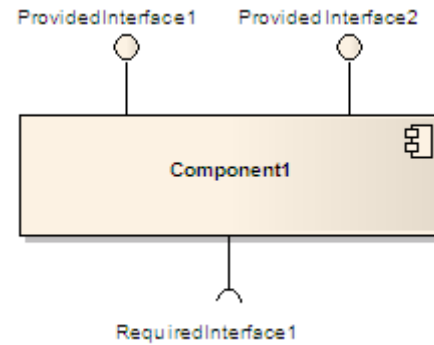


UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

Programación integrativa de componentes web



Giuliana Cristina Umaña Lasluisa

NRC: 16496

pagina1.html

HHHH111111PAGINA 1 COMPONENTE CRISTIAN del Alcazhar Ponce 1

Connectando al Element del HTML DOM desde JS

HHH22222segunda llamada

parrafito insertado

- ▶ Este fragmento de código HTML describe una página web que contiene metadatos esenciales y un cuerpo con un encabezado principal, un componente web personalizado denominado <elemento-uno>, un segundo encabezado y un párrafo. También incluye la importación de un módulo JavaScript llamado "componente1.js", que presumiblemente establece el comportamiento del componente personalizado.

Archivo Editar Ver

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>

<body>
  <h1> HHHH11111111PAGINA 1 COMPONENTE CRISTIAN del Alcazhar Ponce 1</h1>
  <elemento-uno></elemento-uno>
  <h2> HHH22222segunda llamada </h2>

  <p> parrafito insertado</p>

  <script type="module" src="componente1.js">
  </script>
</body>

</html>
```

- Emplea un componente web personalizado llamado `<elemento-uno>` y carga un módulo JavaScript externo "componente1.js" para agregar funcionalidad dinámica a la página.

```
Archivo  Editar  Ver

class Elemento1 extends HTMLElement {
  constructor() {
    super();
    console.log("Elemento1 constructor .... toy constructor ");
    this.p = document.createElement('p'); // nuevo elemento

  } //---fin del constructor
  connectedCallback() {
    this.p.textContent = "Connectando al Element del HTML DOM  desde JS";
    this.appendChild(this.p);

  } // ---fin metodo connectedCallback

} //-----fin de la clase Web Componente

customElements.define('elemento-uno', Elemento1);
```

- Este código JavaScript crea un componente web personalizado llamado <elemento-uno>, que añade dinámicamente un párrafo con texto al unirse al DOM. Utiliza la clase Elemento1 y el método connectedCallback para controlar el contenido y el comportamiento del componente.

pagina2.html

Hola parrafito

Connectando al Element del HTML DOM desde JS

despues del wc

- Esta página HTML incluye un párrafo con el id "par1" que contiene el componente web personalizado <elemento-uno>, el cual agrega contenido dinámico al cargarse. Además, se carga el módulo JavaScript "componente1.js" para manejar el comportamiento del componente.

