Investigación: Desarrollo Web con HTML, CSS, y JavaScript

Introducción

El desarrollo web ha evolucionado significativamente desde sus inicios en la década de 1990. Esta evolución se ha centrado en cómo las páginas web son construidas y cómo interactúan con los usuarios. Tres tecnologías fundamentales que han sido pilares en este desarrollo son HTML, CSS, y JavaScript. Estas tecnologías surgieron para satisfacer diferentes necesidades en la creación de páginas web, ofreciendo estructura, estilo, y funcionalidad respectivamente.

Origen de HTML, CSS, y JavaScript

HTML

El Lenguaje de Marcado de Hipertexto (HTML) fue creado por Tim Berners-Lee en 1990. Berners-Lee, un físico británico, lo desarrolló mientras trabajaba en el CERN (Organización Europea para la Investigación Nuclear) para facilitar el intercambio de información científica en la internet. HTML permitió la creación de páginas web que podían enlazarse entre sí a través de hipervínculos, formando la World Wide Web.

CSS

Las Hojas de Estilo en Cascada (CSS) fueron propuestas por Håkon Wium Lie en 1994. El objetivo de CSS era separar el contenido del diseño dentro de los documentos web, permitiendo a los desarrolladores cambiar el aspecto de las páginas web sin alterar su estructura HTML. CSS fue adoptado como un estándar del W3C (World Wide Web Consortium) en 1996.

JavaScript

JavaScript fue desarrollado por Brendan Eich de Netscape en 1995 bajo el nombre de Mocha, luego renombrado a LiveScript, y finalmente a JavaScript. Este lenguaje de scripting fue creado para añadir interactividad a las páginas web, permitiendo a los desarrolladores crear una experiencia de usuario más dinámica.

Ventajas y Desventajas

Ventajas

HTML: Ofrece una curva de aprendizaje suave y es fácil de usar. Es compatible con todos los navegadores y proporciona la base fundamental para la estructura de las páginas web.

CSS: Facilita la separación entre el contenido y el diseño, lo que mejora la accesibilidad y la mantenibilidad del código. Además, permite la creación de diseños responsivos para adaptarse a diferentes dispositivos.

JavaScript: Agrega interactividad a las páginas web, permitiendo la programación tanto del lado del cliente como del servidor con tecnologías como Node.js. Además, cuenta con una comunidad sólida que ofrece amplio soporte.

Desventajas

HTML: Limitado en cuanto a diseño y dinamismo por sí solo, necesitando de CSS y JavaScript para alcanzar su máximo potencial.

CSS: La gestión de la compatibilidad entre navegadores puede resultar complicada en ciertos casos.

JavaScript: El código puede ser bloqueado por algunos usuarios o navegadores, y prácticas incorrectas pueden conducir a problemas de seguridad como cross-site scripting (XSS).

Ejemplos de Sistema Aplicado

* Formularios Web: Se utiliza HTML para estructurar el formulario, CSS para estilizarlo y JavaScript para validar los datos ingresados por el usuario antes de enviarlos al servidor.
* Páginas Web Responsivas: CSS desempeña un papel crucial en la creación de diseños que se adapten a diferentes dispositivos, mientras que JavaScript puede utilizarse para modificar elementos de la página de manera dinámica en respuesta a la interacción del usuario.
* Aplicaciones Web Interactivas: JavaScript posibilita el desarrollo de aplicaciones web complejas que pueden responder en tiempo real a la interacción del usuario, proporcionando experiencias similares a las aplicaciones de escritorio y mejorando significativamente la usabilidad.

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

La etiqueta <input> en HTML se emplea para crear campos interactivos donde los usuarios pueden ingresar datos dentro de un formulario. Estos campos pueden configurarse para diferentes tipos de entrada, como texto, contraseña, opciones de selección y más, mediante el atributo type.

**2. Ejemplo de un formulario utilizando la etiqueta input:**

Este formulario está diseñado para recolectar información sobre el nombre y la edad del usuario. Una vez que el usuario hace clic en "Enviar", los datos del formulario pueden ser enviados al servidor para su procesamiento. La manera en que estos datos son manejados depende del atributo action de la etiqueta <form> (que no está especificado en este ejemplo) y cómo son procesados ya sea 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 la estructura del formulario y sirve como contenedor para los diferentes campos de entrada (<input>) y sus etiquetas asociadas. Los formularios son utilizados para recolectar información de los usuarios.

**2. `<label for="name">Nombre:</label>`:** La etiqueta <label> proporciona una etiqueta descriptiva para el elemento <input>. El atributo for debe coincidir con el id del <input> al que se refiere, mejorando así la accesibilidad y permitiendo que, al hacer clic en la etiqueta, se enfoque en el 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:**

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, de alto nivel y orientado a objetos, que se utiliza principalmente para crear interactividad en las páginas web.

**5. ¿Para qué sirve el lenguaje JS?**

JS se utiliza para agregar interactividad a las páginas web, lo que permite la manipulación dinámica de los elementos del documento, la gestión de eventos, la validación de formularios, la comunicación con el servidor sin recargar la página, entre otras funcionalidades.

**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 una estructura de árbol de nodos. Esto facilita a los lenguajes de programación, como JavaScript, la manipulación del contenido, la estructura y el estilo de las páginas web de manera dinámica.

**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 hacer referencia a 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

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("Alejandro");

**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)

Referencias para realizar el trabajo:

* <https://uniwebsidad.com/libros/javascript>
* <https://www.w3schools.com/js/default.asp>
* <https://www.w3schools.com/tags/tag_input.asp>