Introducción a elementos semánticos

Nuevas etiquetas de HTML 5

Los elementos semánticos hacen que sea fácil para las computadoras y las personas comprender por igual, el significado y el contexto del contenido de un sitio web. El término “web semántica” no es nuevo, recordemos que el creador de la World Wide Web, Tim Berners-Lee, lo usó por primera vez cuando hablo de la transformación de la World Wide Web en un entorno en el que los documentos publicados están relacionados con información y datos que especifican el contexto semántico en un formato interpretable.

Hay muchas etiquetas que durante el proceso de trabajo aparecieron y otras que dejaron de funcionar, pasamos basándonos en la Web oficial a mostrar las etiquetas más usadas y sus usos actuales

Resúmen de usos, datos, y notas sobre elementos semánticos:

Header:

El elemento está destinado a contener por lo general el encabezamiento de la sección (a h1- h6 elemento o un hgroup element), pero esto no es necesario. El header también puede ser utilizado para envolver la mesa de una sección de contenidos, un formulario de búsqueda, o los logotipos correspondientes.

Ejemplo:

```
<header>
 <p>Bienvenido</p>
 <h1>Mi sitio web</h1>
</header>
```

Nota:

El hgroup es obsoleto ya que los enunciados sólo deben ser títulos principales

(no subtitulos) de secciones en todo caso menos importantes, el HGROUP no debe utilizarse más.

Footer:

El footer representa un pie de página de su ancestro más cercano seccionar el contenido o seccionar la raíz elemento. Un pie de página suele contener información sobre su sección como quién lo escribió, enlaces a documentos relacionados, los datos de derechos de autor, y similares.

El elemento ADDRESS, puede colocarse tanto en el header como en el footer , y el propósito principal de estos elementos es simplemente para ayudar al marcado explica por sí mismo autor de escritura que es fácil de mantener y estilo; no están destinados a imponer estructuras específicas en autores, debe siempre contener información de contacto

```
<footer>
<address> tel: 3456456566767 </address>
</footer>
```

Uso en el root:

```
<!DOCTYPE HTML>
<HTML LANG="en">
<TITLE>Mi ejemplo de footer</TITLE>


<FOOTER>
 <NAV>
  <P><A HREF="/copyright.html">Creadores de la página</A> –
     <A HREF="/terminos.html">Terminos y condiciones</A> –
     <A HREF="/index.html">Blog Index</A></P>
 </NAV>
 <P>Copyright © 2017 EducacionIT</P>
 </FOOTER> </HTML>
```

En una sección:

```
<section>
    <h1>Articulos</h1>
    <p>Duis aute irure dolor in reprehenderit in voluptate velit esse
    cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
    proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
    <a href="articles/primero.html">ver más...</a></p>
   </section>
<footer>
  <p> <small>Autor: Pedro Pablo </small> </p>
  <p><small>Copyright © 2017 Pedro Pablo </small></p>

 </footer>
 </section>
```

Nav:

El HTML <nav>elemento representa una sección de una página que enlaza con otras páginas oa partes dentro de la página: una sección con enlaces de navegación.No todo grupo de links es un NAV, debe considerarse un grupo de links principales

Ejemplo:

```
<nav>
<ul>
<li><a href="DireccionPagina"> Item de Navegación 1 </li>
<li><a href="DireccionPagina"> Item de Navegación 2 </li>
<li><a href="DireccionPagina"> Item de Navegación Etc </li>
</ul>
</nav>
```

Otro ejemplo para que puedas copiar y pegar en tu HTML:

<body>

<header>

<h1>Título Principal</ h1>

<p> Noticias -

 Blog -

 Foros </ p>

```
 <nav>
  <h1> Navegación </ h1>
  <ul>
   <li> <a target="_blank" href=”articulos.html”>Indice </a> </ li >
   <li> <a target="_blank" href="servicios.html"> Servicios</a> </ li>
   <li> <a target="_blank" href="contacto.html"> Contacto< / a> </ li>
  </ ul>
 </ nav>
</ header>
```

Nota:

Los lectores de pantalla, se benefician de estas indicaciones semánticas, para que pueden beneficiarse de la información de navegación que se omiten en la prestación inicial, o que pueden beneficiarse de estar disponible de forma inmediata la información de navegación, puede utilizar este elemento como una forma de determinar qué contenido en el página para saltar al principio o proporcionar a petición (o ambos).

Article:

El elemento article es un contenido independiente y reutilizable, es decir que se puede distribuir de forma independiente o reutilizable, por ejemplo, en una red social. Esto podría ser un mensaje del foro, un artículo de una revista o un periódico, una entrada de blog, un comentario enviado por el usuario, de un widget o gadget interactivo, o cualquier otro elemento independiente del contenido.

