DOM - DOCUMENT OBJECT MODEL

El DOM es un estándar del W3C (World Wide Web Consortium).

El DOM define un estándar para acceder a documentos:

"El Modelo de Objetos de Documento (DOM) del W3C es una plataforma y una interfaz neutral en cuanto al idioma que permite a los programas y scripts acceder y actualizar dinámicamente el contenido, la estructura y el estilo de un documento".

Primer objetivo de los contenidos de aprendizaje, se requieren conocer:

Cómo cambiar el contenido de los elementos HTML

Cómo cambiar el estilo (CSS) de los elementos HTML

Cómo reaccionar ante eventos HTML DOM

Cómo agregar y eliminar elementos HTML

El estándar W3C DOM se divide en 3 partes diferentes:

Core DOM: modelo estándar para todo tipo de documentos

XML DOM: modelo estándar para documentos XML

HTML DOM: modelo estándar para documentos HTML

**METODOS y PROPIEDADES DEL DOM**: Los métodos HTML DOM son acciones que puede realizar (en elementos HTML). Las propiedades HTML DOM son valores (de elementos HTML) que puede configurar o cambiar.

**PRINCIPALES METODOS DEL DOM:**

document.getElementById(id): Retorna el elemento con el ID especificado.

document.getElementsByClassName(class): Retorna todos los elementos con la clase especificada.

document.getElementsByTagName(tag): Retorna todos los elementos con el nombre de etiqueta especificado.

document.createElement(tagName): Crea un nuevo elemento HTML.

document.createTextNode(text): Crea un nuevo nodo de texto.

document.appendChild(child): Añade un nodo como último hijo de un nodo padre.

document.removeChild(child): Remueve un nodo hijo de un nodo padre.

document.replaceChild(newChild, oldChild): Reemplaza un nodo hijo con otro nodo.

document.cloneNode(deep): Clona un nodo. Si deep es true, clona también todos sus hijos.

document.addEventListener(event, function): Añade un evento a un elemento.

document.removeEventListener(event, function): Remueve un evento de un elemento.

document.setAttribute(name, value): Establece un atributo en un elemento.

document.getAttribute(name): Obtiene el valor de un atributo de un elemento.

document.createComment: Método para crear un nuevo nodo de comentario.

document.createDocumentFragment: Método para crear un fragmento de documento.

document.querySelector: Método para obtener el primer elemento que coincide con un selector CSS.

document.querySelectorAll: Método para obtener todos los elementos que coinciden con un selector CSS.

document.adoptNode: Método para adoptar un nodo de otro documento.

document.importNode: Método para importar un nodo de otro documento.

document.contains: Método para verificar si un nodo contiene a otro nodo.

document.compareDocumentPosition: Método para comparar la posición de dos nodos en el documento.

document.normalize: Método para normalizar el árbol del documento.

document.renameNode: Método para renombrar un nodo.

**PRINCIPALES PROPIEDADES DOM:**

document.title: El título del documento.

document.URL: La URL completa del documento.

document.domain: El dominio del documento.

document.body: El elemento `<body>` del documento.

document.head: El elemento `<head>` del documento.

document.cookie: Las cookies del documento.

document.readyState: El estado de carga del documento (e.g., "loading", "interactive", "complete").

document.referrer: La URL del documento de referencia.

document.doctype: El tipo de documento.

document.scripts: Colección de todos los elementos `<script>` en el documento.

document.forms: Colección de todos los formularios en el documento.

document.images: Colección de todas las imágenes en el documento.

document.links: Colección de todos los enlaces en el documento.

document.anchors: Colección de todos los elementos `<a>` con un atributo `name`.

document.characterSet: El conjunto de caracteres del documento.

document.children: Una colección de los hijos del nodo actual.

document.firstElementChild: El primer hijo elemento del nodo actual.

document.lastElementChild: El último hijo elemento del nodo actual.

document.activeElement: El elemento actualmente enfocado en el documento.

document.designMode: Controla si el documento puede ser editado.

document.implementation: Proporciona una interfaz para las funcionalidades del DOM.

document.styleSheets: Colección de todas las hojas de estilo en el documento.

document.visibilityState: Estado de visibilidad del documento.

document.fullscreenElement: El elemento que está en pantalla completa.

document.defaultView: Referencia al objeto window asociado con el documento.

document.scrollingElement: El elemento que está en scroll.

document.dir: Dirección del texto (e.g., "ltr" para izquierda a derecha, "rtl" para derecha a izquierda).

document.location: El objeto location asociado con el documento.

document.documentElement: El elemento raíz del documento (generalmente `<html>`).

document.hasFocus: Verifica si el documento tiene el foco.

document.lastModified: Fecha y hora de la última modificación del documento.

document.embeds: Colección de todos los elementos `<embed>` en el documento.

document.plugins: Colección de todos los plugins del documento.

document.scripts: Colección de todos los scripts del documento.

document.currentScript: El script que se está ejecutando actualmente.

La propiedad InnerHTML: La forma más sencilla de obtener el contenido de un elemento es mediante la propiedad InnerHTML. La propiedad InnerHTML es útil para obtener o reemplazar el contenido de elementos HTML.

La propiedad insideHTML se puede utilizar para obtener o cambiar cualquier elemento HTML, incluidos <html> y <body>.

**PRINCIPALES ATRIBUTOS ESTANDARES DEL DOM**

id=(tipo)= (string): Identificador único del elemento.

class=(tipo)= (string): Lista de clases del elemento.

src=(tipo)= (string): URL de la fuente, común en elementos `<img>`, `<script>`, etc.

href=(tipo)= (string): URL del enlace, común en elementos `<a>`.

alt=(tipo)= (string): Texto alternativo, usado en `<img>`.

title=(tipo)= (string): Texto que aparece como información emergente.

value=(tipo)= (string): Valor del elemento, común en `<input>`, `<textarea>`, etc.

name=(tipo)= (string): Nombre del elemento, común en formularios.

type=(tipo)= (string): Tipo del elemento, usado en `<button>`, `<input>`, etc.

placeholder=(tipo)= (string): Texto de marcador de posición en `<input>` o `<textarea>`.

disabled=(tipo)= (boolean): Indica si el elemento está deshabilitado.

checked=(tipo)= (boolean): Indica si el elemento está seleccionado (en `<input type="checkbox">` o `<input type="radio">`).

readonly=(tipo)= (boolean): Indica si el elemento es de solo lectura.

maxlength=(tipo)= (number): Longitud máxima del valor del elemento.

style=(tipo)= (string): Define los estilos CSS en línea del elemento.

