****

Módulo Profesional 06:  
Desarrollo web en entorno cliente **Actividad UF4**

**CICLO FORMATIVO DE GRADO SUPERIOR EN**

**DESARROLLO DE APLICACIONES WEB**

**MODALIDAD ONLINE**

**JUAN BELLO FERNANDEZ   
2º DAW**

**Actividades evaluables de la UF4**

|  |  |
| --- | --- |
| Objetivos de la actividad | Metodología |
| Evaluar los conocimientos en el desarrollo web en entorno cliente sobre aplicaciones web interactivas y modelo de objetos del documento (DOM). | * Preparación individual |
| Criterios de evaluación | Entrega |
| Conocimiento en temas relacionados con:   * Utilizar mecanismos de comunicación asíncrona entre cliente-servidor. | * Fecha límite el 15/12/2022. * Subir en el aula en formato PDF el documento propuesto para la actividad. |
| Documentos de referencia | |
| Libro de texto y recursos disponibles en el aula. | |

**Descripción de la actividad**

A continuación, se presenta una serie de ejercicios de contenido teórico y práctico. En el primero de los casos, se proponen tanto preguntas de opción múltiple (utilizar color de resaltado de texto para la respuesta elegida), como preguntas de respuesta breve. Por el otro lado, se pide escribir código a través de la actividad propuesta.

**Desarrollo de la actividad**

**Preguntas de opción múltiple (20%).**

1. **Indica si la siguiente afirmación es verdadera o falsa “Un mecanismo de comunicación asíncrona es aquel en el que el proceso que ejecuta una orden espera a que dicha orden llegue a su receptor para continuar su ejecución”.**
2. Verdadera.
3. **Falsa.**
4. **¿Cuál de las siguientes tecnologías no forma parte de AJAX?**
5. XML
6. **PHP**
7. Javascript
8. DOM
9. **Cuando se utiliza el objeto *XMLHttpRequest*, el contenido asíncrono...**
10. **no es necesario que sea XML**
11. no se carga si no es texto plano
12. tiene que ser obligatoriamente XML
13. sólo puede ser XML o JSON
14. **Indica si la siguiente afirmación se verdadera o falsa: “Para navegadores antiguos, como Internet Explorer 6, el objeto *XMLHttpRequest* se tiene que instanciar como un objeto de tipo *ActiveX*”**
15. **Verdadera**
16. Falsa
17. **¿Cuál de las siguientes opciones es un método del objeto *XMLHttpRequest*?**
18. responseXML
19. status
20. onloadstart
21. **open()**
22. **¿Qué valor de la propiedad *readyState* nos indica que la petición se ha completado y por lo tanto ya podemos cargar el contenido?**
23. true
24. **4**
25. ok
26. 1
27. **En cuanto al envío de parámetros con la petición HTTP, ¿cuál de las siguientes afirmaciones es cierta?**
28. Por motivos de seguridad sólo se puede utilizar el método GET
29. El método POST envía los parámetros dentro de la URL
30. No es posible enviar parámetros
31. **Se puede utilizar tanto el método GET como el método POST**
32. **¿Qué formatos se pueden utilizar para el envío y recepción de información en una conexión realizada con *XMLHttpRequest*?**
33. Solo XML
34. Texto plano y XML
35. Solo JSON
36. **Texto plano, XML y JSON**
37. **El formato JSON...**
38. **es más compacto y ligero que el XML**
39. es complejo de procesar para los navegadores
40. tiene la misma estructura de etiquetas que la XML
41. sólo funciona con Javascript
42. **Indica si la siguiente afirmación se verdadera o falsa: “No existen librerías externas para AJAX, debido a que es una tecnología muy fácil de implementar”.**
43. Verdadera
44. **Falsa**

**Preguntas de respuestas breves (40%)**

1. **Responde a las siguientes preguntas relacionadas con la comunicación asíncrona.**
2. **¿Qué diferencia hay entre la comunicación asíncrona y AJAX?**

La comunicación asíncrona es la metodología que define la posibilidad de hacer peticiones al servidor sin necesidad de refrescar la página, y AJAX es la tecnología que utilizamos para hacerlo posible, por lo que es el conjunto de lenguajes, métodos y propiedades que nos permite programar comunicaciones asíncronas.

1. **Denomina las fases básicas por una comunicación asíncrona con AJAX y las instrucciones, propiedades o métodos que se utilizan en cada fase.**

1. **¿Qué diferencia hay entre el método GET y POST? ¿En qué casos se utiliza uno u otro? Qué diferencias hay en la implementación?**

GET y POST son dos métodos del protocolo HTTP que podemos ver como formas de envío de datos a través de internet. Tanto GET como POST son muy conocidos debido a su uso en formularios.

Por ejemplo:  
 <form action="http://www.algunaweb.com " method="get"> indica que los datos del formulario serán enviados a la url especificada usando el método get.

Un resultado usando el método GET, a modo de ejemplo, podría ser el siguiente:

http://www.algunaweb.com/newuser.php?nombre=Pepe&apellido=Flores&email=h52turam%40uco.es&sexo=mujer

El símbolo ? indica dónde empiezan los parámetros que se reciben desde el formulario que ha enviado los datos a la página.

Después del símbolo ? aparecen parejas de datos con su nombre y valor separadas por el símbolo &(ampersand). Las parejas dato1=valor1, dato2=valor2, dato3=valor3… reflejan el nombre y el valor de los campos enviados por el formulario.

Si en un formulario usamos method="post" los datos se transmiten igualmente, pero no son visibles en la url y sólo pueden ser recuperados a través de código de programación.

**TABLA COMPARATIVA**

|  |  |  |
| --- | --- | --- |
|  | **Con GET** | **Con POST** |
| **Los datos son visibles en la url** | Sí | No |
| **Los datos pueden permanecer en el historial del navegador** | Sí | No |
| **Una url puede ser guardada conteniendo parámetros de un envío de datos** | Sí | No |
| **Existen restricciones en la longitud de los datos enviados** | Sí (no se puede superar la longitud máxima de una url) | No |
| **Se considera preferible para envío de datos sensibles (datos privados como contraseñas, números de tarjeta bancaria, etc.)** | No (los datos además de ser visibles pueden quedar almacenados en logs) | Sí (sin que esto signifique que por usar post haya seguridad asegurada) |
| **Codificación en formularios** | application/x-www-form-urlencoded | application/x-www-form-urlencoded ó multipart/form-data. Se usa multipart para envío de datos binarios, por ejemplo ficheros. |
| **Restricciones de tipos de datos** | Sí (sólo admite caracteres ASC-II) | No (admite tanto texto como datos binarios p.ej. archivos) |
| **Se considera preferible para disparar acciones** | No (podría ser accedido por un robot que dispararía la acción) | Sí (sin que esto garantice que no pueda acceder un robot) |
| **Riesgo de cacheado de datos recuperados en los navegadores** | Sí | No |
| **Posibles ataques e intentos de hackeo** | Sí (con más facilidad) | Sí (con menos facilidad) |

**Actividad práctica (40%)**

1. Se desea crear un web que modifique la información de un archivo HTML por el contenido de un archivo \*.txt externo. Se tiene que hacer con comunicación asíncrona y sin refrescar la página.

Escribe el código necesario para hacer posible el enunciado. El archivo HTML original tiene que contener una capa (<div>) con un texto que, al hacer clic sobre un botón, será modificado por el contenido de un archivo externo.

Tendrás que crear un archivo \*.txt con un texto corto, simulando que es el archivo que hay en el servidor.

Implementa las actuaciones necesarias para que funcione en los diferentes navegadores y versiones.