

Universitat
Oberta
de Catalunya

M4.252 - HTML y CSS

PEC 1

Aníbal Santos Gómez

PARTE 1 - TEORÍA

Pregunta 1. Sobre los elementos html y head de un archivo HTML, responded a las siguientes cuestiones:

1. Explicad tres utilidades del elemento title.

<title> es uno de los elementos más importantes de <head>.

- El texto aparece en los navegadores dentro de la barra del título de la aplicación. Es lo primero que verán los usuarios.
- Las tecnologías de asistencia como los lectores de pantalla lo leen para ofrecer una idea sobre qué puede esperar la persona del documento.
- Además es utilizado por motores de búsqueda y posicionamiento SEO.

2. ¿Cómo se indica al navegador el conjunto de caracteres que se debe utilizar en un documento? ¿Cuál es la codificación de caracteres más conveniente para una buena representación de los contenidos?

Un HTML no se puede codificar diferentes partes de un documento en diferentes codificaciones.

La codificación basada en Unicode UTF-8 soporta muchos idiomas y puede acomodar páginas y formularios en cualquier mezcla de esos idiomas. Su uso elimina que el server determine individualmente la codificación de caracteres para cada página servida o formulario entrante. Esto reduce significativamente la complejidad de tratar con un sitio o aplicación multilingüe.

También permite mezclar más idiomas en una sola página que cualquier otra opción de codificación.

Existen tres codificaciones Unicode diferentes: UTF-8, UTF-16 y UTF-32. Sólo UTF-8 debe utilizarse para el contenido de la Web.

Según <https://www.w3.org>, se anima a los autores a utilizar UTF-8. Los verificadores de conformidad pueden aconsejar a los autores que no utilicen codificaciones heredadas. Las herramientas de autoría deberían usar UTF-8 por defecto para los documentos recién creados.

3. ¿Para qué resulta útil la inclusión de la etiqueta "meta" "description"?

La etiqueta <meta> resulta útil para el posicionamiento, conseguimos con ello lograr una mayor visibilidad de los sitios web.

Se utiliza dentro de la cabecera (entre las etiquetas <head> y </head>) de un documento HTML con la finalidad de descubrir que los datos existen y cómo se puede acceder a ellos y para documentar el contenido, calidad y características de los datos, indicando su conveniencia de uso.

Dentro de las propiedades posibles podemos incluir "description", que nos ofrece un breve resumen del contenido del sitio. Un ejemplo sería:

```
<meta name="description" content="brief of a web page">
```

El elemento "name" proporciona a los motores de búsqueda que el elemento del código HTML está especificado. El atributo "content" describe el contenido de este elemento.

Facilitando una descripción del contenido de la página en los resultados de búsqueda, se da a conocer el sitio web y anima a los usuarios a visitarlo.

Antes las palabras clave que se incluían en la meta descripción eran un factor de clasificación importante para los motores de búsqueda. Esto llevó a que cada vez los sitios web se rellenaran de metadatos con palabras clave. Estas meta descripciones no proporcionaban a los usuarios una descripción útil del contenido. Google decidió

entonces que las meta descripciones ya no deberían afectar el ranking de un sitio web.

Aún así, la optimización de sus meta descripciones es importante, ya que tiene un impacto significativo en la tasa de clics. Con una meta description apta y atractiva que anima a los usuarios a hacer clic, puede aumentar el número de estos, incluso si su página tiene un bajo ranking.

Google considera que una página es mucho más relevante si se hace clic con frecuencia y si los usuarios pasan mucho tiempo en ella. Por lo tanto, una buena meta descripción puede mejorar indirectamente el ranking de una página web en Google.

4. **¿En qué etiqueta y con qué atributo se debe indicar el idioma principal de un documento HTML? Mediante un ejemplo, explicad cómo debe indicarse que algún fragmento de texto está en un idioma diferente al declarado para todo el documento.**

¿Por qué resulta importante indicar el idioma del documento y los posibles cambios dentro del texto?

Dentro del elemento html es donde deberemos utilizar el atributo de idioma lang, así estableceremos un idioma predeterminado para el texto en el elemento head del documento.

Esto se hace el elemento html ya que si lo hiciéramos en el body no cubrimos el head del documento. Si tuviéramos cualquier contenido en la página que resulta en idioma distinto al declarado anteriormente, deberemos utilizar los atributos de idioma en los elementos que rodean a ese contenido, lo cual nos permite procesarlo de manera diferente.

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

```
<!doctype html>
```

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

```
<head> </head>
<body>
  <p>Página escrita en español.</p>
  <p lang="en">But this paragraph is written in english.</p>
  <p>Personalizar se traduce al inglés en: <span lang="en">customize</span></p>
</body>
</html>
```

Pregunta 2. Sobre la inclusión de imágenes, y otros elementos multimedia, en un documento HTML, responded a las siguientes cuestiones:

1. ¿Cuál es la mejor manera de incluir imágenes con valor meramente decorativo?

