TEMA 77. LENGUAJES Y HERRAMIENTAS PARA LA UTILIZACIÓN DE REDES GLOBALES. HTML, CSS Y XML. NAVEGADORES WEB Y COMPATIBILIDAD DE ESTÁNDARES

Actualizado a 13/04/2023

1. INTRODUCCIÓN

A principios de los años llega Internet y HTML suponen una revolución tecnológica. En ese entonces, la mayoría de los documentos digitales eran simples archivos de texto plano que no permitían enlaces o formatos especiales. HTML permitió la creación de páginas web interactivas, en las que se podían incluir enlaces, imágenes y otros elementos multimedia. Esto abrió una nueva era en la comunicación digital y marcó el inicio de lo que hoy conocemos como Internet. Desde entonces, HTML ha evolucionado y se ha vuelto más sofisticado, permitiendo la creación de sitios web más complejos y dinámicos.

2. HTTP Y URIS

HTTP (HyperText Transfer Protocol): protocolo de la capa de aplicación que permite la transferencia de información en la World Wide Web. Estandarizado por el IETF. La versión más común es **HTTP/2** (RFC 7540), que pasa a ser un protocolo binario en lugar de texto, ofreciendo mayor rendimiento y seguridad. Características: compresión de cabeceras, posibilidad de usar una única conexión para múltiples solicitudes y respuestas o priorización de flujos.

Su sucesor, **HTTP/3** (RFC 9114), ya se encuentra disponible y es soportado por los navegadores actuales y ya ha sido implementado en un gran número de sitios de internet. HTTP/3 está basado en QUIC, un protocolo de red de la capa de transporte desarrollado inicialmente por Google, de tal forma que se sustituye TCP por UDP, permitiendo conexiones más rápidas. Por defecto realiza cifrado bajo TLS 1.3.

HTTP es un **protocolo sin estado**, es decir, no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto se usan diferentes mecanismos, por ejemplo, las cookies, que son información que un servidor puede almacenar en el sistema cliente. Esto permite a las aplicaciones web instituir la noción de sesión, y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

URL (Uniform Resource Locator): es una cadena de caracteres que identifica los recursos de una red de forma unívoca, siguiendo un formato estándar:

```
<scheme>://<host>:<port>/<path>;<parameters>?<query>#<fragment>
```

3. HTML

HTML (HyperText Markup Language): lenguaje de marcado que permite describir el contenido y el formato de páginas web. Creado por Tim Berners-Lee y estandarizado por el W3C. La primera versión de HTML, conocida como HTML 1.0, fue lanzada en 1991. Fue una versión muy simple que solo permitía la creación de páginas web básicas con texto e imágenes. Las versiones posteriores de HTML, como HTML 2.0 y HTML 3.2, se centraron en mejorar la compatibilidad entre diferentes navegadores y añadir nuevas características como tablas y formularios. HTML 4.01, lanzado en 1999, fue la última versión antes de la llegada de HTML5.

La última versión es **HTML5**. Mejoras: permite diseño adaptativo, simplifica el código permitiendo páginas más ligeras y rápidas, mejora el soporte a contenido multimedia, añade etiquetas para manejar la web semántica y refuerza la separación de presentación y contenido, introduce nuevas APIs (Geolocation, Storage, Drag&Drop ...). Las etiquetas HTML no son case sensitive y existe un set fijo de etiquetas comunes, aunque es posible definir etiquetas personalizadas combinando JavaScript y WebComponents.

Ampliar información en los documentos de HTML de la capeta "Doc Adicional"

