1.3.- Contenido 3: Construir una página web utilizando estilos CSS para la definición de aspectos visuales de la interfaz web

Objetivo de la jornada

1. Modifica el estilo del contenido utilizando propiedades de CSS para la definición de aspectos visuales de la interfaz web.
2. Modifica el posicionamiento de elementos utilizando CSS para la definición de aspectos visuales de la interfaz web.
3. Describe el orden jerárquico de reglas de CSS para la definición de aspectos visuales de la interfaz web.
4. Describe el border-box-model reconociendo sus propiedades modificables para la definición de aspectos visuales de la interfaz web.

1.3.1.- Aplicando CSS

1.3.1.1. Qué es CCS

La intención principal detrás de las hojas de estilo en cascada CSS (siglas en inglés de Cascading Stylesheets) es separar el código a través del cual se definen los contenidos de una página web, del que se escribe para su presentación o aspecto visual. Tal como hemos visto en secciones anteriores, el lenguaje HTML se utiliza para estructurar el contenido de nuestra página web desde el punto de vista semántico (títulos, subtítulos, texto, etc.). CSS es la tecnología que usamos para la estética del mismo o, más concretamente, cómo deben ser desplegados cada uno de los elementos de un documento HTML. Este principio es fundamental por muchos motivos. Un ejemplo claro es el "diseño adaptativo" (responsive design): poder adaptar el mismo contenido a diferentes dispositivos. Es decir, que una misma página web se puede visualizar de una manera diferente en un PC que un móvil, optimizada para cada caso[1].

Desde los inicios del W3C, el concepto y la importancia de separar contenidos de estilos fue determinando la confección de CSS. Éste ha pasado por etapas o versiones que han sufrido actualizaciones como en el caso de otros estándares.

La última versión anterior a CSS3, que es el que actualmente utilizamos, fue CSS2.1, una revisión de la especificación CSS2 original de 1997. Muchos se sorprenden al saber que CSS2 sólo se transformó en una recomendación oficial del W3C en 2011. Además, CSS3 comenzó a discutirse en 1998, un año luego de la aparición de CSS2. CSS2.1 fue una revisión que W3C realizó en base a la forma real en cómo había sido

implementado en realidad en el mercado, que había evolucionado a un estado de muchas inconsistencias entre navegadores en términos de la implementación de CSS2. La creciente carrera de competencia entre navegadores de internet en la última década a favorecido la estandarización de CSS3 que anteriormente, cuando Internet Explorer dominaba el mercado con su implementación CSS2.1 y se resistía a la introducción de CSS3, no había tenido oportunidad de florecer.
CSS3 está estandarizado en documentos individuales que siguen su propio camino y han dado una mejor dinámica al estandar que un solo documento centralizado como fue el caso de CSS2. En la actualidad, la especificación CSS3 es un estandar de la industria en todos los navegadores, posee una gran comunidad de desarrolladores construyendo aplicaciones que éste[2].

Las principales características que presenta CSS3 como mejoras de las versiones anteriores son[3]:

- Soporte de funciones orientadas a diseño responsivo.
- Cógido modular.
- Bordes redondeados.
- Bordes para imágenes.
- Sombras de texto.
- Imágenes de fondo múltiples.
- Mayor velocidad de carga.
- Transformaciones 2D/3D para animaciones y transiciones nativas sin necesidad de Flash o Javascript.
- Nuevos colores y efectos de imagen.

1.3.1.2. Los elementos básicos CSS: reglas, selectores y propiedades

Las hojas de estilo constan, normalmente, de una serie de instrucciones de estilo que definen cómo se han de representar determinados elementos de la página. Los elementos de página se identifican con las etiquetas HTML, identificadores, clases y otros selectores.

Una instrucción o regla CSS tiene la siguiente estructura:

Fuente: https://www.tutorialrepublic.com/css-tutorial/css-syntax.php

Estas instrucciones o reglas nos sirven para definir son en el fondo las definiciones de visualización estética de nuestro documento. Estamos familiarizados a realizar este tipo de definiciones en los típicos procesadores de texto, como Libre Office Write o Microsoft Word, a través de sólo algunos clics. La diferencia es que en este caso lo hacemos de manera literal y en un lenguaje especial (CSS) con el que es posible dar instrucciones similares a la máquina.

A través de estas reglas podemos definir cosas como las siguientes:

- "El título principal de cada página (<h1>) debe estar centrado, ser de color gris oscuro y usar la fuente Montserrat con un tamaño de 40 pixeles".
- "Los subtítulos de cada página de nivel dos (<h2>) deben estar alineados a la izquierda, color negro ligeramente más claro y usar también el tipo de letra Montserrat, pero con un tamaño de 32 pixeles".
- "Las imágenes dentro de la columna principal deben expandirse siempre al ancho máximo de dicha columna."

Estos son ejemplos a modo ilustrativo y pasamos a continuación a traducir las instrucciones anteriores a lenguaje CSS, las que tendrían el siguiente contenido[1]:

```
h1 {
    font-family: Montserrat;
    font-size: 40px;
    color: #333;
    text-align: center;
}

h2 {
    font-family: Montserrat;
    font-size: 32px;
    color: #444;
}

div.contenido img {
    width: 100%;
}
```

REGLAS

Este extracto de hoja de estilos se divide en 3 bloques diferenciados, agrupados en llaves "{…}".

Cada uno de estos grupos es una regla e implementa una directiva concreta de la pequeña lista que expusimos de una manera redactada un poco más arriba.

SELECTORES

Para especificar a qué partes de la página HTML se aplica cada regla en cuestión, le precede un selector. Este selector especifica el ámbito de aplicación de la regla. Recordemos que un documento HTML está organizado de manera jerárquica a modo de árbol. Este ámbito de aplicación van a ser una o varias ramas de ese árbol.

En las dos primeras reglas (selectores "h1" y "h2") el ámbito de aplicación es muy amplio porque los selectores son etiquetas HTML. Es decir, estamos diciendo que la primera regla se aplique a todos los elementos que utilicen la etiqueta <h1> y que la segunda se aplique a todos elementos <h2>.

El último selector es más específico y nos dice que la tercera regla se aplica solamente a elementos <img> que sean hijos de un elemento . Esto se hace con la posición de los elementos, es decir, al poner "img" a la derecha de "div" estamos diciendo que img debe tener un como padre. Pero, además, con el sufijo de ".contenido" estamos definiendo además que el "div" no puede ser cualquiera sino que debe pertenecer a la clase "contenido". Esto es útil para diferenciar distintos bloques , por ejemplo usarlo para diferenciar la columna principal de contenido de otra columna para una barra lateral.

