UT 2.0 - HTML Living Standard

CFGS Desarrollo de Aplicaciones Web
Diseño de Interfaces Web

1. Conceptos básicos

HTML Living Standard, contenidos de: https://html.spec.whatwg.org/

- **Infraestructura común:** Terminología y conceptos básicos.
- **Semántica y estructura:** Elementos y atributos de HTML.
- **Microdatos:** Cómo usar microdatos para anotar contenido.
- **Interacción del usuario:** Eventos y manejo de interacciones.
- **Carga de páginas web:** Procesos y técnicas de carga.
- **APIs de aplicaciones web:** APIs disponibles para desarrolladores.
- **Comunicación:** Métodos de comunicación entre documentos y servidores.
- **Trabajadores web:** Uso de Web Workers para tareas en segundo plano.
- **Almacenamiento web:** Técnicas de almacenamiento local y de sesión.
- **Sintaxis de HTML:** Reglas y convenciones de sintaxis.
- **Sintaxis de XML:** Uso de XML en documentos HTML.
- **Renderizado:** Cómo los navegadores renderizan HTML.
- **Características obsoletas:** Elementos y atributos en desuso.
- **Consideraciones de IANA:** Registro de tipos de medios y otros valores.

1. Conceptos básicos

HTML Living Standard https://html.spec.whatwg.org/

- **Infraestructura común:** Esta sección incluye definiciones de términos clave, como elementos, atributos y nodos, así como una explicación de cómo se estructuran y manipulan los documentos HTML. También aborda temas como el modelo de objetos del documento (DOM) y la serialización de HTML.

- **Semántica y estructura:** Aquí se detallan los elementos y atributos de HTML que se utilizan para definir la estructura y el significado del contenido de una página web. La semántica es crucial para la accesibilidad y la optimización en motores de búsqueda (SEO). Esta sección explica cómo usar correctamente elementos como encabezados, párrafos, listas, enlaces y tablas para crear documentos bien estructurados y comprensibles.

- **Microdatos:** Los microdatos son una forma de anotar el contenido de HTML con información adicional que puede ser utilizada por motores de búsqueda y otras aplicaciones. Esta sección describe cómo usar los atributos itemscope, itemtype y itemprop para agregar metadatos a los elementos HTML. Los microdatos ayudan a mejorar la visibilidad y la relevancia del contenido en los resultados de búsqueda.

Estructura de un Documento HTML

Un documento HTML se organiza en una jerarquía de elementos anidados.

Declaración del tipo de documento (DOCTYPE): Indica al navegador que se está utilizando HTML5.

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

Elemento <head>: Contiene metadatos sobre el documento, como el título, enlaces a hojas de estilo y scripts.

```
<head>
    <meta charset="UTF-8">
    <title>Mi Página Web</title>
    <link rel="stylesheet" href="styles.css">
</head>
```

Elemento <body>: La parte visible de la página, como párrafos, imágenes, enlaces, etc.

HTML

```
<body>
    <h1>Bienvenido a Mi Página Web</h1>
    <p>Este es un párrafo de ejemplo.</p>
</body>
```

Semántica y estructura

La semántica en HTML es crucial para la accesibilidad y la optimización en motores de búsqueda (SEO) mejorando su indexación y relevancia en los resultados de búsqueda. Además, los lectores de pantalla y otras tecnologías de asistencia dependen de estos elementos para proporcionar una experiencia de usuario accesible.

Elementos:

Encabezados: Los elementos <h1> a <h6>

Párrafos: El elemento <p>

Listas: Las listas ordenadas (<ol>) y no ordenadas (<ul>)

Enlaces: El elemento <a> se utiliza para crear hipervínculos

Imágenes: El elemento <img> se utiliza para insertar imágenes.

Secciones: Los elementos <section>, <article>, <nav> y <aside> se utilizan para definir secciones específicas del contenido, como artículos, navegación y contenido lateral.

Encabezado y Pie de Página: Los elementos <header> y <footer> se utilizan para definir el encabezado y el pie de página de un documento o sección.

Atributos Semánticos:

alt: Utilizado en el elemento <img> para proporcionar texto alternativo que describe la imagen.

title: Proporciona información adicional sobre un elemento.

lang: Define el idioma del contenido del elemento.

aria-*: Atributos ARIA (Accessible Rich Internet Applications) que mejoran la accesibilidad proporcionando información adicional a los lectores de pantalla.