3.1. Estructura de HTML

La estructura básica de un fichero HTML5 es la siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Título</title>
</head>
<body>
    <!-- Comentario -->
    ...
</body>
</html>
```

DOCTYPE (indica qué tipo de documento debe abrir el navegador)	• XHTML 1.1: `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">` • HTML 4.01: `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">` • HTML5: `<!DOCTYPE html>`
HTML	Raíz de HTML que incluye `<head>` y `<body>`. Es recomendable el atributo `lang`
HEAD	Cabecera de la página donde se especifican metadatos, recursos y otra información de la página. Los elementos más comunes son: • `<title>`: Título de la página • `<meta>`: Metadatos de la web • `<link>`: Enlaces a hojas de estilos, fuentes e iconos • `<style>`: Estilos incrustados CSS • `<script>`: Añadir ficheros JavaScript o código incrustado
META	Metadatos de la web, utilizados para SEO e indexación, información adicional e información de renderizado y accesibilidad. Los meta elementos más comunes son • `<meta charset="…">`: Codificación de caracteres del documento HTML • `<meta name="description" content="">`: Descripción de la página • `<meta name="author" content="">`: Autor de la página • `<meta name="viewport" content="">`: Definición de viewport, utilizado para diseño responsivo en móviles y pantallas grandes • `<meta name="robots" content="">`: Tipo de indexado de los crawlers
BODY	se utiliza en HTML para definir la sección del documento que contiene el contenido visible de la página
COMENTARIO	Los comentarios se escriben con la sintaxis `<!-- Comentario -->`

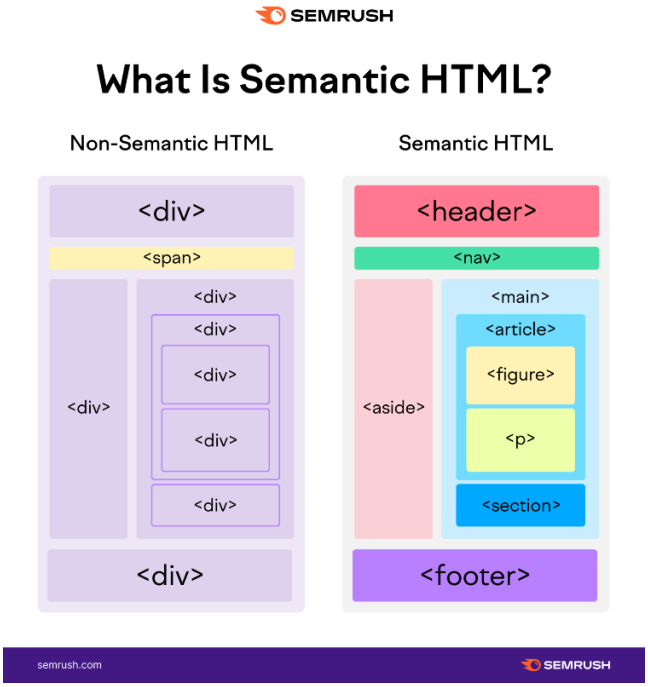
3.2. ETIQUETAS GENERALES

HTML dispone de un gran número de etiquetas para representar el contenido de la web:

Tipo de texto	• : **Texto importante, renderizado en negrita.** • : *Texto enfatizado, renderizado en cursiva.* • <del>: Texto tachado o ~~negado~~ • <ins>: Texto subrayado • <mark>: Texto resaltado. • <small>: Texto pequeño
Contenido	• <p>: Párrafo. • <h1>, <h2> … <h6>: Encabezados. • : Salto de línea. • , : Bloques de contenido. • <iframe>: Web o documento HTML incrustado.
Formato tabular	• <table>…</table>: Elemento raíz de tabla. • <thead>, <tbody>, <tfoot>: Agrupación de contenido de tabla. • <tr>…</tr>: Fila de la tabla. • <th>: Celda de la cabecera. • <td>: Celda de la tabla.
Listas	• … : Lista desordenada. • … : Lista ordenada. • : Celda de la tabla.
Imagenes	• <img src="foto.jpg">: Imagen
Enlaces	• Enlace
Formularios	• <form>…</form>: Elemento raíz de formulario. • <label>: Etiqueta de formulario • <input type="text">: Campo de texto • <input type="number">: Campo numérico • <input type="search">: Campo de búsqueda (**Nuevo**) • <input type="password">: Campo de texto con asteriscos (contraseña) • <input type="date">: Selector nativo de búsqueda de fecha • <input type="radio">: Selector con radio botones • <button>: Botón • <button type="submit">: Botón de envío de formulario
Elementos Obsoletos en HTML5	• Reemplazados por CSS: o <font>, , <big>, <font>, <blink> • Reemplazados por <iframe> o <object>: o <applet>, <frame>, <frameset>, <embed>, <noframes> • Reemplazados por referenciar directamente estilos y no contenido. Siempre que se refiera a estilo debe ser CSS. o <strike>, <tt> • Elementos no recomendados referidos a tipos de texto, en la primera fila se muestra la alternativa o <b>(bold), <i>(italics), <u>(underlined)

3.3. ETIQUETAS HTML5

HTML5 busca una separación más clara de contenido y presentación de la información (tarea de CSS), así como aprovechar al máximo las capacidades de los dispositivos modernos y JavaScript. Por lo que **añade dos novedades importantes, la primera es la creación de elementos semánticos para segmentar de una forma más clara la información, y por otra la creación de elementos dinámicos y multimedia**.



https://www.semrush.com/blog/semantic-html5-guide/

Algunos de los nuevos elementos de HTML5 son:

Elementos Semánticos o Secciones	`<header>`	Define la cabecera de una página o sección
	`<nav>`	Define una sección que solo contiene enlaces de navegación
	`<main>`	Define la sección principal del documento
	`<aside>`	Define la contenido adicional o secundario como anuncios o notas
	`<article>`	Define un contenido independiente y autónomo. Debe ser completo por sí solo.
