

Universitat
Oberta
de Catalunya

M4.252 - HTML y CSS

PEC 2

Aníbal Santos Gómez

PARTE 1 - TEORÍA

Pregunta 1. Sobre la utilización de tablas, responde a las siguientes cuestiones:

- 1. Indica en qué casos es recomendable, según el tipo de datos, usar una tabla, y en qué casos está totalmente desaconsejado. Adjunta una captura de pantalla de una web donde se muestre un ejemplo de uso adecuado de una tabla.**

Las tablas HTML deberán usarse para mostrar datos tabulares, es decir, para datos que puedan mostrarse en una tabla, porque este es su propósito. Podemos decir que el uso principal es representar datos de más de una dimensión.

Las tablas no deben utilizarse como un medio para maquetar el contenido de los documentos, ya que esto puede presentar problemas a la hora de renderizar en medios no visuales. Cuando se utilizan con gráficos, estas tablas pueden obligar a los usuarios a desplazarse horizontalmente para ver una tabla diseñada en un sistema con una pantalla más grande. Para minimizar estos problemas, los autores deben utilizar hojas de estilo para controlar el diseño en lugar de tablas.

Las tablas reducen la accesibilidad para los usuarios con discapacidad visual: ya que los lectores de pantalla, interpretan las etiquetas que existen en una página HTML y leen el contenido al usuario. Debido a que las tablas no son la herramienta correcta para el diseño, y el marcado es más complejo que con las técnicas de diseño CSS, el resultado de los lectores de pantalla será confuso para sus usuarios.

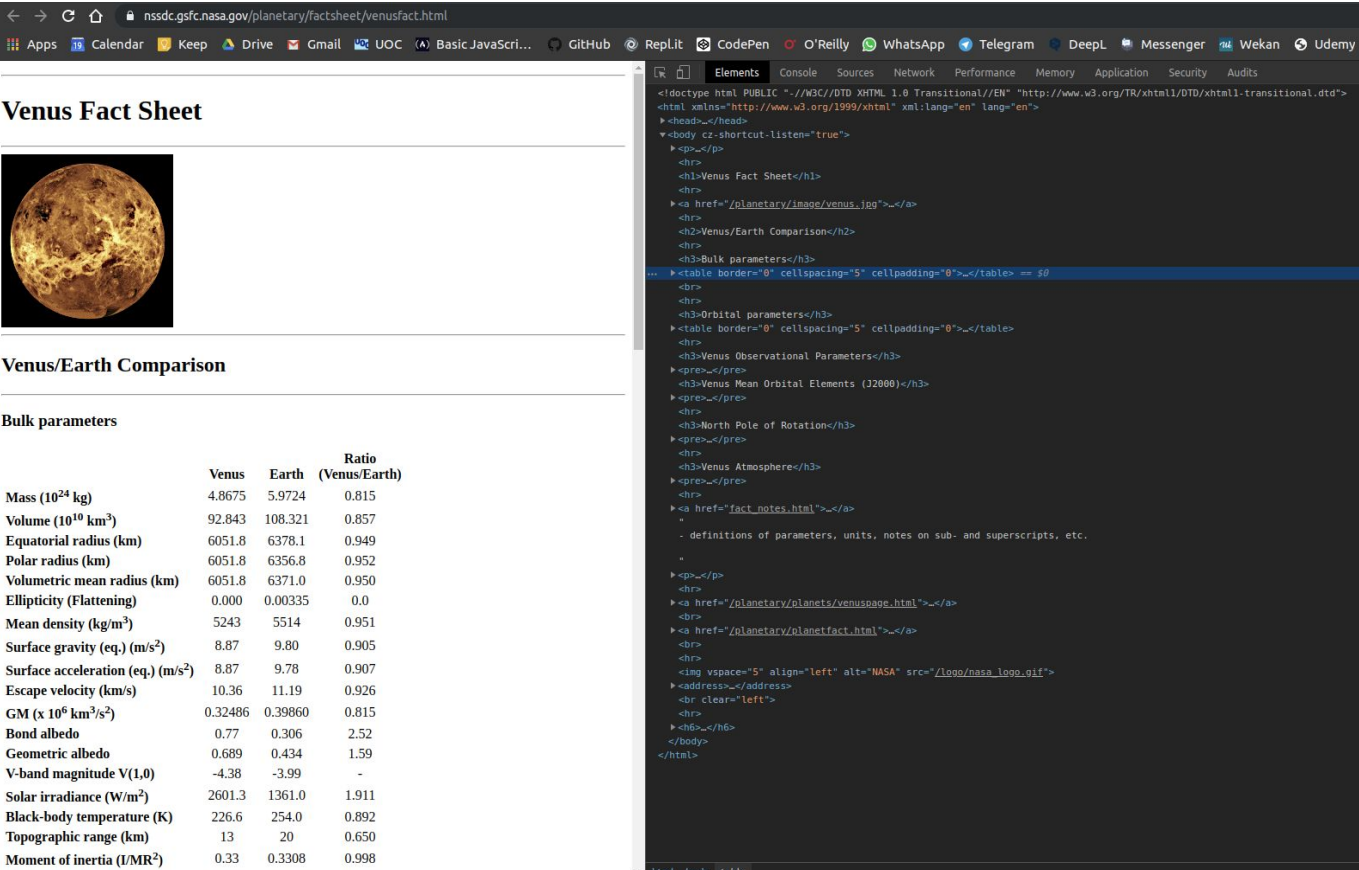
Las tablas generalmente implican estructuras de marcado más complejas que las técnicas de layout. Esto puede hacer que el código sea más difícil de escribir, mantener y depurar.

