Práctica de PHP: Juego de Preguntas y Respuestas

Contexto: En esta práctica, vamos a complementar un proyecto web que habéis o estáis desarrollando en JavaScript. El proyecto consiste en un juego de preguntas y respuestas en una página web (**M6**). En esta práctica se utilizará PHP para almacenar y gestionar la puntuación del juego, mostrar mensajes según el desempeño del jugador y garantizar que las respuestas se almacenen y verifiquen adecuadamente.

Objetivo: El objetivo de esta práctica es utilizar PHP para gestionar la puntuación y las funcionalidades del juego de preguntas y respuestas desarrollado en JavaScript. Se debe complementar el proyecto existente con las siguientes funcionalidades en PHP:

Paso 1: Mejorar el Aspecto Visual con HTML y CSS

1. Mejorar la interfaz web del juego de preguntas y respuestas utilizando HTML y CSS con Flexbox para crear un diseño atractivo y responsive.
2. Agregar estilos CSS para darle un aspecto más atractivo visualmente. Utiliza Flexbox para organizar las preguntas y respuestas en un diseño ordenado.

Ejemplo de mejora del aspecto visual utilizando Flexbox en CSS:

```
<div class="container">
  <div class="row">
    <div class="cell">Pregunta 1:</div>
    <div class="cell">¿Cuál es la capital de Francia?</div>
  </div>
  <div class="row">
    <div class="cell">Respuesta 1:</div>
    <div class="cell">
      <input type="radio" name="respuesta1" value="Paris"> París
    </div>
  </div>
  <!-- Otras preguntas y respuestas -->
</div>
```

```
.container {
  display: flex;
  flex-direction: column;
  width: 100%;
  max-width: 400px; /* Puedes ajustar el ancho máximo según tus
necesidades */
  margin: 0 auto; /* Centra el contenido horizontalmente */
  border: 1px solid #ccc;
  padding: 10px;
}

.row {
  display: flex;
  flex-direction: row;
  margin-bottom: 10px;
}

.cell {
  flex: 1;
  padding: 5px;
  border: 1px solid #ccc;
}
```

Paso 2: Transformación de Datos y Gestión de Sesiones

- **Preparación de la Estructura de Datos:** Ajustar el proyecto existente para manejar las preguntas, respuestas y puntuaciones utilizando arrays y objetos en JavaScript.
- **Incorporación de la Puntuación:**
  - Añadir un campo de puntuación dentro de la lógica de JavaScript.
  - Actualizar este campo en tiempo real, incrementándolo o decrementándolo basado en las respuestas correctas o incorrectas del usuario.
- **Creación y Manejo de JSON:**
  - Convertir estos arrays y objetos, que incluyen las puntuaciones, en objetos JSON.
  - Asegurarse de que los objetos JSON estén bien formados y que contengan todos los datos necesarios, como preguntas, respuestas y puntuaciones.
- **Comunicación con el Servidor:**
  - Adaptar la función que maneja el envío de respuestas en JavaScript.
  - Configurar esta función para que envíe el objeto JSON, que ahora también incluye las puntuaciones, al servidor PHP a través de una solicitud HTTP, probablemente usando el método POST.

- **Validación y Seguridad:**
  - Implementar validaciones para asegurarse de que los datos recibidos en el servidor PHP sean consistentes y seguros.
  - Añadir medidas adicionales de seguridad y validación, como la sanitización de datos, para proteger contra posibles vulnerabilidades.

Ejemplo de Transformación de un Array en JSON en JavaScript:

```
// Definir un array de preguntas y respuestas
const preguntasRespuestas = [
  {
    pregunta: "¿Cuál es la capital de Francia?",
    respuesta: "Paris"
  },
  {
    pregunta: "¿Cuál es el río más largo del mundo?",
    respuesta: "Amazonas"
  }
];

// Convertir el array en formato JSON
const jsonPreguntasRespuestas = JSON.stringify(preguntasRespuestas);

// Enviar el JSON al servidor PHP
fetch("procesar_respuestas.php", {
  method: "POST",
  body: jsonPreguntasRespuestas,
  headers: {
    "Content-Type": "application/json"
  }
})
.then(response => response.json())
.then(data => {
  // Manejar la respuesta desde PHP (por ejemplo, mostrar mensajes)
});
```

