



Taller de Aplicaciones para Internet

Diseño y Programación de Aplicaciones para Internet

NOMBRE DIRECTOR DE ESCUELA

Director de Escuela / Marcelo Lucero

ELABORACIÓN

Experto disciplinar / Hernán Thiers

Diseñador instruccional / Felipe Molina


Jefe de Diseño Instruccional/ Alejandra San Juan

VALIDACIÓN

Andrés del Alcazar

DISEÑO DOCUMENTO

Boris del Campo



1. TEMA 1: CSS	5
1.1 Desarrollo	5
1.1.1 Sintaxis	5
1.1.2 ¿Cómo Funciona?	6
1.1.3 Aplicando Estilos	7
1.1.4 Clases y Estados	9
1.2 Ejemplos/Casos de éxito	12
1.3 Links de interés/material complementario	15
2. TEMA 2: Responsividad	16
2.1 Desarrollo	16
2.1.1 Media Queries	16
2.1.2 Layout	18
2.1.2.1 Layout - Flexbox	18
2.1.2.2 Layout - Grid	20
2.1.2.3 Layout - Float	22
2.2 Ejemplos/Casos de éxito	23
2.3 Links de interés/material complementario	24



Criterios de Evaluación:

- 2.1.- Determina la aplicabilidad del lenguaje CSS, considerando requerimientos técnicos.
- 2.2.- Relaciona estructura de archivo CSS con propiedades básicas de CSS.
- 2.3.- Sintetiza propiedades de CSS y concepto de media queries en CSS, considerando patrones responsivos.
- 2.4.- Aplica lenguaje CSS para la construcción de páginas web responsivas, considerando archivos HTML5.
- 2.5.- Trabaja de forma colaborativa y en red, a través de diversos medios y soportes, adoptando diferentes roles.

1. TEMA 1: CSS

1.1 Desarrollo

Utilizamos HTML para definir una estructura web con sentido semántico, pero careciendo de colores y estilos, para lo cual aparecen las hojas de estilo en cascada o CSS (Cascading Stylesheets). CSS nos permite aplicar colores, definir estilos y tamaños, formar columnas, agregar animaciones entre muchos otros elementos de decoración.

Los navegadores utilizan una capa de estilos interna que aplica un estilo base a los elementos como títulos (más grandes y espaciados), listas (ordenadas y con viñetas), hipervínculos (subrayados y en azul), entre otros. El problema es que esta capa de estilos base es igual para todos y aquí es donde CSS llegó para aplicar tus propios estilos, bajo tu propio diseño.

CSS puede ser usado para aplicar diseño a los documentos de texto plano HTML, por ejemplo para cambiar el color y tamaño de las letras o incluso para definir formas de presentar los elementos (también conocido como “layouts”).

1.1.1 Sintaxis

Las hojas de estilo es un lenguaje basado en reglas. Se definen grupos de reglas para aplicarse a algunos elementos, por ejemplo para definir estilos solo a los títulos del documento.

```
h1 {  
    color: red;  
    font-size: 5em;  
}
```

Figura Nº 1: Grupo de estilos para títulos H1

(MDN Web Docs, 2021)

De acuerdo a la figura anterior, el grupo h1 utiliza dos declaraciones de estilos, para color y tamaño de fuente. El grupo corresponde a un “selector” que en este caso selecciona las etiquetas <h1>. Luego entre llaves se encuentran las dos declaraciones en la forma de “valor: propiedad”, especificando propiedades de estilo para los h1.

Los valores que pueden llevar las propiedades pueden estar en distintas unidades según la necesidad y según la propiedad. De acuerdo a la figura anterior hay una propiedad de color con el valor “red” o rojo. También está la propiedad de tamaño de fuente con el valor “5em”, que también puede representarse en otras unidades.

Las hojas CSS pueden tener varias reglas siempre una sobre otra.

```

h1 {
  color: red;
  font-size: 5em;
}

p {
  color: black;
}

```

Figura Nº 2: Dos grupos de reglas en una hoja de estilos

(MDN Web Docs, 2021)

1.1.2 ¿Cómo Funciona?