**PRINCIPALES ATRIBUTOS NO ESTANDARES DEL DOM**

data= (e.g., data-custom)- : Atributos personalizados para almacenar datos adicionales.

autocapitalize : Controla la capitalización automática del texto.

autocorrect : Activa o desactiva la corrección automática del texto.

autofocus : Indica si un elemento debe recibir el foco al cargarse la página.

inputmode : Proporciona una sugerencia sobre qué tipo de teclado se debe mostrar.

spellcheck : Indica si el corrector ortográfico debe revisarse o no.

draggable : Define si un elemento es arrastrable.

enterkeyhint : Sugerencia para la acción del teclado "Enter".

part : Define el componente de sombra del DOM que es parte del árbol.

translate : Indica si el contenido del elemento debe traducirse o no.

referrerpolicy : Controla la política del referer al cargar recursos.

loading : Sugerencia sobre cómo cargar recursos (e.g., "lazy" para carga diferida).

importance : Sugerencia sobre la importancia de un recurso.

intrinsicsize : Especifica el tamaño intrínseco de un recurso.

nonce : Utilizado para proteger contenido contra ataques de inyección de scripts.

**PRINCIPALES EVENTOS DEL DOM ASOCIADOS A JAVASCRIPT**

onclick: Se dispara cuando un elemento es clicado.

ondblclick: Se dispara cuando un elemento es doble clicado.

onmousedown: Se dispara cuando se presiona un botón del ratón sobre un elemento.

onmouseup: Se dispara cuando se suelta un botón del ratón sobre un elemento.

onmouseover: Se dispara cuando el puntero del ratón se mueve sobre un elemento.

onmouseout: Se dispara cuando el puntero del ratón se mueve fuera de un elemento.

onmousemove: Se dispara cuando el puntero del ratón se mueve mientras está sobre un elemento.

oncontextmenu: Se dispara cuando se abre un menú contextual (usualmente clic derecho).

onkeydown: Se dispara cuando se presiona una tecla.

onkeypress: Se dispara cuando se presiona y se mantiene una tecla.

onkeyup: Se dispara cuando se suelta una tecla.

onfocus: Se dispara cuando un elemento recibe el foco.

onblur: Se dispara cuando un elemento pierde el foco.

oninput: Se dispara cuando el valor de un elemento de entrada cambia.

onchange: Se dispara cuando el valor de un elemento de entrada ha cambiado y el elemento pierde el foco.

onsubmit: Se dispara cuando se envía un formulario.

onreset: Se dispara cuando se reinicia un formulario.

onselect: Se dispara cuando se selecciona el texto dentro de un elemento `<input>` o `<textarea>`.

oninvalid: Se dispara cuando un elemento de entrada es inválido.

onload: Se dispara cuando un objeto ha sido cargado.

onunload: Se dispara cuando la página se descarga.

onbeforeunload: Se dispara antes de que la página se descargue.

onresize: Se dispara cuando el tamaño de la ventana del navegador cambia.

onscroll: Se dispara cuando se desplaza un elemento de scroll.

ondrag: Se dispara cuando un elemento se está arrastrando.

ondragstart: Se dispara cuando comienza a arrastrarse un elemento.

ondragend: Se dispara cuando se termina de arrastrar un elemento.

ondragover: Se dispara cuando un elemento arrastrado está sobre un destino de caída.

ondragenter: Se dispara cuando un elemento arrastrado entra en un destino de caída.

ondragleave: Se dispara cuando un elemento arrastrado deja un destino de caída.

ondrop: Se dispara cuando un elemento arrastrado se suelta en un destino de caída.

**PRINCIPALES ELEMENTOS DEL DOM**

<html>: El elemento raíz de un documento HTML.

<head>: Contiene metadatos/información para el documento.

<title>: Define el título del documento.

<meta>: Define metadatos sobre un documento HTML.

<link>: Define la relación entre el documento actual y un recurso externo.

<style>: Contiene reglas de estilo para el documento.

<script>: Define un script de cliente.

<noscript>: Proporciona contenido alternativo para usuarios que tienen scripts deshabilitados.

<body>: Contiene el contenido visible de la página web.

<header>: Define una cabecera para un documento o una sección.

<nav>: Define un conjunto de enlaces de navegación.

<main>: Representa el contenido principal del `<body>`.

<section>: Define una sección en un documento.

<article>: Representa una pieza independiente de contenido.

<aside>: Contenido aparte del contenido principal (barras laterales, etc.).

<footer>: Define un pie de página para un documento o una sección.

<h1>: Encabezado de primer nivel.

<h2>: Encabezado de segundo nivel.

<h3>: Encabezado de tercer nivel.

<h4>: Encabezado de cuarto nivel.

<h5>: Encabezado de quinto nivel.

<h6>: Encabezado de sexto nivel.

<p>: Párrafo.

<hr>: Representa un cambio temático en el contenido.

<br>: Salto de línea.

<pre>: Define texto preformateado.

<blockquote>: Define una sección que es citada de otra fuente.

<ol>: Lista ordenada.

<ul>: Lista desordenada.

<li>: Elemento de una lista.

<dl>: Lista de descripción.

<dt>: Término en una lista de descripción.

<dd>: Descripción de un término en una lista de descripción.

<figure>: Contenedor autónomo para contenido ilustrativo.

<figcaption>: Pie de foto para el elemento `<figure>`.

<div>: Contenedor genérico para contenido en bloque.

<span>: Contenedor genérico para contenido en línea.

<a>: Define un hipervínculo.

<em>: Define texto enfatizado.

<strong>: Define texto fuerte.

<small>: Define texto pequeño.

<mark>: Define texto resaltado.

<del>: Define texto eliminado.

<ins>: Define texto insertado.

<img>: Representa una imagen.

<iframe>: Representa un marco embebido.

<embed>: Embebe contenido externo (multimedia, por ejemplo).

<object>: Representa un objeto embebido.

<param>: Define parámetros para un elemento `<object>`.

<video>: Embebe contenido de video.