	`<figure>`	Se utiliza para agregar contenido relacionado con una imagen o gráfica.
	`<section>`	Define las partes dentro de un contenido
	`<footer>`	Define el pie de página
Elementos Dinámicos y Multimedia	`<video>`	Representa un video y sus ficheros asociados
	`<audio>`	Representa un sonido o stream de audio
	`<canvas>`	Representa un área de mapa de bits
	`<svg>`	Representa una imagen vectorial
	`<math>`	Define una fórmula matemática
Fecha	`<time>`	Representa un valor de fecha y hora

3.4. Atributos HTML

HTML está basado en XML y por tanto una de las piezas fundamentales son los atributos de los elementos. Estos atributos añaden información acerca de los elementos, eventos o metainformación como por ejemplo de accesibilidad (ARIA - Accessible Rich Internet Applications). Algunos de los atributos más utilizados son:

Atributos básicos	• `class`: Clases de CSS (separadas por espacios). • `id`: Identificador único, utilizado en CSS y JavaScript • `src`: Usado en elementos multimedia para especificar la fuente • `value`: Valor de un campo de entrada • `title`: Información adicional de un elemento
Accesibilidad	• `role`: Papel semántico de un elemento • `alt`: Texto alternativo en imágenes • `tabindex`: Controlar el orden en el que se muestran al pulsar TAB • `aria-hidden`: Contenido oculto para lectores de pantalla • `aria-label`: Texto alternativo para lectores de pantalla • `aria-describedby`: Indica el elemento de descripción
Eventos	• `onclick`: Se ejecuta el código cuando hace clic en un elemento • `onkeydown`: Se ejecuta el código cuando el usuario pulsa una tecla • `onsubmit`: Se ejecuta el código cuando se envía un formulario

3.5. XHTML

XHTML (eXtensible Hypertext Markup Language): es una versión más estricta de HTML desarrollada para hacer HTML más extensible y aumentar la interoperabilidad con otros formatos. Los documentos XHTML están bien formados, lo que permite que sean analizados por parsers XML estándar. Es un estándar de W3C (abandonado en 2009). Última versión: XHTML 1.1.

Las principales diferencias entre XHTML y HTML son:

- Los elementos vacíos deben cerrarse siempre.
- Los valores de los atributos deben siempre ir encerrados entre comillas (simples o dobles).
- Los nombres de elementos y atributos deben ir en minúsculas
- Se pueden incorporar elementos de distintos espacios de nombres XML (como MathML y Scalable Vector Graphics).
- Un navegador no necesita implementar heurísticas para detectar qué quiso poner el autor, por lo que el parser puede ser mucho más sencillo.

4. CSS

CSS (Cascading Style Sheets): lenguaje que describe cómo deben mostrarse/presentarse los elementos de un documento HTML o XML. Permite separar el contenido de la presentación. El modelo en cascada funciona en capas, donde cada capa de estilo afecta a los elementos que están debajo de ella. Además, existe una jerarquía en la cual los estilos más específicos tienen preferencia sobre los generales. Es un estándar W3C. La última especificación es CSS3.

CSS3: Se estructura en módulos. Algunas mejoras introducidas en la versión son:

- Mejora en el tratamiento de bordes, permitiendo bordes redondeados.
- Posibilidad de incluir múltiples imágenes de fondo en una página web.
- Permite incluir cualquier tipografía en una página web mediante la regla @font-face.
- Nuevos atributos para que el navegador se encargue de maquetar texto multicolumna.
- Posibilidad de crear animaciones de los elementos de la pantalla.

4.1. USO DE CSS, PROPIEDADES CSS Y LAYOUTS

CSS es un lenguaje muy versátil para aplicar estilos. Cada instrucción CSS cuenta de 2 partes:

- **Selector**: La primera parte es la expresión en la que definimos a que elemento o a que jerarquía de elementos queremos que se apliquen los estilos. Los estilos se aplican en jerarquía descendente y existen selectores complejos para apuntar a propiedades o relaciones directas/indirectas.
- **Estilos**: Cada instrucción de estilo es de clave-valor separado por ":" aunque existen propiedades combinadas que se pueden dar varios estilos a la vez (Como `margin` que admite una lista de valores en función del margen al que queremos apuntar, si es un único valor es a todos el mismo, si no puede darse un valor especifico por margen). Las instrucciones se separan con ";".

CSS Inline - Aplica un estilo a un único elemento