Las tablas no son responsive tienen un tamaño predeterminado de acuerdo con su contenido, por lo que se necesitan medidas adicionales para que el diseño de la tabla funcione de manera efectiva.

Está por tanto desaconsejado el uso de tablas para maquetar o dar estilos a un documento HTML, esto se realizará mediante CSS.

Podemos dar como válido el siguiente ejemplo sacado de la página web de la NASA:

<https://nssdc.gsfc.nasa.gov/planetary/factsheet/venusfact.html>



The screenshot shows the 'Venus Fact Sheet' page from NASA's NSSDC website. The page features a large image of Venus at the top left. Below it, the 'Venus/Earth Comparison' section contains a table of bulk parameters. The table has four columns: Parameter, Venus, Earth, and Ratio (Venus/Earth). The parameters listed include Mass, Volume, Radii, Density, Gravity, Acceleration, Escape Velocity, GM, Albedo, Temperature, and Moment of Inertia. The right side of the screenshot shows the browser's developer tools, displaying the HTML source code of the page. The code is written in XHTML 1.0 Transitional, which is an older version of HTML. It shows the use of tables for layout and styling, which is discouraged in modern web development. The code also includes various meta tags, links to other pages, and a NASA logo.

	Venus	Earth	Ratio (Venus/Earth)
Mass (10^{24} kg)	4.8675	5.9724	0.815
Volume (10^{10} km ³)	92.843	108.321	0.857
Equatorial radius (km)	6051.8	6378.1	0.949
Polar radius (km)	6051.8	6356.8	0.952
Volumetric mean radius (km)	6051.8	6371.0	0.950
Ellipticity (Flattening)	0.000	0.00335	0.0
Mean density (kg/m ³)	5243	5514	0.951
Surface gravity (eq.) (m/s ²)	8.87	9.80	0.905
Surface acceleration (eq.) (m/s ²)	8.87	9.78	0.907
Escape velocity (km/s)	10.36	11.19	0.926
GM ($\times 10^6$ km ³ /s ²)	0.32486	0.39860	0.815
Bond albedo	0.77	0.306	2.52
Geometric albedo	0.689	0.434	1.59
V-band magnitude V(1,0)	-4.38	-3.99	-
Solar irradiance (W/m ²)	2601.3	1361.0	1.911
Black-body temperature (K)	226.6	254.0	0.892
Topographic range (km)	13	20	0.650
Moment of inertia (I/MR ²)	0.33	0.3308	0.998

En él vemos claramente la exposición de datos multidimensionales y como su uso es el correcto dentro de un estándar en HTML pero que no corresponde a la versión 5. Ya que la versión declarada en el tipo del documento es la 1.0, y carece de las últimas etiquetas semánticas que incorpora HTML 5. Esto es así porque se está utilizando a modo de muestra de datos como documentación.

2. ¿Para qué sirven los atributos colspan y rowspan? ¿Cuándo los usaremos?

- **Colspan:** El atributo colspan define el número de columnas que una celda debe abarcar.

```

<td colspan="number">
<table>
  <tr>
    <th>Mes</th>
    <th>Ahorro</th>
  </tr>
  <tr>
    <td>Enero</td>
    <td>100 €</td>
  </tr>
  <tr>
    <td>Febrero</td>
    <td>80 €</td>
  </tr>
  <tr>
    <td colspan="2">Total: 180 €</td>
  </tr>
</table>

```

- **Rowspan:** El atributo rowspan especifica el número de filas que debe tener una celda.

```

<td rowspan="number">

<table>
<thead>
  <tr>
    <th>Mes</th>
    <th>Ahorro</th>
    <th rowspan="3">Ahorro para verano!</th>
  </tr>
</thead>
<tbody>
  <tr>

```

```

        <td>Enero</td>
        <td>100 €</td>
        <td rowspan="0">100 €</td>
    </tr>
    <tr>
        <td>Febrero</td>
        <td>80 €</td>
    </tr>
</tbody>
</table>

```

3. **¿Para qué sirve la propiedad border-collapse? ¿Qué aplicación tiene en el diseño de una tabla? Pon un ejemplo de su uso que muestre el código HTML y su correspondiente código CSS.**

La propiedad border-collapse en CSS, según el estándar CSS 2.1, nos da la posibilidad de utilizar dos modelos. El primero es el que separa por defecto todas las celdas, es decir, muestra cada celda con cuatro bordes. El segundo combina los bordes de las celdas adyacentes.

Básicamente border-collapse se usa para establecer los bordes de la celda presente dentro de la tabla e indica si estas celdas compartirán un borde común o no.

Ejemplo:

- HTML:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>Border Collapse</title>
    <link href="style.css" rel="stylesheet" type="text/css" />

```

```
</head>
```

```
<body>
```

```
  <h2>
```

```
    border-collapse: Separado
```

```
  </h2>
```

```
  <table id = "separateTable">
```

```
    <tr>
```

```
      <th>Nombre</th>
```

```
      <th>Teléfono</th>
```

```
    </tr>
```

```
    <tr>
```

```
      <td>Pepe</td>
```

```
      <td>923456789</td>
```

```
    </tr>
```

```
    <tr>
```

```
      <td>Juan</td>
```

```
      <td>923456123</td>
```

```
    </tr>
```

```
  </table>
```

```
  <h2>
```

```
    border-collapse: Collapse
```

```
  </h2>
```

```
  <table id = "collapseTable">
```

```
    <tr>
```

```
      <th>Nombre</th>
```

```
      <th>Teléfono</th>
```

```
    </tr>
```

```
    <tr>
```

```
      <td>Pepe</td>
```

```
      <td>923456789</td>
```

```
    </tr>
```

```
    <tr>
```

```
                <td>Juan</td>
                <td>923456123</td>
            </tr>
        </table>
    </body>
</html>
```

- CSS:

