UNIVERSIDAD ADOLFO IBÁÑEZ

Prueba de Simulacro

Parte 1: Corrección de Código HTML y CSS

Una empresa está desarrollando un portal para que sus empleados puedan votar por los representantes más destacados de cada área. Para ello, se ha diseñado un formulario de votación en HTML y CSS, pero este contiene algunos errores que afectan su funcionamiento y presentación.

A continuación, se entrega el código del formulario y sus estilos básicos. Tu tarea consiste en:

Código HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulario de Votación</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Votación de Representantes</h1>
    <form id="voting-form">
        <div class="form-group">
            <label for="name">Nombre del empleado</label>
            <input type="text" id="name" name="name">
        </div>

        <div class="form-group">
            <label for="area">Área de trabajo</label>
            <select id="area" name="area">
                <option value="ventas">Ventas</option>
                <option value="marketing">Marketing</option>
                <option value="desarrollo">Desarrollo</option>
                <option value="rrhh">Recursos Humanos</option>
            </select>
        </div>

        <div class="form-group">
            <button type="submit" class="btn-submit">Votar</button>
        </div>
    </form>
</body>
</html>
```

Código CSS

```css
body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f0f0f0;
}

h1 {
    text-align: center;
    margin-bottom: 20px;
}

.form-group {
    margin-bottom: 15px;
}

.btn-submit {
    background-color: #FF0000;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    text-align: center;
}

.btn-submit:hover {
    background-color: #45a049;
}
```

Recuerda que la finalidad de este formulario es permitir que los empleados seleccionen un área y voten por su representante favorito. El botón de votación debe tener un fondo de color azul. El formulario debe estar alineado de manera que solo el contenido del formulario esté centrado en la página, mientras que el título principal no debe estar centrado junto con él.

Identificar (encerrar en un círculo el error o líneas donde están los errores) y proponer una corrección para cada error, justificando brevemente por qué tu solución es la adecuada.

Parte 2: Análisis de Componente en React

Como parte del sistema de votaciones de la empresa, se ha desarrollado un componente en React que presenta los resultados de la votación. A continuación, se te proporciona el código de este componente junto con una lista de votos emitidos.

Código App.js

```
JS app.js > ...
1   import React from 'react';
2   import VoteResults from './VoteResults';
3
4   const votes = [
5       { name: "Juan", area: "Ventas" },
6       { name: "Ana", area: "Marketing" },
7       { name: "Luis", area: "Desarrollo" },
8       { name: "Sofía", area: "Desarrollo" },
9       { name: "Carlos", area: "RRHH" },
10      { name: "María", area: "Ventas" },
11      { name: "Pedro", area: "Marketing" },
12      { name: "Lucía", area: "Desarrollo" },
13      { name: "Miguel", area: "RRHH" },
14      { name: "Carla", area: "Ventas" }
15  ];
16
17  const App = () => {
18      return (
19          <div>
20              <h1>Portal de Votaciones</h1>
21              <VoteResults votes={votes} />
22          </div>
23      );
24  };
25
26  export default App;
```

Código VoteResults.js