Podríamos haber hecho lo mismo con cualquier otro elemento HTML. En general, las clases (atributo "class" en HTML) son muy útiles para tratar los mismos elementos HTML de una manera diferenciada según su lugar y función en la página. Otro atributo comúnmente utilizado es "id", con el cuál identificamos el elemento en particular, por ejemplo si tenemos un , el "id" nos serviría para referenciar específicamente ese div en particular. En este caso en lugar de utilizar ".contenido" en el selector se usa "#contenido_1", donde "#" hace referencia a que estamos refiriéndonos a un "id".

PROPIEDADES

Tal como ya hemos mencionado, una regla puede tener una serie de declaraciones internas en una sintaxis **propiedad:valor**, por ejemplo, **font-size: 40px;** que establece que a los elementos que caen dentro del ámbito de la regla, según el selector, deben poseer un tamaño de fuente de 40 pixeles. Los elementos deben ser por tanto elementos tipo texto, es decir, elementos <h1>, <h2>, <p>, u otro que podemos encontrar en la sección donde anteriormente hemos tratado las etiquetas disponibles para texto en HTML. De todas formas, en caso que exista algún elemento dentro del ámbito de la regla que no sea tipo texto, la propiedad será ignorada.

1.3.1.3. Utilizar CSS en el archivo HTML y en un archivo externo

Como hemos dicho, el uso de CSS nos permite aplicar estilos a páginas web para que se vean exactamente con el estilo visual que deseamos. Esto es posible porque CSS está conectado al DOM para que pueda rediseñar rápida y fácilmente cualquier elemento. Por ejemplo, si no deseamos utilizar el aspecto predeterminado de <h1>, <h2> y otras etiquetas de encabezado, podemos asignar nuevos estilos para anular la configuración predeterminada de la familia de fuentes y el tamaño utilizado, o en negrita o cursiva debe establecerse, además de muchas otras propiedades.
Hemos creado archivos HTML en secciones anteriores de este módulo, donde incluso hemos de manera básica el uso de CSS con un elemento <style> dentro del elemento <head> del documento HTML.
En la práctica, existen diversas formas de incluir el código CSS en nuestro desarrollo y a continuación los revisaremos, partiendo por la forma que ya hemos utilizado[4].

Como elemento <style> dentro del mismo archivo HTML (Estilos Incrustados)
La primera forma que revisaremos para aplicar estilos a nuestra página web, es agregar el código CSS requerido dentro de el elemento <head> de nuestro código HTML. Por lo tanto, como hemos aplicado en códigos previos del curso, si quisiéramos cambiar el estilo de todos los elementos de etiqueta <h1> que estuvieran presentes en la página, utilizaríamos el siguiente código dentro de <head>:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta name='robots' content='index, follow'>
    <title>The style tag</title>
    <style>
      h1 {
        color:olive;
        font-size :36pt;
        font-family:'Times New Roman';
        font-style :italic;
      }
    </style>
  </head>
  <body>
    <h1>Éste es un Encabezado de Tipo 1</h1>
  </body>
</html>
```

Importando una hoja de estilos

Cuando deseamos aplicar estilos a un sitio web completo y no solamente a una página web individual, una mejor forma de administrar hojas de estilo es extraerlas completamente del código HTML y colocarlas en archivos separados para luego importar los que se requieren en las diversas páginas. Esto permite usar diferentes hojas de estilo para distintos layouts (Como web o impresión) sin necesidad de alterar el código HTML.

1. @import

Una forma de llevar a cabo lo descrito es utilizando la directiva **@import** de CSS como se muestra a continuación:

```
<style>
  @import url('styles.css');
</style>
```

De esta forma instruimos al navegador a obtener la hoja de estilo de nombre **styles.css**. El comando @import entrega la flexibilidad de crear hojas de estilo que a su vez importan otras hojas de estilos. Es importante enfatizar que las etiquetas <style> no deben estar incluidas dentro de ningún archivo CSS.

2. Elemento <link>

También podemos incluir una hoja de estilos CSS con la etiqueta <link> de HTML como sigue:

```
<link rel='stylesheet' type='text/css'
href='styles.css'>
```

Este método tiene el mismo efecto que la directiva @import, con la excepción que <link> es una etiqueta sólo utilizable en HTML, por lo que no puede ser utilizada desde dentro de un archivo CSS. Nótese además que <link> no debe ir incluido dentro de etiquetas <style>…</style>.

Como atributo style de un elemento HTML (Estilo Inline)

Es posible definir o sobreescribir ciertos estilos para la página web de manera caso a caso, insertando atributos de estilo directamente en un elemento HTML, tal como se muestra a continuación:

```
<h1 style='font-style:italic; color:blue;'>Hello there</h1>
```

Sin embargo, este método debe ser excepcional dado que quiebra la separación entre contenido y presentación, que es una de las ventajas principales de la definición de estilos.

1.3.1.4. Modificar estilos

Como vimos más arriba, el manejo de estilos se realiza a través de un listado de instrucciones o reglas, compuestas por selectores, declaraciones y pares propiedad:valor.

En la sección 1.3.1.2, vimos algunos ejemplos ilustrativos de reglas CSS de acuerdo a algunos objetivos básicos de diseño. En los mencionados utilizamos algunos selectores simples del tipo etiqueta HTML, tales como <h1>, <h2> o . De la misma forma incluimos un conjunto muy limitado de propiedades. En la práctica existen múltiples formas de especificar un selector, así como también una infinidad de propiedades CSS.
A continuación haremos un resumen de herramientas imprescindibles que serán suficientes para llevar a cabo una infinidad de personalizaciones de presentación de contenidos en páginas web que desarrollemos[4].

SELECTORES

Existen una gran variedad de selectores. Tal como hemos visto hasta el momento, es posible seleccionar por tipo de elemento HTML, tal como <p>, , <img>, etc. Sin embargo, además podemos seleccionar elementos bajo otros criterios, tales como: Elementos que estén contenidos dentro de otros (Descendientes), elementos que sean hijos directos y por el método muy usado de atributo de elementos HTML: “id”, “class”, “target” y otros. CSS está tan bien pensado que para la selección de lo que queramos personalizar, y su relación con cualquier otro elemento, podemos encontrar los selectores para los resultados deseados.
Nota: No nos referiremos mayormente a las propiedades mostradas en estos ejemplos de selectores.

1. **Selector por tipo de elemento HTML.**
   Hemos visto varios ejemplos de este tipo, y consiste específicamente en indicar el tipo de elemento HTML según su etiqueta. Por ejemplo:

```
p { text-align:justify; }
img { border:1px solid #444; }
h1 { font-size:26pt; }
```

2. **Selector por descendencia.**

Los selectores por descendencia permiten aplicar estilos a elementos que estén contenidos dentro de otro elemento. Por ejemplo, las siguientes reglas establecen que: 1) El color de todos los textos dentro de los elementos sean de color rojo, pero solamente si la aparición de esos textos ocurren dentro de un elemento (Por ejemplo: ** HolaMundo **), y 2) Que el color del texto será azul cuando esté en negrita, siempre que esté dentro de un elemento de lista, y siempre que además esté dentro de una lista sin orden:

```
p b { color:red; }
ul li b { color:blue; }
```

3. **Selector por calidad de hijo.**

Especifica elementos que sólo sean hijos directos de otro elemento. Si consideramos el ejemplo anterior, de texto rojo en negrita, podemos acotarlo especificando que sólo aplicará la regla si es hijo directo del elemento , Por lo tanto, la regla aplicará en este caso: ** HolaMundo **, pero no en ** HolaMundo **: La sintaxis para esto es la siguiente:

```
p > b { color:red; }
```

4. **Selector por ID.**

En caso de existir un elemento HTML con un id determinado, como por ejemplo: ** **, éste puede ser seleccionado directamente desde CSS a través del símbolo # de la siguiente forma, para cambiar el texto a itálico:

```
#mydiv { font-style:italic; }
```

Las ids son para trabajar con secciones de un documento que solo aparecerán una sóla vez. Esto dado que corresponde a un atributo único dentro del DOM.

Es posible acotar el alcance del selector por id, por ejemplo, para que aplique sólo si el elemento es del tipo , con **p#mydiv**. Esto podría ser confuso a primera vista, dado que id es un atributo único. Sin embargo, en sitios dinámicos

esto puede ser útil, cuando el id puede eventualmente ser asignado a distintos tipos de elementos dependiendo de alguna condición al momento de construir la página en el servidor.

5. Selector por Clase.

Cuando existe un número de elementos en una página que se desean personalizar con los mismos estilos, puede asignárseles un nombre de clase, por ejemplo: . Luego puede crearse una única regla que modifique todos estos elementos a la vez. La siguiente regla crea un margen izquierdo de 10 pixeles para todos los elementos de la clase “myclass” que es precedida de “.” como sintaxis que alude a una clase.

```
.myclass { margin-left:10px; }
```

El igual que el selector por id, podemos acotar el alcance de la regla a un tipo de elemento, por ejemplo sólo a elementos del tipo <p> que pertenezcan a esa clase, **con p.myclass**.

6. **Selector por Atributo.**

El selector por atributo nos permite seleccionar los elementos que contengan cierto valor asignado a alguno de sus atributos en el DOM. Por ejemplo, todos los elementos <input type=”submit”> del documento serían seleccionados con:

```
[type='submit'] { width:100px; }
```

También es posible acotar la selección, por ejemplo a elementos como este, pero que sólo estén dentro de un formulario, a través de: **form [type='submit']**.

Este tipo de selector también funciona con los atributos id y class, sin embargo el uso más común es a través de “#” y “.”, respectivamente, según se vio más arriba.

7. Selector Universal.

Existe el selector comodín * que significa seleccionar cualquier elemento. Su uso aislado provoca resultados que no son de mucha utilidad, como aplicar los estilos a todos los elementos de un documento. Sin embargo, combinándolo con otros selectores puede entregar opciones interesantes. Por ejemplo, seleccionar todos los elementos <p> dentro de elementos con id=”boxout”, pero sólo si no son descendientes director de <p>. Esto lo logramos con la siguiente instrucción o regla:

```
#boxout * p {border:1px solid green; }
```

8. **Selector por Grupo.**
Los selectores pueden agruparse utilizando “,” de la siguiente forma:

```
p, #idname, .classname {
    border-bottom:1px dotted orange;
}
```

En este caso todos los elementos del tipo <p>, los que tengan id=”idname”, y los que pertenezcan a la clase class=”classname” serán seleccionados para aplicarles los estilos especificados en la regla.

PROPIEDADES
Es dificultoso recordar y tener en mente permanentemente el gran número de propiedades de CSS3. Hay referencias online muy útiles para consultar este tipo de información, tal como la Referencia CSS provista por Mozilla Developers Network (MDN) https://developer.mozilla.org/es/docs/Web/CSS/Referencia_CSS. La buena noticia es que no hace falta usar permanentemente una gran parte de estas propiedades. En general se cumple la Ley de Pareto del 20/80. Sabiendo una parte menor de ellas se puede lograr una habilidad suficiente para desempeñarse en el mundo del diseño Web. La documentación oficial, libros, tutoriales online y comunidades de desarrolladores siempre estarán disponibles para consultas sobre aplicaciones más específicas. De esta forma, la experiencia de uso logrará la expansión del conocimiento en la materia.

A continuación revisamos un listado de algunas propiedades que son de uso habitual y será de utilidad familiarizarse con ellas:

1. **Textos.**
    - **font-family:** tipo de letra (nombre del tipo => “Montserrat”, “Open Sans”, etc.).
    - **font-size:** tamaño de letra.
    - **font-weight:** peso (normal, negrita, ...).
    - **font-style:** estilo (normal, cursiva, ...).
    - **letter-spacing:** espacio entre letras.
    - **line-height:** espacio entre líneas / altura de la línea.
    - **text-decoration:** cosas como subrayados, tachados, etc.
    - **text-align:** alineación del texto (left, center, right).

    En relación a tamaño de fuentes, éstos pueden estar especificados como dimensiones absolutas o relativas al contexto de la página. Las siguientes tablas son de utilidad para saber en qué casos recurrir a uno u otro modo.

Generalmente en diseños responsivos se utilizan tamaños relativos de fuentes en los textos de la página.

Unidades Absolutas para Tamaño de Fuentes

Unidad	Descripción
in	Pulgadas– 1in equivale a 2.54cm.
cm	Centímetros
mm	Milímetros
pt	Puntos – En CSS, un punto está definido como 1/72 pulgadas (0.353mm).
pc	picas – 1pc equivale a 12pt.
px	pixel – 1px equivale a 0.75pt.

Unidades Relativas para Tamaño de Fuentes

Unidad	Descripción
em	Relativa al tamaño de la fuente del elemento (2em significa 2 veces el tamaño de la fuente actual)
ex	Relativa a la altura de la fuente actual
ch	Relativa al ancho del “0” (Cero)
rem	Relativa al tamaño de la fuente del elemento raíz
vw	Relativa al 1% del ancho del viewport
vh	Relativa al 1% de la altura del viewport
vmin	Relativa al 1% de la menor dimensión del viewport
vmax	Relativa al 1% da la mayor dimensión del viewport
%	Relativa al elemento de jerarquía superior

2. **Colores y Fondos.**
    - **color:** color del elemento. Es importante notar que existen diferentes formatos para especificar el color como palabras predefinidas (“red”, “green”, etc.), formato RGB o valor hexadecimales.
    - **background-color:** color del fondo del elemento.
    - **background-image:** usar una imagen de fondo.
    - **background-repeat:** usar una imagen de fondo como mosaico. Permite diferentes modos de organización de la imagen (ver detalles en material de referencia).
    - **opacity:** opacidad del elemento. Va desde 0 (completamente transparente) hasta 1 (sólido). Un valor de 0.5 sería, por tanto, un nivel de transparencia del 50%.

3. **Bordes**
    - **border:** añade un borde a un elemento y especifica sus propiedades (grosor, estilo de línea, etc.). Ver también el modelo de caja de arriba.
    - **border-color:** color del borde.

- **border-style:** diferentes estilos (sólido, rayitas, puntos, etc.).
- **border-radius:** redondear las esquinas de un elemento.

4. **Otras**
    - **float:** propiedad avanzada que permite una maquetación más sofisticada. Posicionar los elementos de manera "flotante". Ver este tutorial básico sobre posicionamiento flotante.
    - **clear:** controla el comportamiento de los elementos adyacentes a elementos posicionados de forma flotante. Ver también el tutorial anterior sobre posicionamiento flotante.
    - **overflow:** controla el comportamiento de los contenidos que no caben en su elemento contenedor (valores: visible, hidden, scroll, auto, inherit)
    - **display:** controla diferentes aspectos de la visualización de un elemento, permite incluso ocultarlo con el valor "none".
    - **list-style-image:** URL con una imagen que se debe usar como viñeta.
    - **list-style-type:** diferentes estilos de viñetas y numeración para los elementos de la lista.
    - **box-shadow:** aplicar un efecto de sombra a un elemento.

1.3.1.5. Orden jerárquico de CSS.

Dado que las instrucciones de estilo se pueden vincular y combinar de muchas maneras distintas, hay que determinar el procedimiento a seguir en caso de producirse instrucciones contradictorias o conflictivas. Por ejemplo, si tenemos varias instrucciones asignando diferentes valores a una misma propiedad, debe tenerse claridad de cuál tiene mayor prioridad[5].

Se le llama **Cascada** al sistema jerárquico que otorga a las instrucciones de estilo una mayor o menor importancia en función de su procedencia. Es un sistema de regulación para evitar incongruencias entre diferentes hojas de estilo.

A continuación se lista, de menor a mayor prioridad jerárquica, los estilos presentes en un desarrollo web con CSS:

- **Estilos del Navegador:** Cada navegador dispone de su propia hoja de estilo estándar (Browser-Style) que es la que utiliza para representar las páginas HTML. El tipo de letra, el color de fondo y el de la fuente, el color de los vínculos, el tamaño de los títulos, etc., los formatea el navegador.
- **Estilos de Usuario:** Los navegadores modernos permiten al usuario crear sus propias hojas de estilo (hojas de estilo personalizadas o userstyles). El usuario

puede, por tanto, configurar sus preferencias en el navegador y sobrescribir con ellas los ajustes de la hoja de estilo del navegador.