```
table, td, th {
    border: 1px solid black;
}
#separateTable {
    border-collapse: separate;
}
#collapseTable {
    border-collapse: collapse;
}
```

Pregunta 2. Sobre la utilización de formularios:

- 1. ¿Qué relación tiene el atributo for del elemento label con el elemento input de un formulario?**

El elemento <input> se utiliza para crear controles interactivos para formularios basados en web con el fin de aceptar datos del usuario.

El elemento <label> representa un título para un elemento en una interfaz de usuario.

Asociar un <label> a un <input> ofrece las siguientes ventajas:

- El texto de la etiqueta no sólo se asocia visualmente con su correspondiente entrada de texto, sino que también se asocia programáticamente con él. Esto

significa que, un lector de pantalla leerá en voz alta la etiqueta cuando el usuario se centre en la entrada del formulario.

- Se puede hacer clic en la etiqueta asociada para enfocar/activar la entrada, así como la propia entrada. Esta mayor área de golpeo proporciona una ventaja a cualquiera que intente activar la entrada, incluyendo dispositivos de pantalla táctil.

Para asociar `<label>` con un `<input>`, es necesario dar al `<input>` un atributo **id**. `<label>` necesita un atributo **for** cuyo valor es el mismo que el del id del `<input>`.

En definitiva el atributo **for** de un `<label>` nos permite asociar dicho label a un `<input>` mediante su atributo **id**.

2. Indica cuatro propiedades CSS que se pueden aplicar para cambiar el aspecto de un input de texto de formulario.

En primer lugar podemos utilizar el selector `input[type=text]`, que nos seleccionará los campos del formulario que nos aceptan texto. Podemos utilizar de forma genérica `input` asociándolo a un selector de clase (`class=""`) o con un id.

Por ejemplo propiedades que podemos cambiar de un input serían:

- El tamaño, la propiedad **width** nos permite cambiar el ancho de los campos del input.

```
input {  
    width: 100px  
}
```

- El **margin** y **padding**, que nos añaden espacio y alrededor del campo del input para hacerlo más espacioso.

```
input[type=text] {  
    width: 80%;
```



```
padding: 15px 22px;
margin: 10px 5px;
box-sizing: border-box;
}
```

- El borde; **border** controla el grosor, el radio del borde, el estilo y el color de los bordes en los formularios CSS, en el siguiente ejemplo se añade un borde grueso morado:

```
input[type=text] {
border: 4px solid #8842d5;
border-radius: 5px;
}
```

- El color de fondo, **background-color** añade un fondo de un color dentro de la caja del input.

```
input[type=text] {
background-color: #8842d5;
color: white;
}
```

- Imagen, con **background-image** podemos añadir una imagen dentro de la caja del input. Con **background-position**, indicaremos la posición para las imágenes de forma correcta y podremos repetirla o no con **background-repeat**.

```
input[type=text] {
background-color: #9476f7;
background-image: url('search-icon.png');
background-position: 9px 10px;
background-repeat: no-repeat;
padding-left: 50px;
}
```

3. Para qué usaremos la pseudo-clase :focus?

La pseudoclase CSS de :focus representa un elemento que ha recibido el foco es decir donde se centra la atención por parte del usuario al seleccionarlo, por ejemplo, mediante el cursor o mediante el tabulador del teclado.

Los formularios CSS y sus campos de entrada se vuelven más interactivos cuando sus propiedades de estilo cambian una vez que los usuarios hacen clic en ellos.

Se debería añadir la pseudoclase :focus para crear un estilo único para el formulario cuando tiene foco.

```
input[type=text]:focus {  
    background-color: #8842d5;  
}
```

Pregunta 3. Sobre el modelo de cajas:

1. El modelo de cajas es la base del diseño web. ¿Qué representa una “caja”?

El box model supone que cada elemento (caja) que encontramos dentro de un documento HTML se encuentra contenido en una caja rectangular, la cual cuenta con una serie de propiedades que afectarán el cómo se muestran los elementos.

El modelo de cajas es el comportamiento de CSS que hace que todos los elementos de las páginas se representan mediante cajas rectangulares, una caja es por lo tanto el espacio donde será representado el elemento que se encuentra en nuestro documento HTML.

Las cajas de las páginas no son visibles a simple vista porque inicialmente no muestran ningún color de fondo ni ningún borde.

Los navegadores crean y colocan las cajas de forma automática, pero CSS permite modificar todas sus características. Cada una de las cajas está formada por el margen, el color de fondo, la imagen de fondo, el padding, el borde y el contenido.

2. ¿Cómo se comportan los elementos de bloque (block) en el flujo normal del documento? ¿Y los elementos en línea (inline)?

Un **elemento de bloque** ocupa todo el espacio de su elemento padre (contenedor), creando así un "bloque". Los navegadores suelen mostrar el elemento de bloque con una nueva línea tanto antes y después del elemento, se visualizan como una pila de cajas.

Un elemento de nivel de bloque siempre comienza en una nueva línea y ocupa todo el ancho disponible. Pueden contener elementos en línea y otros elementos de bloque. Crean estructuras "más grandes" que los elementos en línea.

Algunos ejemplos son: <header>, <article>, <footer>, <main>, <form>, etc...

Los **elementos en línea** son aquellos que sólo ocupan el espacio delimitado por las etiquetas que definen el elemento, en lugar de romper el flujo del contenido.

Un elemento en línea no comienza en una nueva línea y sólo ocupa el ancho necesario. Los elementos de bloque comienzan en líneas nuevas, pero los elementos en línea pueden comenzar en cualquier parte de una línea.

Algunos ejemplos son: <a>, <iframe>, , <label>, etc...

3. ¿Qué diferencia hay entre display:block, display:inline y display:inline-block? Pon ejemplos.

- display:inline: Muestra en la misma línea todos los elementos y **no acepta las propiedades width, height ni márgenes verticales**.

Los elementos en línea son los siguientes: a, abbr, acronym, b, basefont, bdo, big, br, cite, code, dfn, em, font, i, img, input, kbd, label, q, s, samp, select, small, span, strike, strong, sub, sup, textarea, tt, u, var.

