Módulo 5: Gestión de Objetos del Lenguaje de Guión

Curso de Integración de Componentes Software Web

Explicación de las tecnologías web fundamentales

- html: "Estructura el contenido",
- css: "Da estilo a la presentación",
- javascript: "Aporta la lógica y la interacción"

Métodos para añadir JavaScript a una web

- inline: "Evitar siempre",
- internal: "Útil en páginas pequeñas",
- external: "Método recomendado"

Ejemplo de Script en línea (NO RECOMENDADO)

<button onclick="alert('¡Hola, mundo!')">Haz clic aquí</button>`;

Ejemplo de Script Interno

```
<script>
  document.getElementById("miBoton").addEventListener("click", function() {
    alert("¡Hola, mundo desde un script interno!");
  });
</script>
```

Ejemplo de Script Externo (RECOMENDADO)

```
<script src="scripts/miScript.js" defer></script>
```

Ubicación de los Scripts en el documento HTML

- **head**: "Cargado antes de renderizar la página, puede bloquear el renderizado"
- **body**: "El contenido HTML se carga primero, preferible usar defer o async"

Uso de Defer y Async

HTML se pausa hasta ejecutar el script

- **defer**: "Se descarga en paralelo y se ejecuta después del HTML"
- **async**: "Se descarga en paralelo y se ejecuta tan pronto como se descargue"

Ejemplo de Módulo de JavaScript

Un modulo javascript se utiliza para organizar el código en archivos separados.

<script type="module" src="module.js"></script>

```
// module.js
export function sum(a, b) {
    return a + b
}
```

¿Qué son las DevTools?

- **Descripcion**:

  Herramientas integradas en los navegadores para analizar, depurar y optimizar el código JavaScript y otros elementos web.
- **Utilidad**:

  Facilitan la detección de errores y el análisis del comportamiento de la aplicación en tiempo real.

Métodos para acceder a las DevTools según el navegador

- chrome_edge:

  Clic derecho > Inspeccionar o Ctrl + Shift + I (Windows) / Cmd + Option + I (Mac)
- firefox:

  Clic derecho > Inspeccionar o Ctrl + Shift + I (Windows) / Cmd + Option + I (Mac)
- safari:

  Habilitar en preferencias avanzadas, luego clic derecho > Inspeccionar",

Panel de Consola

```
console.log("Hola desde la consola");
console.error("Este es un mensaje de error");
console.warn("Advertencia: Algo no va bien");
```

Panel de Elementos (DOM y CSS)

Funcionalidad: Permite inspeccionar y modificar el DOM y CSS en tiempo real.
Acciones: Seleccionar elementos, Modificar estilos en vivo

Panel de Depuración (Sources)

```
function sumar(a, b) {
  debugger; // Pausa la ejecución para analizar variables
  return a + b;
}
sumar(3, 4);
```

Panel de Network

Muestra todas las solicitudes realizadas por la aplicación.

Acciones:

- Verificar tiempos de carga
- Inspeccionar datos en solicitudes API
- Identificar errores en respuestas

Panel de Performance

Analiza el rendimiento de la página.

Acciones:

- Registrar tiempo de renderizado
- Identificar cuellos de botella
- Optimizar velocidad de carga

Consejos generales

experimentar:
Modifica estilos y elementos en las DevTools sin afectar el código original.
usarDebugger:
Evita depender de console.log y analiza la ejecución paso a paso.
filtrarMensajes:
Usa filtros en la consola para enfocarte en errores y advertencias específicas.
analizarSolicitudes:
Utiliza el panel Network para detectar errores en la comunicación con APIs.

¿Qué es el DOM?

- El Document Object Model (DOM) es una interfaz de programación para documentos HTML y XML.
- Permite acceder, manipular y modificar la estructura de un documento web desde JavaScript.

Estructura del DOM

- raiz: document
- elementos: "html", "head", "body"
- atributos: Propiedades de los elementos HTML
- texto: Contenido textual dentro de los elementos HTML

Tipos de nodos en el DOM

- **documento**: HTMLDocument - Representa el documento entero.
- **elemento**: HTMLElement - Cada etiqueta HTML.
- **atributo**: Attr - Atributos de los elementos HTML.
- **texto**: Text - Contenido textual de un elemento.
- **comentario**: Comment - Comentarios en el HTML.