Insertamos una imagen en un documento html con la etiqueta ``, si el valor de las mismas es meramente decorativo utilizaremos el atributo `alt=""` en blanco. Por ya que por ejemplo, un lector de pantalla no perdería el tiempo leyendo contenido que no es de necesidad básica para el usuario.

Las imágenes decorativas no pertenecen realmente al html. CSS background images debería ser usado para insertar decoración, pero a veces no puede ser utilizado de otra manera, o determinadas prácticas hacen que se utilice ``, por lo que `alt=""` es la mejor forma de hacerlo.

2. ¿En qué caso podemos dejar el atributo alt de la etiqueta img vacío?

Con `alt=""` como atributo proporcionamos un cierto tipo de información, que indica, si es `alt=""` en blanco, que la imagen es meramente decorativa, si esta formado por un texto, será un texto alternativo a la imagen. En el caso de incluir `` dentro de la etiqueta `<a>` deberemos proporcionar texto accesible que o bien puede ser dentro del mismo tag `<a>` o en el `alt=""` de la ``.

3. Investigad sobre la inserción de video y audio dentro del HTML y responded a las siguientes cuestiones:

- a. ¿Qué etiquetas se utilizan para incluir video y audio, respectivamente, en las páginas web? Ejemplificad vuestra respuesta con dos fragmentos de código que incluyan, además de las etiquetas, los atributos indispensables.**

Para incluir video dentro de las páginas web utilizaremos la etiqueta nativa de <video> de html5, en esta etiqueta deben estar presentes las etiquetas de inicio y las de cierre. Indicaremos la fuente mediante el atributo src="" o mediante elementos <source>, además indicaremos el tipo de formato de video que utilizaremos. Los tres formatos que se deben tener en cuenta para la Web son WebM, MP4 y OGV.

```
<video>
  <source src="movie.mp4" type='video/mp4; codecs="avc1.42E01E,
mp4a.40.2"' />
  <source src="movie.webm" type='video/webm; codecs="vp8, vorbis"' />
</video>
```

El elemento <audio> se utiliza para insertar contenido de audio en un documento html, y forma parte de html5 nativo.

El atributo src está sujeto a controles de acceso de http, en su lugar podemos usar <source> dentro del bloque <audio></audio> para especificar el audio que vamos a insertar.

```
<audio
src="http://developer.mozilla.org/@api/deki/files/2926/=AudioTest_(1).ogg"
  autoplay>
  Your browser does not support the <code>audio</code> element.
</audio>
```

- b. Añadid a los fragmentos de código de la pregunta anterior el código necesario para superar el problema de la incompatibilidad de formato entre navegadores.**

En primer lugar para el tag de <video> implementamos el reproductor:

```
<video width="800" height="374">
```

```
    <source src="my_video.mp4" type="video/mp4" />
```

```
    <source src="my_video.ogv" type="video/ogg" />
```

```
    <source src="my_video.webm" type="video/webm" />
```

Tu navegador no soporta vídeos en HTML5. Descarga el vídeo aquí.

```
</video>
```

Los navegadores antiguos no soportan este tag así que deberemos utilizar flash, donde tendremos ya preparados el video en formato .flv y un reproductor flash.

```
<video width="800" height="374">
```

```
    <source src="my_video.mp4" type="video/mp4" />
```

```
    <source src="my_video.ogv" type="video/ogg" />
```

```
    <object width="800" height="374" type="application/x-shockwave-flash" data="fallback.swf">
```

```
        <param name="movie" value="fallback.swf" />
```

```
        <param name="flashvars" value="autoplay=true&file=video.flv"
```

```
    />
```

```
    </object>
```

```
</video>
```

En el caso del audio, tendremos un problema similar al insertar nuestro reproductor de audio:

```
<audio controls>
```

```
<source src="/media/audio.ogg" type="audio/ogg" />
<source src="/media/audio.mp3" type="audio/mpeg" />
</audio>
```

En Internet Explorer no son compatibles ni ogg ni wav, ni firefox ni opera es compatible mp3, en safari no es compatible ogg, por lo que para estar seguros incluiremos flash:

```
<object                                type="application/x-shockwave-flash"
data="reproductor.swf?soundFile=audio.mp3">
  <param name="movie" value="reproductor.swf?soundFile=audio.mp3" />
</object>
```

- c. **Sabemos que es posible incluir un video de un servicio de videos compartidos, como por ejemplo YouTube, dentro de nuestra página web. Indicad qué etiqueta hay que utilizar. Dentro de esta etiqueta, YouTube incluye un atributo del que podríamos prescindir, indicad cuál y cómo podríamos conseguir el mismo efecto que provoca, mediante el uso de CSS.**