```
span.a {
  display: inline; /* the default for span */
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}
```

```
<h2>display: inline</h2>
```

```
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum
consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span
class="a">Aliquam</span> <span class="a">venenatis</span> gravida nisl sit
amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>
```

- display:block: Muestra los elementos en líneas independientes y acepta las propiedades width, height y márgenes verticales.

Los elementos de bloque son: address, blockquote, center, dir, div, dl, fieldset, form, h1, h2, h3, h4, h5, h6, hr, isindex, menu, noframes, noscript, ol, p, pre, table, ul, dd, dt, frameset, li, tbody, td, tfoot, th, thead, tr.

```
span.b {
  display: block;
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}
```

```
<h2>display: block</h2>
```

```
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum  
consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span  
class="b">Aliquam</span> <span class="b">venenatis</span> gravida nisl sit  
amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>
```

- **display:inline-block:** es una mezcla entre los dos anteriores, se muestran en la misma línea (respetando el flujo) todos los elementos y además, acepta las propiedades width, height y márgenes verticales.

```
span.c {  
  display: inline-block;  
  width: 100px;  
  height: 100px;  
  padding: 5px;  
  border: 1px solid blue;  
  background-color: yellow;  
}
```

```
<h2>display: inline-block</h2>
```

```
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum  
consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span  
class="c">Aliquam</span> <span class="c">venenatis</span> gravida nisl sit  
amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>
```

4. ¿Para qué se utiliza la propiedad box-sizing de CSS con el valor border-box? (box-sizing: border-box)

La propiedad box-sizing nos permite definir el ancho y el alto total de un elemento. Sirve para cambiar el modelo de caja por defecto de los navegadores. El ancho de un elemento se altera si se le aplica un borde o un padding. Eso es porque la anchura del elemento que especificamos con CSS, por defecto no incluye borde ni padding. Por defecto se aplica por el navegador con el valor content-box.

Cuando lo utilizamos con border box, hacemos que el ancho especificado sea el equivalente al ancho total.

Esto es muy útil para elementos fluidos, por ejemplo cuando se necesita que el elemento ocupe determinado.

PARTE 2 - EXPLICACIÓN DE LAS ENTIDADES HTML Y CSS UTILIZADAS

Deberás añadir en el documento de texto donde habrás respondido las preguntas de la primera parte una explicación de las entidades HTML y CSS utilizadas:

1 - Sobre las entidades HTML: debe explicarse el uso de etiquetas elegidas, excepto aquellas que especifican la estructura de la página (doctype, html, head y body):

`<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 de cada documento, la compatibilidad con navegadores, el tipo de caracteres soportados, si soporta viewport. Todo esto se utiliza para el posicionamiento en motores de búsqueda.

`<title>`: 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. Además con él importamos las fuentes de google Roboto y Roboto Slab

`<header>`: representa un grupo de ayudas introductorias o de navegación. Hemos definido la barra de navegación y el logotipo del club de fans en todos los documentos. Los divs utilizados son contenedores para modificar los estilos de forma sencilla y genérica para todos los documentos.

<nav>: representa 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.

<main>: utilizamos main para definir los bloques de contenido, debe de ser uno por documento, después se desgranar en <articles> según las necesidades de cada página html.

<section>: representa una sección genérica de un documento. Sirve para determinar qué contenido corresponde a qué parte de un esquema. Lo hemos utilizado en soci.html para fraccionar el <form>, en función de los grupos de información que maneja dicho <form>

<footer>: esta etiqueta hace referencia al pie de página donde figura mi nombre y apellidos como autor de la imagen y el logo de creative commons.

<h1>: utilizamos h1 como elemento de título de primer nivel único para cada <main> de cada documento, donde indicamos el título del mismo.

<h2>: utilizamos h2 como elemento de título para cada sección del documento.

<div>: div es utilizado para generar contenedores o envoltorios para agrupar diversos elementos mediante clases o ids que actuarán como selectores de css, que manejaremos después con el modelo de cajas.

: 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.

`<ul id="nav-list">`: con `ul` hemos creado listas no ordenadas para la navegación y para los subapartados de la lista ordenada de las obras. Con un identificador para el `css`.

`<ol class="list" type="I">`: con `ol` hemos realizado nuestra lista de elementos ordenados, que hacen referencia a la lista del documento `leyes.html`. Con una `class list` para tratar el `css` y e indicando un tipo `"I"` para la numeración romana.

``: `li` nos sirve para determinar cada elemento de la lista o bien ordenada o sin ordenar.

`<a>`: utilizamos `<a>` para linkear a otros sitios web externos o internos dentro de nuestros documentos.

`<table>`: con `table` representamos datos en dos o más dimensiones, creamos la estructura de la tabla.

`<tbody>`: con `tbody` creamos el cuerpo de la tabla.

`<tr class="primary-table">`: con `tr` definimos cada fila de la tabla.

`<th rowspan="2">`: define cada celda como la cabecera de una columna en una tabla.

`<td>`: define la celda de una tabla que contiene datos.

``: con `strong` damos énfasis en partes importantes de cada documento, como las leyes de la robótica en `leyes.html`, o los `"**"` en los `labels` que hacen referencia a `inputs` requeridos.

`<label for="name">`: representa una etiqueta para la interfaz que enlazamos con un `input` mediante los atributos `for` para `label` e `id` para `input`

`<form>`: con `form` declaramos el bloque que contendrá el formulario con sus `labels`, `inputs` y otro tipo de etiquetas de contenido.

`<input type="text" id="name" name="name" required>`: con `input` registramos los datos del formulario que posteriormente serán enviados al servidor y almacenados en una base de

datos. Contamos con atributos de tipo, como texto o fecha, un id asociado al for de un label un name y un required.

<select id="cuote" name="cuote" required>: representa un control que muestra un menú de opciones. Las opciones contenidas en el menú son representadas por elementos <option>, es usado en el formulario contenido en el documento soci.html dos veces para pedir datos al usuario.

<option value="general">General - 5€/mes</option>: lo utilizamos para representar los items dentro del select en el formulario de soci.html.

2 - Sobre el CSS: debe explicarse todo el CSS utilizado, incluyendo cuando se han utilizado y por qué y para qué se aplican los atributos 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;  
  /* setting font default */  
  font-family: "Roboto";  
  font-size: 18px;  
}
```