Paso 3: Procesamiento de JSON en PHP y Gestión de Sesiones

- **Inicio de Sesión:**
  - Al comienzo de cada script PHP, iniciar una sesión utilizando **`session_start()`** para facilitar el manejo de sesiones y datos del usuario durante su interacción con la aplicación.
- **Recepción y Decodificación de JSON:**
  - Adaptar el servidor PHP para recibir el objeto JSON enviado desde el cliente (JavaScript).
  - Decodificar el JSON para poder acceder y manipular los datos, como preguntas, respuestas y puntuaciones.
- **Procesamiento de Respuestas:**
  - Utilizar PHP para procesar las respuestas proporcionadas por el usuario.
  - Comparar las respuestas del usuario con las respuestas correctas almacenadas y calcular la puntuación basada en la cantidad de respuestas correctas e incorrectas.
- **Gestión de Puntuaciones con Sesiones:**
  - Utilizar variables de sesión, como **`$_SESSION['puntuacion']`**, para almacenar y manejar dinámicamente las puntuaciones de los usuarios.
  - Asegurarse de que la puntuación se actualice adecuadamente dentro de la variable de sesión cada vez que el usuario responda una pregunta.
- **Mantenimiento de la Sesión:**
  - A lo largo de la interacción del usuario con la aplicación, mantener la sesión activa para gestionar la puntuación y otros datos relevantes del usuario.
  - Utilizar las sesiones para mantener una experiencia cohesiva y consistente para el usuario mientras interactúa con diferentes partes de la aplicación, como responder preguntas y ver su puntuación.

Ejemplo de Inicio de Sesión en PHP:

```php
// Iniciar una sesión
session_start();

// Resto del código PHP...
```

Ejemplo de Procesamiento de JSON en PHP con Gestión de Sesiones:

```php
// Iniciar una sesión
session_start();

// Recibir y decodificar el JSON enviado desde JavaScript
$jsonPreguntasRespuestas = file_get_contents("php://input");
$respuestas = json_decode($jsonPreguntasRespuestas);

// Procesar las respuestas y calcular la puntuación
$puntuacion = 0;
foreach ($respuestas as $preguntaRespuesta) {
  $pregunta = $preguntaRespuesta->pregunta;
  $respuesta = $preguntaRespuesta->respuesta;

  // Verificar si la respuesta es correcta y actualizar la puntuación
  if (verificarRespuesta($pregunta, $respuesta)) {
    $puntuacion++;
  }
}

// Actualizar la puntuación en la variable de sesión
$_SESSION['puntuacion'] = $puntuacion;

// Enviar una respuesta JSON de vuelta a JavaScript (por ejemplo, mensaje de éxito o fracaso)
$response = array('mensaje' => 'Respuestas procesadas con éxito');
echo json_encode($response);

// Resto del código PHP...
```

Paso 4: Gestionar Mensajes de Resultado

- **Verificación de Respuestas:**
  - Emplear PHP para revisar las respuestas del usuario, comparándolas con las respuestas correctas almacenadas en el servidor.
  - Implementar la lógica necesaria para determinar si el usuario ha respondido todas las preguntas correctamente o ha cometido algún error.
- **Feedback de Desempeño:**
  - Presentar un mensaje de éxito al usuario si todas las respuestas proporcionadas son correctas, celebrando su logro.

    - En caso de respuestas incorrectas, mostrar un mensaje indicando que ha habido errores y animando al usuario a intentarlo de nuevo.
- **Mensajes Adicionales:**
    - Personalizar los mensajes mostrados al usuario basándose en su desempeño. Ejemplos de mensajes incluyen "Has pasado de ronda" o "Has ganado el juego", entre otros, para hacer la experiencia más dinámica y agradable.
    - Utilizar estos mensajes para guiar al usuario a través de las diferentes etapas del juego y proporcionar claridad sobre su progreso y desempeño.
- **Finalización del Juego:**
    - En caso de que el usuario complete el juego exitosamente, asegurarse de concluir la sesión de manera apropiada y guardar cualquier dato necesario para referencia futura o para la generación de estadísticas.
    - Proporcionar opciones para que el usuario pueda reiniciar el juego, ver su puntuación final o realizar otras acciones post-juego.

Ejemplo de PHP para gestionar mensajes:

```php
// Iniciar una sesión
session_start();

// Resto del código PHP...

if ($_SESSION['puntuacion'] == $totalPreguntas) {
  echo '¡Has respondido todas las preguntas correctamente!';
} else {
  echo '¡Inténtalo de nuevo! Has respondido ' . $_SESSION['puntuacion'] .
' preguntas correctamente.';
}
// Resto del código PHP...
```

Paso 5: Tratamiento de Errores

- **Manejo de Errores:**
    - Incorporar una robusta gestión de errores en el código PHP para anticipar y manejar situaciones inesperadas, como variables indefinidas, falta de respuestas y otros posibles problemas.
    - Establecer mecanismos de control para gestionar de manera adecuada los errores, evitando caídas del sistema o comportamientos inesperados que puedan afectar la experiencia del usuario.