Ejemplo para copiar y pegar en tu HTML:

```
<article>
 <header>
  <h1 >El viaje de tu vida</h1>
<p> El viaje de tu vida que nunca vas a olvidar </p>
<p> Este artículo que leerás será genial </p>
 </header>
 <p>El viaje de tu vida consiste en ...</p>
<footer>
```

 Autor: Pedro Pablo

 Fecha: 2017-10-10

</footer>

<article>

Nota:

Fíjese cómo se reutiliza el elemento footer y header en el ejemplo anterior.

Section:

El section representa una sección genérica de un documento o la aplicación. Una sección, en este contexto, es una agrupación temática de los contenidos. Por lo general tiene un header y puede contener un footer eso no significa que sea obligatorio su uso.

Ejemplos de secciones serían capítulos, las distintas páginas con pestañas en un cuadro de diálogo con fichas o las secciones numeradas de una tesis, lo que se llama OUTLINE de una página. La página principal de un sitio Web podría dividirse en secciones para una introducción, noticias e información de contacto.

```
      <section>
  <h1>
Encabezado
</h1>
  <p>
Un montón de contenido impresionante.</p>
</section>
```

Ejemplo de section en un article:

```
<article>
 <header>
  <h2>Manzanas</h2>
  <p>Deliciosa fruta</p>
 </header>
 <p>La manzana es una fruta de un árbol</p>
 <section>
  <h3>Deliciosa</h3>
  <p>Estas son las que conseguis en el supermercado</p>
 </section>
 <section>
  <h3>Granny Smith</h3>
  <p>Estas son las más jugosas</p>
 </section>
</article>
```

NOTA:

Nótese cómo se utiliza un section anidado en un article, y también se puede anidar section en section. A su vez se puede trabajar con un artículo dentro de una sección y varias secciones pueden anidarse también en una sección en particular. No hay límites en cuanto a temas de anidación, siempre que se respete el espíritu semántico de cada elemento

NO SE DEBEN UTILIZAR ELEMENTO SEMÁNTICOS CON MEROS FINES DE STYLING PARA ESO ESTÁ EL DIV.

Aside:

El contenido del aside está sólo tangencialmente vinculado con el contenido principal. Suele contener publicidades, efemérides, y grupos de links adicionales, tales como post recientes, post por fechas, etc. Suele estar del lado derecho pero no necesariamente debe ser así

```
<header>
 <h1> Mi blog maravilloso </h1>
 <p> Mi Lema </p> </header>
<aside> Publicidad </aside>
```

Main:

El elemento main fue el último en incorporarse a los elementos semánticos

El elemento HTML <main> representa el contenido principal del <body> de un documento o aplicación. El área principal del contenido consiste en el contenido que está directamente relacionado, o se expande sobre el tema central de un documento o la funcionalidad central de una aplicación. Este contenido debe ser único al documento, excluyendo cualquier contenido que se repita a través de un conjunto de documentos como barras laterales, enlaces de navegación, información de derechos de autor, logos del sitio y formularios de búsqueda (a menos, claro, que la función principal del documento sea un formulario de búsqueda).Esto ayudará a los lectores de pantalla para saber dónde comienza y termina la parte central de un documento

La definición de WHATWG <main>difiere de la versión W3C porque el elemento no tiene ningún significado. Es simplemente un gancho de estilo (como una con un nuevo nombre) y representa a sus hijos.

El elemento principal puede ser usado como un contenedor para los contenidos dominantes de otro elemento. Representa sus hijos.

Pero nos parece mucho mejor el uso que le da la W3C. Para ver profundización y discusiones sobre el tema recomendamos

Link de referencia:

http://html5doctor.com/interview-steve-faulkner-html5-editor-new-doctor/

Ejemplo:

```
<header>
 <h1> Mi blog maravilloso </h1>
 <p> Mi Lema </p> </header>
 <main>
     <section>
           <h2> mi sección </h2>
     <p> Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
     tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
     quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
     consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
     cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
     proident, sunt in culpa qui officia deserunt mollit anim id est laborum. </p>
                                                                    </section>
                                                                         </main>
 <footer><p> copyright 2017 </footer>
```

NOTA SOBRE EL ATRIBUTO ROLE:

La respuesta corta es: sirve para mejorar la accesibilidad de los sitios web. El atributo role tiene su origen en el estándar ARIA de accesibilidad web y se ha incorporado también al estándar HTML5. La definición más precisa es la que dan en el documento oficial de los roles de HTML:

Los atributos "role" permite al creador de una página añadir información a sus documentos HTML que luego puede ser extraída automáticamente por una máquina para obtener información sobre el propósito de cada elemento de la página.