Mediante el tag <iframe> podremos insertar un video alojado en Youtube:

```
<iframe                                width="560"                                height="315"
src="https://www.youtube.com/embed/LnflMmVMFVs"                                frameborder="0"
allow="accelerometer; autoplay; encrypted-media; gyroscope;
picture-in-picture" allowfullscreen>
</iframe>
```

Dentro de la etiqueta <iframe> tenemos dos atributos como width y height (ancho y altura donde establecemos la altura en píxeles CSS según html5). Podemos conseguir el mismo efecto utilizando un selector class o id en CSS o utilizando el atributo style="" (CSS también), para indicar los estilos aplicables a la etiqueta <iframe>. Podemos hacerlo de dos maneras:

```
<iframe name="iframe1" id="iframe1" src="empty.htm"
```



```
frameborder="0" border="0" cellspacing="0"
style="border-style: none; width: 100%; height: 120px;">
</iframe>
```

El estilo de la página incrustada en el iframe debe establecerse incluyéndose en la página secundaria:

```
<link type="text/css" rel="Stylesheet" href="Style/simple.css" />
```

O bien cargando desde el padre de la página con Javascript:

```
var cssLink = document.createElement("link");
cssLink.href = "style.css";
cssLink.rel = "stylesheet";
cssLink.type = "text/css";
frames["iframe1"].document.head.appendChild(cssLink);
```

Pregunta 3. Sobre CSS, responded a las siguientes cuestiones:

- 1. Indicad las tres maneras de aplicar CSS a un documento HTML. ¿Cuál de las tres maneras es más ventajosa y por qué? ¿Cuál deberíamos evitar y por qué?**

Existen tres formas de aplicar CSS a un documento HTML:

- CSS externo.
 - CSS Interno.
 - CSS embebido.
- CSS externo: En la cabecera del HTML, el bloque <head></head>, incluimos una relación al archivo CSS:

```
<link rel="stylesheet" type="text/css" href="index.css" />
```

Esta es la manera recomendada de utilizar un archivo CSS. Los navegadores sabrán que deben aplicar los estilos de este archivo al documento HTML actual.

Es la forma más ventajosa porque se obliga así al navegador a aplicar los estilos cuanto antes y eliminar la sensación de que la página no ha cargado por completo.

- CSS interno:

En esta modalidad hay que tener en cuenta que utilizándolo, perdemos la ventaja de tener los estilos en un documento independiente, por lo que es preferible guardarlo en un archivo externo CSS.

```
<!DOCTYPE html>
<html>
<head>
  <title>Título de la página</title>
  <style type="text/css">
    div {
      background:#FFFFFF;
    }
  </style>
</head>
```

- CSS embebido:

Consiste en aplicar CSS directamente sobre las propias etiquetas html del documento:

```
<p>¡Hola <span style="color:#FF0000">qué tal</span>!</p>
```

Su uso se desaconseja también, pero puede sernos útil en algunas ocasiones y para probar determinados estilos sobre ciertos elementos, aunque luego lo independicemos a nuestros documentos externos de CSS.

Sin duda alguna la forma menos ventajosa es cargarlo en cada etiqueta directamente, esta situación solo se aplica cuando queremos estilizar algo de forma muy concreta. La mejor opción sin duda es externalizar nuestro código CSS a un archivo externo y cargarlo en nuestro documento HTML, entre otras cosas, será siempre más ordenado y mantenible nuestro código.

2. ¿Qué son las "reglas at" (@rules)? Explicad dos usos de estas reglas mediante un ejemplo donde se muestre el código que es necesario utilizar para implementarlas.

Las Reglas Condicionales (At-rules) es un módulo de CSS que permite definir un conjunto de reglas que solo aplicarán con base en las capacidades del procesador o del documento al cual la hoja de estilos está siendo aplicada.

Es una declaración CSS que comienza con el símbolo arroba, '@', un identificador, e incluye todo el contenido hasta el siguiente punto y coma, o el siguiente bloque CSS.