Cuando un navegador presenta una página, debe combinarlo con él o los documentos de estilo, generando un proceso de varias etapas. Si bien este proceso puede variar de acuerdo al navegador, se demuestra el comportamiento habitual de la mayoría de los navegadores.

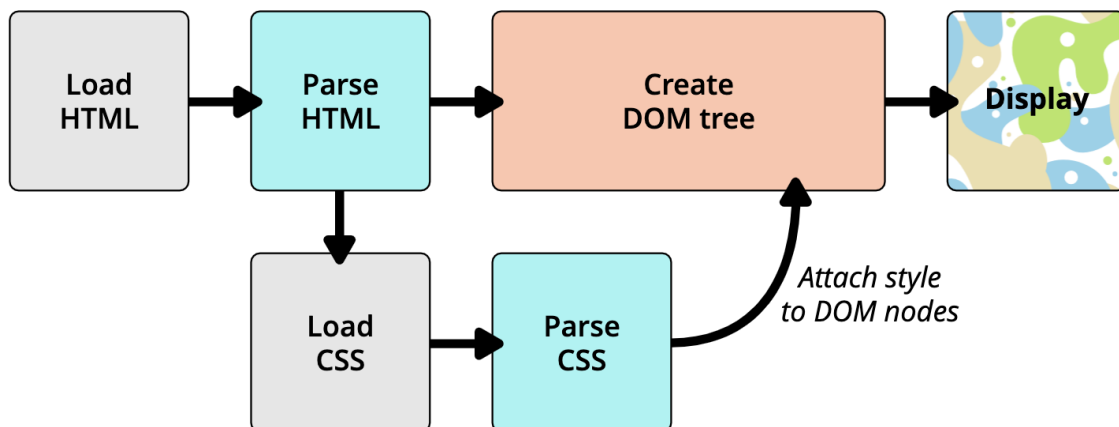



Figura Nº 3: Proceso de presentación HTML + CSS en navegadores

(MDN Web Docs, 2021)

De acuerdo a la figura anterior, se detallan los siguientes pasos:

1. El navegador carga el documento HTML.
2. Convierte el HTML en la estructura lógica DOM (HTML en memoria).
3. El navegador recolecta todos los recursos utilizados por el documento, como imágenes, videos, audios y archivos de estilo CSS.
4. Lee y revisa el documento CSS y ordena las reglas de acuerdo a los selectores. De acuerdo a los selectores que encuentra, aplica las reglas en los nodos del DOM que correspondan.

-
- 
5. Una vez aplicadas las reglas se renderiza un árbol DOM actualizado con los estilos.
 6. Se visualiza la página en pantalla.

En el tercer paso el navegador valida las instrucciones y es posible que no entienda o no soporte alguna propiedad. Si esto ocurre simplemente lo ignora y sigue al siguiente. Esto en realidad es una oportunidad para CSS, puesto que permite repetir reglas aplicando distintas propiedades y valores que puedan funcionar en uno u otro navegador.

1.1.3 Aplicando Estilos

Existen tres maneras de incorporar CSS y una de ellas es utilizando un archivo externo de extensión CSS. Si consideramos un documento HTML, debemos vincular el archivo CSS en las etiquetas de cabecera del HTML <head> y utilizar el elemento <link> para generar el vínculo.

```
<link rel="stylesheet" href="styles.css">
```

Figura Nº 4: Vincular archivo CSS a página HTML

(MDN Web Docs, 2021)

El elemento <link> le avisa al navegador que hay un archivo de estilos con el atributo “rel” e indica el nombre del archivo con el atributo href.

Consideremos el siguiente código HTML:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Getting started with CSS</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>I am a level one heading</h1>
  <p>This is a paragraph of text. In the text is a <span>span element</span>
and also a <a href="http://example.com">link</a>.</p>
  <p>This is the second paragraph. It contains an <em>emphasized</em> element.</p>
  <ul>
    <li>Item <span>one</span></li>
    <li>Item two</li>
    <li>Item <em>three</em></li>
  </ul>
</body>
</html>
```



Tras vincular el archivo “styles.css” podemos mantener todas las reglas en este archivo, por lo cual si se agrega una regla para cambiar de color los títulos <h1>, al actualizar el navegador debería reflejarse de inmediato.