Las personas ciegas y con otros tipos de discapacidad utilizan dispositivos y navegadores especiales. Si una página utiliza los atributos "role" para describir el propósito de cada elemento, la navegación de esas personas será mucho más agradable, ya que su navegador especial es capaz de "entender" cómo está creada la página y puede leer los contenidos al usuario de manera mucho más lógica.

En cuanto al atributo muchas veces recomienda lo siguiente

Ejemplo para copiar y pegar en tu HTML:

<main role="main">

  ...

</main>

Figure:

El <figure>elemento es utilizado para marcar diagramas, ilustraciones, fotografías y ejemplos de código que ayuden a la comprensión del contenido contextualmente ubicado más cercano. Pero debe ser independiente del flow, por lo tanto lo puedo redistribuir y retirar del contexto sin alterar la estructura del mismo

Hay discusiones también sobre si es correcto la utilización del figure con el logo en el header principal de un sitio web.

La realidad es que no se debería utilizar para tal fin, ya que este no es explicativo es simplemente un logo , marca o entidad, no me ayuda a comprender el contenido alrededor.

Ejemplo para copiar y pegar en tu HTML:

<figure>

<img> <figcaption> </figcaption> </figure>

 contexto

El figcaption tiene como intención ser una leyenda o subtítulo dentro de la figura que representamos con figure. No es obligatorio, y puede ir antes o después del elemento gráfico que está contenido en el figure.

Se intentó previamente trabajar con otros elementos tales como legend, listas de definición etc, pero fracasaron

Un figure , puede contener varias imágenes por ejemplo,

```
<body>
<figure>
  <img src="imagenes/imagen.jpg" alt="texto alternativo">
  <figcaption>Esta imagen tiene un <a href="http://www.flickr.com/photos/
  misfotos/">Vínculo</a></figcaption>
</figure>
<body>
```

Ejemplo con un video para copiar y pegar en tu HTML:

```
<figure>
 <video src="mivideo.avi"></video>
 <figcaption>Este es un video, te gusta?</figcaption>
</figure>
```

Ejemplo en un artículo para copiar y pegar en tu HTML:

```
<article>
 <h1>El dia de las votaciones</h1>
 <figure>
  <img src="candidato.jpeg" alt="Candidato oficial">
  <figcaption>Este es el candidato oficial</figcaption>
 </figure>
 <p>Los candidatos se pusieron de acuerdo para...</p>

</article>
```

Empezando a trabajar con los Estilos:

Qué es CSS? :

CSS (Cascading Style Sheet) me permite darle styling o decir como se van a mostrar los elementos previamente

estructurados en HTML.

Por caso yo sé que con HTML puedo decir que un elemento por ejemplo es un enunciado pero el cómo se verá , qué tipografía tendrá , cuál será su tamaño lo hago con CSS

De qué manera aplicar CSS en HTML? :

CSS se puede implementar de tres maneras en nuestro HTML

En línea:

Esta forma de trabajo se hace dentro del mismo elemento de HTML al cual estoy modificando. Por ejemplo como veremos en el siguiente ejemplo al párrafo se le ha cambiado el color de fondo con la propiedad que más adelante profundizaremos llamada background-color.

Ejemplo para copiar y pegar en tu HTML:

```
<body>

<p style="background-color: red;"> Mi párrafo con un fondo rojo </p>

</body>
```

Interna:

Esta manera de trabajo es más cómoda para el desarrollador y permite una modificación más eficaz así como un mantenimiento más ágil y rápido.

Se trabaja de la siguiente manera, ejemplo para copiar y pegar en tu HTML:

```
<head>

<style>

/*acá escribimos css*/

</style>

</head>
```

Nota:

Como se ha visto en el ejemplo anterior, lo hacemos a través del elemento style, este elemento siempre debe estar

en el head de nuestro documento.

Puede que vean una versión de esta etiqueta de la siguiente manera, ejemplo:

```
<head>
<style type="text/css">
/*acá escribimos css*/
</style>
</head>
```

El atributo type es opcional en la versión de HTML5 por tanto el omitirlo esta más que permitido.

También noten que hemos trabajado con comentarios, recuerden que los comentarios permiten transmitir información o mismo dejar una sentencia o regla de estilo en este caso sin acción . Los comentarios en CSS no importa cuantas líneas tenga se realizan con

/* comentario */

Externo:

Esta forma es la más recomendada. Por qué?. Básicamente, imaginemos un sitio mediano de 30 archivos html , que pasaría si debemos modificar algo o realizar algún cambio, que tal entrar a 40 archivos?, la verdad es una pesadilla no?. Bueno , pero si de pronto tenemos un sólo archivo.css que modifica todo nuestro sitio web? Genial no?

