UD01UT01 Arquitecturas y Lenguajes de Programación en Clientes Web

Autor: Pedro Martín Escuela

Actividad 1

1. **¿En qué se diferencian esencialmente ambos modelos en términos de dónde se ejecuta el código y quién tiene acceso a los resultados?**

   Las dierencias entre la ejecución en entorno **servidor** y **cliente**, comienzan en que, el codigo que se ejecuta en entorno del **cliente**, ocurre en el **navegador** del usuario que interactúa con él, mientras que el código en el lado del **servidor** se ejecuta en un **servidor remoto** para gestionar datos, lógica de negocio y la generación de contenido antes de **enviarlo al cliente**. Por esto mismo, el codigo ejecutado en entorno del **cliente**, **puede ser accedido por el usuario**, mientras que el que se ejecuta en entorno de **servidor**, es procesado por este, así que **no es accesible**.

2. **Desde la consola del navegador (F12 → Console), ejecuta el siguiente código JavaScript: console.log('Hola desde el cliente'). Luego, intenta imaginar y explicar por qué no sería posible ejecutar un código PHP directamente en esa misma consola**

   No sería posible por que el codigo de **php se ejecuta en entorno de servidor** y no es accesible por el usuario, al contrario que el de **JavaScript**, que como acabamos de comprobar, **se ejecuta en el cliente**.

3. **Reflexión: Basándote en lo anterior, enumera al menos una ventaja y una desventaja clave de la programación en el cliente (JavaScript) frente a la programación tradicional en el servidor.**

   Una **desventaja** de la ejecución en servidor respecto al cliente, es su **alto coste de mantenimiento** y la **dependencia** de todos los usuarios de este. Aun que por otro lado, una ventaja que tiene, es la **centralización de datos**, lo que implica **más seguridad** y **evita la inconsistencia de datos**.

Actividad 2

La actividad es realizada en el directorio:

```
/ut1/tareas/tarea1/actividad2
```

Actividad 3

1. **Copia y pega el siguiente código HTML y JavaScript en un nuevo archivo. Contiene un error intencionado.**

2. **Abre el archivo en tu navegador y observa el comportamiento incorrecto.**

3. **Usa las Herramientas de Desarrollo (F12) para encontrar y solucionar el error. Sigue el flujo:**

a) Revisa la pestaña "Consola" para ver si hay errores.

b) Si no hay errores evidentes, usa la pestaña "Sources"/"Debugger" para establecer un punto de ruptura (breakpoint) en la función validarFormulario() y ejecuta el código paso a paso, observando los valores de las variables.

La condición del if/else es que si el caracter del arroba está en una posición mayor que 0, este se ejecuta correctamente, y si no, da un error. Como tal, no es un fallo, ya que estaría mal si fuera > -1 por que podríamos poner un correo cuyo primer caracter fuese @ y dejaría enviarlo. Otro error podría ser que no detecta si el correo tiene extensión (.org, .com, etc), o que al enviarse el formulario correctamente, se refresca la página, cosa que impide ver el mensaje de que se ha enviado correctamente.

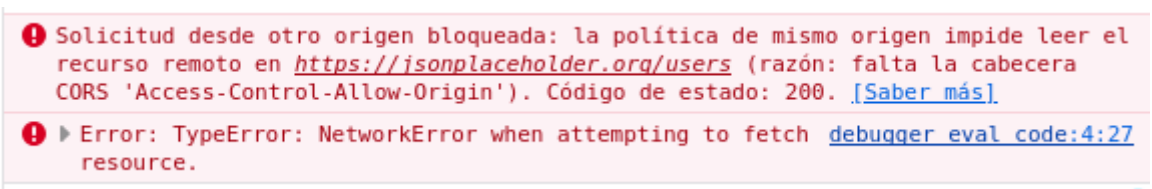
Actividad 4

Los navegadores web ejecutan código JavaScript en un entorno seguro y "enjaulado" conocido como sandbox. Este mecanismo es fundamental para las capacidades y limitaciones de ejecución. Investiga y realiza lo siguiente:

1. Experimento Práctico - La Política del Mismo Origen (Same-Origin Policy):

```
a) Abre las herramientas de desarrollo de tu navegador (F12) y ve a la
pestaña "Consola".

b) Visita un sitio web popular como https://www.wikipedia.org.

c) Intenta ejecutar el siguiente código en la consola para intentar leer
información de otro origen (dominio):

fetch('https://jsonplaceholder.org/users')
.then(response => response.json())
.then(json => console.log(json))
.catch(error => console.error('Error:', error));

d) Observa y analiza: ¿Qué sucede? Si obtienes un error, cópialo y
explícalo brevemente. ¿Qué mecanismo de seguridad del navegador está
impidiendo o permitiendo esta acción?

e) Contrasta: Ahora visita la página de prueba de JSONPlaceholder
https://jsonplaceholder.org y ejecuta el mismo código en su consola.
¿Ocurre lo mismo? ¿Por qué?
```

El error que aparece es de CORSS, que es un bloqueo de seguridad que implementa el navegador cuando una página web intenta acceder a recursos de un dominio diferente al de su origen.



Cuando ejecutamos el codigo en la otra página, esta no da error ya que estos recursos si pertenecen al mismo dominio de su origen. En lugar del error, nos devuelve un array de usuarios pedidos.