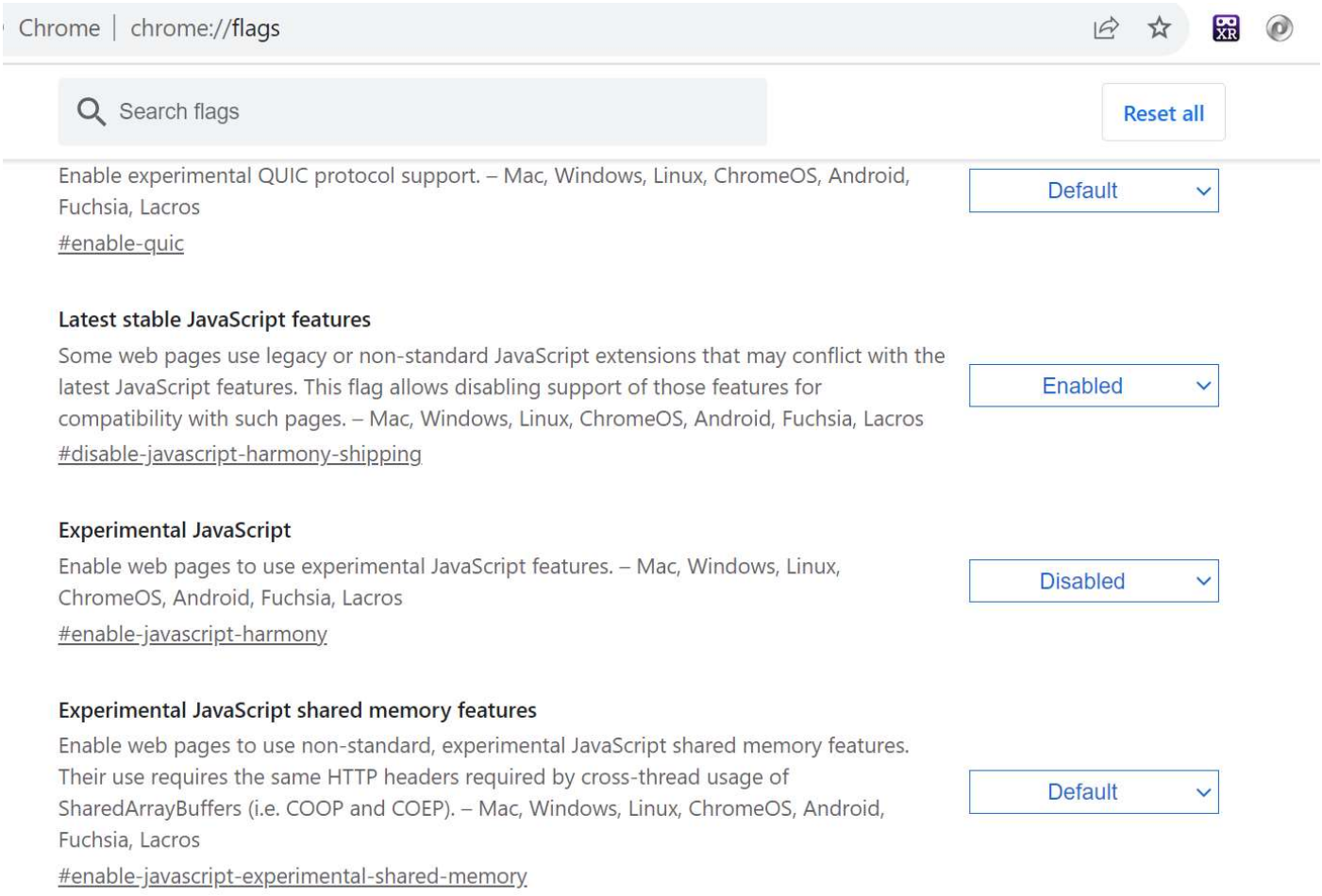
Observa que Chrome no utiliza estas definiciones de CSS, lo hace a través de las famosas flags (banderas).