Aquí hacemos un reset de todos los estilos que puedan ser aplicados por el navegador u otros agentes externos, quitando márgenes, paddings y configurando la fuente predeterminada del documento Roboto y su tamaño a 18px.

```
html, body {  
  height: 100%;  
}
```

En este caso estamos fijando la altura del documento y del cuerpo a 100%.

```
/***** SET COLORS THEME *****/
```

```
:root {  
  --primary: #306291;  
  --primary-dark: rgba(69, 141, 209, 0.5);  
  --primary-white: rgba(69, 141, 209, 0.2);  
  --primary-text: #1f1f1f;  
  --primary-text-white: #666666;  
  --contrast: #d9645d;  
  --white: #ffffff;  
}
```

Mediante custom variables fijamos los colores que utilizaremos durante el resto de la plantilla para no tener que indicar el color de forma individual y así poder reutilizar estas variables.

```
/***** SET FONTS THEME *****/
```

```
h1, h2, h3, h4, h5, h6 {  
  font-family: "Roboto Slab";  
  color: var(--primary);  
}
```

```
h1 {  
  font-size: 40px;  
  line-height: 48px;  
}
```

```
h2 {
```

```
font-size: 24px;
line-height: 32px;
text-transform: uppercase;
}
```

```
footer, article {
font-size: 16px;
}
```

```
.text-home {
font-size: 24px;
}
```

En esta parte definimos que todas las etiquetas h1 a h6 tendran la fuente Robo Slab y el color primario definido previamente. Además h1 por defecto tendrá 40px de tamaño y un interlineado de 48px, mientras que h2 será 24px de tamaño y 32px de interlineado, aprovechamos también para transformarla a mayúsculas. El footer y los articles (contenido), tendrán un tamaño de fuente de 16px y el texto que aparece en el index.html tendrá 24px.

```
/***** SET HEADER *****/
```

```
header {
width: 100%;
}
```

El ancho del header será 100%

```
.logo-container {
max-width: 1200px;
margin-right: auto;
margin-left: auto;
padding: 20px 0px 10px 0px;
}
```

En el contenedor del logo hemos fijado un ancho maximo de 1200px para que cuadre con el contenido del main (que tendrá este ancho máximo), y no se descentre en ningún momento, y junto con los márgenes establecemos el centrado del mismo contenedor o caja.

```
.nav-container {  
  max-width: 1200px;  
  margin-right: auto;  
  margin-left: auto;  
  padding: 10px 0px 20px 0px;  
}
```

En cuanto a la barra de navegación lleva el mismo patrón pero con un padding diferente para amoldar las cajas por si hubiera que customizar cada una de forma independiente.

```
nav>ul>li {  
  display: inline;  
}
```

Con este grupo de selectores hacemos que el display de la lista no ordenada sea de todos los elementos li en linea.

```
nav>ul>li>a {  
  margin: 0px 40px 0px 0px;  
  text-transform: uppercase;  
  text-decoration: none;  
  color: var(--primary);  
}
```

En este punto estamos dotando de estilos primarios a los enlaces de la barra de navegación, transformando a mayúsculas, quitando el subrayado y seteando el color primario por defecto.

```
nav>ul>li>a:hover{  
  color: var(--contrast);  
}
```

En este caso cuando hacemos hover sobre los elementos de la barra de navegación se cambiará al color de contraste.

```
nav>ul>li>.current {  
  color: var(--contrast);  
}
```

Aquí indicamos la ruta seleccionada el menú con una clase current y mediante el color contraste.

```
.line-nav {  
  width: 100%;  
  height: 6px;  
  background: linear-gradient( 90deg, var(--primary) 0%, var(--primary-white) 100%);  
}
```