```
h1 {  
  color: red;  
}
```

La regla anterior se basó en el selector por elemento (elemento de título en este caso). Podemos hacer lo mismo para aplicar estilos a los dos párrafos del HTML:

```
p {  
  color: green;  
}
```

Incluso, asignar la misma propiedad a más de un elemento al mismo tiempo:

```
p, li {  
  color: green;  
}
```




En este último caso, aplicamos el color verde a todos los párrafos y listas que hayan en todo el HTML.

El resultado tras aplicar los cambios anteriores sería el siguiente:

I am a level one heading

This is a paragraph of text. In the text is a span element and also a [link](#).

This is the second paragraph. It contains an *emphasized* element.

- Item one
- Item two
- Item *three*

Figura Nº 5: Página HTML con estilos CSS

(MDN Web Docs, 2021)

Como se puede apreciar en la figura anterior, tanto los vínculos como las listas tienen un estilo predefinido, vale decir que se ven como tal sin haberlo indicado en nuestra hoja de estilos.

Por ejemplo, queremos evitar las viñetas en las listas, para eso se utiliza el selector del elemento `` de la siguiente manera:


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

Incluso es posible cambiar el tipo de viñeta, utilizando los valores que ofrece CSS para la propiedad “list-style-type”.

1.1.4 Clases y Estados

Hasta ahora se ha revisado la aplicación de reglas en base al selector de elementos, esto suele funcionar sin problema cuando todos los elementos del mismo tipo se deben ver igual. Para el otro escenario, donde a pesar de ser el mismo elemento no se deben ver igual, existe la oportunidad de aplicar distintos estilos para los mismos elementos y para esto una de las maneras más utilizadas son las “Clases”.

Como ejemplo, se agrega el atributo “class” a la lista de la estructura HTML anterior, dando un nombre a la clase (ej: “special”):



```
<ul>
  <li>Item one</li>
  <li class="special">Item two</li>
  <li>Item <em>three</em></li>
</ul>
```

En el archivo CSS se debería ahora agregar la regla para esta clase, dando algunas propiedades de estilo (ej: letra naranja y negrita).

```
.special {
  color: orange;
  font-weight: bold;
}
```

El resultado de aplicar esta Clase de estilos sería el siguiente:

I am a level one heading

This is a paragraph of text. In the text is a span element and also a [link](#).

This is the second paragraph. It contains an *emphasized* element.

- Item one
- **Item two**
- Item *three*

Figura Nº 5: Distintos estilos para mismos Elementos

(MDN Web Docs, 2021)

Esta misma clase “special” se puede replicar en otros elementos HTML del documento, cuyo resultado será aplicar los mismos estilos en otras secciones.

Por otro lado, es posible utilizar el selector de elemento junto con la Clase, para aplicar los estilos a todos los elementos de un mismo tipo. Ejemplo, aplicar los estilos de la Clase “special” a todos los elementos de lista :

```
li.special {
  color: orange;
  font-weight: bold;
}
```



```
}
```

Con lo anterior, se lograría aplicar el color naranja y letra negrita a cualquier lista que se agregue al documento.

Para facilitar aún más la aplicación de estilos, es posible indicar varios elementos y/o clases en la misma regla. Ejemplo; aplicar el color naranja y letra negrita a todas las listas y a todos los elementos `` que hayan en el documento:

```
li.special,  
span.special {  
  color: orange;  
  font-weight: bold;  
}
```

Otro alcance para aplicar estilos son los Estados. Uno de los ejemplos más comunes son los estados de un Hipervínculo `<a>` (o “ancla”). Este elemento tiene efectivamente distintos estados:

1. Nunca ha sido visitado → “link”
2. Ya fue visitado → “visited”
3. El cursor está sobre él → “hover”
4. Está seleccionado (usando tab) → “active”
5. Está activo con un clic encima (sin soltar) → “focus”

Con CSS se pueden aplicar distintos estilos para cada uno de estos estados. En el siguiente ejemplo, se aplican dos reglas de estilos para el elemento `<a>` para dos distintos estados:

```
a:link {  
  color: pink;  
}  
  
a:visited {  
  color: green;  
}  
  
a:hover {  
  text-decoration: none;  
}
```

Si se aplican estos estilos al mismo documento HTML usado anteriormente, el resultado sería el siguiente:



I am a level one heading

This is a paragraph of text. In the text is a span element and also a [link](#).

This is the second paragraph. It contains an *emphasized* element.

- Item one
- Item two
- Item *three*

Figura Nº 6: Estilos por Estado

(MDN Web Docs, 2021)

El vínculo se ve de color rosado (color: pink). Si ubicamos el cursor encima, el subrayado debería desaparecer (text-decoration: none) y si hacemos clic encima quedará de color verde (color: green).


Cabe agregar que a veces no es recomendable quitar el subrayado en su estado inicial (link) puesto que sirve para indicar al usuario que este texto es un vínculo, vale decir es “clickeable”.

1.2 Ejemplos/Casos de éxito

Un sitio de ventas requiere mejorar la apariencia en su listado de productos, sobre todo para asegurar que los usuarios vean las ofertas y productos destacados. Para esto se ha requerido mejorar los estilos implementados.

Se tiene la siguiente estructura HTML:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Mi Negocio .CL</title>
    <meta charset="UTF-8">
    <link href="style.css" rel="stylesheet" type="text/css"/>
  </head>
  <body>
    <header>
      <h1>Mi Negocio .CL</h1>
    </header>
    <section>
      <article>
        <h2>Listado de Productos</h2>
```



```
<div>
    
    </div>
    <ul>
        <li>Manzanas - $2500</li>
        <li>Zanahorias - $1500</li>
        <li>Naranjas - $2500</li>
        <li>Peras - $2500</li>
        <li>Duraznos - $1500</li>
        <li>Damascos - $1500</li>
        <li>Kiwis - $2500</li>
    </ul>
</article>
</section>
<footer>
    <h4>Avisos legales</h4>
    <a href="#">Política de cookies</a>
    <h4>Redes sociales</h4>
    <a href="#">Mi Facebook</a>
</footer>
</body>
</html>
```

Se tiene el siguiente archivo style.css:

```
body {
    font-family: Helvetica;
}

h1 {
    color: green;
}
```



El dueño solicita las siguientes características:

- Dentro de la misma lista se deben destacar aquellos productos que están con descuento
- Estos deben verse en rojo
- En negrita
- Cuando el usuario pase el cursor por encima, se agranda el tamaño de la letra.

Para lograr los requisitos se creará una clase de nombre “oferta” y se aplicará en los productos que el dueño indicó como productos en oferta:

```
<ul>
  <li>Manzanas - $2500</li>
  <li class="oferta">Zanahorias - $1500</li>
  <li>Naranjas - $2500</li>
  <li>Peras - $2500</li>
  <li class="oferta">Duraznos - $1500</li>
  <li class="oferta">Damascos - $1500</li>
  <li>Kiwis - $2500</li>
</ul>
```

Luego, se crearán dos reglas para esta clase: color rojo + fuente en negrita y tamaño de fuente en 18px al estar en estado “hover”:

```
.oferta {
  color: red;
  font-weight: bold;
}

.oferta:hover {
  font-size: 18px;
}
```

El resultado es el siguiente, tomando en cuenta que el cursor se ubica encima del ítem de Zanahorias:

Listado de Productos



- Manzanas - \$2500
- **Zanahorias - \$1500**
- Naranjas - \$2500
- Peras - \$2500
- **Duraznos - \$1500**
- **Damascos - \$1500**
- Kiwis - \$2500

Figura N° 7: Estilo para productos en oferta

(MDN Web Docs, 2021)

1.3 Links de interés/material complementario

Cambiando estilos de listas: <https://developer.mozilla.org/en-US/docs/Web/CSS/list-style-type>



2. TEMA 2: Responsividad

2.1 Desarrollo

La responsividad es una de las características del CSS que hoy son prácticamente una necesidad. Un sitio web que se visualice de manera correcta sin importar las dimensiones del dispositivo (PC, Celular, Tablet, otros), es fundamental.

