Unidad 1: Arquitecturas y Lenguajes de Programación en Clientes Web

Modelo Cliente-Servidor:

Es un modelo de comunicación en el que los clientes solicitan servicios y los servidores los proporcionan.

- Se basa en peticiones y respuestas.
- Se utiliza en aplicaciones como correo electrónico, navegación web y aplicaciones empresariales.

- **Ventajas**: centralización de recursos, escalabilidad y seguridad.

- **Desventajas**: coste y complejidad.

El Navegador (Browser):

El navegador web es el cliente web más común.

- Permite a los usuarios escribir una URL y genera solicitudes web.
- Los servidores web almacenan recursos y responden a las solicitudes.
- Los navegadores interpretan lenguajes de programación, como JavaScript.

- **Estadísticas de Uso de Navegadores:**

    - Se pueden verificar las estadísticas de uso de navegadores, siendo Chrome uno de los más populares.

Compatibilidad de Tecnologías Web:

La página "Can I use" permite verificar la compatibilidad de tecnologías web con diferentes navegadores.

Flags de Chrome:

- Chrome tiene funciones experimentales llamadas "flags" que pueden activarse para probar nuevas características.

Modelo de Ejecución de Código en el Cliente:

- El navegador es un programa en la máquina del cliente.
- Se inicia con la solicitud de un recurso web a través de una URL.
- Los servidores DNS encuentran la dirección IP del recurso.
- Se genera una petición web mediante un protocolo como HTTP.

Lenguajes de Scripting del Lado del Cliente y JavaScript:

- Son lenguajes que se ejecutan en el navegador web del usuario.
  Se utilizan para agregar funcionalidad a las páginas web.
- **Ejemplos** incluyen JavaScript, VBScript, ActionScript, CoffeeScript, TypeScript y Dart.

Limitaciones de los Lenguajes del Lado del Cliente:

- Rendimiento y seguridad

Historia de JavaScript:

- Fue desarrollado por Brendan Eich en 1995.
- Ha evolucionado y se ha convertido en el lenguaje de scripting más popular para el desarrollo web.

ECMAScript:

- Especificación que estandariza el comportamiento de JavaScript en diferentes navegadores.

Frameworks de Desarrollo del Lado del Cliente:

- Proporcionan estructuras para el desarrollo de aplicaciones web.
- **Ejemplos** incluyen React, Angular, Vue.js, Ember.js y Backbone.js.

Páginas Estáticas vs. Páginas Dinámicas:

- Páginas Estáticas: Contenido que no cambia después de la carga inicial.
- Páginas Dinámicas: Utilizan lenguajes de scripting para modificar el contenido.

Cómo Agregar Código JavaScript a una Página Web:

- Puede agregarse mediante etiquetas <script> incrustadas o referenciando archivos externos.
- <scritp src=”script.js”/>

Consola:

- Herramienta de desarrollo para interactuar y depurar código JavaScript en el navegador.

Aplicar CSS a la Consola:

- Se pueden aplicar estilos CSS a los mensajes de la consola.

Vocabulario Relacionado:

- Frontend, Backend, Full Stack, DevOps, jQuery, Polyfill, Fallback.
- Esta unidad introduce conceptos clave sobre el modelo cliente-servidor, el papel del navegador web, lenguajes de scripting en el lado del cliente (con énfasis en JavaScript), la historia de JavaScript, ECMAScript, frameworks de desarrollo y más.

ASYNC VS DEFER JAVASCRIPT

- **"Async"** se utiliza cuando el script no depende del contenido de la página y se puede ejecutar en cualquier momento.

- **"Defer"** se utiliza cuando el script debe ejecutarse después de que se haya analizado todo el contenido HTML de la página.

Ambos atributos **ayudan a evitar bloqueos de renderización** y **mejoran el rendimiento de la página** al permitir la **descarga paralela de recursos**, pero tienen diferentes tiempos de ejecución. La elección entre ellos **depende de las necesidades específicas del script** y su **interacción con el contenido de la página**.