En esta última parte de la cabecera establecemos un div con una clase line-nav donde creamos una línea divisoria con una altura en píxeles y un background con gradientes que van desde el color primario hasta el color primario claro. Me parecía más adecuado utilizar esta gama de colores puesto que reutilizamos las variables y queda en la misma armonía de colores.

```
/***** SET MAIN *****/
```

```
article {  
  max-width: 1200px;  
}
```

Aquí fijamos cada bloque de contenido genérico dentro del main, con un ancho máximo de 1200px.

```
h2 {  
  margin: 35px 0px 15px 0px;  
}
```

Fijamos un margen para todos los h2 de todos los documentos.

```
article>p {  
  margin: 15px 0px 0px 0px;  
}
```

Fijamos un margen superior de 15px para todos los párrafos de todos los artículos de todos los documentos.

```
.footer-article {  
  padding-top: 10px;  
  padding-bottom: 10px;  
  font-size: 0.8rem;  
}
```

```
.footer-article>a{  
  font-size: 0.9rem;  
}
```

```
.p-it {  
  font-style: italic;  
  font-size: 0.9rem;  
}
```

Estos tres selectores de clase afectan a los footers propios de cada main, donde adaptamos de forma genérica los paddings, las fuentes (que son algo más pequeñas y la estética cursiva).

```
.content-main {  
  padding: 30px 0px 20px 0px;  
}
```

Este padding separa el contenido global del main.

```
.content-article {  
  margin-left: auto;  
  margin-right: auto;  
}
```

Aquí centramos todo el contenido de cada article de cada documento.

```
/* INDEX */
```

```
.background-index {  
  background-image: url(../img/fonsrobot.jpg);  
  background-size: cover;  
  width: 100%;  
  height: calc(100% - (143px + 139px));  
}
```

Dentro del documento index.html, hemos definido un contenedor que hará de fondo utilizando la imagen proporcionada de fondo mediante background-image y con cover cubriremos todo el contenedor. El ancho es del 100% del contenedor y la altura se calcula en función del 100% del documento - el alto del header y el alto del footer con la función calc.

```
.card-wrapper {  
  max-width: 1200px;  
  margin-left: auto;  
  margin-right: auto;  
}
```

Hemos creado un envoltorio para la caja que mostrará la presentación por delante de la imagen. Está fijada a un máximo de 1200px para preservar la posición con respecto a las otras páginas y el contenido prefijado anteriormente y centrado con margin-left y right auto.

```
.container-info-index {  
  background-color: var(--white);  
  border-left: 1px solid var(--primary);  
  border-right: 1px solid var(--primary);
```

```
border-bottom: 1px solid var(--primary);
max-width: 350px;
padding: 30px;
}
```

El contenedor de la informativo tiene un fondo blanco, y todos los bordes menos el superior de un pixel de ancho en color primario. Hemos fijado un ancho máximo de 250px y un padding de 30px

```
.item-index {
  margin: 20px 0px 20px 0px;
}
```

Aquí hemos fijado un margen superior de 20px y uno inferior del mismo tamaño.

```
/* LEYES */
```

```
cite {
  font-style: normal;
}
```

Quitamos la cursiva de la cita referente al título del Círculo Vicioso, porque le damos estilos con una clase creada y reutilizada en el footer.

```
.list-wrapper {
  width: 60%;
  margin: 0 auto;
}
```

Creamos un contenedor de tamaño 60% de ancho respecto del bloque con un margin 0 auto para centrar el mismo.

```
.list {
  padding: 15px 30px 15px 30px;
  display: inline-block;
```



```
background-color: var(--primary-white);
border: 2px solid var(--primary);
margin: 35px 0px 35px 0px;
}
```

Creamos la clase list de la lista ordenada en número romanos, con un padding superior e inferior de 15px e izquierdo y derecho de 30px, fijamos los bordes a 2px del color primario y un fondo de color primario claro. Establecemos además un margen de 35px superior e inferior.

```
.list>li {
  margin-top: 5px;
}
```

Con el selector de elemento de lista fijamos una separación de margen superior de 5px.

```
#wikipedia{
  font-size: 0.8rem;
  font-style: normal;
}
```

Con el selector de id wikipedia, cambiamos el tamaño de la fuente y quitamos la cursiva de la clase footer-article.

```
/* DATOS */
```

```
.data-text {
  padding: 40px 0px 10px 0px;
}
```

Fijamos la separación del h1 y el párrafo de datos.html

```
.table-text {  
  text-transform: uppercase;  
  padding: 10px 0px 10px 0px;  
}
```

Damos estilos de mayúsculas y fijamos el padding del texto que se encuentra encima de la tabla.

```
.table-wrapper {  
  width: 80%;  
  margin: 0 auto;  
  padding: 40px;  
}
```

El ancho del contenedor de la tabla será de un 80%, un margen auto 0 para centrar el contenedor y un padding de 40px para separarlo del resto de contenedores.

```
.table-wrapper>p {  
  text-align: center;  
}
```

Centraremos todos los textos con text-align (del contenedor de la tabla), ya que align center como propiedad de table está obsoleto.

```
.table-wrapper>table {  
  width: 100%;  
  border: 1px solid var(--primary-dark);  
  text-align: center;  
  border-collapse: collapse;  
}
```

Fijamos los estilos de la tabla con un ancho del 100% del contenedor, un borde de 1px de color primario oscuro, centramos el texto y quitamos la separación de los bordes entre celdas con border-collapse: collapse.

```
.primary-table {  
  background-color: var(--primary-dark);  
  color: var(--white);  
  font-weight: bold;  
  text-transform: uppercase;  
}
```