<button aria-label="Cerrar">X</button>

Microdatos

Los microdatos son una forma de anotar el contenido HTML con información adicional que **puede ser utilizada por motores de búsqueda** y otras aplicaciones para entender mejor el contenido de una página web. Esta técnica permite agregar metadatos a los elementos HTML mediante el uso de atributos específicos:

Itemscope: Este atributo se utiliza para crear un nuevo "item" de microdatos. Al añadir itemscope a un elemento, se indica que este elemento y sus descendientes contienen un conjunto de pares nombre-valor que describen un objeto.

```
<div itemscope> <!-- Contenido del item --> </div>
```

Itemtype: Este atributo se utiliza junto con itemscope para especificar el tipo de item mediante una URL que define el vocabulario utilizado. Esto ayuda a los motores de búsqueda a entender el contexto del item.

```
<div itemscope itemtype="https://schema.org/Person"><!-- Contenido del item --></div>
```

Itemprop: Este atributo se utiliza para definir una propiedad del item. Se añade a los elementos descendientes del elemento que tiene itemscope.

```
<div itemscope itemtype="https://schema.org/Person">
  <span itemprop="name">John Doe</span>
  <img itemprop="image" src="johndoe.jpg" alt="John Doe">
</div>
```

Microdatos

Elemento Principal: El con itemscope y itemtype define un nuevo item de tipo Person según el vocabulario de Schema.org.

Propiedades: Los elementos descendientes con itemprop definen propiedades del item, como name, image, jobTitle, worksFor y email.

Item Anidado: El con itemscope y itemtype dentro de worksFor define un item anidado de tipo Organization, con su propia propiedad name.

```
<div itemscope itemtype="https://schema.org/Person">
    <h1 itemprop="name">John Doe</h1>
    <img itemprop="image" src="johndoe.jpg" alt="John Doe">
    <p>Job: <span itemprop="jobTitle">Software Engineer</span></p>
    <p>Works at: <span itemprop="worksFor" itemscope itemtype="https://schema.org/Organization">
        <span itemprop="name">Tech Company</span>
    </span></p>
    <p>Email: <a href="mailto:johndoe@example.com" itemprop="email">johndoe@example.com</a></p>
</div>
```

Beneficios de los Microdatos:

Mejora del SEO: SEO significa Search Engine Optimization (Optimización para motores de búsqueda). Se trata del conjunto de estrategias y técnicas de optimización que se hacen en una página web para que aparezca orgánicamente en buscadores de Internet como Google, Yahoo o Youtube…

Interoperabilidad: Los datos estructurados pueden ser utilizados por diferentes aplicaciones y servicios para proporcionar funcionalidades adicionales, como tarjetas enriquecidas en los resultados de búsqueda.

1. Conceptos básicos

HTML Living Standard https://html.spec.whatwg.org/

- **Interacción del usuario:** Esta parte del documento se centra en los eventos y el manejo de interacciones del usuario en una página web. Explica cómo los desarrolladores pueden usar eventos como clics, desplazamientos y entradas de teclado para crear experiencias interactivas. También aborda la gestión de eventos y la propagación de eventos, proporcionando una base sólida para el desarrollo de interfaces de usuario dinámicas.

- **Carga de páginas web:** La carga eficiente de páginas web es esencial para una buena experiencia de usuario. Esta sección cubre los procesos y técnicas para optimizar la carga de páginas, incluyendo la gestión de recursos, el uso de caché y la minimización de solicitudes HTTP. También se discuten estrategias como la carga diferida y la carga condicional para mejorar el rendimiento**.**

- **APIs de aplicaciones web:** HTML proporciona varias APIs que los desarrolladores pueden usar para crear aplicaciones web ricas y funcionales. Esta sección describe APIs como la API de Canvas para gráficos, la API de Geolocalización para obtener la ubicación del usuario y la API de Historial para gestionar el historial de navegación. Estas APIs amplían las capacidades de HTML y permiten la creación de aplicaciones web avanzadas.

Manipulación de Documentos HTML

La manipulación de documentos HTML se realiza principalmente a través del Document Object Model (DOM), que es una representación estructurada del documento.

Acceder a Elementos: Puedes acceder a los elementos del DOM utilizando métodos como getElementById, getElementsByClassName, getElementsByTagName, y querySelector.