La interfaz de programación DOM : Se puede acceder al HTML DOM con JavaScript (y con otros lenguajes de programación). En el DOM, todos los elementos HTML se definen como objetos.

La interfaz de programación son las propiedades y métodos de cada objeto.

Una propiedad es un valor que puede obtener o establecer (como cambiar el contenido de un elemento HTML). Un método es una acción que puedes realizar (como agregar o eliminar un elemento HTML).

La propiedad InnerHTML: La forma más sencilla de obtener el contenido de un elemento es mediante la propiedad InnerHTML. La propiedad InnerHTML es útil para obtener o reemplazar el contenido de elementos HTML.

La propiedad insideHTML se puede utilizar para obtener o cambiar cualquier elemento HTML, incluidos <html> y <body>.

**PRINCIPALES APIs COMO HERRAMIENTAS PARA WEB COMPONENTS**

Estas APIs proporcionan las herramientas necesarias para la creación, manipulación y gestión de Web Components, permitiendo a los desarrolladores construir componentes web reutilizables y modulares.

1. Custom Elements API

- Concepto: Permite la definición de nuevos elementos HTML personalizados.

- Funcionalidad: Proporciona `customElements.define()` para registrar nuevos elementos y la clase `HTMLElement` para crear componentes personalizados.

2. Shadow DOM API

- Concepto: Crea un DOM separado y encapsulado dentro de un elemento.

- Funcionalidad: Usa `element.attachShadow({mode: 'open'|'closed'})` para crear un shadow root y `shadowRoot.innerHTML` para manipular su contenido.

3. HTML Templates

- Concepto: Define plantillas de contenido reutilizable.

- Funcionalidad: Utiliza la etiqueta `<template>` para definir fragmentos de HTML que no se renderizan hasta ser clonados.

4. HTML Imports (deprecated)

- Concepto: Importar HTML en otros documentos.

- Funcionalidad: Utilizaba la etiqueta `<link rel="import">` para importar documentos HTML. (Ahora se reemplaza con módulos JavaScript).

5. Document Fragment API

- Concepto: Crea nodos de documento sin elementos de contenedor.

- Funcionalidad: Usa `document.createDocumentFragment()` para crear un fragmento de documento para operaciones de DOM más eficientes.

6. Mutation Observer API

- Concepto: Observa cambios en el DOM.

- Funcionalidad: Utiliza `new MutationObserver(callback)` para crear un observador que detecta cambios en el DOM y `observer.observe(target, options)` para especificar qué observar.

7. Template Content API

- Concepto: Accede al contenido de una plantilla.

- Funcionalidad: Usa `template.content` para obtener el contenido de un elemento `<template>`.

8. HTMLSlotElement API

- Concepto: Maneja ranuras (slots) en Shadow DOM.

- Funcionalidad: Usa `slot` para definir puntos de inserción y `assignedNodes()` para obtener los nodos asignados a un slot.

9. Element.attachInternals()

- Concepto: Proporciona un objeto internals para componentes personalizados.

- Funcionalidad: Usa `element.attachInternals()` para acceder a funcionalidades internas como `form-associated custom elements`.

10. Element.closest()

- Concepto: Encuentra el ancestro más cercano que coincide con un selector.

- Funcionalidad: Usa `element.closest(selector)` para obtener el elemento ancestro más cercano.

11. Element.getRootNode()

- Concepto: Obtiene el nodo raíz de un elemento.

- Funcionalidad: Usa `element.getRootNode(options)` para obtener el nodo raíz, útil en Shadow DOM.

12. Document.createElement()

- Concepto: Crea un nuevo elemento HTML.

- Funcionalidad: Usa `document.createElement(tagName)` para crear un nuevo elemento.

13. Document.importNode()

- Concepto: Importa un nodo de otro documento.

- Funcionalidad: Usa `document.importNode(node, deep)` para importar nodos.

14. HTMLElement.attachShadow()

- Concepto: Crea un Shadow DOM para el elemento.

- Funcionalidad: Usa `element.attachShadow({mode: 'open'|'closed'})` para crear un Shadow DOM.

15. HTMLElement.shadowRoot

- Concepto: Obtiene el shadow root de un elemento.

- Funcionalidad: Accede con `element.shadowRoot` al shadow root adjunto.

16. Element.innerHTML

- Concepto: Establece o devuelve el HTML dentro de un elemento.

- Funcionalidad: Usa `element.innerHTML` para manipular el contenido HTML.

17. Element.setAttribute()

- Concepto: Establece el valor de un atributo.

- Funcionalidad: Usa `element.setAttribute(name, value)` para definir un atributo.

18. Element.getAttribute()

- Concepto: Obtiene el valor de un atributo.

- Funcionalidad: Usa `element.getAttribute(name)` para obtener el valor de un atributo.

19. Element.removeAttribute()

- Concepto: Elimina un atributo de un elemento.

- Funcionalidad: Usa `element.removeAttribute(name)` para eliminar un atributo.

20. Element.hasAttribute()

- Concepto: Comprueba si un elemento tiene un atributo.

- Funcionalidad: Usa `element.hasAttribute(name)` para verificar la presencia de un atributo.

21. Node.cloneNode()

- Concepto: Clona un nodo.

- Funcionalidad: Usa `node.cloneNode(deep)` para clonar un nodo y sus hijos opcionalmente.

22. Node.appendChild()

- Concepto: Añade un nodo como último hijo de un nodo padre.

- Funcionalidad: Usa `node.appendChild(child)` para añadir un nodo hijo.

23. Node.removeChild()

- Concepto: Elimina un nodo hijo de un nodo padre.

- Funcionalidad: Usa `node.removeChild(child)` para eliminar un nodo hijo.

24. Node.insertBefore()

- Concepto: Inserta un nodo antes de otro nodo.

- Funcionalidad: Usa `node.insertBefore(newNode, referenceNode)` para insertar un nodo.

25. Node.replaceChild()

- Concepto: Reemplaza un nodo hijo con otro nodo.

- Funcionalidad: Usa `node.replaceChild(newChild, oldChild)` para reemplazar un nodo hijo.

26. NodeList

- Concepto: Colección de nodos, similar a un array.

- Funcionalidad: Usado en métodos como `querySelectorAll` para manejar múltiples nodos.

27. HTMLCollection

- Concepto: Colección de elementos, similar a un array.

- Funcionalidad: Usado en propiedades como `children` para manejar múltiples elementos.