```
<h1
style="color:red;margin:0">
Este texto va en rojo y no
tiene márgenes.
</h1>
```

CSS Interno – Define un estilo a un HTML dentro de <style>

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p {color: red;}
  </style>
</head>
<body>
  <p>Párrafo rojo</p>
</body>
</html>
```

CSS Externo – Referencia externa a un fichero CSS reutilizable.

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet"
href="estilos.css">
</head>
<body>
  <p>Párrafo rojo</p>
</body>
</html>
```

Las propiedades CSS se pueden utilizar para cambiar márgenes(`margin`), espaciado (`padding`), colores (`color`), fondos(`background`), fuentes de texto(`font-family`) y modificaciones como el tamaño o decoraciones, así como elementos más complejos de CCS3 como animaciones o selectores que pueden hacer cálculos o revisar propiedades internas.

CSS se utiliza también para determinar la posición de los elementos en el documento y como se ajustan al espacio disponible. Estos estilos se aplican con la propiedad `display` que puede tomar diversos valores. Los valores clásicos son `inline` y `block`. Así como los más novedosos que son Flexbox (`flex`) que permite controlar de forma fina el espacio entre elementos y como se comportan al redimensionar una página y Gridbox (`grid`) que define la página como una tabla virtual flexible y personalizable, este último es el más novedoso y no esta soportado en algunos navegadores como Internet Explorer.

4.2. Preprocesadores CSS

El preprocesador de CSS toma el código fuente escrito en la sintaxis del preprocesador y lo procesa, resolviendo las variables, funciones, mixins, anidamiento y cualquier otra característica de la sintaxis utilizada. **Luego, el preprocesador de CSS compila todo el código fuente en un archivo CSS plano**.

El archivo CSS resultante puede ser utilizado en cualquier sitio web, de la misma manera que un archivo CSS plano escrito manualmente. La ventaja de utilizar un preprocesador de CSS es que se puede escribir un código más eficiente y fácil de mantener, gracias a la utilización de variables, funciones y otras características avanzadas.

Los preprocesadores de CSS más utilizados son:

- Sass (Syntactically Awesome Style Sheets) (El más utilizado en la industria)
- Less (Leaner Style Sheets)
- Stylus
- PostCSS

4.3. Diseño responsivo, adaptativo y Media Queries

Tanto el diseño adaptativo como el diseño responsivo tienen como objetivo hacer que los sitios web se vean bien en cualquier dispositivo, pero existen algunas diferencias clave entre ambos:

- En el diseño responsivo, se utilizan patrones de diseño fluido que permiten que los elementos de la página se ajusten y cambien de tamaño según el tamaño de la pantalla del dispositivo. En cambio, en el diseño adaptativo, se utilizan diseños preestablecidos para diferentes tamaños de pantalla.
- El diseño responsivo es más flexible y puede adaptarse a una amplia variedad de dispositivos y tamaños de pantalla, mientras que el diseño adaptativo se enfoca en proporcionar una experiencia específica para dispositivos específicos.
- **El diseño responsivo se basa en una estructura de cuadrícula flexible que se adapta al ancho de la pantalla, mientras que el diseño adaptativo se basa en diferentes diseños preestablecidos que se activan según el ancho de la pantalla**.

En resumen, el diseño responsivo es más flexible y adaptable a cualquier dispositivo, mientras que el diseño adaptativo se enfoca en proporcionar experiencias específicas para dispositivos específicos. Para ello CSS nos ofrece de una herramienta muy potente: las media queries.

Las media queries son una función de CSS que permite aplicar estilos diferentes según las características de un dispositivo, como el tamaño de la pantalla, la orientación o la resolución. Con las media queries, se pueden definir reglas de estilo para dispositivos específicos o para rangos de tamaños de pantalla. Por ejemplo, se puede establecer una media query para dispositivos con un ancho de pantalla menor a 768 píxeles, y definir un conjunto de estilos específicos para ese rango de tamaño de pantalla. De esta forma, se pueden crear diseños y estilos diferentes para diferentes dispositivos, sin necesidad de crear múltiples versiones del sitio web.