Son agrupaciones de selectores que nos permiten aplicarlas en bloque a un objeto determinado, su sintaxis es parecida a la de un selector, con la diferencia que están precedidas por una coma, y pueden contener algunos atributos antes de el primer bracket ({}):

@import rule

La regla @import permite importar estilos desde otra hoja de estilo. Debe aparecer justo al principio de la hoja de estilo antes de cualquiera de las reglas, y su valor es una URL. Se puede escribir de una de las dos maneras siguientes:

```
<style type = "text/css">  
    @import url("mystyle.css");  
</style>
```

```
<style type = "text/css">  
    @import "mystyle.css";  
</style>
```

@font-face rule

La regla @font-face se utiliza para describir exhaustivamente una cara de fuente para su uso en un documento. @font-face también puede usarse para definir la ubicación de una fuente para descargar, aunque esto puede tener límites específicos de implementación.

```
<style type = "text/css">
  @font-face {
    font-family: "Scarborough Light";
    src: url("http://www.font.site/s/scarbo-lt");
  }
  @font-face {
    font-family: Santiago;
    src: local ("Santiago"),
    url("http://www.font.site/s/santiago.tt")
    format("truetype");
    unicode-range: U+??,U+100-220;
    font-size: all;
    font-family: sans-serif;
  }
</style>
```

3. ¿En qué consisten los siguientes mecanismos de resolución de conflictos entre reglas CSS?

En CSS los mecanismos de resolución de conflictos funcionan de la siguiente manera; si dos declaraciones tienen la misma importancia, la especificidad de las reglas decidirá cuál se debe aplicar. Si las reglas tienen la misma especificidad, el orden de las fuentes controla el resultado.

a. Importancia.

La importancia de una declaración de CSS depende de dónde se ha especificado. Las declaraciones contrapuestas se aplicarán en el orden siguiente: las nuevas anularán a las más antiguas.

Cuando empleamos `important` en un estilo, esta declaración sobrescribe a todas las demás, es un uso considerado mala práctica y rompe la cascada de estilos. Casos en los cuales podemos aplicarlos, porque no existe otra forma, son aquellos como declaraciones específicas que sobrescriben css externo como el de frameworks como bootstrap.

```
* {  
  font-family: "Comic Sans MS" !important;  
}
```

b. Especificidad.

La especificidad hace referencia a la relevancia que tiene un estilo concreto de CSS sobre un elemento de la página al cual le están afectando varios estilos de CSS al mismo tiempo. Es decir, hace referencia al grado de importancia de un estilo sobre otro.

Cuanto mayor sea la especificidad que le estemos dando a un estilo, mayor será la probabilidad de que ese estilo sea el aplicado finalmente. Para ello, las reglas de CSS siguen un orden de prioridad.

El orden de prioridad va de la siguiente manera, de menor a mayor especificidad:

- Selectores de título (p.ej: `p`) y pseudo-elementos (p.ej: `:before`)
- Selectores de clase (p.ej: `.ejemplo`), selectores de atributos (p.ej: `[type="text"]`) y pseudo-clases (p.ej: `:focus`)
- Selectores ID (p.ej: `#ejemplo`)

Sin embargo, además de todas estas especificidades, si utilizamos estilos inline estos sobrescribirán cualquier estilo de las páginas externas de CSS. Se podría decir que los estilos inline son los que tienen una mayor especificidad, por lo tanto, nunca debemos utilizar estilos inline en nuestra página.

```
<p id="parrafo" class="parrafo">Esto es una prueba</p>
<p id="parrafo2" style="color: red">Esto es una prueba</p>
```

```
#parrafo{
  color: green;
}
```

```
#parrafo2{
  color: green;
}
```

```
.parrafo{
  color: red;
}
```

c. Orden de las fuentes

Si dos declaraciones afectan al mismo elemento, tienen la misma importancia y la misma especificidad, lo que las distingue es el orden en las fuentes. La declaración final se aplica sobre las anteriores.

Las declaraciones al final del fichero anularán a las que sucedan antes al fichero en caso de conflicto. Las contrapuestas también pueden suceder en diferentes hojas de estilo.

Por ejemplo:

```
p {
  color: cyan;
```

```
}
```

```
p {  
  background-color: yellow;  
  color: black;  
}
```

La última regla especifica color:black y anulará a color:cyan de la regla anterior.

4. Indicad el selector a utilizar para conseguir afectar a los elementos siguientes:

a. Todos los elementos de la página.

Seleccionamos todos los elementos de la página con el selector *

```
* {  
  background-color: yellow;  
}
```

b. Todos los h2 que son hermanos de un h1.

Utilizamos el selector de hermano adyacente, que son los elementos que siguen a otros. +

```
h1 + h2 {  
  font-style: italic;  
}
```

c. Todos los span descendientes de un p.

Seleccionamos todos los descendientes de otro elemento con el selector (representado por un espacio).

```
p span {  
    background-color: DodgerBlue;  
}
```

- d. Todas las imágenes de clase "warning" que dentro de su atributo src contengan un archivo con extensión ".jpg"; este selector debería prever que ".jpg" puede estar escrito también en mayúsculas.**

```
img .warning[src$=".jpg"] {  
    width: 50%;  
}
```


PARTE 2. PARTE PRÁCTICA.

Explicación de las entidades HTML y CSS utilizadas.