Subtipos de HTMLElement

- **enlace**: HTMLAnchorElement - <a>
- **div**: HTMLDivElement -
- **imagen**: HTMLImageElement - <img>
- **input**: HTMLInputElement - <input>
- **boton**: HTMLButtonElement - <button>
- **parrafo**: HTMLParagraphElement - <p>
- **encabezado**: HTMLHeadingElement - <h1>-<h6>
- **tabla**: HTMLTableElement - <table>

Para qué sirve el DOM

seleccionar: Seleccionar elementos del documento.
navegar: Navegar entre elementos relacionados.
modificar: Modificar contenido y atributos de los elementos.
agregar_eliminar: Añadir o quitar elementos en el DOM.
cambiar_estilos: Modificar los estilos CSS dinámicamente.

Actualización del DOM

- Manipulacion: Cada vez que se cambia un elemento, el navegador actualiza la representación visual.
- Optimizacion: Evitar cambios directos frecuentes para mejorar rendimiento.
- Fragmentos: Uso de DocumentFragment para minimizar manipulaciones.
- virtualDOM: Algunos frameworks usan un DOM virtual para optimizar cambios.

Ejemplo de manipulación del DOM

```
// Crear un nuevo elemento
    let newElement = document.createElement("div");
    newElement.textContent = "¡Hola, Mundo!";
    newElement.style.color = "blue";
    newElement.style.fontSize = "24px";
    newElement.style.fontWeight = "bold";
    newElement.style.textAlign = "center";
    document.body.appendChild(newElement);
```

Ejemplo de manipulación del DOM

```
// Eliminar un elemento
    document.body.removeChild(newElement);

// Modificar un elemento
    newElement.textContent = "¡Hola, JavaScript!";
    document.body.appendChild(newElement);
```

Ejemplo de manipulación del DOM

```
// Reemplazar un elemento
    let oldElement = document.createElement("p");
    oldElement.textContent = "¡Hola, Mundo!";
    document.body.replaceChild(oldElement, newElement);
// Clonar un elemento
    let cloneElement = oldElement.cloneNode(true);
    document.body.appendChild(cloneElement);
```

Ejemplo de manipulación del DOM

```
// Mover un elemento
    document.body.insertBefore(cloneElement, oldElement);
// Obtener un elemento
    let element = document.getElementById("elementId");
    console.log(element);
// Seleccionar elementos
    let elements = document.querySelectorAll(".elementClass");
    console.log(elements);
```

Ejemplo de manipulación del DOM

```
// Modificar atributos
    element.setAttribute("class", "newClass");
    element.removeAttribute("class");
// Modificar estilos
    element.style.color = "red";
    element.style.backgroundColor = "yellow";
```

Ejemplo de manipulación del DOM

```
// Modificar contenido
    element.textContent = "¡Hola, Mundo!";
    element.innerHTML = "<strong>¡Hola, Mundo!</strong>";
// Modificar eventos
    element.addEventListener("click", function () {
    alert("¡Hola, Mundo!");
});
```

Ejemplo de manipulación del DOM

```
// Modificar clases
    element.classList.add("newClass");
    element.classList.remove("oldClass");
// Modificar datos
    element.dataset.id = "1";
    console.log(element.dataset.id);
```

Jerarquía de Objetos

- Estructura de objetos en el navegador:
  - window: Objeto global.
  - document: Representa el HTML.
  - navigator, history, location.
- Ejemplo: window.alert('Hola');

```
Ejemplos:
    console.log(window.location.href);
    alert('Navegador: ' + navigator.userAgent);
```

Manipulación del DOM

- Document Object Model (DOM): Estructura en árbol de HTML.
  - Métodos: getElementById(), querySelector().
  - Propiedades: innerHTML, style.
- Ejemplo: document.getElementById('titulo').innerHTML = 'Nuevo Título';

Ejemplos:

```
document.body.style.backgroundColor='lightblue';
```

Prácticas del Módulo 5

- Actividades:
  - Mostrar propiedades del navegador.
  - Cambiar contenido HTML con scripts.
  - Agregar elementos dinámicos al DOM.
  - Usar métodos de document.