```
JS VoteResults.js > ...
1   import React from 'react';
2
3   const VoteResults = ({ votes }) => {
4       const voteCount = votes.reduce((acc, vote) => {
5           if (!acc[vote.area]) {
6               acc[vote.area] = { count: 0, employees: [] };
7           }
8           acc[vote.area].count += 1;
9           acc[vote.area].employees.push(vote.name);
10          return acc;
11      }, {});
12
13      const sortedAreas = Object.keys(voteCount).sort();
14
15      return (
16          <div>
17              <h2>Resultados de la Votación</h2>
18              <ul>
19                  {sortedAreas.map((area) => (
20                      <li key={area}>
21                          <strong>{area}:</strong> {voteCount[area].count} votos
22                          <ul>
23                              {voteCount[area].employees.map((name, index) => (
24                                  <li key={index}>{name}</li>
25                              ))}
26                          </ul>
27                      </li>
28                  ))}
29              </ul>
30          </div>
31      );
32  };
33
34  export default VoteResults;
```

Muestra el resultado esperado del componente en el navegador, tal como fue programado en el código.

Parte 3: Diseño de una API para Registro de Votos

Como parte del sistema de votación, se necesita una API que permita registrar los votos de los empleados en una base de datos. Esta API recibirá los datos del formulario de votación, incluyendo el nombre del empleado y el área por la que vota.

Desarrolla una API en JavaScript que permita registrar un voto enviando los datos recibidos a una base de datos.

Puedes incluir todos los supuestos que creas necesarios, como por ejemplo asumir una base de datos en específico.

Muestra únicamente el código de la API, incluyendo un endpoint para recibir los datos y un ejemplo de la lógica de almacenamiento de los votos en la base de datos.

Requisitos adicionales:

- La API debe recibir los datos en formato JSON.
- Asegúrate de que el código esté estructurado de manera clara y organizada, usando buenas vistas en clases.

Parte 4 - Preguntas conceptuales

Todas las preguntas pueden tener más de una alternativa correcta, debe marcar todas las correctas.

1.- Sobre el uso de Local Storage en aplicaciones web, ¿cuáles de las siguientes afirmaciones son correctas?

A. Local Storage permite almacenar datos de forma persistente en el navegador, incluso después de cerrar la página.
B. Los datos en Local Storage se eliminan automáticamente cuando se cierra el navegador.
C. Los datos almacenados en Local Storage pueden ser accedidos desde cualquier dominio.
D. La información almacenada en Local Storage se envía automáticamente en cada solicitud HTTP al servidor.
E. Local Storage sólo permite almacenar datos en formato texto.

2.- En relación al diseño responsive, ¿cuáles de las siguientes prácticas ayudan a lograr que una página web sea adaptativa/responsive?

A. Uso de media queries (@) en CSS para ajustar el diseño según el tamaño de la pantalla.
B. Usar unidades de medida relativas como % o em en lugar de valores fijos en píxeles.
C. Emplear flex o grid para estructurar el contenido en vez de layouts basados en tablas.
D. Permitir que los elementos del DOM se reordenen automáticamente en función del ancho de la pantalla.
E. Utilizar frameworks de CSS como Bootstrap o Tailwind que incluyen clases para diseño responsive.
F. Evitar imágenes de grandes tamaños (superiores a 1000 pixeles) para que se ajuste a todos los dispositivos.

3.- ¿Cuáles de las siguientes afirmaciones describen correctamente las diferencias conceptuales entre React y Next.js?

A. React es una biblioteca de JavaScript para construir interfaces de usuario, mientras que Next.js es un framework que extiende las capacidades de React.
B. Next.js soporta rendering del lado del servidor (SSR) de manera nativa, mientras que React es client-side por defecto.
C. Las aplicaciones de React requieren configurar el server-side rendering manualmente, mientras que en Next.js es opcional pero está integrado.
D. Next.js incluye herramientas para routing de manera nativa, mientras que React depende de bibliotecas adicionales como React Router para el routing.
E. React tiene un sistema de archivos específico para la estructura de rutas, mientras que Next.js usa un sistema de archivos opcional.
F. Solo se puede usar Next.js con un backend en Node.js, mientras que React funciona con cualquier backend.

Pregunta 4: ¿Cuáles son las prácticas correctas para el uso de hashing en contraseñas dentro de formularios de registro en aplicaciones web?

A. Agregar una Sal (*salt*) aleatorio a cada contraseña antes de hacer el hashing para aumentar la seguridad.
B. Aplicar el hashing en el cliente, enviando la contraseña hasheada al servidor en lugar de la contraseña en texto plano.
C. Usar algoritmos de hashing específicos para contraseñas, como bcrypt, para asegurar las contraseñas almacenadas.
D. Guardar tanto la contraseña en texto plano como el hash en la base de datos para comparar en caso de errores.
E. Asegurar que las contraseñas hasheadas en el servidor no se envíen de vuelta al cliente por razones de seguridad.
F. Hashing y “salting” son procesos equivalentes y cumplen la misma función en seguridad de contraseñas.
G. Almacenar las contraseñas anteriores en texto plano para lograr comparar y evitar que los usuarios las reutilicen.

Pregunta 5: Sobre el uso y ventajas del almacenamiento en caché en aplicaciones web, ¿cuáles de las siguientes afirmaciones son correctas?

A. El almacenamiento en caché permite almacenar respuestas de solicitudes, reduciendo el tiempo de carga en visitas posteriores.
B. El caché sólo almacena archivos de imagen para mejorar la velocidad de carga.
C. Las aplicaciones pueden usar el caché para almacenar datos que no cambian con frecuencia, como archivos CSS y JavaScript.
D. El caché en el lado del cliente permite reducir la carga en el servidor y mejorar el rendimiento.
E. Al actualizar el contenido, es importante invalidar el caché para asegurar que el usuario reciba los datos más recientes.
F. Los datos en el caché permanecen almacenados de manera indefinida, incluso después de cerrar el navegador.
G. El almacenamiento en caché es únicamente efectivo en dispositivos móviles.