- **Validación de Respuestas:**
  - Implementar procesos de validación que revisen las respuestas y entradas del usuario, asegurándose de que estén en un formato aceptable y sean seguras antes de ser procesadas.
  - Proteger el sistema contra posibles ataques o entradas maliciosas, filtrando y limpiando las respuestas recibidas para mantener la integridad y seguridad del juego.
- **Seguridad:**
  - Fortalecer las medidas de seguridad dentro del código, asegurándose de que los datos del usuario y las respuestas estén protegidos contra accesos no autorizados o manipulaciones malintencionadas.
  - Implementar prácticas de codificación segura para garantizar que el juego funcione de manera confiable y segura, protegiendo tanto la información del usuario como la lógica del juego.

Ejemplo de manejo de errores en PHP:

```php
// Iniciar una sesión
session_start();

// Resto del código PHP...
if (isset($_POST['respuesta1'])) {
  $respuesta = $_POST['respuesta1'];
  // Validar la respuesta aquí antes de usarla
} else {
  echo 'Debes seleccionar una respuesta.';
}
// Resto del código PHP...
```

Paso 6: Respuestas Aleatorias y Más Preguntas

- **Aleatorización de Preguntas y Respuestas:**
  - Modificar el código actual para que las respuestas posibles de cada pregunta se muestren en un orden aleatorio cada vez que se carga una pregunta. Esto evitará que los jugadores memoricen la posición de las respuestas correctas.
  - Adaptar el código PHP para asegurar que las preguntas se seleccionen y muestren de manera aleatoria en cada sesión de juego. Esto contribuye a hacer cada juego único y aumenta su rejugabilidad.
- **Agregar Nuevas Preguntas:**
  - Enriquecer el banco de preguntas añadiendo al menos cuatro nuevas cuestiones. Asegurarse de que estas preguntas, junto con sus respuestas correctas e incorrectas, estén correctamente integradas en el array de preguntas existente o en el archivo JSON.

- Asegurarse de que estas nuevas preguntas sean seleccionadas y mostradas aleatoriamente dentro del juego, manteniendo la integridad y fluidez de la experiencia de juego.

- **Testing y Verificación:**
  - Probar exhaustivamente todas las nuevas implementaciones, asegurándose de que funcionen como se espera. Esto incluye la correcta aleatorización de preguntas y respuestas, así como la correcta incorporación y funcionamiento de las nuevas preguntas.
  - Realizar múltiples sesiones de juego para verificar que la lógica del juego, el cálculo de puntuaciones y la presentación de preguntas y respuestas funcionen de manera fluida y correcta.

Ejemplo de PHP para mostrar preguntas aleatorias:

```php
// Obtener preguntas desde una matriz
$preguntas = array(
  "¿Cuál es la capital de Francia?",
  "¿Cuál es el río más largo del mundo?",
  // Otras preguntas
);

// Mezclar las preguntas aleatoriamente
shuffle($preguntas);

// Mostrar las primeras cuatro preguntas
for ($i = 0; $i < 4; $i++) {
  echo $preguntas[$i];
}

// Resto del código PHP...
```

Puntuación de la Práctica: La práctica se evaluará según los siguientes criterios:

- La web muestra las preguntas y se puede interactuar con ellas correctamente.
- Las respuestas correctas incrementan la puntuación del jugador y se almacenan en una variable de sesión. La puntuación se muestra en la interfaz.
- El juego muestra mensajes según el desempeño del jugador (éxito, fracaso, mensajes adicionales) utilizando PHP.
- Existe un tratamiento adecuado de errores en el código PHP para evitar fallos.
- Las respuestas y las preguntas se muestran de forma aleatoria en la interfaz.
- Se han añadido al menos cuatro nuevas preguntas y se seleccionan aleatoriamente en cada intento de juego.
- La mejora del aspecto visual con HTML y CSS es notable, y la página es responsiva.
  TOT 6 puntos

- **Diseño y Creatividad:** Estética y originalidad en la presentación visual.
- **Funcionalidad y Navegación:** Eficiencia y cohesividad.
- **Responsive:** Adaptabilidad y rendimiento.
- **Código y Tecnologías:** Claridad, organización y aplicación adecuada de tecnologías.
  TOT 4 puntos

Formato de Entrega: Entregar los documentos HTML, CSS y JavaScript que componen el proyecto web, además de los archivos PHP creados para esta práctica. Los archivos deben estar comprimidos en un archivo RAR y se deben cargar en la plataforma de entrega en la fecha acordada. Los retrasos en la entrega tendrán una penalización en la nota.