El ejemplo que vemos a continuación muestra cómo vincularlo, siempre recuerden que debemos tener un archivo.css para vincular, este archivo debe guardar las reglas de nomenclatura de cualquier archivo.html (no debe contener espacios, no puede tener caracteres extraños, puede trabajarse con – o _ y también es case sensitive, es decir no da lo mismo mayúscula que minúscula)

```
<head>
<link href="archivo.css" rel="stylesheet">
</head>
```

Explicación:

href => me dice cual será el lugar donde está alojado mi archivo.css.

Puede ser una ruta absoluta, por caso cuando trabajamos con frameworks (como bootstrap) donde nos valemos de los servidores donde están alojadas esas hojas de estilo ejemplo para copiar y pegar en el HTML:

<head>

<link target="_blank" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
 rel="stylesheet">

</head>

Muchas veces tenemos un html con varios archivos externos vinculados, esto es normal, pues trabajamos con hojas propias y también con frameworks como el caso anterior o quizás con archivos.css de algún plugin de Javascript con el cual estamos trabajando.

La pregunta será cuál de los archivos tendrá más prioridad? Bueno esa es una pregunta fácil de responder.

En realidad, no importa si el archivo es externo o si es un estilo interno. Tampoco si tengo varios archivos externos vinculados al mismo tiempo, si por caso estoy modificando el color de fondo del body, en varios archivos o incluso como mencioné anteriormente en un estilo interno siempre el orden decide la jugada, pues el que va a quedar es el que está último en cuanto a orden. Este tema lo vamos a dirimir al final de esta clase con mayor detenimiento.

Vamos a comenzar a trabajar!

Que tal si empezamos a ver como hacer nuestro CSS:

Siempre el CSS trabaja con reglas de estilo:

Cómo es esto? :

```
<head>
<style>
selector { propiedad:valor; }

</style>

</head>
```

El selector qué es?:

Es aquello sobre lo cual implemento la regla de estilo. Es decir aquello que estoy afectando con mi propiedad y valor.

La historia se hace un poco más compleja cuando nos encontramos con varios tipos de selectores.

Selector de elemento o etiqueta:

Este tipo de selector, está creado a partir del mismo elemento de HTML que estamos modificando. Veamos el siguiente ejemplo, para copiar y pegar en el HTML:

```
<style>

p { background-color: red; }

</style>
```

En el ejemplo anterior hemos cambiado el color de fondo de un párrafo, el tema es el siguiente. Todos los párrafos se verán afectados, es decir que este tipo de selector es genérico, afecta a todos los elementos de su tipo.

Selector de ID :

Este tipo de selector permite modificar un elemento de manera puntual. En realidad en nuestro CSS lo veremos de la siguiente forma, ejemplo para copiar y pegar en el HTML:

```
<style>

#parrafo { background-color: red; }

</style>
```

Este tipo de selector se compone en nuestra regla con un # que precede el nombre de id. El nombre de ID debe estar formado por letras o números, - o _ (guión de abajo underscore o guión del medio) , no puede contener espacios y tampoco números al comienzo (esto es una regla no colocar nunca números al principio, pues así sino no funcionará en algunos navegadores).

Luego es importante informar a qué elemento modificaré con este ID, pues a diferencia del selector de elemento o etiqueta anteriormente visto, no sabemos a qué afectamos con este ID salvo que lo informemos.

Cómo lo haremos?, simplemente le diremos al navegador que ID afectará puntualmente a ese elemento a través del atributo ID.

Ejemplo para copiar y pegar en el HTML:

```
<head>

<style>
```

```
#parrafo { background-color: red; }

</style>

</head>

<body>

<p id="parrafo"> Mi párrafo </p>

<p> Otro párrafo si ese ID </p>

</body>
```

El ID tiene una característica peculiar que es básicamente que sólo puede utilizar un mismo nombre de ID una sola vez por HTML.

Por tanto si hacemos caso del ejemplo anterior el ID párrafo en mi index.html será utilizado una sola vez. Si de pronto deseo utilizarlo en otro archivo.html por ejemplo contacto.html puedo hacerlo y no necesariamente debo aplicarlo a un párrafo puedo aplicarlo al elemento que desee pero siempre utilizándolo una vez por HTML.