28. HTMLElement.innerHTML

- Concepto: Establece o devuelve el HTML dentro de un elemento.

- Funcionalidad: Usa `element.innerHTML` para manipular el contenido HTML.

29. HTMLElement.outerHTML

- Concepto: Establece o devuelve el HTML completo del elemento.

- Funcionalidad: Usa `element.outerHTML` para obtener o establecer el HTML completo de un elemento.

30. HTMLElement.toggleAttribute()

- Concepto: Alterna un atributo en un elemento.

- Funcionalidad: Usa `element.toggleAttribute(name, force)` para agregar o eliminar un atributo basado en una condición.

**CLASES E INTERFACES DE JAVASCRIPT POR LA IMPORTANCIA DE SU IMPORTANCIA EN EL DESARROLLO DE SOFTWARE**

1. Event

- Concepto: Representa cualquier tipo de evento en el DOM.

- Funcionalidad: Proporciona propiedades y métodos comunes para cualquier evento.

- Importancia: Fundamental para el manejo de eventos en la web.

- Superclases: Object

- Subclases: UIEvent, MouseEvent, KeyboardEvent, etc.

- Clases asociadas: EventTarget

2. Element

- Concepto: Clase base para todos los elementos en un documento HTML o XML.

- Funcionalidad: Proporciona métodos y propiedades para la manipulación del DOM.

- Importancia: Crucial para cualquier manipulación del DOM.

- Superclases: Node

- Subclases: HTMLElement, SVGElement

- Clases asociadas: Document, Node

3. HTMLElement

- Concepto: Representa cualquier elemento HTML.

- Funcionalidad: Proporciona propiedades y métodos específicos para elementos HTML.

- Importancia: Vital para la creación y manipulación de elementos HTML.

- Superclases: Element

- Subclases: HTMLDivElement, HTMLSpanElement, etc.

- Clases asociadas: Document

4. Document

- Concepto: Representa todo el documento HTML o XML.

- Funcionalidad: Proporciona acceso al contenido del documento y su estructura.

- Importancia: Esencial para acceder y manipular el DOM.

- Superclases: Node

- Subclases: HTMLDocument, XMLDocument

- Clases asociadas: Window, Element

5. Node

- Concepto: Representa un nodo en el árbol DOM.

- Funcionalidad: Proporciona propiedades y métodos comunes para todos los nodos del DOM.

- Importancia: Base de todos los objetos del DOM.

- Superclases: EventTarget

- Subclases: Element, Document, Text, Comment

- Clases asociadas: NodeList, NamedNodeMap

6. EventTarget

- Concepto: Interfaz implementada por objetos que pueden recibir eventos y pueden tener listeners.

- Funcionalidad: Proporciona métodos para manejar eventos (`addEventListener`, `removeEventListener`, `dispatchEvent`).

- Importancia: Fundamental para el sistema de eventos en el DOM.

- Superclases: Object

- Subclases: Node, Window, XMLHttpRequest

- Clases asociadas: Event

7. NodeList

- Concepto: Colección de nodos, generalmente el resultado de métodos de selección de DOM.

- Funcionalidad: Proporciona acceso a un conjunto de nodos.

- Importancia: Importante para manipular conjuntos de elementos.

- Superclases: Object

- Subclases: StaticNodeList, LiveNodeList

- Clases asociadas: Document, Element

8. HTMLCollection

- Concepto: Colección de elementos HTML.

- Funcionalidad: Proporciona acceso a un conjunto de elementos.

- Importancia: Similar a NodeList, pero específicamente para elementos HTML.

- Superclases: Object

- Subclases: None

- Clases asociadas: Document, Element

9. CSSStyleDeclaration

- Concepto: Representa un conjunto de propiedades CSS.

- Funcionalidad: Permite acceder y modificar estilos CSS.

- Importancia: Esencial para la manipulación de estilos en la web.

- Superclases: Object

- Subclases: None

- Clases asociadas: HTMLElement, CSSStyleRule

10. HTMLDivElement

- Concepto: Representa un elemento `<div>`.

- Funcionalidad: Proporciona propiedades específicas para `<div>`.

- Importancia: Muy común en la construcción de estructuras de página.

- Superclases: HTMLElement

- Subclases: None

- Clases asociadas: HTMLSpanElement, HTMLParagraphElement

11. HTMLSpanElement

- Concepto: Representa un elemento `<span>`.

- Funcionalidad: Proporciona propiedades específicas para `<span>`.

- Importancia: Utilizado para aplicar estilos a partes del texto.

- Superclases: HTMLElement

- Subclases: None

- Clases asociadas: HTMLDivElement, HTMLParagraphElement

12. HTMLInputElement

- Concepto: Representa un elemento `<input>`.

- Funcionalidad: Proporciona propiedades y métodos específicos para `<input>`.

- Importancia: Crucial para formularios y entrada de datos.

- Superclases: HTMLElement

- Subclases: None

- Clases asociadas: HTMLFormElement, HTMLButtonElement

13. HTMLButtonElement

- Concepto: Representa un elemento `<button>`.

- Funcionalidad: Proporciona propiedades y métodos específicos para `<button>`.

- Importancia: Utilizado ampliamente en interacciones de usuario.

- Superclases: HTMLElement

- Subclases: None

- Clases asociadas: HTMLFormElement, HTMLInputElement

14. HTMLAnchorElement

- Concepto: Representa un elemento `<a>`.

- Funcionalidad: Proporciona propiedades y métodos específicos para `<a>`.

- Importancia: Esencial para la navegación en la web.

- Superclases: HTMLElement

- Subclases: None

- Clases asociadas: HTMLLinkElement, HTMLAreaElement

15. HTMLFormElement

- Concepto: Representa un elemento `<form>`.

- Funcionalidad: Proporciona propiedades y métodos específicos para `<form>`.

- Importancia: Crucial para la recopilación de datos de usuario.

- Superclases: HTMLElement

- Subclases: None

- Clases asociadas: HTMLInputElement, HTMLButtonElement

16. Text

- Concepto: Representa nodos de texto en el DOM.

- Funcionalidad: Permite la manipulación de nodos de texto.

- Importancia: Fundamental para la manipulación del contenido de texto.

- Superclases: Node

- Subclases: None

- Clases asociadas: Comment, DocumentFragment

17. Comment

- Concepto: Representa comentarios en el DOM.