Las media queries se escriben en la hoja de estilos CSS, dentro de bloques @media que contienen reglas de estilo específicas para cada rango de tamaño de pantalla o dispositivo.

Ampliar información en el Tema 44 de PreparaTIC - Accesibilidad y usabilidad. W3C. Diseño universal. Diseño web adaptativo.

5. XML

XML (eXtensible Markup Language): lenguaje de marcado que permite describir contenido de forma independiente a su presentación. Es un metalenguaje subconjunto de SGML que permite definir la gramática de nuevos lenguajes. Estandarizado por el W3C. La última versión es XML 1.1. Las etiquetas XML son case sensitive.

Las partes principales de un fichero XML son:

- **Prólogo**: es opcional y contiene una sentencia que declara el documento como XML, una declaración del tipo de documento y comentarios e instrucciones de procesamiento.
    - `<?xml version="1.0" encoding="UTF-8"?>`
- **Cuerpo**: es obligatorio, debe contener solo un elemento raíz (imprescindible para que el documento esté bien formado).
- **Elementos**: pueden tener contenido o ser elementos vacíos.
- **Atributos**: permiten incorporar propiedades a los elementos de un documento. Deben ir entre comillas.
- **Entidades predefinidas**: representan caracteres especiales, de manera que no sean interpretados como marcado por el procesador XML.
- **Secciones CDATA**: permiten especificar datos usando cualquier carácter sin que se interprete como marcado XML.
- **Comentarios**: son ignorados por el procesador XML.

5.1. Validación XML

Para que un documento XML sea válido es necesario que cumpla una serie de condiciones. De esta forma se pueden procesar de forma rápida, correcta y eficiente:

- **Documento XML bien formado**: aquél que respeta la estructura y sintaxis definidas por la especificación de XML:
    - Debe comenzar con una declaración XML.
    - Debe tener un único elemento raíz.
    - Los elementos deben estar cerrados y correctamente anidados.
    - Los valores de los atributos deben ir entre comillas.
    - Los elementos son case-sensitive.
- **Documento XML válido**: documento que está bien formado y es conforme a una DTD o XML Schema.

Para ayudarnos en la validación semántica y de lógica de datos disponemos de varias herramientas:

- **DTD (Document Type Definition)**: describe la estructura y sintaxis de un documento XML, definiendo los tipos de elementos, atributos y entidades permitidas. Puede ubicarse en un fichero externo o estar contenida en el propio XML.
- **XML Schemas**: es un documento XML que define los elementos y atributos que puede contener un documento XML. A diferencia de los DTD, se basan en sintaxis XML y son más potentes al permitir definir nuevos tipos de datos, soportar herencia de tipos y permitir usar namespaces.

5.2. PROCESAMIENTO XML

A la hora de procesar un documento XML existen diversas APIs para realizar consultas y modificaciones de documentos XML.

- **DOM (Document Object Model)**: es una API que permite representar objetos de un documento HTML o XML de forma jerárquica. El parser procesa el documento XML completo y elabora un árbol en memoria del mismo, permitiendo a la aplicación recorrer y modificar el árbol. Inconveniente: consume mucha memoria.

- **SAX (Simple API for XML)**: el parser va procesando el documento XML elemento a elemento. Es rápido, pero no permite modificar los datos.

5.3. XSL

XSL (eXtensible Stylesheet Language): lenguaje que permite definir el formato de un documento XML para su presentación, de manera análoga a CSS y HTML. Es un conjunto de estándares del W3C:

- **XSLT (XSL Transformations)**: plantillas con una serie de reglas de transformación y formato que se disparan cuando los elementos coinciden con la plantilla. Transforma un documento XML en otro formato (PDF, HTML...).

