1. ¿Qué hace la etiqueta input de HTML y cómo se utilizan?

La etiqueta `<input>` de HTML se utiliza para crear campos de entrada de datos para que los usuarios puedan introducir información dentro de un formulario. Se puede configurar para diferentes tipos de entrada como texto, contraseña, radio button, checkbox, entre otros, mediante el atributo `type`.

2. Ejemplo de un formulario utilizando la etiqueta input:

Este formulario está diseñado para recoger el nombre y la edad del usuario. Cuando el usuario hace clic en "Enviar", los datos del formulario se pueden enviar a un servidor para su procesamiento. La manera exacta en que se manejan los datos depende del atributo `action` de la etiqueta `<form>` (que no se ha especificado en este ejemplo) y de cómo se procesan esos datos en el servidor o mediante JavaScript en el lado del cliente.

```html
<form>
  <label for="name">Nombre:</label>
  <input type="text" id="name" name="name">
  <label for="age">Edad:</label>
  <input type="number" id="age" name="age">
  <input type="submit" value="Enviar">
</form>
```

1. **`<form>`:** Esta etiqueta define el formulario en sí y actúa como contenedor para todos los campos de entrada (`<input>`) y etiquetas asociadas. Los formularios son utilizados para recoger entradas de los usuarios.

2. **`<label for="name">Nombre:</label>`:** La etiqueta `<label>` especifica una etiqueta para el elemento `<input>`. El atributo `for` debe ser igual al atributo `id` del `<input>` al que está asociada, lo que mejora la accesibilidad y permite que al hacer clic en la etiqueta, el foco se mueva al campo correspondiente.

- **`for="name"`:** Indica que esta etiqueta está asociada con el campo de entrada cuyo `id` es "name".

3. `<input type="text" id="name" name="name">`: Este es un campo de entrada donde los usuarios pueden escribir texto.

- **`type="text"`:** Especifica que el tipo de entrada es texto, permitiendo cualquier carácter.

- **`id="name"`:** Define un identificador único para el campo de entrada, que se utiliza para asociarlo con su etiqueta (`<label>`) correspondiente.

- **`name="name"`:** El nombre del campo de entrada, que se utiliza para identificar los datos en el formulario cuando se envían a un servidor.

4. `<label for="age">Edad:</label>`: Similar a la primera etiqueta `<label>`, pero esta vez para el campo de edad.

5. `<input type="number" id="age" name="age">`: Un campo de entrada para números.

- **`type="number"`:** Indica que el campo acepta solo números. Los navegadores pueden mostrar flechas para aumentar o disminuir el valor.

- **`id="age"` y `name="age"`:** Funcionan de manera similar a los explicados anteriormente, pero para el campo de edad.

6. `<input type="submit" value="Enviar">`: Este es un botón de envío.

- **`type="submit"`:** Define un botón que, al hacer clic, envía el formulario a un servidor o desencadena una acción definida.

- **`value="Enviar"`:** Especifica el texto que aparecerá en el botón de envío. En este caso, "Enviar".

3. Muestrario de al menos 10 tipos de inputs diferentes:

¡Entendido! Vamos a detallar un poco más sobre el muestrario de inputs del punto 3:

1. `<input type="text">`: Campo de texto para que el usuario escriba texto libre, como nombre, dirección, etc.

2. `<input type="password">`: Campo para escribir una contraseña. Los caracteres introducidos se muestran como puntos o asteriscos para mantener la privacidad.

3. `<input type="radio">`: Botones de radio que permiten al usuario seleccionar una única opción entre varias disponibles. Todos los botones de radio asociados deben tener el mismo nombre (`name`) pero valores (`value`) diferentes.

4. `<input type="checkbox">`: Casillas de verificación que permiten al usuario seleccionar múltiples opciones de un conjunto.

5. `<input type="submit">`: Botón para enviar los datos del formulario al servidor. El atributo `value` especifica el texto del botón.

6. `<input type="reset">`: Botón para restablecer todos los campos del formulario a sus valores predeterminados.

7. `<input type="file">`: Permite al usuario seleccionar uno o más archivos de su dispositivo para ser subidos a un servidor.