- Funcionalidad: Permite la manipulación de nodos de comentario.

- Importancia: Útil para documentación y depuración.

- Superclases: Node

- Subclases: None

- Clases asociadas: Text, DocumentFragment

18. DocumentFragment

- Concepto: Representa un fragmento de documento.

- Funcionalidad: Utilizado para realizar manipulaciones del DOM fuera del DOM real.

- Importancia: Mejora el rendimiento en manipulaciones de DOM complejas.

- Superclases: Node

- Subclases: None

- Clases asociadas: Text, Comment

19. XMLHttpRequest

- Concepto: Proporciona una forma de interactuar con servidores.

- Funcionalidad: Permite enviar y recibir datos a través de HTTP.

- Importancia: Fundamental para la comunicación asíncrona (AJAX).

- Superclases: EventTarget

- Subclases: None

- Clases asociadas: Fetch API, Promise

20. FormData

- Concepto: Representa datos de formulario.

- Funcionalidad: Permite construir fácilmente conjuntos de pares clave/valor para enviar con XMLHttpRequest.

- Importancia: Simplifica la manipulación de formularios.

- Superclases: Object

- Subclases: None

- Clases asociadas: XMLHttpRequest, HTMLFormElement

21. URL

- Concepto: Representa una URL.

- Funcionalidad: Proporciona métodos para trabajar con URLs.

- Importancia: Esencial para la manipulación de URLs.

- Superclases: Object

- Subclases: None

- Clases asoci

adas: URLSearchParams

22. URLSearchParams

- Concepto: Representa los parámetros de búsqueda de una URL.

- Funcionalidad: Proporciona métodos para trabajar con los parámetros de búsqueda de una URL.

- Importancia: Facilita la manipulación de parámetros en URLs.

- Superclases: Object

- Subclases: None

- Clases asociadas: URL

23. Window

- Concepto: Representa una ventana que contiene un documento DOM.

- Funcionalidad: Proporciona propiedades y métodos para controlar la ventana del navegador.

- Importancia: Fundamental para la interacción con el navegador.

- Superclases: EventTarget

- Subclases: None

- Clases asociadas: Document, Navigator

24. Navigator

- Concepto: Representa el estado y la identidad del agente de usuario.

- Funcionalidad: Proporciona información sobre el navegador y el dispositivo.

- Importancia: Esencial para obtener información del entorno del usuario.

- Superclases: Object

- Subclases: None

- Clases asociadas: Window

25. History

- Concepto: Proporciona una interfaz para interactuar con el historial del navegador.

- Funcionalidad: Permite la navegación en el historial del navegador.

- Importancia: Crucial para la navegación del lado del cliente.

- Superclases: Object

- Subclases: None

- Clases asociadas: Window, Location

26. Location

- Concepto: Representa la URL de la ventana actual.

- Funcionalidad: Proporciona métodos para cambiar la URL de la ventana actual.

- Importancia: Esencial para la manipulación de la URL.

- Superclases: Object

- Subclases: None

- Clases asociadas: Window, History

27. Screen

- Concepto: Representa la pantalla de salida.

- Funcionalidad: Proporciona propiedades sobre la pantalla de salida del navegador.

- Importancia: Útil para la manipulación de pantallas y ventanas.

- Superclases: Object

- Subclases: None

- Clases asociadas: Window

28. Storage

- Concepto: Proporciona acceso a una sesión o almacenamiento local.

- Funcionalidad: Permite almacenar pares clave/valor.

- Importancia: Crucial para el almacenamiento de datos en el navegador.

- Superclases: Object

- Subclases: None

- Clases asociadas: LocalStorage, SessionStorage

29. LocalStorage

- Concepto: Proporciona acceso al almacenamiento local.

- Funcionalidad: Permite almacenar pares clave/valor que persisten a través de sesiones.

- Importancia: Esencial para el almacenamiento persistente en el navegador.

- Superclases: Storage

- Subclases: None

- Clases asociadas: SessionStorage

30. SessionStorage

- Concepto: Proporciona acceso al almacenamiento de sesión.

- Funcionalidad: Permite almacenar pares clave/valor que persisten solo durante la sesión de la página.

- Importancia: Crucial para el almacenamiento temporal en el navegador.

- Superclases: Storage

- Subclases: None

- Clases asociadas: LocalStorage

31. Promise

- Concepto: Representa un valor que puede estar disponible ahora, en el futuro, o nunca.

- Funcionalidad: Facilita la programación asíncrona.

- Importancia: Fundamental para la gestión de operaciones asíncronas.

- Superclases: Object

- Subclases: None

- Clases asociadas: AsyncFunction

32. ServiceWorker

- Concepto: Proporciona una forma de interceptar y manejar las solicitudes de red.

- Funcionalidad: Permite la creación de aplicaciones web offline.

- Importancia: Esencial para aplicaciones progresivas.

- Superclases: Worker

- Subclases: None

- Clases asociadas: Navigator, Promise

33. Worker

- Concepto: Proporciona un entorno para ejecutar scripts en hilos de fondo.

- Funcionalidad: Permite realizar tareas de manera concurrente sin bloquear la interfaz de usuario.

- Importancia: Crucial para operaciones de larga duración.

- Superclases: EventTarget

- Subclases: ServiceWorker, SharedWorker

- Clases asociadas: WorkerGlobalScope

34. BroadcastChannel

- Concepto: Proporciona una forma simple de comunicar entre diferentes contextos de navegación.

- Funcionalidad: Permite la comunicación entre ventanas, iframes, workers, etc.

- Importancia: Útil para sincronización de datos en múltiples contextos.

- Superclases: EventTarget

- Subclases: None

- Clases asociadas: MessageChannel

35. MessageChannel

- Concepto: Crea un canal de mensajes para comunicación entre diferentes contextos.

- Funcionalidad: Permite la comunicación bidireccional entre diferentes contextos.

- Importancia: Esencial para la comunicación entre workers y ventanas.

- Superclases: Object

- Subclases: None

- Clases asociadas: BroadcastChannel, MessagePort

36. MessagePort

- Concepto: Representa un extremo de un canal de mensajes.

- Funcionalidad: Permite el envío y recepción de mensajes.

- Importancia: Crucial para la comunicación entre diferentes partes de una aplicación web.