- **XPath (XML Path Language)**: lenguaje que permite construir expresiones para recorrer y procesar un documento XML.

- **XSL-FO (XSL Formatting Objects)**: es un documento XML que especifica cómo formatear unos datos para su presentación.

- **XQuery**: es un lenguaje que permite acceder a los datos contenidos en los XML.

- **XLink**: permite crear elementos XML que describen relaciones cruzadas entre documentos, imágenes y archivos.

- **XPointer**: método de direccionamiento de fragmentos de un documento XML. Está construido sobre XPath.

5.4. LENGUAJES BASADOS EN XML

Sindicación web o sindicación de contenidos: reenvío de contenidos desde una fuente original (sitio web origen) a un sitio web de destino, habitualmente codificados en XML.

- **RSS (Really Simple Syndication)**: formato XML para distribuir contenido en la web.

- **RDF (Resource Description Framwork)**: es un modelo estándar para intercambio de datos en la web, siendo una de las tecnologías esenciales de la web semántica. El modelo de datos RDF se basa en hacer declaraciones sobre los recursos en forma de expresiones sujeto-predicado-objeto (tripleta).

Envío de información económica y financiera:

- **XBRL (eXtensible Business Reporting Language)**: formato estándar basado en XML para intercambio de información contable, financiera y tributaria.

- **XAdES (XML Advanced Electronic Signatures)**: formato estándar basado en XML para firma electrónica avanzada.

- **Facturae**: formato estándar basado en XML para intercambio de facturas electrónicas. Es el formato requerido para la presentación de facturas por parte de los proveedores en el punto general de entrada de facturas de la AGE, FACe.

Herramientas técnicas:

- **XAML**: Un lenguaje desarrollado por Microsoft, utilizado para crear interfaces de usuario en aplicaciones Windows e híbridas.

- **Maven**: Una herramienta de gestión de proyectos de software del entorno Java desarrollada por Apache. Utiliza un archivo XML llamado "pom.xml" para definir las dependencias, configuración y construcción del proyecto

- **SVG (Scalable Vector Graphics)**: Es un formato de imagen vectorial basado en XML que permite crear gráficos y animaciones vectoriales escalables, es decir, que pueden ser redimensionados sin perder calidad. A diferencia de los formatos de imagen rasterizados (como JPG o PNG), los archivos SVG no están compuestos por píxeles, sino por vectores y fórmulas matemáticas, lo que permite una mayor nitidez y flexibilidad en su uso.

- **SOAP (Simple Object Access Protocol)**: es un protocolo de comunicación que se utiliza en arquitecturas orientadas a servicios (SOA) para intercambiar datos en formato XML entre sistemas distribuidos.

6. JSON Y OTROS FORMATOS

JSON (JavaScript Object Notation) es un formato de archivo estándar abierto y un formato de intercambio de datos que utiliza texto legible por humanos para almacenar y transmitir objetos de datos consistentes en pares atributo-valor y matrices (u otros valores serializables). Es un formato de datos común con diversos usos en el intercambio electrónico de datos, incluido el de las aplicaciones web con servidores. El estándar JSON se define en **ECMA-404** e **ISO/IEC 21778:2017**

JSON es un formato de datos independiente del lenguaje. Se derivó de JavaScript, pero muchos lenguajes de programación modernos incluyen código para generar y analizar datos en formato JSON. Al igual que XML, podemos utilizar JSONSchema para validar las reglas de datos y JPath para realizar consultas internas. Una de las grandes diferencias con XML es que no soporta comentarios excepto en JSON5 dado que los parseadores son más lentos cuando existen comentarios y JavaScript por defecto no los admite.

En los últimos años se han popularizado alternativas a XML más compactas, pero manteniendo las capacidades de verificación, personalización y legibilidad.