Las flags de Chrome son funciones experimentales que Google prueba en el navegador Chrome. Estas funciones aún no están disponibles en la versión estable de Chrome, pero puedes activarlas si quieres probarlas. Para acceder a las flags de Chrome, escribe chrome://flags en la barra de direcciones de Chrome. Verás una lista de todas las flags disponibles. Para activar una flag, haz clic en el botón "Activar".

Debes tener en cuenta que las flags son funciones experimentales y pueden causar problemas en el navegador. Si tienes problemas con una flag, puedes desactivarla volviendo a la página de flags y haciendo clic en el botón "Desactivar".

Aquí hay algunos ejemplos de flags de Chrome:

- **WebXR**: Esta flag permite usar realidad virtual y realidad aumentada en Chrome.
- **WebAssembly**: Esta flag permite usar código de máquina en el navegador.
- **Native Client**: Esta flag permite ejecutar aplicaciones nativas en el navegador.
- **Últimas funcionalidades de JavaScript, etc.**: Herramientas con nuevas incorporaciones procedentes de los estándares. Vea la siguiente imagen:




De hecho, Chrome tiene un navegador propio que actualiza a diario para testear últimas novedades, es el Google Chrome Canary. Por cuestiones de tiempo no probaremos su funcionamiento, pero están invitados a probar su funcionamiento.

Más sobre Colores Relativos: https://lenguajecss.com/css/colores/relative-color/

## Modelo de ejecución de código en el cliente

El navegador no deja de ser un programa en la máquina del cliente. El proceso de ejecución se inicia con la solicitud de un recurso web mediante la descripción de su URL.

**Formato de las URL**: servicio://maquina.dominio:puerto/ruta/recurso

Determinadas máquinas llamadas servidores DNS (Domain Name Servers) se encargan de averiguar la dirección IP de la máquina que contiene el recurso solicitado.

Una vez obtenido este dato se genera una petición en la Web a través del protocolo utilizado (que en la URL corresponde a la parte de servicio). Típicamente se trata del protocolo http, aunque pueden ser muchos otros protocolos (ftp, https, etc.).

El siguiente paso pone en marcha la operativa de la Web. Nuestros paquetes http viajarán a través de diferentes máquinas, hubs, switchs, y unas máquinas especiales llamadas routers que lo irán redirigiendo hasta que alcancen el servidor.

El servidor recibe la petición por un puerto de entrada (habitualmente en web utilizamos el puerto 8080). Comprueba si existe /ruta/recurso y en caso afirmativo se vuelve a generar el mismo proceso pero en este caso enviando el recurso hasta la máquina del cliente.

Una vez que el cliente recibe el recurso se carga en el navegador web y aquí se atraviesan varias fases de procesamiento. El navegador se compone de varias partes:

- El **motor de búsquedas** aporta una forma sencilla de comunicación entre el usuario y la máquina. Es la ventana de nuestro navegador, que debe tener una interfaz agradable e intuitiva.
- El **motor de visualización**. También llamado **motor de renderizado**. Se encargará de mostrar los recursos en la ventana del navegador. Para este cometido tendrá que interpretar y mostrar código HTML o incluso XML (en este caso mostrará el árbol sintáctico). También tendrá que aplicar los estilos CSS. Incorpora por lo tanto un **analizador XML**, un **analizador CSS**, y en su caso distintos analizadores para los lenguajes del cliente (típicamente un **analizador JavaScript**).
  - Procesamiento de HTML: El motor de renderizado analiza el código HTML de una página web y construye un árbol de elementos (DOM) que representa la estructura y contenido de la página. Esto incluye etiquetas HTML como encabezados, párrafos, imágenes y enlaces.
  - Estilo y diseño con CSS: El motor de renderizado también procesa las reglas de estilo CSS de una página web y aplica los estilos correspondientes a los elementos HTML. Esto incluye aspectos como colores, fuentes, márgenes y tamaños de elementos.
  - Ejecución de JavaScript: Si una página web contiene código JavaScript, el navegador ejecuta ese código para agregar interactividad y funcionalidad a la página. Esto puede incluir acciones como validación de formularios, animaciones y manipulación del contenido de la página.
- El **motor de comunicaciones**. Es la parte del navegador capaz de trabajar y soportar los distintos protocolos utilizados en el modelo de red. A este modelo se le suele conocer como **modelo TCP/IP** y trabaja con diferentes protocolos como el TCP, UDP, IP, HTTP, HTTPS, FTP, …
- Los **plugins**. Cada fabricante lucha por hacer su navegador más popular. Esto pasa porque el navegador soporte y reconozca el mayor número de tecnologías posible. Para evitar complicar más el programa se suelen añadir a posteriori pequeños módulos que actúan como intérpretes. A cada uno de estos módulos agregados lo llamamos plugin y tiene como misión soportar una tecnología diferente. Por ejemplo podemos tener plugins para soportar otros lenguajes de marcas como VRML, Math-Ml, etc.