- Superclases: EventTarget

- Subclases: None

- Clases asociadas: MessageChannel

37. URLPattern

- Concepto: Permite trabajar con patrones de URL.

- Funcionalidad: Facilita el análisis y la coincidencia de URLs.

- Importancia: Útil para el enrutamiento en aplicaciones web.

- Superclases: Object

- Subclases: None

- Clases asociadas: URL

38. Request

- Concepto: Representa una solicitud HTTP.

- Funcionalidad: Proporciona propiedades y métodos para manejar solicitudes HTTP.

- Importancia: Fundamental para la API Fetch.

- Superclases: Object

- Subclases: None

- Clases asociadas: Response, Headers

39. Response

- Concepto: Representa una respuesta HTTP.

- Funcionalidad: Proporciona propiedades y métodos para manejar respuestas HTTP.

- Importancia: Crucial para la API Fetch.

- Superclases: Object

- Subclases: None

- Clases asociadas: Request, Headers

40. Headers

- Concepto: Representa los encabezados de una solicitud o respuesta HTTP.

- Funcionalidad: Permite la manipulación de los encabezados HTTP.

- Importancia: Esencial para la API Fetch.

- Superclases: Object

- Subclases: None

- Clases asociadas: Request, Response

41. Fetch API

- Concepto: Proporciona una interfaz para hacer solicitudes HTTP.

- Funcionalidad: Facilita la obtención de recursos a través de la red.

- Importancia: Fundamental para la comunicación con servidores.

- Superclases: Object

- Subclases: None

- Clases asociadas: Promise, Request, Response, Headers

42. Blob

- Concepto: Representa un objeto de datos binarios.

- Funcionalidad: Permite la manipulación de datos binarios.

- Importancia: Crucial para trabajar con archivos.

- Superclases: Object

- Subclases: File

- Clases asociadas: FileReader, URL

43. File

- Concepto: Representa un archivo.

- Funcionalidad: Permite la manipulación de archivos en el cliente.

- Importancia: Esencial para la manipulación de archivos.

- Superclases: Blob

- Subclases: None

- Clases asociadas: FileReader, FormData

44. FileReader

- Concepto: Permite leer el contenido de archivos.

- Funcionalidad: Proporciona métodos para leer archivos de manera asíncrona.

- Importancia: Crucial para la lectura de archivos en el cliente.

- Superclases: EventTarget

- Subclases: None

- Clases asociadas: Blob, File

45. Performance

- Concepto: Proporciona acceso a la API de rendimiento.

- Funcionalidad: Permite medir el rendimiento de las aplicaciones web.

- Importancia: Esencial para optimización y depuración.

- Superclases: Object

- Subclases: None

- Clases asociadas: PerformanceEntry, PerformanceObserver

46. PerformanceEntry

- Concepto: Representa una entrada de rendimiento.

- Funcionalidad: Proporciona datos sobre una métrica de rendimiento específica.

- Importancia: Crucial para analizar el rendimiento.

- Superclases: Object

- Subclases: PerformanceMark, PerformanceMeasure

- Clases asociadas: Performance, PerformanceObserver

47. PerformanceObserver

- Concepto: Observa eventos de rendimiento y ejecuta un callback cuando ocurren.

- Funcionalidad: Permite la observación de métricas de rendimiento en tiempo real.

- Importancia: Esencial para el monitoreo continuo del rendimiento.

- Superclases: Object

- Subclases: None

- Clases asociadas: Performance, PerformanceEntry

48. Crypto

- Concepto: Proporciona acceso a criptografía avanzada.

- Funcionalidad: Permite realizar operaciones criptográficas seguras.

- Importancia: Crucial para la seguridad en la web.

- Superclases: Object

- Subclases: None

- Clases asociadas: SubtleCrypto

49. SubtleCrypto

- Concepto: Proporciona métodos criptográficos de alto nivel.

- Funcionalidad: Permite realizar operaciones criptográficas complejas.

- Importancia: Esencial para la implementación de seguridad.

- Superclases: Object

- Subclases: None

- Clases asociadas: Crypto

50. Geolocation

- Concepto: Proporciona acceso a la ubicación geográfica del dispositivo.

- Funcionalidad: Permite obtener la latitud y longitud del usuario.

- Importancia: Crucial para aplicaciones basadas en ubicación.

- Superclases: Object

- Subclases: None

- Clases asociadas: Position, PositionError

***API - HTMLElement para Web Components***

**ATRIBUTOS DEL API – HTMLElement para Web Components :** Los Web Components son elementos personalizados que pueden tener sus propios atributos y estándar de HTMLElement

1. id: Define un identificador único para el elemento.

2. class: Especifica una o más clases para el elemento (utilizado para estilos CSS).

3. style: Define estilos en línea para el elemento.

4. data: Permite almacenar datos personalizados en el elemento.

5. aria: Proporciona información accesible sobre el elemento para tecnologías de asistencia.

6. role: Define el papel semántico del elemento para tecnologías de asistencia.

7. tabindex: Especifica el orden de tabulación del elemento.

8. title: Define un texto de información sobre herramientas para el elemento.

9. hidden: Oculta el elemento.

10. contenteditable: Indica si el contenido del elemento es editable.

11. draggable: Define si el elemento es arrastrable.

12. autofocus: Indica si el elemento debe obtener automáticamente el foco al cargar la página.

13. disabled: Indica si el elemento está deshabilitado.

14. required: Indica si el elemento es obligatorio en un formulario.

15. readonly: Indica si el elemento es de solo lectura.

16. placeholder: Define un texto de marcador de posición para el elemento.

17. checked: Indica si el elemento está marcado (para elementos de entrada de tipo checkbox o radio).

18. value: Define el valor del elemento (para elementos de entrada).

19. src: Especifica la URL de origen del recurso.

20. href: Especifica la URL de destino del hipervínculo.

21. alt: Define texto alternativo para elementos de imagen.

22. title: Define el texto de información sobre herramientas para elementos de imagen.

23. srcset: Especifica una lista de fuentes para imágenes.

24. download: Indica que el recurso debe descargarse cuando se hace clic en el enlace.

25. rel: Define la relación entre el documento actual y el recurso enlazado.

26. target: Especifica dónde abrir el enlace vinculado.

27. type: Especifica el tipo de elemento de entrada.