```
<!DOCTYPE html>
<html>

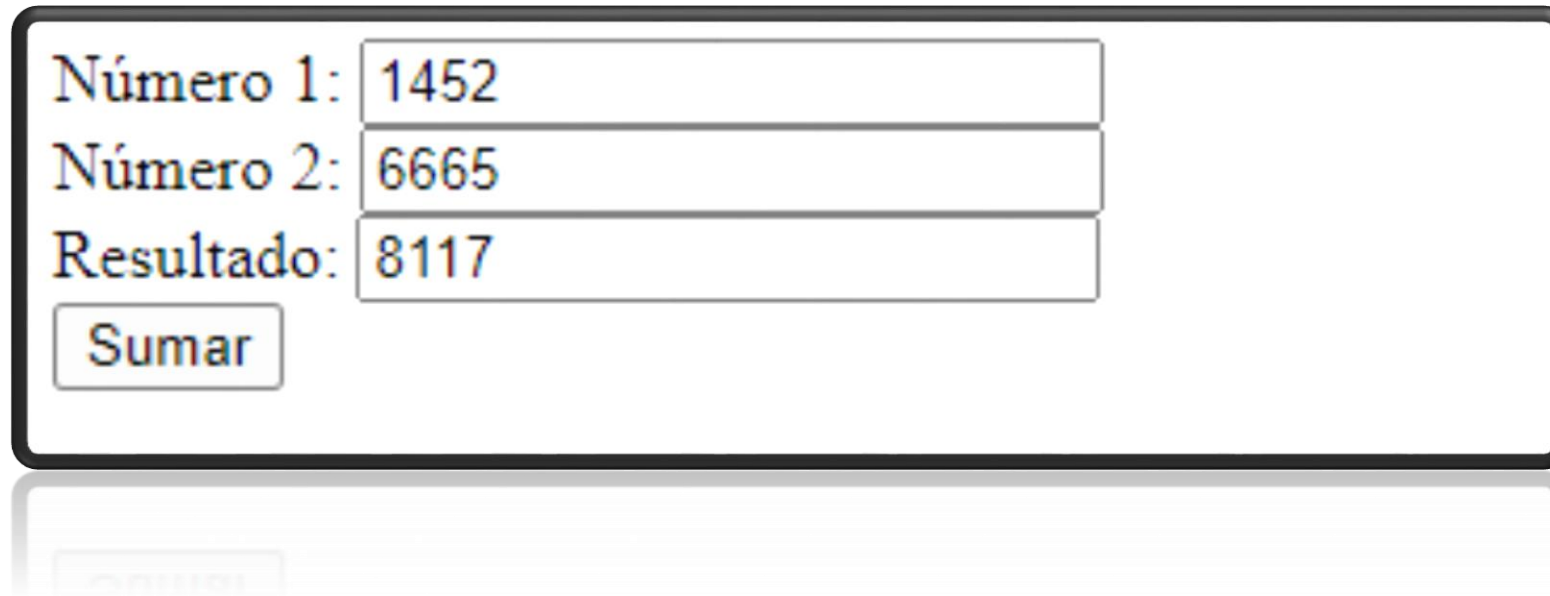
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>

<body>
  <p id="par1">Hola parrafito
    <elemento-uno></elemento-uno>
  </p>
  <h3>despues del wc</h3>
  <script type="module" src="componente1.js">
  </script>
</body>

</html>
```

- La página HTML incorpora un componente web personalizado `<elemento-uno>` dentro de un párrafo, mostrando cómo es posible mejorar el contenido HTML con elementos dinámicos. Además, incluye un encabezado `<h3>` y un módulo JavaScript externo "componente1.js", que ayuda a separar el comportamiento de la estructura de la página.

pagina5.html



A screenshot of a web form titled 'pagina5.html'. The form is enclosed in a black border and contains three input fields and a button. The first input field is labeled 'Número 1:' and contains the value '1452'. The second input field is labeled 'Número 2:' and contains the value '6665'. The third input field is labeled 'Resultado:' and contains the value '8117'. Below the input fields is a button labeled 'Sumar'. The form is reflected below it.

Número 1:	1452
Número 2:	6665
Resultado:	8117

Sumar

- Esta página HTML contiene un formulario sencillo para sumar dos números, usando etiquetas `<label>` e `<input>` para ingresar los datos y un botón para ejecutar la suma. También integra un componente web personalizado `<componente-suma>` y un script externo "componente5.js" que, presumiblemente, gestiona la lógica de la suma.

- Este código JavaScript crea un componente personalizado <componente-suma> que permite sumar dos números proporcionados por el usuario. Emplea Shadow DOM para encapsular los estilos y funcionalidades, garantizando un diseño autónomo.