**Modelo básico de ejecución de código en el cliente:**

1. Descargar el recurso web.
2. Si se trata de una página web se interpreta el código xml o html (utilizando su analizador correspondiente).
3. Se interpretan los estilos asociados para producir la visualización (renderizado) final.
4. Se dispara el motor de JavaScript una vez finalizada la carga y análisis del recurso.

La última parte (el motor de JavaScript) corresponde a la ejecución de un verdadero lenguaje de programación en el cliente web. Para ser exactos no tendríamos que hablar sólo de JavaScript sino de **lenguajes de scripting**.

## Lenguajes de scripting en el cliente

Un lenguaje de scripting en el cliente es un lenguaje de programación que se ejecuta en el navegador web del usuario. Se utilizan para agregar funcionalidad a las páginas web, como la interacción con el usuario, el procesamiento de datos y la visualización de contenido dinámico.

Los lenguajes de scripting en el cliente se diferencian de los lenguajes de programación del lado del servidor en que se ejecutan en el navegador del usuario, mientras que los lenguajes de programación del lado del servidor se ejecutan en el servidor web.

Algunos ejemplos de lenguajes de scripting en el cliente incluyen:

- JavaScript: el lenguaje de scripting más popular para el desarrollo web.
- VBScript: un lenguaje de scripting desarrollado por Microsoft.
- ActionScript: un lenguaje de scripting desarrollado por Adobe. Compila a JavaScript y se utilizaba principalmente con flash.
- CoffeeScript: un lenguaje de script que se compila en JavaScript.
- TypeScript: un superconjunto de JavaScript que añade tipado estático.
- Dart: un lenguaje de programación que se puede utilizar para desarrollar aplicaciones web, móviles y de escritorio. Desarrollado por Google.

Los lenguajes de scripting en el cliente se utilizan para una amplia gama de tareas, como:

- Agregar interacción con el usuario: JavaScript se puede utilizar para crear formularios interactivos, botones de acción y menús desplegables.
- Procesar datos: JavaScript se puede utilizar para validar los datos de los formularios, calcular resultados y almacenar datos en el navegador.
- Visualizar contenido dinámico: JavaScript se puede utilizar para mostrar imágenes, videos y animaciones en las páginas web.

Los lenguajes de scripting en el cliente son una herramienta poderosa que puede ayudar a crear páginas web más interactivas y atractivas.

### Restricciones de los lenguajes clientes

Estas son algunas de las restricciones más comunes:

- **Rendimiento**: Los lenguajes de scripting suelen ser menos eficientes que los lenguajes de programación compilados, como C++ o Java. Esto se debe a que los lenguajes de scripting se interpretan en tiempo de ejecución, en lugar de compilarse en código máquina.
- **Seguridad**: Los lenguajes de scripting pueden ser menos seguros que los lenguajes de programación compilados. Esto se debe a que los lenguajes de scripting suelen ser más flexibles, lo que puede hacer que sean más vulnerables a ataques. Por eso **NO pueden acceder** libremente a los recursos de la máquina del cliente. Imagina la situación: eres un usuario de la web, te conectas y descargas una página. En medio de esa página hay código programado y ese código accede tranquilamente a tu disco duro y borra archivos. Si esto fuera posible dejaríamos un importante hueco de seguridad y nadie navegaría tranquilo.