Selector de CLASS:

Los dos selectores anteriores nos han generado mucho material para trabajar pero qué pasa si de pronto tengo no uno ni todos los elementos a modificar, sino solamente varios, que haré?. En este caso surge un selector que puede utilizarse la cantidad de veces que desee en un mismo HTML. De hecho es el selector que más se utilizará por su característica de ser de esta manera, pero esto no significa que no usaremos ID o SELECTORES DE ELEMENTO, estos se utilizan en menor medida pero siempre están presentes en nuestra hoja de estilo.

El ejemplo siguiente muestra con detenimiento cómo funciona, ejemplo para copiar y pegar en el HTML:

```
<head>

<style>

.color { background-color: red; }

</style>

</head>

<body>

<h1 class="color"> Enunciado con la clase </h1>

<p class="color"> Mi párrafo  con clase</p>

<p>Otro elemento sin clase </p>
```

</body>

Nota:

En el ejemplo anterior, hemos aplicado la misma clase más de una vez y también sin importar el elemento al cual se lo aplicamos.

El atributo class es el que nos permite comunicar a qué elemento afectamos. Las reglas de nomenclatura para los nombres de clases son las mismas que para los ID vistas anteriormente (ojo! No te olvides que no empiezan nunca con números)

Selectores COMPUESTOS:

Qué pasaría si quisiera afectar a varios elementos o afecta a elemento según su posición su lugar , etc.

Bueno tenemos otros selectores, técnicamente son un producto de los selectores anteriores, eso significa que se generan a partir de la unión de dos o más de selectores de elemento , ID o CLASS.

La realidad es que tenemos varios de estos selectores y los iremos viendo a partir de que vayamos avanzado con propiedades y valores. Por lo tanto los invito a conocer las primeras propiedades:

Propiedades de fuente

Font-family:

Esta propiedad es muy interesante, no permite decir que tipografía tendrá el elemento al cual estamos afectando (según el selector que hayamos utilizado) Tiene como valores posibles tipografías que pueden ser seguras de Web (aquellas instaladas en los sistemas operativos de manera predeterminada) o también se pueden incluir tipografías de fantasía.

Cómo es esto? Bueno vamos a profundizar sobre este tema:

La verdad es que en un principio la premisa es que si el usuario no tiene instalado en su panel de control en su carpeta fuentes la tipografía que por caso otro utiliza en su página web, este no la puede ver. Eso es cierto, pero también hay una posibilidad de subir nuestra propia tipografía a la web y de esta forma saltear esta limitación. Vamos de a poco!

La propiedad en principio se trabaja de la siguiente manera, ejemplo para copiar y pegar en el HTML:

<style>

body { font-family: 'Arial Black', Arial, sans-serif; }

</style>

En el ejemplo anterior hay varios puntos a explicar:

Hemos decidido que la tipografía será aplicada al selector de elemento o etiqueta body, por qué?. Bueno en general se verá que este selector está al comienzo de nuestra hoja de estilo básicamente porque estamos trabajando con la 'ley de herencia'.

Esta ley establece que los elementos hijos (anidados, o dentro de otros) heredarán propiedades de fuente de los elementos padres (que anidan o que tienen otros dentro). Es decir que todo lo que esté dentro del body tendrá ahora esta tipografía. Esto en general suele ser así pues trabajamos con una línea estética en nuestro diseño que seguramente pide que la mayoría de los elementos de nuestra página web tengan la misma tipografía.

Entendiendo esto continuemos. Por qué colocamos varias opciones, la idea es acotar al máximo la brecha que pueda llegar a existir entre el diseño original y lo que finalmente ve el usuario.

Por tanto en el ejemplo anterior le decimos al navegador que busque la tipografía Arial Black (este nombre está entre comillas pues cuando el nombre de tipografía está conformado por más de una palabra debe ir entre comillas simples o dobles es indistinto), luego le decimos que si no encuentra esa tipografía en la computadora o dispositivo del usuario entonces vamos a trabajar con la tipografía Arial y sino que por lo menos trabajo con una tipografía que pueda encontrar en la computadora o dispositivo del usuario con el mismo tipo genérico.

Qué es un tipo genérico?:

El tipo genérico agrupa tipografías afines, existen varios tipos pero vamos a explicar los más conocidos:

Serif:

Ejemplos: Times, Times New Roman, Georgia, etc

Estas tipografías tienen terminaciones en los glifos esas significa es decir son más dibujadas que las que veremos a continuación

Sans-serif:

Ejemplos de este tipo de fuentes son, Verdana, Trebuchet, Arial,etc , estas tipografías al no tener terminaciones en los glifos como las anteriores.

Monospace:

Ejemplos:

Courier, Lucida, etc

Se llaman así porque todos sus caracteres ocupan el mismo ancho por tanto parecen como letras de máquinas de escribir.