```
class Suma extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({ mode: 'open' });

    // Crear un contenedor para los estilos y la funcionalidad
    const container = document.createElement('div');
    container.innerHTML = `
    <style>
      /* Estilos del componente */
      :host {
        display: block;
        margin-top: 10px;
      }
    </style>
    <slot></slot>
    `;

    this.shadowRoot.appendChild(container);
  }

  connectedCallback() {
    // Obtener referencia al botón de suma y añadir el listener de clic
    const btnSumar = document.querySelector('#btnSumar');
    if (btnSumar) {
      btnSumar.addEventListener('click', () => this.sumar());
    }
  }
}
```



```

sumar() {
  // Obtener los valores de los inputs
  const n1 = parseFloat(document.querySelector('#txtn1').value);
  const n2 = parseFloat(document.querySelector('#txtn2').value);

  // Comprobar si los valores son números válidos
  if (isNaN(n1) || isNaN(n2)) {
    alert('Por favor, introduce números válidos');
    return;
  }

  // Realizar la suma
  const resultado = n1 + n2;

  // Mostrar el resultado en el input de resultado
  document.querySelector('#txtres').value = resultado;
}

// Definir el nuevo elemento personalizado
customElements.define('componente-suma', Suma);

```

- Al cargarse, el componente añade un listener al botón de suma (#btnSumar) para ejecutar el método sumar(), el cual toma los valores de los campos de entrada (#txtn1 y #txtn2), realiza la suma y muestra el resultado en el campo de texto de solo lectura (#txtres).

pagina6.html

- Esta página HTML contiene un encabezado principal y un componente web personalizado `<componente-calcular>`, demostrando el uso de Custom Elements y Shadow DOM para encapsular tanto la funcionalidad como los estilos. Un script externo "componente6.js" gestiona el comportamiento del componente.

CUSTOM ELEMENT + SHADOW DOM

5	
6	
30	
Sumar	
Restar	
Multiplicar	
Dividir	

- Este archivo JavaScript crea un componente web personalizado <componente-calcular> que facilita la realización de operaciones matemáticas básicas (suma, resta, multiplicación y división) entre dos números ingresados por el usuario. Emplea Shadow DOM para encapsular los estilos y la funcionalidad del componente.

```
class Calcular extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({ mode: 'open' });

    // Crear un contenedor para los estilos y la funcionalidad
    const container = document.createElement('div');
    container.innerHTML = `
    <style>
      /* Estilos del componente */
      :host {
        display: block;
        margin-top: 10px;
        background-color: yellow;
        text-align: center;
      }
    </style>
    <div>
      <input type="text" id="txtn1" placeholder="Número 1">
      <br/>
      <input type="text" id="txtn2" placeholder="Número 2">
      <br/>
      <input type="text" id="txtres" placeholder="Resultado" readonly>
      <br/>
      <button id="btnSumar">Sumar</button>
      <br/>
      <button id="btnRestar">Restar</button>
      <br/>
      <button id="btnMultiplicar">Multiplicar</button>
      <br/>
      <button id="btnDividir">Dividir</button>
    </div>
  `;
  }
```

- El componente incluye botones para cada operación, y cuando se hace clic en alguno de ellos, se realiza la operación correspondiente y se muestra el resultado en un campo de texto de solo lectura.

```
    calcular(operacion) {  
      // Obtener los valores de los inputs  
      const n1 = parseFloat(this.shadowRoot.querySelector('#txtn1').value);  
      const n2 = parseFloat(this.shadowRoot.querySelector('#txtn2').value);  
  
      // Comprobar si los valores son números válidos  
      if (isNaN(n1) || isNaN(n2)) {  
        alert('Por favor, introduce números válidos');  
        return;  
      }  
  
      // Realizar la operación correspondiente  
      let resultado;  
      switch (operacion) {  
        case 'sumar':  
          resultado = n1 + n2;  
          break;  
        case 'restar':  
          resultado = n1 - n2;  
          break;  
        case 'multiplicar':  
          resultado = n1 * n2;  
          break;  
        case 'dividir':  
          resultado = n1 / n2;  
          break;  
        default:  
          return;  
      }  
  
      // Mostrar el resultado en el input de resultado  
      this.shadowRoot.querySelector('#txtres').value = resultado;  
    }  
  }  
}
```

pagina7.html

Formulario con Web Components

despues del web component

- Esta página HTML presenta un encabezado principal y un componente web personalizado `<custom-button>`, ilustrando cómo se puede mejorar un formulario utilizando Web Components. Después del componente, hay otro encabezado, y el comportamiento del componente es controlado por un script externo llamado "componente7.js".

- Este archivo JavaScript crea un componente web personalizado <custom-button>. La clase CustomButon hereda de HTMLElement y, al ser instanciada, muestra un mensaje en la consola confirmando la creación del botón. El componente se registra en el navegador utilizando customElements.define.