  Para evitar esto se impuso a los diseñadores de lenguajes de scripting esta restricción: el acceso a la máquina del cliente debe estar limitada y ser muy restringida. Sí se puede, por ejemplo, escribir pequeños ficheros de texto con información del servidor (lo que se conoce como cookie).

Otra restricción, los scripts siguen una política llamada **mismo origen**. Esto implica que los scripts de un sitio web no tendrán acceso a los datos de otros sitios web (envío de información en formularios, contraseñas, cookies). Las páginas web de distintos dominios no son accesibles entre sí.

- **Escalabilidad**: Los lenguajes de scripting pueden ser menos escalables que los lenguajes de programación compilados. Esto se debe a que los lenguajes de scripting suelen ser más ligeros, lo que puede dificultar su uso en aplicaciones complejas.

## Historia de JavaScript

JavaScript fue desarrollado originalmente por Brendan Eich de Netscape con el nombre de Mocha, el cual fue renombrado posteriormente a LiveScript, para finalmente quedar como JavaScript.

La primera versión de JavaScript fue lanzada en 1995 como parte del navegador web Netscape Navigator 2.0. JavaScript rápidamente se convirtió en un lenguaje popular para el desarrollo web, ya que permitía a los desarrolladores agregar funcionalidad interactiva a sus páginas web.

En 1996, Microsoft lanzó su propio lenguaje de scripting para el navegador web Internet Explorer 3.0, llamado JScript. JScript era compatible con JavaScript, pero tenía algunas características adicionales.

En 1997, la European Computer Manufacturers Association (ECMA) publicó el estándar ECMAScript 1.0, que definía la sintaxis y las características de JavaScript. Este estándar ha sido actualizado varias veces desde entonces, la última versión es ECMAScript 2023.

A lo largo de los años, JavaScript ha seguido evolucionando y ha agregado nuevas características y funcionalidades. Hoy en día, JavaScript es el lenguaje de scripting más popular para el desarrollo web, y se utiliza en una amplia gama de aplicaciones, desde sitios web simples hasta aplicaciones web complejas.

Aquí hay algunos de los hitos más importantes en la historia de JavaScript:

- 1995: Brendan Eich crea Mocha, que más tarde se renombra a LiveScript y finalmente a JavaScript.
- 1996: Microsoft lanza JScript, un lenguaje de scripting compatible con JavaScript.
- 1997: ECMA publica el estándar ECMAScript 1.0.
- 2003: Mozilla lanza Firefox, un navegador web que utiliza el motor de JavaScript Gecko.
- 2004: Google lanza Chrome, un navegador web que utiliza el motor de JavaScript V8.
- 2009: Node.js se lanza al público, lo que permite ejecutar JavaScript en el lado del servidor.
- 2015: ECMA publica el estándar ECMAScript 6, que introduce nuevas características, como clases y módulos.
- 2023: ECMA publica el estándar ECMAScript 2023, que introduce nuevas características, como funciones de flecha y módulos de importación. Se puede encontrar las nuevas especificaciones en https://tc39.es/ecma262/

JavaScript ha tenido un impacto significativo en el desarrollo web. Ha permitido a los desarrolladores crear aplicaciones web más interactivas y atractivas, y ha democratizado el desarrollo web, haciéndolo más accesible a un público más amplio.

### ECMAScript

ECMAScript es la especificación donde se describe los detalles de cómo debe funcionar y comportarse JavaScript en un navegador. Es un intento por unificar el comportamiento de los navegadores. Ocurre que el nivel de cumplimiento de los estándares suele ser un porcentaje de todo lo que se detalla en estos estándares.

En cada nueva versión de ECMAScript, se modifican detalles sobre Javascript y/o se añaden nuevas funcionalidades, manteniendo Javascript vivo y con novedades que lo hacen un lenguaje de programación moderno y cada vez mejor preparado para utilizar en el día a día.