(JavaScript) *var elemento = document.getElementById("miElemento");*

Modificar Contenido: Puedes cambiar el contenido de un elemento utilizando la propiedad innerHTML o textContent.

(JavaScript) *elemento.innerHTML = "Nuevo contenido";*

Cambiar Atributos: Puedes modificar los atributos de un elemento utilizando setAttribute o accediendo directamente a las propiedades del elemento.

(JavaScript) *elemento.setAttribute("class", "nuevaClase");*

(JavaScript) *elemento.href = "https://nueva-url.com";*

Añadir y Eliminar Elementos: Puedes crear nuevos elementos con createElement y añadirlos al DOM con appendChild o insertBefore. Para eliminar elementos, puedes usar removeChild.

Eventos: Puedes añadir manejadores de eventos a los elementos para responder a las interacciones del usuario.

(JavaScript) *elemento.addEventListener("click", function()*
{alert("Elemento clickeado!");});

Serialización de HTML

Se denomina al proceso de **convertir un documento HTML** o una parte de él en una **cadena de texto** que puede ser almacenada, transmitida o procesada**.** Este proceso es esencial para varias operaciones, como guardar el estado de una página, enviar datos a través de la red o generar contenido dinámico.

En JavaScript, la serialización de HTML se puede realizar utilizando el método:

outerHTML devuelve una cadena que representa el elemento seleccionado y todo su contenido, incluyendo las etiquetas de apertura y cierre

(JavaScript) *var htmlSerializado = elemento.outerHTML;*

innerHTML: devuelve el contenido interno del elemento

(JavaScript) *var contenidoInterno = elemento.innerHTML;*

La deserialización es el proceso inverso: convertir una cadena de texto en un documento HTML. Para deserializar, puedes usar el método innerHTML para insertar la cadena en un elemento del DOM:

(JavaScript) *nuevoElemento.innerHTML = htmlSerializado;*

Gestión de recursos

Carga Asíncrona y Diferida de Scripts: Utilizar los atributos async y defer en las etiquetas <script> para cargar scripts de manera asíncrona o diferida, permite que el navegador continúe procesando el HTML mientras se cargan los scripts.

<script src="script.js" async></script>

Optimización de Imágenes: Utilizar formatos de imagen adecuados y comprimir las imágenes para reducir su tamaño. Además, se recomienda el uso de atributos width y height para evitar el reflujo durante la carga.

<img src="imagen.jpg" width="600" height="400" alt="Descripción de la imagen">

Control de Caché: Configurar encabezados HTTP adecuados para controlar la caché del navegador, como Cache-Control y Expires. Esto permite que los recursos estáticos se almacenen en caché y se reutilicen en futuras visitas.

Cache-Control: max-age=31536000

Expires: Wed, 21 Oct 2025 07:28:00 GMT

Service Workers: Implementar Service Workers para gestionar la caché de manera más avanzada, permitiendo el almacenamiento en caché de recursos y la creación de aplicaciones web que funcionen sin conexión

Minimizar peticiones HTTP

Combinar archivos: Combinar y reducir Archivos: Combinar múltiples archivos CSS y JavaScript en uno solo y minificarlos para reducir el tamaño de los archivos y el número de solicitudes HTTP.

 # Ejemplo de uso de herramientas como UglifyJS para minificar JavaScript

uglifyjs script1.js script2.js -o script.min.js

Uso de CDNs: Utilizar Redes de Distribución de Contenidos (CDNs) para servir recursos estáticos desde ubicaciones geográficas cercanas al usuario, reduciendo la latencia y mejorando los tiempos de carga.

Carga Condicional: Cargar recursos solo cuando sean necesarios, utilizando técnicas como la carga diferida (lazy loading) para imágenes y otros recursos multimedia.

<img src="imagen.jpg" loading="lazy" alt="Descripción de la imagen">

1. Conceptos básicos

HTML Living Standard https://html.spec.whatwg.org/

- **Comunicación:** La comunicación entre documentos y servidores es fundamental para las aplicaciones web modernas. Esta sección cubre métodos como XMLHttpRequest y Fetch para realizar solicitudes HTTP, así como técnicas para la comunicación entre ventanas y marcos mediante postMessage. También se discuten los WebSockets para la comunicación en tiempo real y las técnicas de CORS para manejar solicitudes entre orígenes.