1. Sobre las entidades HTML: debe explicarse el uso de las etiquetas elegidas.
 - `<!DOCTYPE html>`: Declaramos el tipo de documento, nos sirve para saber que nuestro documento está siguiendo la estructura determinada por un DTD, que determina los atributos que pueden tener las etiquetas.
 - `<html>`: Utilizamos esta etiqueta para declarar el documento html y en él, el idioma que está escrito, "es".
 - `<head>`: declaramos la cabecera del documento, donde quedan reflejadas etiquetas de título de página, y los metas, además de la importación de los estilos css mediante la etiqueta link.
 - `<meta>`: aportamos información del documento, como la descripción del sitio, la compatibilidad con navegadores, el tipo de caracteres soportados, si soporta viewport. Todo esto se utiliza para el posicionamiento en motores de búsqueda.
 - `<title>`: es un elemento de la cabecera que describe el título del documento.
 - `<link>`: es un elemento que carga un recurso externo, en este caso cargamos nuestros estilos css haciendo referencia mediante el atributo rel a stylesheet y con href indicamos la ruta del archivo.
 - `<body>`: representa el contenido de un documento HTML.
 - `<header>`: representa un grupo de ayudas introductorias o de navegación. Hemos definido las etiquetas h1, y nav como cabeceras de todas los documentos. Los divs utilizados son contenedores para modificar los estilos de forma sencilla y genérica para todos los documentos.

- `<nav>`: epresenta una sección de una página cuyo propósito es proporcionar enlaces de navegación. Aquí hemos utilizado para componer la navegación una lista no ordenada que incluye una etiqueta `<a>` para referirnos a cada documento mediante su atributo href.
- `<section>`: representa una sección genérica de un documento. Sirve para determinar qué contenido corresponde a qué parte de un esquema. Lo hemos utilizado en nuestros documentos para definir el contenido semántico que engloba la información que consultará el usuario. Está envuelto por un div a modo de wrapper para modificar los estilos de forma universal a través del archivo estilos.css.
- `<footer>`: esta etiqueta hace referencia al pie de página donde figura mi nombre y apellidos como autor de cada documento y la asignatura para la cual se presenta el trabajo realizado.
- `<hr>`: hemos utilizado hr para separar cada sección, header, section y footer.
- `<h1>`: utilizamos h1 como elemento de título de primer nivel único para cada documento, donde indicamos el título del mismo.
- `<h2>`: utilizamos h2 como elemento de título para cada sección del documento.
- `<h3>`: utilizamos h3 como elemento de título para cada subsección del documento.
- `<div>`: div es utilizado para generar contenedores para agrupar diversos elementos mediante clases o ids que actuarán como selectores de css.
- ``: al igual que div, utilizamos span como contenedor de línea para modificar algunas características de los estilos de algunas palabras o pequeños conjuntos de estas.
- `<p>`: utilizamos p para distribuir los textos de los diferentes documentos.

- ``: em es utilizado para dar énfasis a diferentes palabras del documento, sobre todo títulos del autor o fechas reseñables.
- ``: para insertar las imágenes en el documento utilizamos el tag `img`.
- `<iframe>`: para insertar el video de youtube, utilizamos la etiqueta `iframe`, que nos embebe un video de la fuente correspondiente.
- ``: con `ul` hemos creado listas no ordenadas para la navegación y para los subapartados de la lista ordenada de las obras.
- ``: con `ol` hemos realizado nuestra lista de elementos ordenados, que hacen referencia a las obras.
- ``: `li` nos sirve para determinar cada elemento de la lista o bien ordenada o sin numerar.
- `<dl>`: creamos un contenedor de lista descriptiva para las reseñas de las obras.
- `<dt>`: creamos por cada título el grupo de términos que determinarán cada una de las reseñas.
- `<dd>`: hace referencia al contenido y las descripciones propias de las reseñas.
- `<blockquote>`: hace referencia a una cita de bloque contenida en el documento del artículo.
- `<q>`: utilizamos `q` para crear citas contenidas dentro de párrafos.
- `<a>`: utilizamos `<a>` para linkear a otros sitios web externos o internos dentro de nuestros documentos.

2. Sobre el CSS: debe explicarse todo el CSS utilizado, incluyendo cuando se ha utilizado y por qué y para qué se aplican los selectores y propiedades definidos.