Nota que de ECMAScript 6 en adelante, se toma como regla nombrar a las diferentes especificaciones por su año, en lugar de por su número de edición. Aunque en los primeros temas los mencionaremos indiferentemente, ten en cuenta que se recomienda utilizar ECMAScript 2015 en lugar de ECMAScript 6 (ES6).

Una buena forma de conocer en que estado se encuentra un navegador concreto en su especificación de ECMAScript es consultando la tabla de compatibilidad Kangax (https://kangax.github.io/compat-table/es6/). En dicha tabla, encontramos una columna «Desktop browsers» donde podemos ver el porcentaje de compatibilidad con las diferentes características de determinadas especificaciones de ECMAScript.

Y en qué versión comenzamos a programar: se pueden tener políticas diferenciadas a este respecto, se puede programar a una versión ampliamente implementada por los navegadores, con lo que no tendíamos que considerar posibles excepciones. Utilizar la última versión y que sea otro software el que hace una traducción a una versión más reconocida (https://babeljs.io/). También podemos utilizar un framework

Para ver más: https://lenguajejs.com/javascript/introduccion/ecmascript/

## Framework para desarrollos en cliente

Un framework es un software que proporciona una estructura para el desarrollo de aplicaciones web. Un framework típicamente incluye una serie de componentes personalizables y reutilizables que se pueden utilizar para construir y mantener un sitio web.

Los frameworks son muy populares entre los programadores porque permiten centrarse en la lógica del funcionamiento de un sitio web, en lugar de gastar tiempo en la escritura de códigos que se pueden reutilizar. Además, los frameworks hacen que sea más fácil la organización de los módulos web y disminuyen los intentos de prueba y error para comprobar el funcionamiento del código.

En general, un framework consta de una serie de librerías, extensiones, plugins y otras funcionalidades que se pueden utilizar para construir y mantener un sitio web. Los frameworks también suelen incluir plantillas que hacen más fácil la organización de los módulos web.

Algunos de los frameworks más populares para el desarrollo web son:

Aquí hay algunos otros frameworks populares para el desarrollo web del lado del cliente:

1. **React**: Técnicamente, React es más una librería JavaScript de front-end que un framework de desarrollo. Sin embargo, muchos desarrolladores web consideran que también encaja en esta última categoría, gracias a su arquitectura basada en componentes. Fue uno de los primeros frameworks en aprovechar un Modelo de Objetos del Documento (DOM) virtual, que permite realizar cambios en los datos de un documento sin afectar a su presentación en el navegador.

Creado en 2013 y mantenido por Facebook, React sirve como un marco de trabajo de código abierto que se compone de un ecosistema de bibliotecas y componentes reutilizables de la Interfaz de Usuario (UI), que se pueden utilizar tanto para proyectos de desarrollo web del lado del servidor como del lado del cliente. Por ejemplo, es una poderosa herramienta para construir sitios de comercio electrónico y de venta al por menor.

Parte de la popularidad de React también se puede atribuir a su rápido rendimiento y escalabilidad. Además de funcionar bien junto con los esfuerzos de optimización de motores de búsqueda (SEO), utiliza el renderizado del lado del servidor (SSR) en el que se basan muchas plataformas sociales, incluyendo Instagram y Twitter.

Otro de los beneficios únicos de usar React es que es fácil de aprender si ya estás familiarizado con HTML y CSS. También utiliza la sintaxis JSX que hace que sea fácil de aprender, y está respaldado por una comunidad de desarrolladores sólida y fiable.

2**. Angular**: También conocido como AngularJS, el framework Angular proviene de Google, lo que ayuda a consolidarlo como una de las opciones más populares que hay. Este framework JavaScript de código abierto es particularmente útil si estás construyendo aplicaciones web y sitios web a gran escala y de alto rendimiento. Incluso se ha utilizado para crear sitios como Netflix.

Aunque no es del todo un marco de trabajo de pila completa, Angular se puede utilizar como un marco de trabajo front-end para la construcción de aplicaciones del lado del cliente y páginas web de aplicación única (tanto para móviles como para escritorio). Además de HTML y CSS, Angular utiliza Typescript, que es un superconjunto de JavaScript. También viene con herramientas que te ayudan a lograr una codificación consistente en tus proyectos.