- **Estilos de Autor (Desarrollador):** Las hojas de estilo creadas por nosotros como desarrolladores (O de terceros), sobrescriben, por regla general, diferentes instrucciones tanto del navegador como del usuario (siempre que existan); las hojas de estilo se combinan. Si importamos varias hojas de estilo de autor a nuestra página, en caso de conflicto siempre serán las instrucciones de la última hoja de estilo cargada las que tengan preferencia. Su hoja de estilo personalizada, esto es, la hoja de estilo de autor, no puede ser sobrescrita por la del usuario, ya que la hoja de estilo de autor tiene mayor prioridad. Esta prioridad jerárquica sólo puede cambiarse mediante la regla !important.

  ORDEN DE CARGA DE ARCHIVOS CSS Y PRIORIDADES
  Las Hojas de Estilo de Autor importadas a través de @import o con <link> en el archivo HTML, se cargan secuencialmente y, en caso de haber propiedades duplicadas para mismos elementos, los archivos CSS cargados en último lugar sobreescribirán los anteriores en aquellas propiedades, a no ser que existan criterios, por ejemplo !important, que lo eviten.

- **Estilos Incrustados:** Por otro lado, en el área <head> del documento se pueden incluir instrucciones de Estilos Incrustados. Las instrucciones incrustadas sobrescriben las instrucciones externas o importadas, pero sólo en el documento actual (esto es, el documento en que están incrustadas las instrucciones).
- **Estilos Inline:** En la etiqueta HTML también podrían aparecer directamente (Inline) instrucciones de estilo dentro del atributo **style** del elemento. Esas instrucciones tienen entonces prioridad ante todas las demás instrucciones, aunque sólo en la instancia actual (Esto es, en el elemento HTML en que se han escrito las instrucciones).

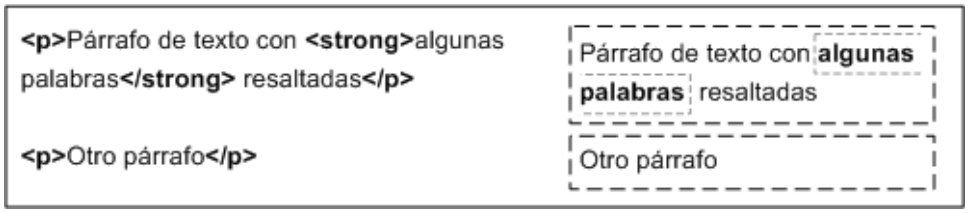
El uso de la regla **!important** y el cálculo de **Specificity** que utiliza el navegador, corresponden a conceptos más avanzados, que invitamos al lector revisar en la bibliografía.

1.3.1.6. El Modelo de Caja (Box Model).

El modelo de cajas o "box model" es seguramente la característica más importante del lenguaje de hojas de estilos CSS, ya que condiciona el diseño de todas las páginas web. El modelo de cajas es el comportamiento de CSS que hace que todos los elementos de las páginas se representen mediante cajas rectangulares.

Las cajas de una página se crean automáticamente. Cada vez que se inserta una etiqueta HTML, se crea una nueva caja rectangular que encierra los contenidos de

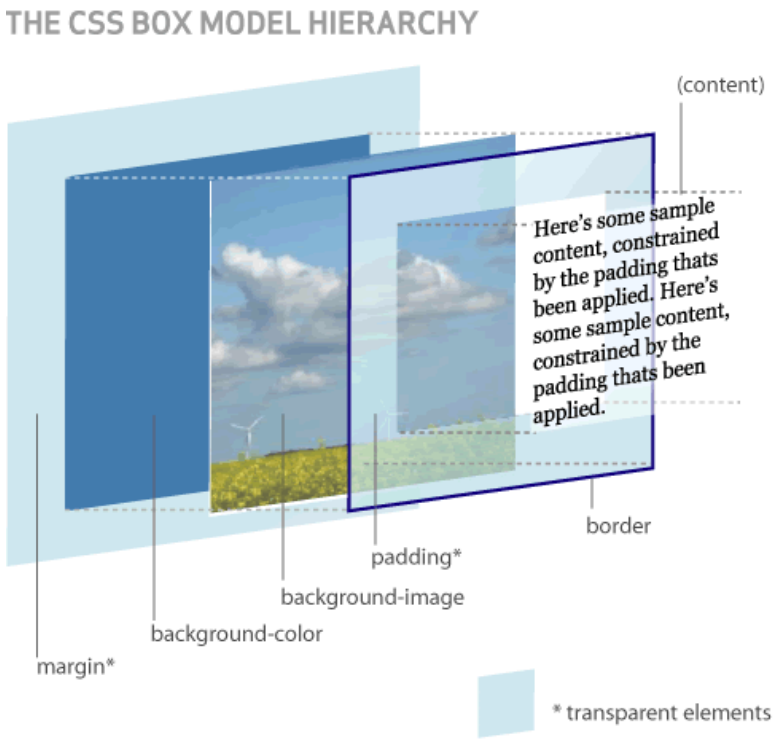
ese elemento. La siguiente imagen muestra las tres cajas rectangulares que crean las tres etiquetas HTML que incluye la página[6]:



Las cajas de las páginas no son visibles a simple vista porque inicialmente no muestran ningún color de fondo ni ningún borde. La siguiente imagen muestra las cajas que forman una página web de ejemplo, después de forzar a que todas las cajas muestren su borde:

Los navegadores crean y colocan las cajas de forma automática, pero CSS permite modificar todas sus características.

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 seis partes, tal y como muestra la siguiente imagen:



Fuente: https://hicksdesign.co.uk/boxmodel/

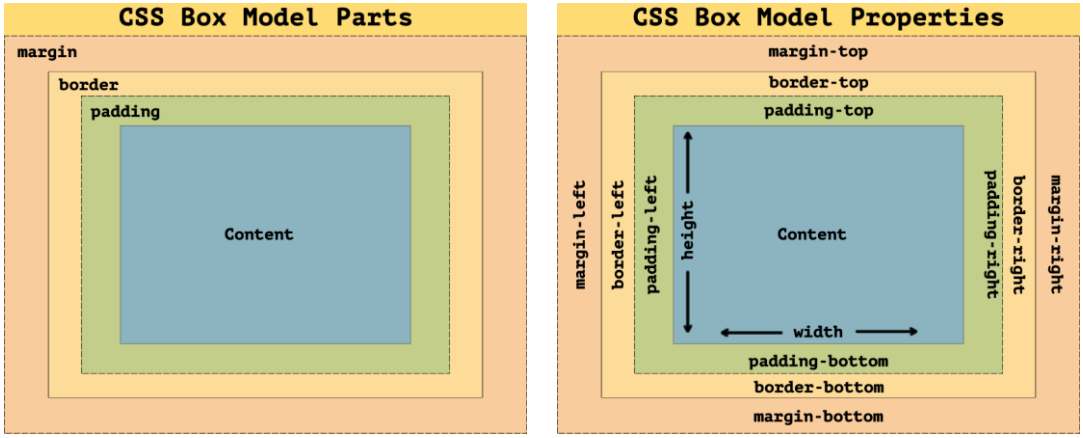
Las partes que componen cada caja y su orden de visualización desde el punto de vista del usuario son las siguientes:

- **Contenido (content):** se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)
- **Relleno (padding):** espacio libre opcional existente entre el contenido y el borde.
- **Borde (border):** línea que encierra completamente el contenido y su relleno.
- **Imagen de fondo (background image):** imagen que se muestra por detrás del contenido y el espacio de relleno.
- **Color de fondo (background color):** color que se muestra por detrás del contenido y el espacio de relleno.
- **Margen (margin):** separación opcional existente entre la caja y el resto de cajas adyacentes.

El relleno y el margen son transparentes, por lo que en el espacio ocupado por el relleno se muestra el color o imagen de fondo (si están definidos) y en el espacio ocupado por el margen se muestra el color o imagen de fondo de su elemento padre (si están definidos). Si ningún elemento padre tiene definido un color o imagen de fondo, se muestra el color o imagen de fondo de la propia página (si están definidos).

Si una caja define tanto un color como una imagen de fondo, la imagen tiene más prioridad y es la que se visualiza. No obstante, si la imagen de fondo no cubre totalmente la caja del elemento o si la imagen tiene zonas transparentes, también se visualiza el color de fondo. Combinando imágenes transparentes y colores de fondo se pueden lograr efectos gráficos muy interesantes.

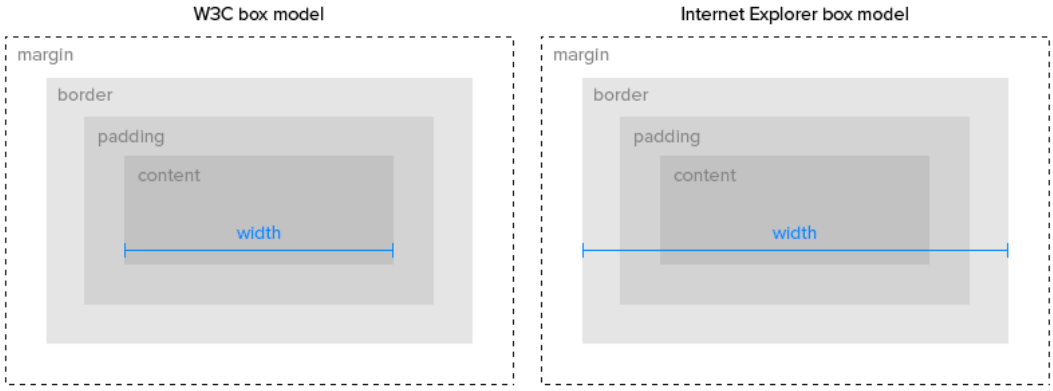
Propiedades del Modelo de Caja



Fuente imagen[7]

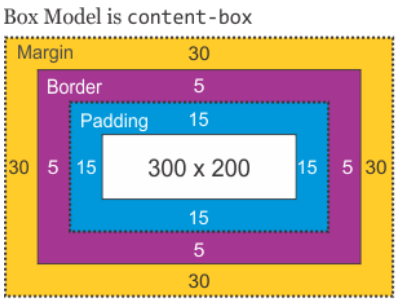
Para ilustrar las distintas propiedades del Modelo de Caja, tomaremos como ejemplo un elemento HTML de la forma ** **, que podemos pensar como la figura recién expuesta. Iremos revisando progresivamente cómo podemos personalizar y se relacionan sus distintas propiedades[8].

Debemos tener en cuenta que existe una propiedad que hace variar la interpretación del ancho/alto del elemento en relación a la posición de padding y border. Esta propiedad se denomina **box-sizing** y tiene un valor por defecto de **content-box** y opcionalmente podemos configurarlo como **border-box.** Además existe el valor padding-box, aunque es menos utilizado, por lo que no lo consideraremos aquí. Uno u otro valor es escogido según la comodidad en la interpretación intuitiva del desarrollador. La siguiente figura da a entender gráficamente ambas visiones:

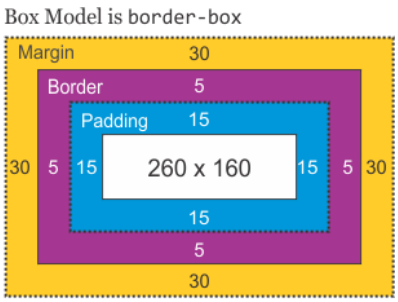


Fuente: *https://www.abeautifulsite.net/box-sizing-border-box-explained*

En términos de código, podemos visualizar la diferencia con el siguiente ejemplo que posee todas las propiedades configuradas con los mismos valores, pero en un caso con **box-sizing:content-box**, y el otro con **box-sizing:border-box**:




```
div{
      width: 300px;
      height: 200px;
      padding: 15px;
      border: 5px solid grey;
      margin: 30px;
      -moz-box-sizing: content-box;
      -webkit-box-sizing: content-box;
      box-sizing: content-box;
    }
```




```
div{
      width: 300px;
      height: 200px;
      padding: 15px;
      border: 5px solid grey;
      margin: 30px;
      -moz-box-sizing: border-box;
      -webkit-box-sizing: border-box;
      box-sizing: border-box;
    }
```

Fuente: *https://crypt.codemancers.com/posts/2013-11-17-box-model-behaviour/*

Es usual considerar más intuitivo trabajar con **border-box**, pues la variación de valores de padding o border no aumentan el tamaño del elemento que estamos intentando diseñar. Así, todos nuestros ejemplos están configurados con **border-box**. Es una práctica habitual hacer la siguiente definición en base a un selector universal que defina esté modo para todos los elementos del documento:

```
* {
  box-sizing: border-box;
}
```

ANCHO Y ALTO

Las propiedades height y width, que son configuraciones básicas de CSS para diversos elementos, podemos utilizarlas en este caso de la misma forma:

```
div.box-ejemplo {
    width: 150px;
    height: 100px;
}
```

De esta forma definimos que el ancho y el alto del contenido de nuestro div perteneciente a la clase box-ejemplo, son 150 y 100px, respectivamente.

RELLENO (PADDING)

Las propiedades de relleno de CSS definen el espacio entre el elemento de borde y el elemento de contenido. CSS soporta versiones explícitas y versiones abreviadas para especificar las propiedades espaciales de este tipo, y de otras que veremos más adelante. Por ejemplo, un elemento que deseemos diseñar con Padding (Relleno): superior de 25px, derecho de 50px, inferior de 75px e izquierdo de 100px; puede definirse la siguiente regla:

```
div.box-ejemplo {
    width: 150px;
    height: 100px;
    padding-top: 25px;
    padding-right: 50px;
    padding-bottom: 75px;
    padding-left: 100px;
}
```

Si revisamos las versiones resumidas de la misma regla, una versión equivalente puede expresarse de la siguiente forma:

```
div.box-ejemplo {
```

```
    width: 150px;
    height: 100px;
    padding: 25px 50px 75px 100px;
}
```

Si hacemos simétricos los rellenos izquierdo y derecho, en 50px, podemos resumir de la siguiente forma:

```
div.box-ejemplo {
    width: 150px;
    height: 100px;
    padding:25px 50px 75px;
}
```

Si adicionalmente agregamos simetría al relleno superior e inferior, podemos expresarla como:

```
div.box-ejemplo {
    width: 150px;
    height: 100px;
    padding:25px 50px;
}
```

Por último, si igualamos todos los rellenos a 25px, podemos resumir aún más nuestra regla y queda de la siguiente forma:

```
div.box-ejemplo {
    width: 150px;
    height: 100px;
    padding:25px;
 }
```

BORDE (BORDER)

Las propiedades de borde de CSS permiten especificar el estilo y el color del borde del elemento.

- **border-width (Ancho del borde):** Nos permite configurar el ancho del borde en cantidad de pixeles, o por medio de los tres valores predefinidos:
  - **thin**,
  - **medium**,

- **thick**.

- border-color (Color del borde): Es utilizada para configurar el color de borde. Se puede configurar color por:
  - **name** – specify a color name, like “red”
  - **RGB** – specify a RGB value, like “rgb(255,0,0)”
  - **Hex** – specify a hex value, like “#ff0000”

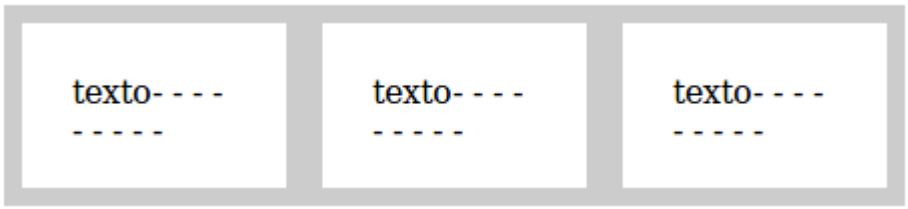
- border-style (Estilo del borde): Permite definir qué tipo de trazo poseerá el borde de la caja. Los posibles valores de esta propiedad son:

  - **dotted:** Define un borde punteado.
  - **dashed:** Define un borde de rayas.
  - **solid:** Define un borde sólido.
  - **double:** Define dos bordes. El ancho de dos bordes es el mismo que el valor border-width.
  - **groove:** Define un borde estriado (grooved) 3D. El efecto depende del valor de border-color.
  - **ridge:** Define un borde puntiagudo (ridged) 3D. El efecto depende del valor de border-color.
  - **inset:** Define un borde de inset 3D. El efecto depende del valor de border-color.
  - **outset:** Define un borde de outset 3D. El efecto depende del valor de border-color.

La versión resumida de configuración de bordes en una sola línea es como se indica a continuación, para un ejemplo de cobre de espesor de 10px, de estilo sólido y de color #ccc.