La otra gran alternativa a XML es **YAML** (acrónimo recursivo para "YAML Ain't Markup Language"). **Es un formato de serialización de datos legible por humanos que se utiliza para representar estructuras de datos complejas**. Es similar a JSON en el sentido de que ambos son formatos de intercambio de datos, pero se diferencian en su sintaxis y enfoque.

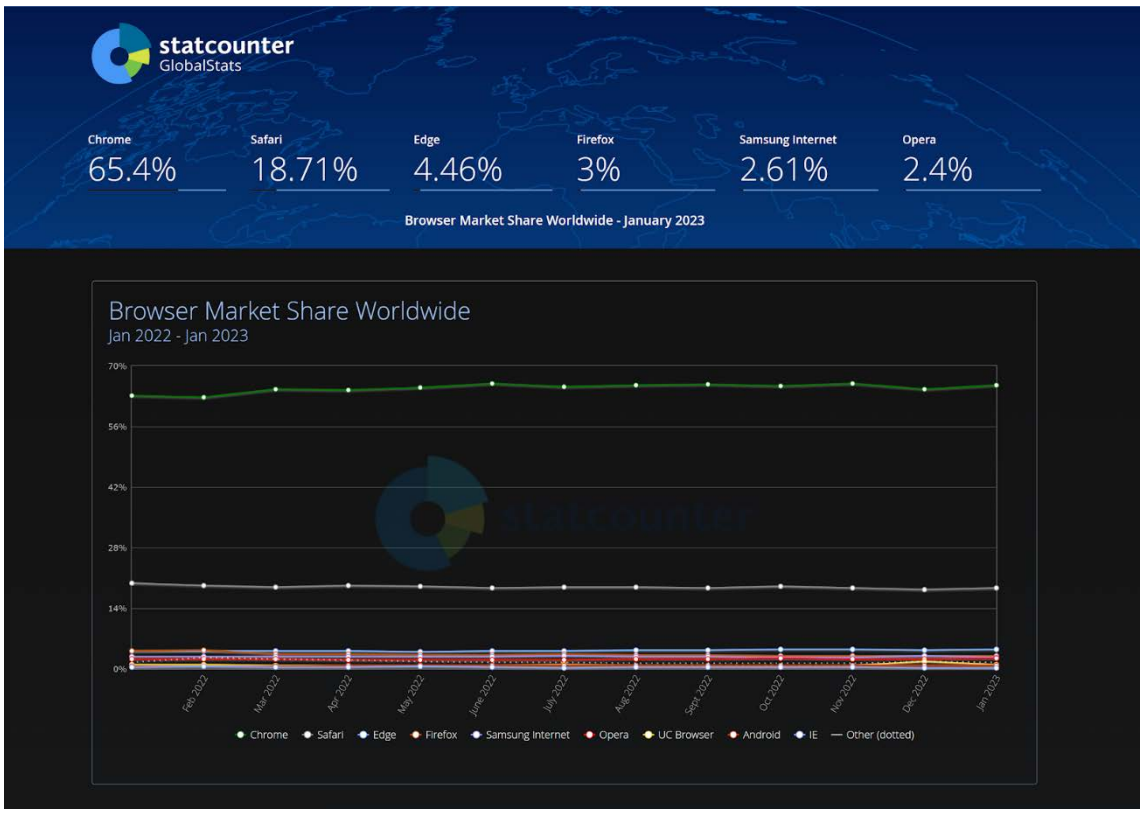
Mientras que JSON utiliza una sintaxis basada en llaves y comillas para representar objetos y arrays, YAML utiliza una sintaxis basada en indentación. Esto hace que YAML sea más fácil de leer y escribir para los humanos, ya que no requiere tantos caracteres de puntuación.

Además, YAML es más flexible que JSON en términos de los tipos de datos que puede representar. Por ejemplo, YAML permite representar fechas, expresiones regulares y referencias a objetos en la misma estructura de datos, lo que lo hace más adecuado para representar datos complejos en sistemas informáticos.

7. NAVEGADORES WEB

Un navegador web o agente de usuario es un programa que permite el acceso a la web, interpretando la información de distintos tipos de archivos y sitios web para que estos puedan ser vistos por un usuario.

A nivel global, el porcentaje de uso de los navegadores en enero de 2023 es:



https://gs.statcounter.com

En los últimos años, Chrome ha sido el navegador dominante sobrepasando a Internet Explorer 11 que en 2022 perdió soporte oficial en favor de Microsoft Edge. Microsoft Edge originalmente era un navegador independiente, pero en 2020 pasa a ser una variante de Chromium (la versión abierta de Chrome).

Un navegador web típicamente consta de varias partes, incluyendo:

- **Interfaz de usuario**: es la parte visible del navegador que permite a los usuarios interactuar con él. Incluye elementos como la barra de direcciones, la barra de herramientas, los botones de navegación, las pestañas, los menús desplegables, etc.
- **Motor de renderizado**: es el componente que se encarga de interpretar el código HTML, CSS y JavaScript de una página web y convertirlo en una representación visual que pueda ser mostrada en la pantalla.
- **Motor de JavaScript**: es el componente que se encarga de interpretar y ejecutar el código JavaScript de una página web.
- **Motor de seguridad**: es el componente que se encarga de garantizar la seguridad y privacidad de los usuarios mientras navegan por la web, incluyendo el bloqueo de sitios web maliciosos y la protección contra el seguimiento en línea.
- **Elementos secundarios**: Elementos de soporte, acciones secundarias o mejora de la experiencia de usuario como: gestores de descargas, gestores de pestañas y marcadores, extensión mediante complementos, integraciones con servicios de terceros…

7.1. Motores de renderizado

El motor de renderizado es el componente principal de un navegador web que se encarga de interpretar y procesar el código HTML, CSS y otros recursos de una página web, para luego mostrarla de manera visual en la pantalla del usuario.

El motor de renderizado es responsable de determinar cómo se deben visualizar los elementos de la página, como el tamaño y la posición de los textos, las imágenes, los videos, los formularios, entre otros elementos. Esto genera problemas de compatibilidad dado que algunas instrucciones de CSS complejas o novedosas no se reflejan igual en todos o no existe soporte.

Existen diferentes motores de renderizado que se utilizan en los navegadores web. Algunos de los más conocidos son:

- **Blink**: es un motor de renderizado de código abierto que se utiliza en el navegador web Google Chrome, así como en otros navegadores basados en Chromium, como Opera.
- **WebKit**: es un motor de renderizado de código abierto que se utiliza en el navegador web Safari de Apple, así como en otros navegadores como Chrome y Opera.
- **Gecko**: es un motor de renderizado de código abierto que se utiliza en el navegador web Firefox.
- **Trident**: es un motor de renderizado desarrollado por Microsoft que se utilizó en versiones anteriores del navegador web Internet Explorer.