Hay algunas desventajas en Angular. Por ejemplo, es mucho más grande que otros frameworks, y tampoco es tan amigable con el SEO. Sin embargo, hay un montón de recursos y tutoriales disponibles en línea que pueden ayudarle a sacar el máximo provecho de sus características.

**3. Vue.js**: Vue.js es otro popular framework de JavaScript para el desarrollo web. Se considera una opción progresiva que incluye una arquitectura de componentes, y un ecosistema escalable que es útil para la construcción de aplicaciones front-end.

Una de las mayores ventajas de utilizar Vue.js es su versatilidad. Dependiendo de la complejidad de tu proyecto, puedes utilizarlo como una librería JavaScript o como un completo framework front-end.

Una desventaja a tener en cuenta es que no está soportado por algunos de los mayores gigantes digitales como Facebook y Google. Sin embargo, como marco de trabajo, es generalmente fácil de usar.

Si ya sabes HTML, CSS y JavaScript, puedes comenzar a construir tus aplicaciones casi de inmediato. También hay un amplio surtido de cursos en vídeo, guías y documentación a la que puedes acceder para guiarte.

4. **Ember.js**: Este framework de JavaScript, lanzado por primera vez en 2011, es popular entre los desarrolladores experimentados. Cuenta con un impresionante conjunto de características y herramientas básicas que facilitan la construcción de aplicaciones web complejas.

Utilizado por empresas como Netflix, LinkedIn y Apple, Ember se jacta de ser un «framework para desarrolladores ambiciosos». Cuenta con una comunidad masiva, solidaria y útil que se compone de desarrolladores dedicados y apasionados por su capacidad para mejorar su productividad.

Algunas de las mayores ventajas de usar Ember son sus características organizativas y su avanzado sistema de gestión de versiones. El framework también incluye útiles capacidades de vinculación, como ofrecer la opción de crear propiedades a partir de funciones.

Por supuesto, teniendo en cuenta que es un framework orientado a desarrolladores experimentados para construir aplicaciones modernas, puede no ser la mejor opción para los principiantes. Sin embargo, si quieres darle una oportunidad a Ember, hay un montón de artículos de documentación y recursos incluidos en el sitio web principal para empezar.

5. **Backbone.js**: Backbone.js es un framework de JavaScript que se utiliza para construir aplicaciones web del lado del cliente. Backbone.js utiliza un enfoque basado en eventos para la construcción de aplicaciones web, lo que significa que las aplicaciones web responden a eventos específicos del usuario. Backbone.js también es muy popular entre los desarrolladores web debido a su facilidad de uso y su capacidad para crear aplicaciones web dinámicas y receptivas.

Es importante tener en cuenta que la elección del framework correcto depende del proyecto específico y del conjunto de habilidades del desarrollador. Sin embargo, estos frameworks son algunos de los más populares y utilizados por los desarrolladores web.

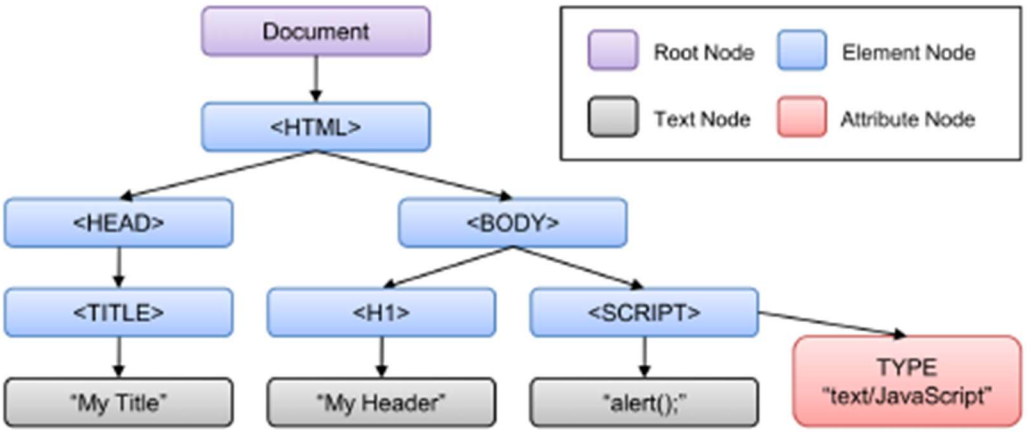
## Páginas Estáticas vs Páginas Dinámicas