- **Trabajadores (Workers) web:** Los Web Workers permiten ejecutar scripts en segundo plano sin bloquear la interfaz de usuario. Esta sección explica cómo crear y usar Web Workers para realizar tareas intensivas en computación, como el procesamiento de datos y la manipulación de imágenes, de manera eficiente. Los Web Workers mejoran el rendimiento y la capacidad de respuesta de las aplicaciones web.

- **Almacenamiento web:** HTML proporciona varias técnicas para almacenar datos en el navegador del usuario. Esta sección cubre el almacenamiento local y de sesión, que permiten guardar datos de manera persistente o temporal. También se discuten las APIs de IndexedDB y Web SQL para el almacenamiento de datos estructurados. Estas técnicas son esenciales para aplicaciones web que necesitan almacenar grandes cantidades de datos o funcionar sin conexión.

Comunicación (Javascript)

XMLHttpRequest es una **API** que permite realizar solicitudes HTTP para intercambiar datos entre un cliente y un servidor. Es ampliamente utilizada para realizar solicitudes AJAX.

- **open(method, url, async):** Inicializa la solicitud.
- **onreadystatechange:** Define una función que se ejecuta cuando cambia el estado de la solicitud.
- **send():** Envía la solicitud al servidor.

Fetch es una API que proporciona una forma más sencilla y flexible de realizar solicitudes HTTP.

- **fetch(url, options):** Realiza una solicitud HTTP.
- **then():** Maneja la respuesta de la solicitud.
- **catch():** Maneja cualquier error que ocurra durante la solicitud.

PostMessage permite la comunicación entre ventanas y marcos, incluso si están en diferentes dominios. Es útil para enviar mensajes entre un documento principal y un iframe.

- **postMessage(message, targetOrigin):** Envía un mensaje al contexto de destino.
- **addEventListener('message', handler):** Escucha los mensajes entrantes.

Comunicación (Javascript)

WebSockets permiten la comunicación en tiempo real entre un cliente y un servidor a través de una conexión persistente.

- **new WebSocket(url):** Crea una nueva conexión WebSocket.
- **onopen**: Evento que se dispara cuando se abre la conexión.
- **onmessage:** Evento que se dispara cuando se recibe un mensaje.
- **onclose:** Evento que se dispara cuando se cierra la conexión.

CORS (Cross-Origin Resource Sharing) es una técnica que permite que los recursos de una página web se soliciten desde otro dominio distinto al que sirvió la página. Esto se logra mediante el uso de encabezados HTTP específicos.

Access-Control-Allow-Origin: https://otro-dominio.com

Access-Control-Allow-Methods: GET, POST, PUT, DELETE

Access-Control-Allow-Headers: Content-Type

Web Workers

Son una herramienta poderosa para mejorar el rendimiento de las aplicaciones web al permitir la ejecución de tareas intensivas en segundo plano

Permiten **ejecutar scripts en segundo plano**, sin bloquear la interfaz de usuario. Esto es especialmente útil para tareas intensivas en computación, como el procesamiento de datos y la manipulación de imágenes.

Primero, crea un archivo JavaScript que contenga el código del worker. Este archivo se ejecutará en un hilo separado.

Después, en el archivo principal de tu aplicación, crea una instancia del worker y define cómo manejar los mensajes entrantes y salientes.

Beneficios:

No Bloquean la UI: Los Web Workers permiten que la interfaz de usuario permanezca receptiva mientras se realizan tareas intensivas en segundo plano.

Comunicación Asíncrona: Utilizan postMessage para enviar y recibir datos, lo que facilita la comunicación entre el hilo principal y el worker.

Aislamiento: Los Web Workers tienen su propio contexto global, lo que significa que no pueden acceder directamente al DOM del documento principal, mejorando la seguridad y la estabilidad.

Nota: No pueden acceder directamente al DOM ni a ciertas APIs del navegador, pero pueden utilizar APIs como XMLHttpRequest y Fetch.

Almacenamiento web

Técnicas y APIs proporcionan diversas opciones para el almacenamiento de datos en aplicaciones web

Almacenamiento Local (localStorage)

Persistencia: Los datos almacenados en localStorage persisten incluso después de cerrar el navegador. Esto es útil para guardar información que debe estar disponible en futuras visitas.

Alcance: Los datos son accesibles desde todas las pestañas y ventanas que cargan la misma página web.