```
div.box-ejemplo {
    width: 150px;
    height: 100px;
    padding:25px;
    border:10px solid #ccc;
 }
```

Si colocamos 3 elementos texto- - - - - - - - , adyacentes veríamos que nuestra regla de estilo produce el siguiente resultado hasta el momento:



MARGEN (MARGIN)

Las propiedades margen (margin) de CSS define el espacio externo alrededor del elemento. El margen limpia un área alrededor de un elemento (Fuera del borde). El margen no tiene color de fondo, y es completamente transparente.

La versión extendida para definir los valores de las propiedades de margen en CSS, para nuestro ejemplo de div de clase box-ejemplo, si deseamos un margenes: superior de 25px, derecho de 50px, inferior de 75px e izquierdo de 100px, tendríamos:

```
div.box-ejemplo {
    width: 150px;
    height: 100px;
    padding:25px;
    border:10px solid #ccc;
    margin-top:25px;
    margin-bottom:50px;
    margin-right:75px;
    margin-left:100px;

 }
```

De la misma forma que en el caso de padding, es posible utilizar versiones abreviadas para expresar los etilos del margen. Así, la misma regla de estilo podríamos expresarla como:

```
div.box-ejemplo {
    width: 150px;
    height: 100px;
    padding:25px;
    border:10px solid #ccc;
    margin:25px 50px 75px 100px;
 }
```

Si los márgenes izquierdo y derecho los igualamos al valor de 50px, entonces obtenemos:

```
div.box-ejemplo {
```

```
        width: 150px;
        height: 100px;
        padding:25px;
        border:10px solid #ccc;
        margin:25px 50px 75px;
    }
```

Si adicionalmente, igualamos los márgenes superior e inferior a 25px, abreviaremos la regla a:

```
div.box-ejemplo {
    width: 150px;
    height: 100px;
    padding:25px;
    border:10px solid #ccc;
    margin:25px 50px;
}
```

Y por último si simplificamos nuestro requerimiento a necesitar un margen de 25px en las cuatro direcciones, entonces tendremos nuestra regla de la forma:

```
div.box-ejemplo {
    width: 150px;
    height: 100px;
    padding:25px;
    border:10px solid #ccc;
    margin:25px;
}
```

Y tendremos el siguiente resultado visualizándolo en nuestro navegador:

Se deja al lector investigar sobre el concepto de **colapso del margen** en CSS. Y hacer algunos ejemplos de práctica para comprender en qué casos ocurre.

FONDO (BACKGROUND)

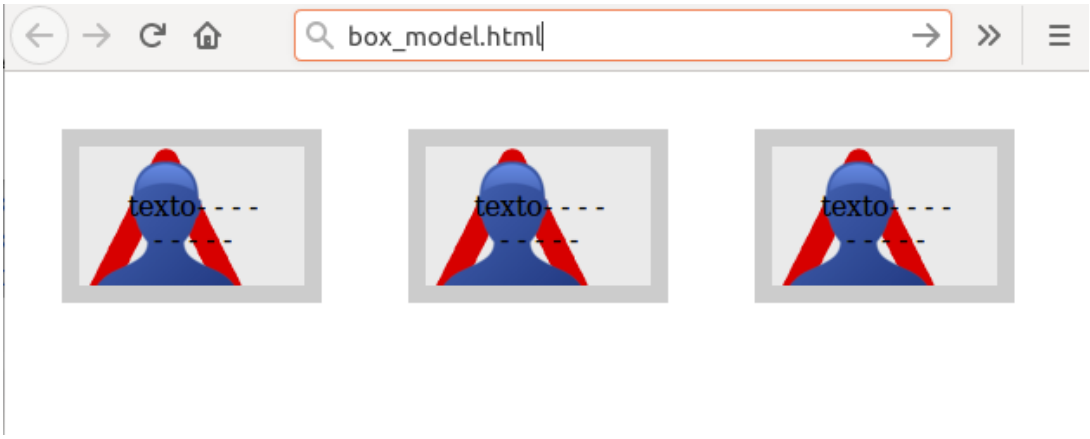
En CSS3 es posible la inclusión de uno o varios fondos detrás de un elemento seleccionado. En relación al modelo Border-Box, podemos definir varias propiedades relacionadas con el aspecto del fondo, entre las cuales, algunas comunes son:

Propiedad	Descripción
background-image	Permite indicar la fuente de la imagen a insertar. Pueden ser varias, unas sobre otras, donde la primera mencionada estará por delante de las siguientes. El formato es: **background-image: url("imagen_1.png"), url("imagen_2.png") ;**
background-repeat	Permite repetir continuamente la imagen de fondo si el área disponible es de mayor dimensión. Los valores más comunes son: **repeat, repeat-x, repeat-y, no-repeat, space, round.**
background-clip	Define el alcance de la imagen de fondo, dentro del modelo border-box, determinando si cubrirá sólo el área de contenido, o hasta la zona de padding o border. Las opciones más comunes son: **border-box, padding-box y content-box**
background-color	Es un color de fondo, que puede estar definido en RGB, Hexadecimal o nombre del color.

A modo de ilustración, para incorporar estas propiedades de nuestro ejemplo progresivo de Box Model, escogemos dos imágenes: image_person.png, image_sign.png y color gris #ccc. Además, para la imagen se agregan algunas otras propiedades que se detallan a continuación:

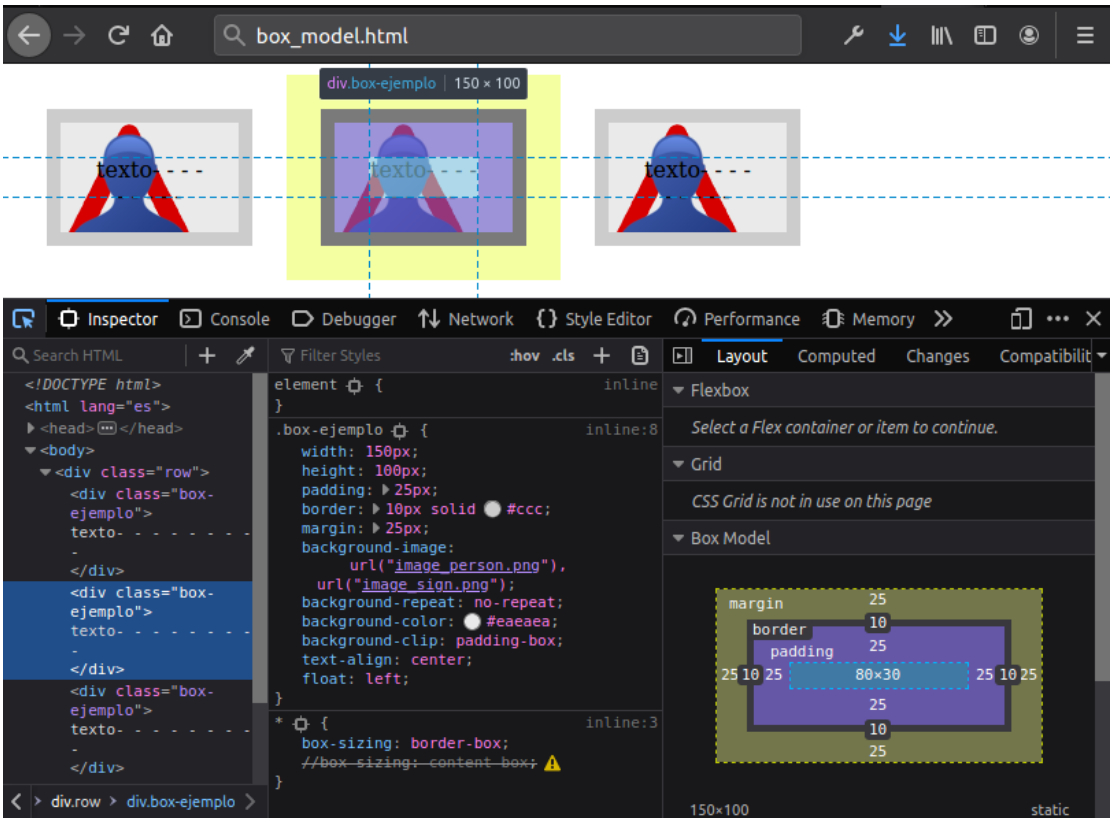
```
div.box-ejemplo {
    width: 150px;
    height: 100px;
    padding:25px;
    border:10px solid #ccc;
    margin:25px;
    background-image: url("image_person.png"), url("image_sign.png") ;
    background-repeat: no-repeat;
    background-color: #eaeaea;
    background-clip: padding-box;
    text-align: center;
 }
```

De esta forma obtenemos el siguiente resultado, donde se observa la superposición de imágenes y el color de fondo, detrás del texto de contenido.



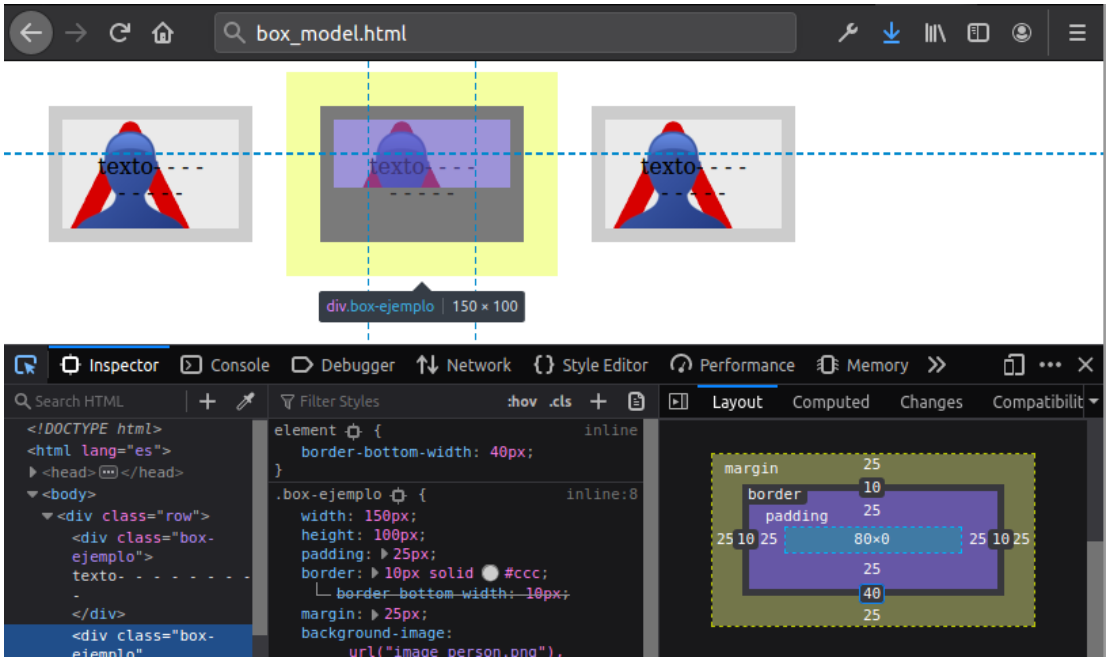
Es importante notar que en este caso utilizamos imágenes PNG con transparencia para lograr notar el efecto de superposición entre imágenes y el color de fondo, además del texto de contenido.

Volviendo a las herramientas de desarrollador disponibles en nuestro navegador web, podremos ir a la herramienta inspector y observar cómo se muestra el Modelo de Caja (Box Model) de manera gráfica del ejemplo que hemos construido a lo largo de las últimas páginas:



Al seleccionar con el mouse uno de los elementos div que hemos implementado, podemos observar que en el lado inferior derecho se muestra gráficamente un esquema de la caja correspondiente al div en particular con los valores de cada propiedad. Es interesante la funcionalidad de la herramienta de desarrollo que nos

permite editar en el esquema gráfico los valores de propiedades CSS del elemento y podemos previsualizar cómo se verían eventuales cambios. Esta funcionalidad es muy útil en la etapa de diseño de nuestras páginas, pues podemos experimentar con ideas y llegar a diseños optimizados. A continuación mostramos, a modo ilustrativo, el cambio realizado dinámicamente del borde inferior y la previsualización superior respectiva.



1.3.2.- Referencias

[1] Tutorial para aprender CSS básico desde cero
https://www.hostingatope.com/tutorial-aprender-css-basico-manual-pdf/

[2] Peter Gasston, The Book of CSS3, 2nd edition, 2015

[3] CSS vs CSS3
https://dzone.com/articles/css-vs-css3-find-out-the-most-awesome-differences

[4] Robin Nixon, CSS & CSS3 20 Lessons to Successful Web Development, 1st edition, 2015

[5] Ralph G. Schulz, Diseño Web con CSS, 1ra edición, 2009

[6] Modelo de Cajas CSS
https://uniwebsidad.com/libros/css/capitulo-4

[7] Inside the Box Model
https://webskillsbootcamp.teachable.com/courses/anyone-can-add-css-internet-skills/lectures/2465387

[8] CSS. Entender las propiedades de borde, margen y relleno
https://www.templatemonster.com/help/es/css-understanding-border-margin-and-padding-properties.html