28. min, max: Define los valores mínimo y máximo para elementos de entrada numérica.

29. step: Define el incremento o decremento para elementos de entrada numérica.

30. cols, rows: Especifica el número de columnas y filas para elementos `<textarea>`.

**METODOS MAS COMUNES DEL API HTMLElement para web components**

1. addEventListener(event, handler): Registra un controlador de eventos en el elemento.

2. removeEventListener(event, handler): Elimina un controlador de eventos del elemento.

3. querySelector(selector): Devuelve el primer elemento que coincide con un selector CSS dentro del elemento.

4. querySelectorAll(selector): Devuelve una colección de todos los elementos que coinciden con un selector CSS dentro del elemento.

5. appendChild(child): Agrega un nuevo hijo al elemento.

6. removeChild(child): Elimina un hijo del elemento.

7. setAttribute(name, value): Establece el valor de un atributo del elemento.

8. getAttribute(name): Obtiene el valor de un atributo del elemento.

9. removeAttribute(name): Elimina un atributo del elemento.

10. focus(): Da foco al elemento.

11. blur(): Hace que el elemento pierda el foco.

12. scrollIntoView(): Desplaza el contenido del elemento al área visible del contenedor más cercano.

13. hasAttribute(name): Devuelve un valor booleano que indica si el elemento tiene un atributo específico.

14. classList.add(className): Agrega una clase al elemento.

15. classList.remove(className): Elimina una clase del elemento.

16. classList.toggle(className): Alterna la presencia de una clase en el elemento.

17. classList.contains(className): Devuelve un valor booleano que indica si el elemento tiene una clase específica.

18. matches(selector): Comprueba si el elemento coincide con un selector CSS dado.

19. closest(selector): Devuelve el ancestro más cercano del elemento que coincide con el selector CSS especificado.

20. insertAdjacentHTML(position, text): Inserta HTML adyacente al elemento en una posición específica.

21. scrollTo(x, y): Desplaza el contenido del elemento a las coordenadas especificadas.

22. scrollBy(x, y): Desplaza el contenido del elemento por una cantidad específica.

23. scrollTo(options): Desplaza el contenido del elemento utilizando opciones de desplazamiento.

24. scrollIntoView(options): Desplaza el contenido del elemento al área visible del contenedor más cercano utilizando opciones de desplazamiento.

25. requestFullscreen(): Pide al navegador que entre en modo de pantalla completa con el elemento.

26. webkitRequestFullscreen(): Pide al navegador que entre en modo de pantalla completa con el elemento (para navegadores basados en WebKit).

27. getAttributeNames(): Devuelve una matriz de todos los nombres de atributos del elemento.

28. insertAdjacentElement(position, element): Inserta un elemento adyacente al elemento en una posición específica.

29. insertAdjacentText(position, text): Inserta texto adyacente al elemento en una posición específica.

30. insertBefore(newNode, referenceNode): Inserta un nuevo nodo antes de un nodo de referencia en la lista de hijos del elemento.

**PRINCIPALES PROPIEDADES DEL API HTMLElement para desarrollo de web components**

1. id: Retorna o establece el ID del elemento, permitiendo la identificación única en el DOM.

2. classList: Proporciona acceso a las clases CSS del elemento como un objeto `DOMTokenList`, permitiendo la manipulación dinámica de clases.

3. style: Retorna un objeto `CSSStyleDeclaration` que representa los estilos en línea aplicados al elemento, permitiendo la manipulación directa de los estilos CSS.

4. dataset: Retorna un objeto `DOMStringMap` que contiene todos los atributos de datos (`data-\*`) del elemento, facilitando el almacenamiento de datos personalizados.

5. aria-\*: Proporciona acceso a los atributos ARIA (Accessible Rich Internet Applications) del elemento, permitiendo mejorar la accesibilidad.

6. role: Retorna o establece el papel semántico del elemento para tecnologías de asistencia, mejorando la accesibilidad.

7. innerHTML: Retorna o establece el contenido HTML del elemento como una cadena, permitiendo la manipulación del contenido interno del elemento.

8. outerHTML: Retorna el contenido HTML del elemento, incluyendo el propio elemento.

9. textContent: Retorna o establece el texto sin formato del contenido del elemento y sus descendientes, permitiendo la manipulación del contenido textual.

10. offsetWidth: Retorna el ancho del elemento, incluyendo el padding, el borde y la barra de desplazamiento vertical si está presente.

11. offsetHeight: Retorna el alto del elemento, incluyendo el padding, el borde y la barra de desplazamiento horizontal si está presente.

12. offsetParent: Retorna el elemento primario desplazado más cercano del elemento.

13. parentElement: Retorna el nodo padre del elemento.

14. children: Retorna una colección de los elementos secundarios del elemento, facilitando la manipulación de los hijos.

15. firstElementChild: Retorna el primer hijo del elemento que es un elemento, omitiendo los nodos de texto y comentarios.

16. lastElementChild: Retorna el último hijo del elemento que es un elemento, omitiendo los nodos de texto y comentarios.

17. previousElementSibling: Retorna el elemento hermano anterior del elemento.

18. nextElementSibling: Retorna el elemento hermano siguiente del elemento.

19. tagName: Retorna el nombre de la etiqueta del elemento en mayúsculas.

20. scrollLeft: Retorna o establece la cantidad de desplazamiento horizontal del elemento.

21. scrollTop: Retorna o establece la cantidad de desplazamiento vertical del elemento.

22. clientWidth: Retorna el ancho interior del elemento, excluyendo el padding pero incluyendo la barra de desplazamiento vertical si está presente.

23. clientHeight: Retorna el alto interior del elemento, excluyendo el padding pero incluyendo la barra de desplazamiento horizontal si está presente.

24. offsetLeft: Retorna la distancia del borde izquierdo del elemento al borde izquierdo del `offsetParent`.

25. offsetTop: Retorna la distancia del borde superior del elemento al borde superior del `offsetParent`.

26. offsetParent: Retorna el elemento primario desplazado más cercano del elemento.

27. tagName: Retorna el nombre de la etiqueta del elemento en mayúsculas.

28. scrollWidth: Retorna el ancho total del contenido del elemento, incluyendo el contenido que no es visible debido a desbordamiento.

29. scrollHeight: Retorna el alto total del contenido del elemento, incluyendo el contenido que no es visible debido a desbordamiento.