Capacidad: Generalmente, localStorage permite almacenar hasta 5MB de datos por origen (dominio).

(Javascriopt) localStorage.setItem('nombre', 'Juan');

Almacenamiento de Sesión (sessionStorage)

Persistencia: Los datos almacenados en sessionStorage solo persisten durante la sesión de la página. Se eliminan cuando se cierra la pestaña o ventana del navegador.

Alcance: Los datos son accesibles solo desde la pestaña o ventana en la que se almacenaron.

Capacidad: Similar a localStorage, generalmente permite hasta 5MB de datos por origen.

(Javascriopt) sessionStorage.setItem('nombre', 'Juan');

Almacenamiento web

Técnicas y APIs proporcionan diversas opciones para el almacenamiento de datos en aplicaciones web

IndexedDB es una API para almacenar grandes cantidades de datos estructurados. Es una base de datos NoSQL que permite realizar consultas complejas y transacciones

Persistencia: Los datos persisten incluso después de cerrar el navegador.

Capacidad: Permite almacenar grandes cantidades de datos (varios megabytes o más).

Transacciones: Soporta transacciones para asegurar la integridad de los datos.

Consultas: Permite realizar consultas complejas utilizando índices.

(Javascriopt) var request = indexedDB.open('miBaseDeDatos', 1);

Web SQL es una API que permite almacenar datos en una base de datos SQL en el navegador. Aunque está en desuso y no es recomendada para nuevos proyectos, aún es soportada por algunos navegadores**.**

Persistencia: Los datos persisten incluso después de cerrar el navegador.

Capacidad: Permite almacenar grandes cantidades de datos.

Consultas SQL: Utiliza consultas SQL para interactuar con la base de datos.

*(Javascriopt) var db = openDatabase('miBaseDeDatos', '1.0', 'Base de datos de ejemplo', 2 * 1024 * 1024);*

Almacenamiento web

No confundir el almacenamiento de datos en el lado cliente y en el lado servidor:

Lado Cliente: (IndexedDB)

Se usa en aplicaciones web que necesitan almacenamiento local, aplicaciones web progresivas (PWA), almacenamiento de datos en el cliente. Almacena pares clave-valor.

Lado Servidor. (MongoDB, Firebase)

Son BBDD NoSQL es la abreviatura de **"Not only SQL"** y se refiere a una categoría de DBMS que no utilizan SQL como lenguaje de consulta principal. Organizan los datos en documentos, normalmente de formato JSON. Se usan en Aplicaciones que requieren flexibilidad en el esquema y grandes volúmenes de datos.

1. Conceptos básicos

HTML Living Standard https://html.spec.whatwg.org/

- **Sintaxis de HTML:** La sintaxis de HTML define las reglas y convenciones para escribir documentos HTML válidos. Esta sección describe la estructura básica de un documento HTML, incluyendo el uso de etiquetas de apertura y cierre, atributos y entidades de caracteres. También se abordan las diferencias entre HTML y XHTML, y se proporcionan ejemplos de buenas prácticas para escribir código HTML limpio y mantenible.

- **Sintaxis de XML:** Aunque HTML5 no requiere el uso de XML, es posible escribir documentos HTML en una sintaxis compatible con XML. Esta sección explica cómo usar la sintaxis de XML en documentos HTML, incluyendo el uso de declaraciones XML, espacios de nombres y entidades de caracteres. La compatibilidad con XML permite una mayor interoperabilidad con otras tecnologías basadas en XML.

- **Renderizado:** El renderizado es el proceso mediante el cual los navegadores interpretan y muestran el contenido de un documento HTML. Esta sección describe cómo los navegadores construyen el árbol de renderizado a partir del DOM y aplican estilos CSS para presentar el contenido visualmente. También se discuten temas como el flujo de diseño, el reflujo y el repintado, que afectan el rendimiento del renderizado.

Sintaxis de HTML:

Existen herramientas que ayudan a identificar y corregir errores en el código HTML, asegurando que cumpla con los estándares web y sea accesible.

W3C Markup Validation Service: Esta es la herramienta oficial del World Wide Web Consortium (W3C) para validar documentos HTML, XHTML, SMIL, MathML...

Nu HTML Checker (v.Nu): Otra herramienta del W3C que se enfoca en la validación de HTML5 y otros formatos modernos.

Total Validator: Ofrece validación HTML junto con comprobaciones de accesibilidad y ortografía.