A continuación se describe todo el código utilizado para generar los estilos y maquetar los documentos html.

```
/****** BODY RESET *****/
```

```
* {  
  box-sizing: border-box;  
  margin: 0;  
  padding: 0;  
}
```

Aquí hacemos un reset de todos los estilos que puedan ser aplicados por el navegador u otros agentes externos, quitando márgenes, paddings.

```
/****** HEADER & NAV *****/
```

```
header {  
  padding: 30px 30px 0 30px;  
  text-align: center;  
  font-family: Arial, Helvetica, sans-serif;  
}
```

Con el selector header establecemos un padding, centramos el texto y seteamos la fuente que utilizaremos en toda la etiqueta.

```
header>img {  
  width: 50px;  
}
```

Aquí fijamos la imagen del logo a un ancho fijo para cada documento.

```
header>h1 {  
    margin-top: 20px;  
    font-size: 35px;  
}
```

En este fragmento de css, establecemos una separacion con el margin-top y fijamos un tamaño determinado para la fuente.

```
#nav-list li {  
    list-style-type: none;  
}
```

Con el selector id nav-list y li quitamos los bolos de cada uno de los elementos li del nav.

```
#nav-list li a {  
    display: inline-block;  
    text-align: center;  
    font: normal 11px Arial;  
    text-transform: uppercase;  
    text-decoration: none;  
}
```

Aquí con display inline-block hacemos que el comportamiento sea en bloque y se renderiza en la línea con el resto de los elementos seleccionados. Centramos el texto, seteamos la fuente a normal en Arial de 11 px, la transformamos a mayúsculas y le quitamos el subrayado del hipervínculo.

```
#nav-list a:hover {  
    font-weight: bold;  
}
```

Cuando pasemos el puntero del ratón la fuente se vuelve negrita.

```
#nav-list a:visited {  
    color: black;  
}
```

Seteamos el color en negro para que sea uniforme visitado o no visitado.

```
.header-item {  
    margin-top: 30px;  
}
```

Esta clase selector es aplicada como envoltorio a la etiqueta nav para desplazarla con margin-top.

```
/****** CONTENT SITE *****/
```

```
.content-space {  
    margin: auto;  
    width: 50%;  
    font-family: 'Arial Narrow Bold';  
}
```

Aquí definimos el contenedor de las secciones, fijamos el ancho a la mitad, los márgenes automáticos y la fuente.

```
.content-title {  
    margin-top: 40px;  
    text-align: center;  
}
```

Definimos las características de espacio que tendrán los títulos de el contenido.

```
.content-subtitle {  
    margin: 20px 120px 0 120px;  
    text-align: center;
```

```
    font-size: 18px;
}
```

Definimos las características de espacio que tendrán los subtítulos de el contenido.

```
.content-credits {
    margin: 25px;
    text-align: center;
}
```

Definimos las características de espacio que tendrán los créditos de el contenido.

```
.author {
    text-decoration: underline;
}
```

Definimos las características de los autores de los créditos del contenido.

```
.content-img img {
    display: block;
    margin-top: 20px;
    margin-left: auto;
    margin-right: auto;
    margin-bottom: 20px;
    width: 50%;
}
```

Definimos las características de espacio y posición de las imágenes del contenido.

```
.content-img p {
    text-align: center;
}
```

Centramos el párrafo que se sitúa dentro del contenedor de imagen.

```
ol {  
    padding: 20px 30px 40px 30px;  
}
```

Establecemos un padding de separación en la lista ordenada, del resto de elementos.

```
ul {  
    padding: 0 50px 20px 50px;  
    font-style: italic;  
    text-decoration: underline;  
}
```

Establecemos una separación de los elementos no ordenados, subrayándolos y modificándolos a cursiva.

```
dt {  
    margin-top: 30px;  
    margin-bottom: 15px;  
    font-size: 18px;  
}
```

Fijamos los espacios entre cada elemento dt y el tamaño de la fuente.

```
dd {  
    padding: 0 0px 0 40px;  
    line-height: 150%;  
}
```

Fijamos la separación hacia la izquierda de cada descripción y el interlineado.

```
dd>p {  
    margin-top: 15px;  
}
```


Separamos el párrafo contenido en el contenedor de la lista descriptiva dd.

```
iframe {  
    display: block;  
    margin-top: 25px;  
    margin-left: auto;  
    margin-right: auto;  
    margin-bottom: 25px;  
}
```

Establecemos la posición de la etiqueta iframe, de la misma manera que hemos definido las imágenes.

```
.content-iframe {  
    text-align: center;  
    font-size: 12px;  
}
```

Fijamos el tamaño de la fuente y alineamos el texto del contenedor del iframe.

```
.content-text p {  
    margin-top: 30px;  
    margin-bottom: 30px;  
    line-height: 150%;  
}
```

Fijamos el espacio entre párrafo y el interlineado de cada uno de forma general.

```
.content-title-text {  
    font-size: 20px;  
}
```

Fijamos el tamaño de la fuente de cada título de cada texto.

```
.content-referece-text {  
    margin-left: 65px;  
    margin-right: 200px;  
    border-left-width: 5px;  
    border-left-style: solid;  
    border-left-color: black;  
}
```