30. draggable: Retorna o establece si el elemento es arrastrable, permitiendo arrastrar y soltar funcionalidades.

PRINCIPALES EVENTOS ASOCIADOS AL API HTMLElement para desarrollo web components

1. click: Se activa cuando se hace clic en el elemento.

2. dblclick: Se activa cuando se hace doble clic en el elemento.

3. mousedown: Se activa cuando se presiona un botón del mouse sobre el elemento.

4. mouseup: Se activa cuando se suelta un botón del mouse sobre el elemento.

5. mouseover: Se activa cuando el puntero del mouse se mueve sobre el elemento

6. mouseout: Se activa cuando el puntero del mouse sale del área del elemento.

7. mousemove: Se activa cuando el puntero del mouse se mueve sobre el elemento.

8. mouseenter: Se activa cuando el puntero del mouse entra en el área del elemento

9. mouseleave: Se activa cuando el puntero del mouse sale del área del elemento

10. keydown: Se activa cuando una tecla se presiona mientras el elemento tiene el foco del teclado.

Estos eventos permiten una interacción dinámica con los Web Components, lo que facilita la creación de componentes interactivos y receptivos.

PRINCIPALES METODOS DE CUSTOM ELEMENT PARA WEB COMPONENTS

1. constructor(): El constructor es un método especial que se llama automáticamente cuando se crea una instancia del Custom Element. Aquí puedes realizar inicializaciones y configuraciones iniciales, como la creación de la sombra DOM, la vinculación de eventos, y la configuración de propiedades y atributos iniciales.

class MiElemento extends HTMLElement {

constructor() {

super();

// Inicializaciones aquí

}

}

```

2. connectedCallback(): Este método se llama automáticamente cuando el Custom Element se conecta al DOM principal. Es útil para realizar tareas de inicialización que necesitan acceso al DOM, como la recuperación de elementos secundarios o la configuración de observadores de mutaciones.

class MiElemento extends HTMLElement {

connectedCallback() {

// Tareas de inicialización aquí

}

}

```

3. disconnectedCallback(): Se llama automáticamente cuando el Custom Element se desconecta del DOM principal. Puedes utilizar este método para limpiar recursos, detener procesos en segundo plano o realizar otras tareas de limpieza.

class MiElemento extends HTMLElement {

disconnectedCallback() {

// Limpiar recursos aquí

}

}

```

4. attributeChangedCallback(name, oldValue, newValue): Este método se llama automáticamente cuando se cambia el valor de un atributo observado del Custom Element. Puedes definir qué atributos observar mediante el método `observedAttributes()`.

class MiElemento extends HTMLElement {

attributeChangedCallback(name, oldValue, newValue) {

// Reaccionar a cambios de atributos aquí

}

static get observedAttributes() {

return ['atributo1', 'atributo2'];

}

}

```

5. adoptedCallback(): Se llama automáticamente cuando el Custom Element se adopta en un nuevo documento. Esto puede ocurrir cuando se mueve el elemento a través del DOM, por ejemplo, con `document.adoptNode()`.

class MiElemento extends HTMLElement {

adoptedCallback() {

// Tareas adicionales al adoptar el elemento en un nuevo documento

}

}

```

6. addEventListener(type, handler): Este método está disponible en todos los elementos y se utiliza para agregar un controlador de eventos al Custom Element. Permite que el elemento reaccione a eventos específicos, como clics de ratón, cambios de teclado, etc.

miElemento.addEventListener('click', () => {

// Acciones al hacer clic en el elemento

});

```

7. removeEventListener(type, handler): Se utiliza para eliminar un controlador de eventos previamente agregado al Custom Element.

miElemento.removeEventListener('click', handlerFunction);

```

8. dispatchEvent(event): Este método dispara un evento personalizado en el Custom Element. Puedes usarlo para crear y despachar tus propios eventos personalizados que otros elementos pueden escuchar.

const miEvento = new CustomEvent('mi-evento', { detail: { mensaje: 'Hola!' } });

miElemento.dispatchEvent(miEvento);

```

9. setAttribute(name, value): Se utiliza para establecer el valor de un atributo en el Custom Element.

miElemento.setAttribute('atributo', 'valor');

```

10. removeAttribute(name): Se utiliza para eliminar un atributo del Custom Element.

miElemento.removeAttribute('atributo');

```

**PRINCIPALES PROPIEDADES DE CUSTOM ELEMENT PARA WEB COMPONENTS**

1. value: Representa el valor actual del elemento, útil para elementos interactivos como campos de entrada.

2. disabled: Indica si el elemento está deshabilitado o no.

3. innerHTML: Contiene el HTML interno del elemento, permitiendo la manipulación dinámica del contenido.

4. textContent: Contiene el texto sin formato del contenido del elemento y sus descendientes.

5. checked: Indica si el elemento está marcado o no, comúnmente utilizado en elementos de entrada de tipo checkbox o radio.

6. classList: Proporciona acceso a las clases CSS del elemento como una lista para manipulación dinámica de clases.

7. style: Permite establecer estilos en línea para el elemento mediante una interfaz de estilo CSS.

8. id: Define un identificador único para el elemento, útil para referenciarlo desde el código JavaScript.

9. dataset: Proporciona acceso a los atributos de datos (`data-\*`) del elemento como un objeto, permitiendo almacenar datos personalizados.

10. innerText: Contiene el texto visible del elemento, excluyendo cualquier contenido HTML.

PRINCIPALES ATRIBUTOS DE CUSTOM ELEMENT PARA WEB COMPONENTS

1. classList: Proporciona acceso a las clases del elemento como una lista.

2. children: Devuelve una colección de los elementos secundarios del elemento.

3. parentNode: Devuelve el nodo padre del elemento.

4. innerHTML: Obtiene o establece el HTML interno del elemento.

5. outerHTML: Obtiene o establece el HTML del elemento, incluyendo el propio elemento.

6. textContent: Obtiene o establece el texto sin formato del contenido del elemento y sus descendientes.

7. offsetWidth: Devuelve el ancho del elemento, incluido el padding y el borde, en píxeles.

8. offsetHeight: Devuelve el alto del elemento, incluido el padding y el borde, en píxeles.

9. offsetParent: Devuelve el elemento primario desplazado más cercano del elemento.

10. style: Proporciona acceso a los estilos CSS del elemento.