```
class CustomButon extends HTMLElement {  
  constructor() {  
    super();  
    console.log('boton creado');  
  }  
}  
window.customElements.define('custom-button', CustomButon);
```

```
window.customElements.define('custom-button', CustomButon);
```

boton.html

- ▶ Esta página HTML incluye un formulario simple con un campo de texto y un botón de envío, además de un componente web personalizado <mi-boton>. El formulario tiene una etiqueta y un campo de entrada para ingresar un mensaje. Se incluye un script externo "boton.js" que gestiona el comportamiento del componente personalizado.

The image shows a web browser window with a form titled "boton.html". On the left, there is a label "Mensaje:" followed by a text input field. Below the input field is a small button labeled "Enviar". To the right of the input field is a large yellow button labeled "Botón Personalizado". On the right side of the browser window, there is a dark notification box. Inside this box, the text "servicios-integrados.net dice" is displayed in a light color, followed by "¡Botón personalizado clickeado!". At the bottom right of the notification box is a blue button labeled "Aceptar".

- Este archivo JavaScript define un componente web personalizado <mi-boton>. La clase MiBoton extiende HTMLElement y utiliza Shadow DOM para encapsular el botón personalizado. Al ser instanciado, crea un botón con texto "Botón Personalizado", aplica estilos específicos y añade un evento de clic que muestra un mensaje de alerta al hacer clic en el botón.

```
class MiBoton extends HTMLElement {
  constructor() {
    super();
    // Attach a shadow root to the element.
    this.attachShadow({ mode: 'open' });

    // Create a button element.
    const button = document.createElement('button');
    button.textContent = 'Botón Personalizado';

    // Apply styles to the button.
    const style = document.createElement('style');
    style.textContent = `
      button {
        background-color: yellow;
        border: none;
        color: white;
        padding: 15px 32px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: 4px 2px;
        cursor: pointer;
        border-radius: 12px;
      }
    `;

    // Append the style and button to the shadow DOM.
    this.shadowRoot.append(style, button);

    // Add event listener to handle button click.
    button.addEventListener('click', this.handleClick);
  }
}
```


index2.html

- ▶ Esta página HTML incluye un encabezado principal y un componente web personalizado <elemento-uno>. El script "my-element.js", que probablemente define el comportamiento del componente personalizado, es cargado al final del documento usando el atributo type="module".

Web Components - Custom Elements

CONTENIDOS DEL PARRAFO !

Parrafo 1: template fuera de la Clase

Parrafo 2

Parrafo 3

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Custom Elements</title>
</head>

<body>
  <h1 id="titulo1">Web Components - Custom Elements</h1>
  <elemento-uno></elemento-uno>
  <script type="module" src="my-element.js"></script>
</body>

</html>
```

- ▶ Este archivo JavaScript define un componente web personalizado <elemento-uno>. La clase myElement extiende HTMLElement y, al ser instanciada, crea un nuevo párrafo (<p>) dentro del constructor.
- ▶ En el método connectedCallback, se añade texto al párrafo y se agrega al DOM del componente.
- ▶ Además, se incluye un template HTML estático definido fuera de la clase myElement, el cual contiene estilos CSS y dos párrafos (<p>). Este template se añade al DOM del componente durante el connectedCallback.

```
const template = document.createElement("div");
template.innerHTML = `
  <style>
    .texto {
      color: red;
    }
    p {
      color: blue;
    }
  </style>
  <p class="texto" id="template1">Párrafo 1:  template fuera de la Clase</p>
  <p id="template2">Párrafo 2</p>
`;

class myElement extends HTMLElement {
  constructor() {
    super();
    this.p = document.createElement("p");
  }
  connectedCallback() {
    this.p.textContent = "CONTENIDOS DEL PARRAFO !";
    this.appendChild(this.p);
    this.appendChild(template);
  }
}
customElements.define("elemento-uno", myElement);
```