Nota:

Existen otros tipos genéricos para profundizar más en el tema recomendamos el siguiente link https://es.wikipedia.org/wiki/Familias_de_tipos_de_letra_(HTML)

Por otro lado, qué pasa si he diseñado una tipografía o si por caso quiero trabajar con una tipografía que no se

encuentra en el grupo de las seguras.

Bueno para eso podemos trabajar con la regla @font-face

Vamos a de poco! Qué es una regla o por qué utilizamos el @, bueno tenemos varias reglas en nuestro CSS , nos permiten hacer cosas especiales por ejemplo en este caso incorporar una tipografía especial, en un ejemplo sencillo sin profundizar demasiado ya que esto excede los alcances de este curso:

Cómo funciona?, ejemplo para copiar y pegar:

```
<style>

@font-face {

font-family: 'miFuente';

src: url('fuente.ttf')}


h1 { font-family: miFuente; }


</style>
```

Sin embargo , en este caso intentaremos hacer lo mismo a través de Google Fonts, donde de manera sencilla solo vinculando la fuente a nuestra página podemos utilizar la fuente que gustemos.

Link de referencia:

https://fonts.google.com

Font-weight:

Esta propiedad me permite indicar el peso de fuente de un tipografía. Sus valores son:

normal (la tipografía no tiene peso de fuente es decir es normal)

bold (la tipografía está en negrita)

lighter (la tipografía tiene menos peso de fuente que su elemento padre o elemento que la anida)

bolder (la tipografía tiene más peso de fuente que su elemento padre o elemento que la anida)

Nota:

Tenemos también como valores posibles números del 100 al 900. Estos valores no son lineales o generales es decir que dependiendo que opciones me ofrezca una tipografía puntualmente será las variantes de peso de fuente que tendrá y qué valores específicamente será cada una. Este tema si bien forma parte de este curso se profundizará

cuando se trabajar de manera más avanzada con tipografías especiales que como habíamos mencionado anteriormente excede a este curso y manual.

Ejemplo de uso para copiar y pegar en el HTML:

<style>

p { font-weight: bold; }

strong { font-weight: normal; }

</style>

En este caso hemos puesto a los párrafo en negrita y al strong (elemento que junto con los enunciados y el elemento b cómo así otros tienen de manera predeterminada su peso de fuente en bold) como normales es decir ya no está más en negrita.

Esto cambia el valor semántico del strong?:

Claro que no, recuerden que el CSS no afecta en nada el SEO o posicionamiento orgánico en los resultados de búsqueda.

Por otro lado hemos tocado el tema de predeterminado qué es esto? Claro los elementos más allá que apliquemos estilos o no vienen con cierto styling, cómo es esto? Noten el ejemplo siguiente:

En el ejemplo estamos viendo el inspector de elementos , cómo lo activamos con f12 o botón derecho del mouse sobre la pantalla del navegador y seleccionamos inspector o inspeccionar.

Este elemento es sumamente útil para los desarrolladores, y en este caso puntual nos muestra de un lado la interpretación del navegador de nuestro HTML (Que no es lo mismo que la vista código) y por otro lado algo así como el historial de CSS que hay sobre los elementos que seleccionamos en la primera vista. En el ejemplo anterior se muestra que el h1 es un elemento de bloque, que está en negrita y que tiene márgenes. Esto lo modificamos desde nuestro propio estilo de las formas que hemos visto anteriormente (interna, en linea, externa) pero siempre es bueno saber que es lo predeterminado para poder saber cómo empezar a trabajar.

Font-style:

Esta propiedad nos permite modificar el estilo de nuestra tipografía o si queremos decirlo de manera menos técnica si estará en itálica o no.

Los valores posibles son:

normal (la tipografía no está en itálica)

italic => transforma la tipografía en itálica

oblique => la diferencia con italic, es que está en realidad no es itálica, está como torcida un par de grados.

El valor oblique inclina más la tipografía que el valor italic, observemos el ejemplo siguiente, ejemplo para copiar y pegar en el HTML:

```
<style>

p { font-style: italic; }

</style>
```

Los elementos como em, i, address y otros tienen de manera predeterminada el valor de font-style en italic, si queremos modificarlo lo haremos de la siguiente forma, ejemplo para copiar y pegar en el HTML:

```
<style>

address, i, em { font-style: normal; }

</style>
```

En el ejemplo anterior hay varios puntos a explicar por un lado, por qué hemos trabajado con este selector tan extraño, bueno este es nuestro primer selector compuesto, y puntualmente se llama selector grupal. Está formado a partir de selectores que como comporten la misma propiedad y valor los hemos agrupado para simplificar nuestra estructura.