HTML Validator de Google Chrome DevTools: Integrado en las herramientas de desarrollo de Google Chrome, permite validar HTML directamente desde el navegador.

Tidy HTML: Una herramienta que no solo valida HTML, sino que también puede corregir y limpiar el código.

Sintaxis de HTML y XHTML:

HTML es más flexible y tolerante con los errores, mientras que XHTML sigue una sintaxis más estricta y precisa, similar a XML, no tolera errores de sintaxis. XHTML se utiliza cuando se busca un código más robusto, limpio y compatible con diferentes tecnologías y dispositivos.

XHTML es ideal cuando necesitas que tu documento web sea compatible con otras aplicaciones basadas en XML. Esto facilita la interoperabilidad entre diferentes sistemas y tecnologías

Un código XHTML bien estructurado puede ayudar a mejorar la indexación en los motores de búsqueda, lo que puede ser beneficioso para SEO (SEO es la sigla para Search Engine Optimization, que significa "optimización para motores de búsqueda")

- <!DOCTYPE html> para XHTML.
- El elemento <html> incluye el atributo xmlns para definir el espacio de nombres XML.
- Todos los elementos, incluidos los vacíos, deben cerrarse correctamente (
, <img />).
- Todos los atributos deben estar entre comillas dobles.

Renderizado:

Los navegadores siguen un proceso específico para construir el árbol de renderizado a partir del DOM y aplicar estilos CSS para presentar el contenido visualmente.

1. Construcción del DOM (Document Object Model)

- Parsing del HTML: El navegador analiza el código HTML y crea el DOM, una estructura en forma de árbol que representa el contenido del documento1.
- Nodos del DOM: Cada elemento HTML se convierte en un nodo del DOM. Por ejemplo, una etiqueta se convierte en un nodo HTMLDivElement.

DOM Tree:

- html
  - head
    - meta
    - title
    - style
  - body
    - div.container
    - div.item (“Item 1”)
    - div.item (“Item 2”)

Renderizado:

Los navegadores siguen un proceso específico para construir el árbol de renderizado a partir del DOM y aplicar estilos CSS para presentar el contenido visualmente.

2. Construcción del CSSOM (CSS Object Model)

- Parsing del CSS: El navegador analiza las hojas de estilo CSS y crea el CSSOM, otra estructura en forma de árbol que representa las reglas de estilo aplicables2.
- Reglas de Estilo: Cada regla CSS se convierte en un nodo del CSSOM, que contiene información sobre los selectores y las propiedades de estilo.

CSSOM Tree:

- body { font-family: Arial, sans-serif; }
- .container { display: flex; }
- .item { margin: 10px; padding: 20px; border: 1px solid #000; }

Renderizado:

4. Cálculo del Diseño (Layout)

- **Posición y Tamaño:** El navegador calcula la posición y el tamaño exactos de cada nodo visible en el árbol de renderizado. Este proceso se conoce como “layout” o "reflow"2.
- **Modelo de Caja:** Utiliza el modelo de caja CSS para determinar cómo se distribuyen los elementos en la página.

5. Pintura (Painting)

- **Renderizado de Píxeles:** Finalmente, el navegador toma el árbol de renderizado y pinta los píxeles en la pantalla. Este proceso incluye dibujar bordes, colores de fondo, texto, sombras, etc2.

Render Tree:

- RenderBlock (body)
  - RenderBlock (div.container)
  - RenderBlock (div.item) (“Item 1”)
  - RenderBlock (div.item) (“Item 2”)

1. Conceptos básicos

HTML Living Standard https://html.spec.whatwg.org/

- **Características obsoletas:** Con el tiempo, algunos elementos y atributos de HTML se vuelven obsoletos y ya no se recomiendan para su uso. Esta sección lista las características obsoletas y proporciona alternativas modernas. El uso de características actualizadas mejora la compatibilidad y la accesibilidad de los documentos HTML, asegurando que sigan siendo relevantes y funcionales en el futuro.

- **Consideraciones de IANA:** La Internet Assigned Numbers Authority (IANA) gestiona varios registros relacionados con HTML, como los tipos de medios y los valores de atributos. Esta sección describe las consideraciones y procedimientos para registrar nuevos tipos de medios y otros valores en los registros de IANA. La coordinación con IANA asegura que los estándares de HTML se mantengan consistentes y ampliamente reconocidos.