8. `<input type="hidden">`: Campo que no es visible para el usuario. Se utiliza para enviar información que no puede ser modificada o que no necesita ser vista por el usuario, como un ID de usuario o un token.

9. `<input type="date">`: Proporciona un control para ingresar una fecha (año, mes y día, sin horario).

10. `<input type="color">`: Muestra un selector de color, permitiendo al usuario elegir un color específico.

4. ¿Qué es el lenguaje JavaScript?

JavaScript es un lenguaje de programación interpretado, orientado a objetos, de alto nivel, que se utiliza principalmente para la creación de páginas web interactivas.

5. ¿Para qué sirve el lenguaje JS?

Sirve para añadir interactividad a las páginas web, permitiendo la manipulación de los elementos del documento, manejo de eventos, validación de formularios, comunicación con el servidor sin recargar la página, y mucho más.

6. ¿A qué se refiere el DOM (Document Object Model) en el desarrollo Web?

El DOM es una interfaz de programación que representa los documentos HTML y XML como un árbol de nodos. Esto permite a los lenguajes de programación, como JavaScript, manipular el contenido, estructura y estilo de las páginas web.

7. ¿Cuál es la diferencia entre los archivos HTML, CSS y JS?

- HTML: proporciona la estructura básica de la página.
- CSS: se utiliza para controlar la presentación, el formato y el diseño.
- JavaScript: se usa para añadir interactividad y funcionalidades dinámicas.

8. ¿Para qué sirve la etiqueta `<script>`?

La etiqueta `<script>` se utiliza para incluir o referenciar código JavaScript en un documento HTML, permitiendo agregar scripts directamente en el código HTML o cargar archivos JS externos.

9. Métodos para cambiar elementos tipo Tag Name, ID y Class Name con JS:

- Por Tag Name: `document.getElementsByTagName("p")` selecciona todos los elementos `<p>`.
- Por ID: `document.getElementById("demo")` selecciona el elemento con el ID `demo`.

- Por Class Name: `document.getElementsByClassName("test")` selecciona todos los elementos con la clase `test`.

10. Programa básico “Hola mundo” con JS en Replit:

```html
<!DOCTYPE html>
<html>
<body>

<h2>Hola Mundo</h2>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hola Mundo!";
</script>

</body>
</html>
```

Funcionamiento: Este código utiliza JavaScript para cambiar el contenido del elemento `<p>` a "Hola Mundo!".

11. Sintaxis de la programación en JS:

La sintaxis de JS incluye declaraciones de variables, estructuras de control, funciones, operadores, y otros elementos que permiten crear scripts. Ejemplo básico:

```javascript
let numero = 5; // Declaración de variable
```

```javascript
if (numero > 0) { // Estructura de control
  console.log("El número es positivo.");
}
```

12. Cómo se declaran variables en JS:

Las variables en JS pueden declararse usando `var`, `let`, o `const`, seguido por el nombre de la variable y, opcionalmente, un valor inicial.

```javascript
let nombre = "Carlos";
const PI = 3.14;
```

13. Cómo se declaran funciones en JS:

Las funciones se declaran con la palabra clave `function`, seguida de un nombre, paréntesis que pueden incluir parámetros, y un bloque de código.

```javascript
function saludar(nombre) {
  console.log("Hola, " + nombre);
}
```

14. Cómo se llaman funciones en JS:

Se llaman especificando el nombre de la función seguido de paréntesis, los cuales pueden incluir argumentos.

```javascript
saludar("Carlos");
```

15. Declarar una función interna de un HTML con JS y declarar un documento externo con JS:

- Función interna: Dentro de `<script>` en HTML.

  ```html
  <script>
  function mostrarMensaje() {
    alert("Hola Mundo");
  }
  </script>
  ```

- Documento externo: Usando la etiqueta `<script>` con el atributo `src`.

  ```html
  <script src="ruta/al/archivo.js"></script>
  ```

16. Ciclos de programación disponibles en JS:

- `for`
- `while`
- `do...while`
- `for...in`
- `for...of`

17. Funciones de condicional disponibles en JS:

- `if`...`else`
- `switch`
- Operador ternario (condición ? expr1 : expr2)