Fijamos los estilos del blockquote con un contenedor, donde establecemos los márgenes, y la línea que nos indica la cita.

```
.creative-commons {  
    margin: auto;  
    width: 90%;  
    text-align: center;  
    font-size: 10px;  
    margin-bottom: 30px;  
}
```

Definimos como será el contenedor que contendrá la información de creative commons, centramos el texto, fijamos el tamaño de la fuente y los espacios, así como el ancho total.

```
.creative-commons img {  
    display: block;  
    margin-top: 20px;  
    margin-left: auto;  
    margin-right: auto;  
    margin-bottom: 10px;  
    width: 12%;  
}
```

Fijamos como se situa y se fija la imagen de creative commons con el selector img, que ademas de lo anteriormente aplicado, se aplicarán estas reglas css.

```
.creative-commons a {  
    color: black;  
}
```

Los enlaces contenidos dentro de este contenedor serán en color negro.

```
#big-timonel {  
    margin-top: 30px;  
}
```

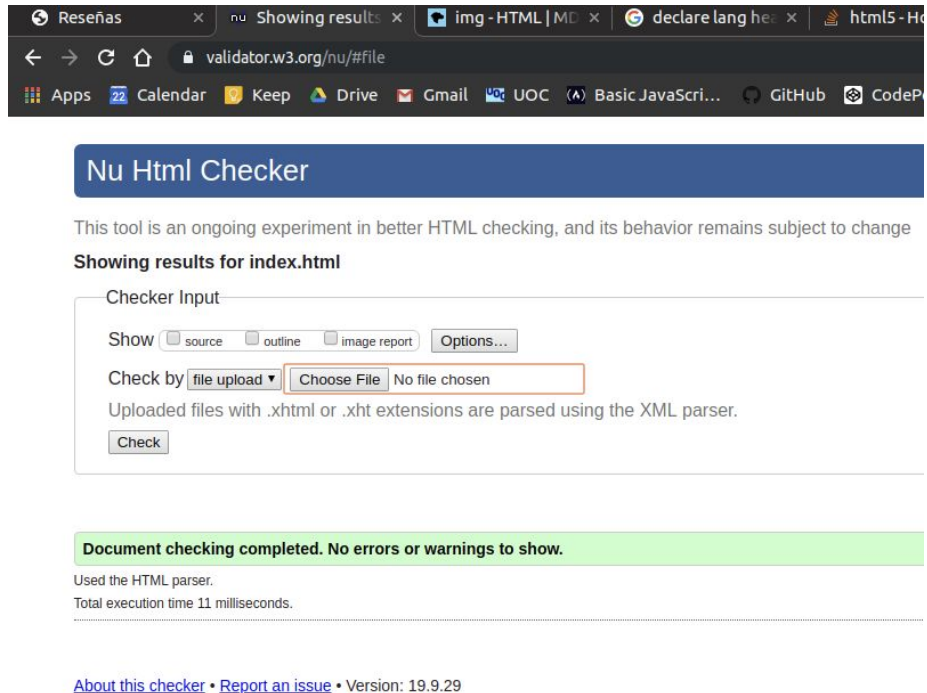
Aplicamos un selector por id, en este caso donde nos falta un poco de separacion entre el contenedor anterior y el siguiente.

```
.underline-reference {  
    text-decoration: underline;  
}
```

Aplicamos un subrayado a un span contenido en
orwell-en-tiempo-de-reconocimiento-facial.html

Validación y accesibilidad de las páginas.

Validación <https://validator.w3.org/>



The screenshot shows the W3C Nu Html Checker interface in a browser. The browser's address bar displays 'validator.w3.org/nu/#file'. The page title is 'Nu Html Checker'. Below the title, a message states: 'This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change'. The main heading is 'Showing results for index.html'. Under 'Checker Input', there are options to 'Show' (source, outline, image report) and a 'Check by' dropdown set to 'file upload'. A 'Choose File' button is highlighted with a red box, and the text 'No file chosen' is visible. A 'Check' button is at the bottom of the input section. Below this, a green box contains the message: 'Document checking completed. No errors or warnings to show.' Further down, it says 'Used the HTML parser.' and 'Total execution time 11 milliseconds.' At the bottom, there are links for 'About this checker' and 'Report an issue', and the version '19.9.29'.