Refuerzo teórico para entender las sesiones:

Las sesiones en PHP son una forma fundamental de mantener la información del usuario y rastrear su actividad durante una visita a un sitio web.

Ejemplo de cómo se podrían usar las sesiones en esta plataforma:

1. Inicio de Sesión en PHP:

```php
// Iniciar una sesión
session_start();

// Verificar si el usuario ha iniciado sesión
if (!isset($_SESSION['usuario'])) {
  // Si no ha iniciado sesión, redirigirlo a la página de inicio de sesión
  header('Location: inicio_sesion.php');
  exit();
}

// Resto del código PHP...
```

En este ejemplo, cuando un usuario visita el sitio, se inicia una sesión en PHP utilizando **`session_start().`** Luego, se verifica si el usuario ha iniciado sesión. Si no lo ha hecho, se lo redirige a la página de inicio de sesión. Esto garantiza que solo los usuarios autenticados puedan acceder a ciertas partes del sitio.

2. Almacenar Información del Usuario:

```php
// Iniciar una sesión
session_start();

// Almacenar información del usuario en la sesión
$_SESSION['usuario'] = 'nombre_de_usuario';
$_SESSION['puntuacion'] = 0;

// Resto del código PHP...
```

En este ejemplo, se almacena información del usuario en la sesión. Por ejemplo, se guarda el nombre de usuario y su puntuación inicial en **`$_SESSION`**. Esto permite que la **información** del usuario esté **disponible** en **todas** las páginas del sitio mientras dure su sesión.

3. Actualizar la Información de la Sesión:

```php
// Iniciar una sesión
session_start();

// Actualizar la puntuación del usuario
$_SESSION['puntuacion'] += 10;

// Resto del código PHP...
```

Cuando el usuario responde correctamente a una pregunta, su puntuación se actualiza en **`$_SESSION`**. Cada vez que responda correctamente, su puntuación se incrementará en 10 puntos. Esto demuestra cómo las **sesiones** pueden utilizarse para **rastrear** el **progreso** del usuario **en toda la plataforma**.

4. Cerrar Sesión:

```php
// Iniciar una sesión
session_start();

// Cerrar sesión al hacer clic en "Cerrar sesión"
if (isset($_POST['cerrar_sesion'])) {
  session_destroy(); // Eliminar toda la información de la sesión
  header('Location: inicio_sesion.php'); // Redirigir al usuario a la
página de inicio de sesión
  exit();
}

// Resto del código PHP...
```

Si un usuario decide **cerrar sesión**, puedes destruir la sesión actual utilizando **`session_destroy()`.** Esto **eliminará toda** la **información** de la **sesión** y redirigirá al usuario a la página de inicio de sesión.

Refuerzo teórico para entender file_get_contents y file_put_contents:

file_get_contents y **file_put_contents** son **funciones** muy útiles en PHP para **leer** y **escribir** contenido en archivos.

Uso de file_get_contents para Leer un Archivo:

Supongamos que en tu plataforma web tienes un archivo llamado "preguntas.json" que contiene preguntas en formato JSON. Quieres cargar estas preguntas desde el archivo y mostrarlas en tu sitio web.

```php
// Leer el contenido del archivo "preguntas.json"
$preguntasJSON = file_get_contents('preguntas.json');
// Decodificar el JSON en un arreglo PHP
$preguntas = json_decode($preguntasJSON, true);
// Mostrar las preguntas en la interfaz web
foreach ($preguntas as $pregunta) {
    echo 'Pregunta: ' . $pregunta['texto'] . '<br>';
    echo 'Opciones: ' . implode(', ', $pregunta['opciones']) .
'<br><br>';
}
// Resto del código PHP...
```

En este ejemplo, **file_get_contents** se utiliza para leer el contenido del archivo "preguntas.json". Luego, **json_decode** se usa para convertir el **JSON** en un arreglo PHP que puede ser fácilmente manejado. Después, se pueden mostrar las preguntas y opciones en la interfaz web.