Cuando se define una página web en modo estático su contenido no se modifica una vez se ha renderizado en el navegador. Básicamente son páginas que utilizan para mostrar información que no cambia en un breve período de tiempo. Cualquier modificación requiere que se actualice le código HTML + CSS a mano.

Los lenguajes de scrpting permiten modificar dicha página para que pueda modificarse todo o parte de la información sin tener que modificar, a mano, el código HTML de la página. A esto se les conoce como páginas dinámicas. Para ello el navegador representa el código HTML como una estructura en árbol (árbol DOM) a la que se le modifica su comportamiento.

arbol_dom.html

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Mi título</title>
  </head>
  <body>
    <h1>Mi titulo</h1>
    <script type="text/JavaScript">
      alert("Hola Mundo");
    </script>
  </body>
</html>
```




## ¿Cómo agregar código JavaScript a una página web?

Aquí hay algunos ejemplos de cómo agregar scripts a una página web HTML:

**Script incrustado**: Puede agregar un script directamente en el archivo HTML utilizando la etiqueta `<script>`. Aquí hay un ejemplo de cómo agregar un script que muestra una ventana emergente cuando se carga la página:

```
<!DOCTYPE html>
<html>
<head>
        <title>Script incrustado</title>
</head>
<body>

        <h1>Script incrustado</h1>

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

</body>
</html>
```

**Script externo**: Puede agregar un script externo a su archivo HTML utilizando la etiqueta `<script>` y el atributo `src`. Aquí hay un ejemplo de cómo agregar un script externo que muestra una ventana emergente cuando se carga la página:

```html
<!DOCTYPE html>
<html>
<head>
        <title>Script externo</title>
        <script src="mi_script.js"></script>
</head>
<body>

        <h1>Script externo</h1>

</body>
</html>
```

En este ejemplo, el archivo `mi_script.js` contiene el siguiente código:

```
alert("¡Hola, mundo!");
```

### ¿Y dónde se debe invocar a un script dentro del HTML?

Lee el siguiente artículo, te ayudará a entender:
https://blog.desafiolatam.com/javascript-en-el-head-o-en-el-cierre-del-body-estas-equivocado/#comments

## La consola

La consola proporciona una serie de funciones que te permiten interactuar con ella y depurar tu código. A continuación, se muestra una lista de algunas de las funciones más comunes asociadas a la consola:

- **console.log():** Imprime un mensaje en la consola .
- **console.error():** Imprime un mensaje de error en la consola.
- **console.warn():** Imprime un mensaje de advertencia en la consola .
- **console.info():** Imprime un mensaje informativo en la consola .
- **console.debug():** Imprime un mensaje de depuración en la consola .
- **console.assert():** Imprime un mensaje y una traza de pila en la consola si la primera expresión es falsa .
- **console.clear():** Limpia la consola.
- **console.count():** Registra el número de veces que se ha llamado a esta línea con la etiqueta especificada.
- **console.countReset():** Restablece el valor del contador con la etiqueta especificada.
- **console.dir():** Muestra una lista interactiva de las propiedades de un objeto JavaScript especificado
- **console.table()**: Muestra datos tabulares como una tabla.

Estas son solo algunas de las funciones disponibles en la consola. Puedes encontrar más información sobre estas funciones y otras en la documentación oficial de MDN Web Docs 1.

### Aplicar CSS a la consola

Para agregar estilo CSS a los mensajes de `console.log`, puedes usar el parámetro `%c` para pasar estilos CSS. Por ejemplo, para imprimir un mensaje en rojo con un fondo amarillo, puedes hacer lo siguiente:

```
console.log("%cEste es un mensaje en rojo con fondo amarillo", "color: red; background-color: yellow;");
```

Esto imprimirá el mensaje en la consola con el estilo CSS especificado.
También puedes usar múltiples estilos CSS en un solo mensaje de `console.log`. Para hacer esto, simplemente agrega más parámetros `%c` y estilos CSS correspondientes. Por ejemplo:

```
console.log("%cEste es un mensaje en rojo", "color: red;", "%c y este es un mensaje en azul", "color: blue;");
```

Esto imprimirá dos mensajes en la consola, uno en rojo y otro en azul.

## Vocabulario relacionado con la unidad de trabajo

- **Frontend:** se refiere a la parte de la aplicación web que el usuario ve e interactúa. Esto incluye el HTML, CSS y JavaScript que forman la interfaz de usuario. Un desarrollador frontend es responsable de la apariencia y la sensación de la aplicación web. Trabajan con HTML, CSS y JavaScript para crear la interfaz de usuario. Los desarrolladores frontend necesitan ser creativos y tener una buena comprensión de la experiencia del usuario.

- **Backend:** se refiere a la parte de la aplicación web que el usuario no ve. Esto incluye el código del lado del servidor que maneja los datos, la lógica y la seguridad de la aplicación. es responsable de los datos, la lógica y la seguridad de la aplicación web. Trabajan con lenguajes del lado del servidor como PHP, Python o Java para crear el backend.

- **Full stack:** se refiere a un desarrollador que es hábil en ambos aspectos del desarrollo web: frontend y backend. Esto significa que puede desarrollar la aplicación web completa, desde la interfaz de usuario hasta la base de datos.

  Aquí hay algunos ejemplos de proyectos que pueden ser desarrollados por cada tipo de desarrollador:
  - Frontend: Un sitio web de noticias, una aplicación de redes sociales o una aplicación de juegos.
  - Backend: Un sistema de gestión de bases de datos, una aplicación de procesamiento de pagos o un servidor web.
  - Full stack: Un sitio web de comercio electrónico, una aplicación móvil o un sistema de gestión de contenido.

- **DevOps** son profesionales que trabajan para integrar los departamentos de desarrollo y operaciones de TI. Su objetivo es automatizar los procesos de desarrollo, implementación y operaciones de software para mejorar la eficiencia y la confiabilidad.

  Las funciones específicas de un ingeniero de DevOps pueden variar de una organización a otra, pero generalmente incluyen las siguientes:

  Automatizar la entrega de software: Los ingenieros de DevOps utilizan herramientas y procesos para automatizar la entrega de software, desde la codificación hasta la implementación en producción. Esto ayuda a garantizar que el software se pueda entregar de forma rápida y confiable.

  - Gestionar la infraestructura: Los ingenieros de DevOps son responsables de gestionar la infraestructura de TI, incluyendo servidores, redes, almacenamiento y bases de datos. Esto incluye tareas como el aprovisionamiento, la configuración y la optimización de la infraestructura.
  - Implementar la seguridad: Los ingenieros de DevOps son responsables de implementar la seguridad en los sistemas de software. Esto incluye tareas como la detección y prevención de amenazas, así como la gestión de la seguridad de la información.
  - Monitorear y optimizar los sistemas: Los ingenieros de DevOps monitorizan los sistemas de software para detectar problemas potenciales. También optimizan los sistemas para mejorar el rendimiento y la eficiencia.

- **JQuery:** jQuery es una biblioteca de JavaScript que simplifica la interacción con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Query es una biblioteca poderosa y flexible que puede ser utilizada para una amplia gama de tareas. Actualmente está en desuso porque parte de sus funcionalidades se han incluido en el estándar EMAC 6 de JavaScript.

- **Polyfill:** librería o código Javascript que actúa de «parche» o «relleno» para dotar de una característica que el navegador aún no posee, hasta que una actualización del navegador la implemente.

- **Fallback**: un fragmento de código que el programador prepara para que en el caso de que algo no entre en funcionamiento, se ofrezca una alternativa.