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for index.html

Checker Input

Show ☐ source ☐ outline ☐ image report

Check by file upload No file chosen

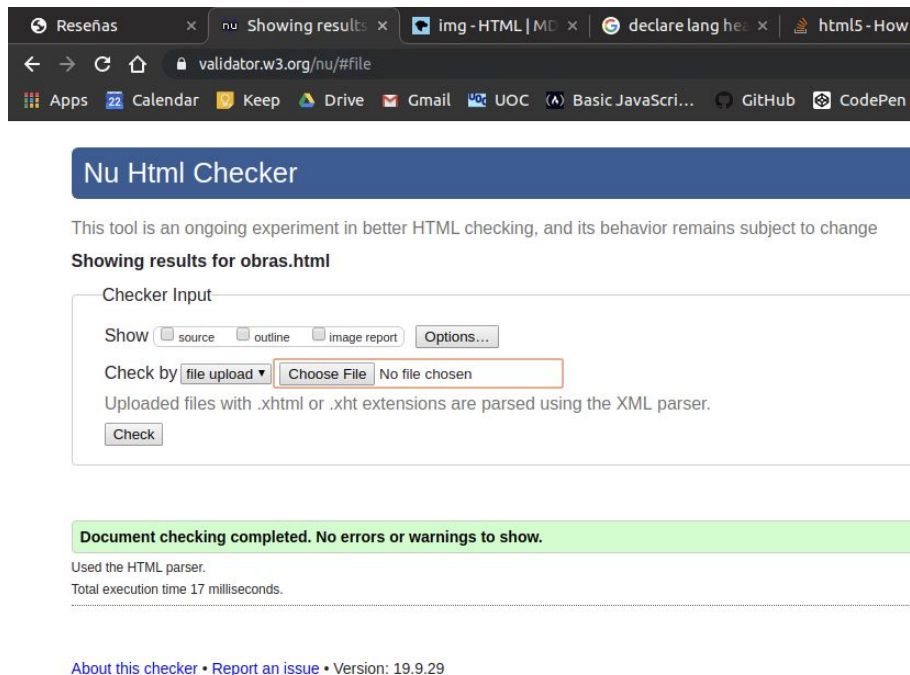
Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

Document checking completed. No errors or warnings to show.

Used the HTML parser.

Total execution time 11 milliseconds.

[About this checker](#) • [Report an issue](#) • Version: 19.9.29



This screenshot is identical to the one above, but the main heading is 'Showing results for obras.html'. The 'Total execution time' is now '17 milliseconds'.

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for obras.html

Checker Input

Show ☐ source ☐ outline ☐ image report

Check by file upload No file chosen

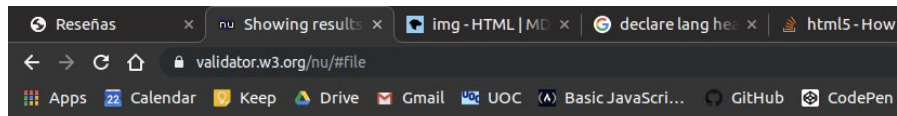
Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

Document checking completed. No errors or warnings to show.

Used the HTML parser.

Total execution time 17 milliseconds.

[About this checker](#) • [Report an issue](#) • Version: 19.9.29



Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for orwell-en-tiempo-de-reconocimiento-facial.html

Checker Input

Show ☐ source ☐ outline ☐ image report

Check by No file chosen

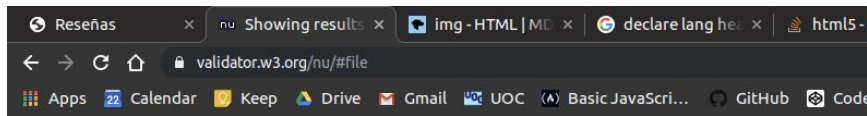
Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

Document checking completed. No errors or warnings to show.

Used the HTML parser.

Total execution time 148 milliseconds.

[About this checker](#) • [Report an issue](#) • Version: 19.9.29



Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for estilos.css

Checker Input

Show ☐ source ☐ outline ☐ image report

Check by No file chosen

Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

Document checking completed. No errors or warnings to show.

Total execution time 4 milliseconds.

[About this checker](#) • [Report an issue](#) • Version: 19.9.29

Validacion <https://achecker.ca>



Web Accessibility Checker
atutor.ca/achecker

Saturday October 26, 2019 16:53:15

Source Title: Orwell

Accessibility Review (Guidelines: WCAG 2.0 (Level AA))
Report on known problems (0 found):

 **Congratulations! No known problems.**



Web Accessibility Checker
atutor.ca/achecker

Saturday October 26, 2019 16:54:02

Source Title: Reseñas

Accessibility Review (Guidelines: WCAG 2.0 (Level AA))
Report on known problems (0 found):

 **Congratulations! No known problems.**



Web Accessibility Checker
atutor.ca/achecker

Saturday October 26, 2019 16:54:48

Source Title: Artículo

Accessibility Review (Guidelines: WCAG 2.0 (Level AA))
Report on known problems (0 found):

 **Congratulations! No known problems.**