- **EdgeHTML**: es un motor de renderizado desarrollado por Microsoft que se utilizó en el navegador web Microsoft Edge antes de su cambio a Chromium.

Cada motor de renderizado tiene sus propias características y capacidades, lo que puede afectar el rendimiento y la compatibilidad del navegador web. Los desarrolladores de navegadores web eligen el motor de renderizado que mejor se adapte a sus necesidades y objetivos.

7.2. Interpretes JavaScript

Un motor de JavaScript es un componente de un navegador web que se encarga de interpretar y ejecutar el código JavaScript de una página web. El motor de JavaScript es responsable de convertir el código JavaScript en instrucciones que la computadora pueda entender y ejecutar.

Los motores de JavaScript utilizan varias técnicas para interpretar y ejecutar el código JavaScript de una página web de manera eficiente y rápida como compilación JIT, optimización de código en tiempo real y predictivo, recolectores de basura y paralelización.

Existen varios motores de JavaScript que se utilizan en los navegadores web. A continuación, se presentan algunos de los más conocidos:

1. V8: es un motor de JavaScript de código abierto desarrollado por Google. Se utiliza en el navegador web Google Chrome, así como en otros navegadores basados en Chromium. También se utiliza en otros productos como Node.js.
2. SpiderMonkey: es un motor de JavaScript de código abierto desarrollado por Mozilla. Se utiliza en el navegador web Firefox.
3. JavaScriptCore: es un motor de JavaScript de código abierto desarrollado por Apple. Se utiliza en el navegador web Safari y otros productos de Apple.
4. Chakra: es un motor de JavaScript desarrollado por Microsoft. Se utilizó en el navegador web Microsoft Edge antes de su cambio a Chromium.
5. Nashorn: es un motor de JavaScript desarrollado por Oracle. Se utiliza en la plataforma Java SE.
6. Rhino: es un motor de JavaScript de código abierto desarrollado por Mozilla. Se utiliza en la plataforma Java SE.

Cada motor de JavaScript tiene sus propias características y capacidades, lo que puede afectar el rendimiento y la compatibilidad del navegador web y al estándar de ECMAScript que soportan.

8. ESTÁNDARES WEB Y SEGURIDAD

Para comprobar la compatibilidad de un navegador con los estándares actuales podemos utilizar páginas como HTML5Test o CSS3Test, aunque actualmente todos los grandes navegadores tienen una compatibilidad con los estándares muy alta. Si somos desarrolladores de sitios web, tenemos que intentar conseguir el cross-browsing, nombre con el que se denomina al hecho de que una página o sitio web se visualicen de igual forma y funcionen correctamente en todos los navegadores.

En JavaScript es estándar ECMAScript es quien determina las nuevas funcionalidades y los navegadores progresivamente dan el soporte. En CSS el W3C propone las ampliaciones oficiales pero muchas veces son de uso de general antes de que una funcionalidad este estandarizada. Dado que cada vendedor lo implementa de una forma concreto, es común utilizar prefijos en instrucciones CSS para las versiones iniciales y mantener la compatibilidad entre navegadores.

Actualmente, todos los navegadores ofrecen la posibilidad de realizar sesiones de navegación en modo privado, lo cual significa que el navegador no guardará nuestro historial de búsqueda y navegación, las cookies o la información introducida en formularios al navegar en este modo. Hay que remarcar que el modo privado no nos hace anónimos para los sitios web o para nuestro proveedor de servicios de Internet, si no que su finalidad es hacer que sea más fácil mantener en privado nuestra actividad en línea respecto a cualquier otra persona que utilice el mismo equipo.

Si realmente queremos realizar una navegación anónima, de tal modo que sea realmente difícil rastrear nuestra IP al movernos por la red, debemos utilizar mecanismos como servidores http proxy (TOR) o redes P2P privadas (I2P, Freenet).