Puede ser que se componga de selectores de elemento , ID o CLASS, más allá que este ejemplo puntual sólo tiene selectores de elemento podríamos ver lo siguiente, ejemplo para copiar y pegar en el HTML:

```
<style>

p, .elementos, #contenedor { font-style: italic; }

</style>
```

Vamos a continuar, ya falta poquito para terminar!:

Font-size:

Esta propiedad es un poco más compleja pues tiene valores que merecen observar todo con mayor detenimiento, esta propiedad me permite establecer los tamaños de fuente que tendrán los elementos de texto que voy a trabajar:

Medidas relativas:

Estas dependen del medio a través del cual estoy viendo mi página web.

PX:

El PX es una medida que depende la resolución de pantalla del dispositivo a través del cual se ve mi página web.

%:

Es una medida referente a su vez a una medida base. Es decir el 100% es la medida actual de referencia y a medida que cambiamos este valor ese valor porcentual será en referencia a la medida base. Cuál es la medida base, pues los párrafos, los párrafos miden de manera predeterminada 16px , esa medida es el 100%. Por tanto el siguiente ejemplo hará que los enunciados tengan el 50% del 100% que es 16px.

Ejemplo para copiar y pegar en el HTML:

```
<style>

h1 { font-size: 50%}

</style>
```

EM:

El em es equivalente a la medida base, si seguimos con la explicación anterior, entonces 1em = 100% = 16px

Claro está que esto es así pues aún no hemos modificado el tamaño de los párrafos. Pero si por caso hago lo siguiente, ejemplo para copiar y pegar en el HTML:

```
<style>

body { font-size: 12px;}

</style>
```

Esto es así pues hemos modificado la base, ahora es 12px no más 16px, ojo a no confundir, hemos dicho que la

base son los párrafos para poner un ejemplo pero en realidad son todos los elementos de texto que tienen 16px de manera predeterminada salvo los enunciados, estos tienen justamente medidas proporcionales a la medida base.

Esto explica por qué no es suficiente poner a los párrafos una medida puntualmente para modificar la base, sino que debemos ir al elemento padre, en este caso lo hacemos bien general, y decidimos que todo lo que esté dentro de nuestro body tendrá la medida 12px de base.

Para ilustrar mejor esta situación , si se fijan en el inspector de elementos que previamente hemos indagado, verán que los h1 por ejemplo son el doble (2em) de la medida base.

Es decir que a menos que yo modifique esa relación , los h1 siempre serán el doble de la base.

Una variante de em es el rem, que hoy por hoy se recomienda aún más por qué ? Bueno si yo por caso digo que algo es 2em es en base a la medida base de su padre, supongamos que en un párrafo que tiene 14px de medida base, trabajamos con un em al cual le decimos que será 2em, bueno en este caso será el doble de 14px, esto parece algo bastante útil en muchos caso pero en otros casos no queremos que los elementos cambien según la medida del padre sino del root, es decir del html. Por esa razón la pequeña gran diferencia entre rem y em es que rem es la medida en base al root y no al padre directo del elemento.

Ejemplo para copiar y pegar en el HTML:

```
<style>

html { font-size: 62.5%; }

body { font-size: 14px; font-size: 1.4rem; } /* =14px */

h1   { font-size: 24px; font-size: 2.4rem; } /* =24px */

</style>
```

EX:

Es equivalente a la altura de la letra x, no es tan utilizada, la verdad es que ex y em son medidas tipográficas anteriores a CSS por tanto muchas veces son opciones válidas pero en el caso del ex poco usada por eso no profundizaremos en la misma.

También tenemos la posibilidad de usar palabras como smaller, larger, xx-large, x-large, xx-small, x-small, medium, smaller, larger, estas palabras no son absolutas es decir que dependen del elemento padre del elemento al cual le aplicamos este valor de font-size.

Por otro lado también tenemos medidas Absolutas, a diferencia de las relativas estas no dependen del medio a través del cual estoy viendo mi HTML, los ejemplos más típicos son cm, mm, pt, pc, in, etc.

Nota:

Luego tenemos medidas más actuales y vinculadas con la medida de la pantalla del dispositivo a través del cual estamos viendo nuestra página, este tema excede los temas a trabajar en este manual y curso, pero sin embargo les contaremos cuales son:

vw Relativo al 1% del ancho del tamaño de pantalla

vh Relativo al 1% del alto del tamaño de pantalla

existen otras variantes como vmin y vmax, como mencionamos anteriormente estos no son temas propios de este curso.