Uso de file_put_contents para Escribir en un Archivo:

Supongamos que deseas guardar las respuestas de los usuarios en un archivo llamado "respuestas.csv" para mantener un registro de las respuestas correctas e incorrectas.

```php
// Obtener la respuesta del usuario desde el formulario
$respuestaUsuario = $_POST['respuesta'];
// Comprobar si la respuesta es correcta
$esRespuestaCorrecta = verificarRespuesta($respuestaUsuario);
// Guardar la respuesta y su estado en el archivo "respuestas.csv"
$registroRespuesta = "Respuesta: $respuestaUsuario - " .
($esRespuestaCorrecta ? 'Correcta' : 'Incorrecta') . "\n";
file_put_contents('respuestas.csv', $registroRespuesta, FILE_APPEND);
// Resto del código PHP...
```

En este ejemplo, después de verificar si la respuesta del usuario es correcta, se utiliza **file_put_contents** para agregar un registro de la respuesta en el archivo "respuestas.txt". La opción **FILE_APPEND** se utiliza para agregar el registro al final del archivo en lugar de sobrescribirlo.

Refuerzo teórico para entender JSON:

El formato JSON (JavaScript Object Notation) es ampliamente utilizado en el desarrollo web para transmitir datos entre un servidor y un cliente, ya que es **fácil** de leer y escribir tanto para humanos como para máquinas.

JSON en una Plataforma Web:

Supongamos que en tu plataforma web de preguntas y respuestas, deseas almacenar y transmitir preguntas y respuestas en formato JSON. Esto facilitaría la transferencia de datos entre el servidor y el cliente.

1. Almacenamiento de Preguntas en un Archivo JSON:

Primero, podrías almacenar las preguntas en un archivo JSON llamado "preguntas.json". El archivo podría tener la siguiente estructura:

```
[
    {
      "id": 1,
      "pregunta": "¿Cuál es la capital de Francia?",
      "opciones": ["París", "Londres", "Berlín", "Madrid"],
      "respuesta_correcta": "París"
    },
    {
      "id": 2,
      "pregunta": "¿Cuál es el río más largo del mundo?",
      "opciones": ["Amazonas", "Nilo", "Misisipi", "Yangtsé"],
      "respuesta_correcta": "Amazonas"
    },
    // Otras preguntas
  ]
```

En este ejemplo, cada pregunta se representa como un objeto JSON con propiedades como "pregunta", "opciones" y "respuesta_correcta". Esto hace que sea fácil de almacenar y administrar preguntas en un archivo estructurado.

2. Carga de Preguntas desde un Archivo JSON:

Luego, en el código PHP de tu plataforma web, puedes cargar estas preguntas desde el archivo JSON y utilizarlas para mostrar preguntas a los usuarios:

```php
// Leer el contenido del archivo "preguntas.json"
$preguntasJSON = file_get_contents('preguntas.json');

// Decodificar el JSON en un arreglo PHP
$preguntas = json_decode($preguntasJSON, true);

// Mostrar una pregunta aleatoria en la interfaz web
$preguntaAleatoria = $preguntas[array_rand($preguntas)];

echo 'Pregunta: ' . $preguntaAleatoria['pregunta'] . '<br>';
echo 'Opciones: ' . implode(', ', $preguntaAleatoria['opciones']) .
'<br>';

// Resto del código PHP...
```

En este código PHP, utilizamos **`file_get_contents`** para cargar el contenido del archivo "preguntas.json" y luego usamos **`json_decode`** para convertir el JSON en un arreglo PHP. Luego, seleccionamos una pregunta aleatoria del arreglo y la mostramos en la interfaz web.

3. Respuestas en Formato JSON:

Cuando un usuario envía una respuesta, puedes recopilar y enviar la respuesta en formato JSON al servidor para su procesamiento:

```javascript
// En el lado del cliente (JavaScript)
const respuestaUsuario = "París"; // Supongamos que el usuario selecciona
esta respuesta

// Crear un objeto JSON con la respuesta
const respuestaJSON = {
  pregunta_id: 1, // Identificador de la pregunta
  respuesta: respuestaUsuario
};

// Enviar el objeto JSON al servidor (por ejemplo, utilizando AJAX)

// Resto del código JavaScript...
```

En el **lado del servidor** (PHP), puedes recibir y procesar la respuesta JSON:

```php
// En el lado del servidor (PHP)
$respuestaJSON = json_decode(file_get_contents('php://input'), true);

// Verificar la respuesta y actualizar la puntuación del jugador

// Resto del código PHP...
```

En este ejemplo, utilizamos **`file_get_contents('php://input')`** para leer el JSON enviado por el cliente y luego usamos **`json_decode`** para convertirlo en un arreglo PHP. Luego, puedes verificar la respuesta y actualizar la puntuación del jugador según corresponda.