Fijamos de las cabeceras y el lateral izquierdo, con esta clase, el fondo de color primario oscuro, el color del texto en blanco, el ancho de las letras en negrita y lo transformamos a mayúsculas.

```
.secondary-table {  
  background-color: var(--primary-white);  
}
```

Fijamos el fondo en color blanco del resto.

```
th, td {  
  padding: 10px;  
  border: 1px solid var(--primary-dark);  
}
```

Establecemos para todas las celdas un mismo padding dfe 10px y un borde de primario oscuro.

```
.result {  
  color: var(--primary);  
  font-weight: bold;  
}
```

Para los resultados fijamos un color primario para el texto y un ancho en negrita.

```
#total {  
  color: var(--primary-text);  
  font-weight: bold;  
}
```

Para el selector de total, hacemos lo mismo que en el párrafo anterior pero fijando un color blanco.

```
/* SOCI */
```

En el documento soci.html tenemos un formulario que hemos fraccionado en tres sections, para poder diferenciar cada grupo de datos.

```
section {  
  border-radius: 10px;  
  border-color: var(--primary-dark);  
  background-color: var(--primary-white);  
  padding: 50px 50px 40px 50px;  
  width: 60%;  
  margin-top: 40px;  
  margin-bottom: 40px;  
  margin-left: auto;  
  margin-right: auto;  
}
```

Cada section tendrá un 60% de ancho, un radio de 10px, un color de borde primario oscuro, un padding superior, izquierdo y derecho de 50px y uno inferior de 40px. Unos márgenes superior e inferior de 40px y auto para el resto.

```
section>h5{  
  color: var(--black);  
  font-family: var(--roboto);  
  margin-top: -65px;  
  padding-bottom: 20px;  
  font-size: 1.3rem;
```

```
font-weight: bold;
}
```

Cada título irá posicionado en el borde de cada contenedor section con un margen superior negativo, que empuja al elemento fuera de la caja. El resto de propiedades son comunes a las que usamos en el documento css, padding genérico de 10px, tamaño de fuente ajustado de 1.3rem y un ancho de fuente en negrita.

```
.data-title>span {
font-size: 1.3rem;
color: var(--primary);
}
```

Los numeros van con una etiqueta span donde fijamos el tamaño y el color primario.

```
.data-subtitle {
padding-bottom: 20px;
font-size: 0.8rem;
font-style: italic;
color: var(--primary-text-white);
}
```

La especificación de requerido va con un color primario texto claro, en cursiva, 0.8 rem de tamaño y con un padding inferior de 20px.

```
label {
padding: 12px 12px 12px 0;
display: inline-block;
}
```

Cada label del documento va con un padding superior, izquierdo e inferior de 12px y con una disposición en línea de bloque.

A continuación se expone un modelo creado para organizar el contenido de los inputs y los labels de forma coherente, mediante un custom grid creando filas y columnas que puedan adaptarse a los contenedores genéricos anteriormente vistos.

```
.row:after {  
  content: "";  
  display: table;  
  clear: both;  
}
```

En primer lugar con la clase row estamos creando filas por cada label e input, donde el display será table.

```
.col-35 {  
  float: left;  
  width: 35%;  
  margin-top: 6px;  
}
```

Las clases de label, serán lo que se conoce como columnas que en este caso hemos fijado un ancho del 35% y flotantes a la izquierda.

```
.col-65 {  
  float: left;  
  width: 65%;  
  margin-top: 6px;  
}
```

Las clases de input, serán lo que se conoce como columnas que en este caso hemos fijado un ancho del 65% y flotantes a la izquierda.

```
.card-text {  
  margin: 20px 0px 15px 0px;  
  padding: 30px 0px 0px 20px;
```

```
width: 70%;  
border: 1px solid var(--primary-dark);  
}
```

Para la tarjeta de los radio buttons, hemos creado un contenedor del 70% con un borde de color primario oscuro y con unos márgenes y paddings de la misma manera que el resto de cajas.

```
#peri {  
  margin-top: -42px;  
  margin-left: 5px;  
}
```

Con el selector de id hemos colocado el título con un margen negativo en el borde de la caja anterior.

```
.submit {  
  margin-top: 20px;  
  text-align: center;  
}
```

Con esta clase separamos el contenedor del botón y centramos el mismo en el section correspondiente.

```
input[type=text], input[type=date], select {  
  width: 100%;  
  padding: 10px;  
  background-color: var(--white);  
  border: 1px solid var(--primary-dark);  
  border-radius: 4px;  
  resize: vertical;  
}
```

Con este selector, indicamos que todos los inputs de tipo texto, fecha y los selects del formulario tendrán un ancho del 100%, un padding de 10px, un fondo blanco, un borde

primario oscuro, un radio de 4px y un resize que nos permite controlar el cambio vertical de los elementos.

```
input[type=text]:focus, input[type=date]:focus, select:focus {  
  border: 1px solid var(--contrast);  
}
```

Con este selector indicaremos que cuando el input esté en el foco cambie el borde a contraste.

```
input[type=submit] {  
  background-color: var(--contrast);  
  color: var(--white);  
  padding: 12px 80px;  
  border: none;  
  border-radius: 4px;  
  cursor: pointer;  
  text-transform: uppercase;  
  font-size: 22px;  
}
```

Mediante el selector de input tipo submit indicamos que el botón de envío tendrá un fondo contraste, un color blanco, ningún borde, con un borde radio de 4px, activamos cursor con cursor pointer, letras mayúsculas y tamaño especificado a 22px.

```
/****** SET FOOTER *****/
```

```
footer {  
  width: 100%;  
  background-color: var(--primary-white);  
}
```