Color:

La propiedad color me permite modificar el color de un elemento de texto para eso es importante conocer qué valores posibles tenemos:

Cantidad de RGB:

Nosotros en la web trabajamos con el modelo RGB, esto es cantidad de RED, GREEN Y BLUE. Cada canal tiene una intensidad del 0 al 255 por esa razón dependiendo la intensidad de cada canal será el resultado final , por caso veamos el siguiente ejemplo para copiar y pegar en el HTML:

```
<style>

p { color: rgb(255,0,0;} /*color rojo*/

p { color: rgb(0,255,0} /*color verde*/

p { color: rgb(0,0,255} /*color azul*/

</style>
```

Nota:

En el ejemplo anterior, se escribió de manera poco práctica claro está tres veces el mismo selector, pero igualmente vamos a dirimir la cuestión , si fuese esto en que color quedaría mi párrafo, bueno quedaría en azul pues si tengo el mismo selector, misma propiedad, diferente valor siempre queda el último.

Valores HEXADECIMALES:

Estos valores precedidos por un # indican la cantidad de intensidad en cada canal con letras de la A a la F y números del 0 al 9

Ejemplo para copiar y pegar en el HTML:

<style>

p { color: #333}

</style>

Valores PALABRAS:

No tenemos palabras para todos los colores pero sí tenemos varias con las cuales podemos trabajar, hay una lista bastante interesante en el siguiente link:

https://www.w3schools.com/cssref/css_colors.asp

Ejemplo para copiar y pegar en el HTML:

<style>

p { color: blue;}

</style>

Nota:

Existen otras variantes válidas como rgba, hsl y hsla pero por ser temas que exceden este curso serán tratados más adelante en otros cursos.

Por otro lado, se notará que esta propiedad es heredable eso significa que los elementos hijos toman el color de los elementos padres, pero en el caso de los vínculos eso no sucede.

Si queremos puntualmente modificar el color de los vínculos debemos hacerlo directamente:

```
<head>
<style>
a { color: white; text-decoration: none; }

</style>

</head>
```

Lamentablemente se verá que ahora el vínculo no importa su estado (si fue visitado o no) tendrá el mismo color para afectar al mismo en sus variantes,lo haremos a través de las pseudo clases, eso será visto en próximas clases.

Font-variant:

Esta propiedad me permite cambiar el valor de la tipografía y transformarla en versalitas o letras de imprenta. Su valores son:

- normal (la tipografía no está en versalitas)
- small-caps (la tipografía está en versalitas)

Ejemplo para copiar y pegar en el HTML:

<style>

p { font-variant: small-caps; }

</style>

Nota:

Muchas veces vemos en múltiples propiedades del valor de initial, es valor setea la propiedad a su valor inicial y es una valor que no tiene compatibilidad en IE 11 y siguientes, por tanto es poco utilizada.

MÁS PROPIEDADES:

Text-transform:

Esta propiedad me permite transformar un texto de mayúscula a minúscula y viceversa.

Los valores posibles son:

- none (este es el valor predeterminado, es decir no tiene la propiedad aplicada)
- uppercase (transforma las minúsculas en mayúsculas)
- lowercase (transforma las mayúsculas en minúsculas)
- capitalize (transforma la primer letra de cada palabra en mayúscula)

Text-indent:

Esta propiedad me permite indicar la tabulación de la primer línea de un texto. Los valores pueden ser establecidos en medidas relativas o absolutas lo cual también incluye % , esto se relaciona con la anchura del elemento en el cual se encuentra el texto.

Line-height:

Esta propiedad me permite indicar el interlineado de un elemento.

Por caso, los valores posibles son:

normal (valor predeterminado)

número (se multiplica por el tamaño de fuente y eso da el resultado final)

medidas de longitud (acá el % se relaciona con la medida de fuente)

Nota:

Si se trabaja con un número o un % este dependerá del tamaño de fuente es decir es que va ir variando, pero si trabajo por caso con px es una medida fija que no irá variando.

Text-decoration:

Hace referencia a la decoración de texto, sumamente útil cuando se trabaja con vínculos.

Valores posibles:

- none (valor predeterminado salvo de los vínculos)
- underline (valor predeterminado de los vínculos es un subrayado)
- line-through (es una línea que atraviesa el texto)
- overline (es una línea sobre el texto)

Antes de finalizar, vamos a saber que podemos simplificar la escritura de nuestros estilos, trabajando con con shorthand o declaración de una sola línea:

```
<head>
<style>
selector { font: weight variant style tamaño/line-height tipografia}


</style>

</head>
```

En el caso de fuente el mismo será de la siguiente manera, ejemplo para copiar y pegar en el HTML:

<head> <style>

p { font: bold small-caps normal 12px/13px Arial, sans-serif}

</style> </head>