Fijamos un 100% de ancho para el footer con un fondo en color claro primario.


```
footer>span {  
  padding: 15px;  
  text-align: center;  
}
```

Establecemos una separación con el padding por cada span.+

```
.footer-item {  
  display: block;  
  text-align: center;  
  margin-left: auto;  
  margin-right: auto;  
  padding: 10px;  
  color: var(--primary-text-white);  
}
```

Con esta clase establecemos que el display será por bloques, texto centrado, color primario claro de texto.

```
.line-footer {  
  width: 100%;  
  height: 6px;  
  background: linear-gradient( 90deg, var(--primary-white) 0%, var(--primary) 100%);  
}
```

Definimos la línea divisoria de la misma manera que la de la cabecera pero invirtiendo los colores del gradiente.

```
.author {  
  color: var(--primary);  
  border-bottom: 2px solid var(--primary);  
  text-decoration: none;  
  font-style: normal;  
  font-size: 0.9rem;  
}
```

Con esta clase damos estilo a los enlaces, en color primario con un subrayado de 2px, sin decoración de enlace, sin estilo de fuente y con un tamaño de 0.9rem.

```
.author:hover {  
  color: var(--contrast);  
  border-bottom: 2px solid var(--contrast);  
}
```

Damos estilo a los hover de cada enlace de la clase anterior, con un color contraste en el texto y en el subrayado.

Validación de las páginas.

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

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and

Showing results for datos.html

Checker Input

Show ☐ source ☐ outline ☐ image report

Check by

file upload ▾

Choose File

No file chosen

Uploaded files with .xhtml or .xht extensions are parsed using

Document checking completed. No errors or warnings to show.

Used the HTML parser.

Total execution time 6 milliseconds.

[About this checker](#) • [Report an issue](#) • Version: 19.11.19

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and

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

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: 19.11.21

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and it

Showing results for leyes.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

Document checking completed. No errors or warnings to show.

Used the HTML parser.

Total execution time 13 milliseconds.

[About this checker](#) • [Report an issue](#) • Version: 19.11.22

Nu Html Checker

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

Showing results for soci.html

Checker Input

Show ☐ source ☐ outline ☐ image report

Options...

Check by

file upload

Choose File

 No file chosen

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

Check

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

Message Filtering

1.

Warning

 The

date

 input type is not supported in all browsers. Please be sure to test, and consider using a polyfill.
- From line 94, column 33; to line 94, column 83
- <input type="date" id="birthdate" name="birthdate">

Document checking completed.

Used the HTML parser.

Total execution time 12 milliseconds.

[About this checker](#) • [Report an issue](#) • Version: 19.11.22

19	:root	Error de análisis sintáctico	--primary: #306291;
20	:root	Error de análisis sintáctico	--primary-dark: rgba(69, 141, 209, 0.5);
21	:root	Error de análisis sintáctico	--primary-white: rgba(69, 141, 209, 0.2);
22	:root	Error de análisis sintáctico	--primary-text: #1f1f1f;
23	:root	Error de análisis sintáctico	--primary-text-white: #666666;
24	:root	Error de análisis sintáctico	--contrast: #d9645d;
25	:root	Error de análisis sintáctico	--white: #ffffff;
26	:root	Error de análisis sintáctico	--black:#000000;
27	:root	Error de análisis sintáctico	--roboto: font-family: "Roboto";
28	:root	Error de análisis sintáctico	}
34	h1, h2, h3, h4, h5, h6	Propiedad no válida : color Error de análisis sintáctico	var(--primary)
84	nav > ul > li > a	Propiedad no válida : color Error de análisis sintáctico	var(--primary)
88	nav > ul > li > a: hover	Propiedad no válida : color Error de análisis sintáctico	var(--contrast)
92	nav > ul > li > .current	Propiedad no válida : color Error de análisis sintáctico	var(--contrast)
98	.line-nav	Propiedad no válida : background Error de análisis sintáctico	var(--primary) 0%, var(--primary-white) 100%
155	.container-info-index	Propiedad no válida : background-color Error de análisis sintáctico	var(--white)
156	.container-info-index	Propiedad no válida : border-left Error de análisis sintáctico	var(--primary)
157	.container-info-index	Propiedad no válida : border-right Error de análisis sintáctico	var(--primary)
158	.container-info-index	Propiedad no válida : border-bottom Error de análisis sintáctico	var(--primary)
181	.list	Propiedad no válida : background-color Error de análisis sintáctico	var(--primary-white)
182	.list	Propiedad no válida : border Error de análisis sintáctico	var(--primary)
218	.table-wrapper > table	Propiedad no válida : border Error de análisis sintáctico	var(--primary-dark)
224	.primary-table	Propiedad no válida : background-color Error de análisis sintáctico	var(--primary-dark)
225	.primary-table	Propiedad no válida : color Error de análisis sintáctico	var(--white)
231	.secondary-table	Propiedad no válida : background-color Error de análisis sintáctico	var(--primary-white)
236	th, td	Propiedad no válida : border Error de análisis sintáctico	var(--primary-dark)
240	.result	Propiedad no válida : color Error de análisis sintáctico	var(--primary)
245	#total	Propiedad no válida : color Error de análisis sintáctico	var(--primary-text)
253	section	Propiedad no válida : border-color Error de análisis sintáctico	var(--primary-dark)
254	section	Propiedad no válida : background-color Error de análisis sintáctico	var(--primary-white)