Antiguamente el objetivo era trabajar bajo dimensiones preestablecidas de pantalla, situación ya poco efectiva frente a los dispositivos móviles de múltiples tamaños. Por lo tanto, o se busca una dimensión ideal de pantalla o se buscaba alguna flexibilidad. Precisamente este último término es el que resultó más práctico.

Las primeras respuestas de flexibilidad se establecen en el año 2004 donde la mayoría de los métodos recurrían a código Javascript para detectar la pantalla y cargar otros estilos para otros tamaños.

No fue sino hasta el año 2010 cuando nace el concepto de “Responsividad” o Diseño Responsivo, que describe el uso de tres técnicas:

1. Uso de Grillas Fluidas o Fluid Grids.
2. Uso de Imágenes Fluidas o Fluid Images.
3. Uso de Media Queries.

Hoy, las buenas prácticas y estándares de la industria toman como base la responsividad a la hora de implementar y desarrollar estilos en UI.

2.1.1 Media Queries

El diseño responsivo se logra gracias a las Media Queries, que permiten aplicar utilizar la detección de pantalla sin la necesidad de código adicional por Javascript, desde el mismo CSS. Así se aplican distintos estilos para distintas dimensiones.

Por ejemplo, el siguiente código muestra un “test” para comprobar si la página se está presentando en una pantalla (no en una impresión) y que el “viewport” es al menos de 800px de ancho:

```
@media screen and (min-width: 800px) {  
  .container {  
    margin: 1em 2em;  
  }  
}
```




El concepto de “viewport” corresponde al área del polígono (o rectángulo) que se encuentra a la vista de la pantalla. Normalmente se refiere a la parte de una página web que se está viendo en el navegador. Todo lo que esté fuera de la vista (fuera del polígono) estaría fuera del “viewport”.

En cuanto al ejemplo, la regla se establece con “@media”, seguido de “screen” (para hacer referencia a la pantalla), seguido de un “and” para combinar con la regla de un ancho mínimo (min-width: 800px). En su interior, para una clase de nombre “container” se aplica la propiedad “margin” con los valores allí indicados.

Otro ejemplo puede resultar útil para aplicar formatos especiales a artículos de un sitio, por lo cual utilizar la regla con selectores sería ideal para aplicar un efecto en todos los elementos “article” solo cuando la pantalla tenga un ancho mínimo de 900px.

```
@media screen and (min-width: 900px) {  
  article {  
    padding: 1rem 3rem;  
  }  
}
```

Tomemos el siguiente ejemplo donde se tiene un documento HTML con un artículo y un CSS con distintos estilos según el tamaño de la pantalla:

```
<article>  
  Este es un artículo  
</article>
```

```
article {  
  background-color: yellow;  
  padding: 10px;  
}  
  
@media screen and (min-width: 900px){  
  article {  
    background-color: green;  
    padding: 30px;  
  }  
}
```



Para el elemento “article” se aplica un efecto de un color de fondo amarillo y un espaciado interior de 10px pero si la pantalla supera los 900px entonces su fondo será verde y tendrá mayor espaciado.

Resultado esperado para una pantalla menor a los 900px de ancho:

Este es un artículo

Resultado esperado para una pantalla superior a 900px de ancho:

Este es un artículo

Es posible agregar múltiples reglas @media para soportar múltiples dimensiones y/o tipos de layout para estas.

2.1.2 Layout

Existen distintas alternativas de presentación que responden a las las necesidades de la responsividad, algunas de estas son:

- Flexbox

Presentación de una dimensión, basada en filas o columnas con la característica de que su contenido es flexible al espacio que tenga para rellenar.

- Grids

Sistema de presentación en dos dimensiones, basado en filas y columnas con muchas opciones para confeccionar grillas complejas.

- Floats


Una de las propiedades más utilizadas para hacer “flotar” elementos dentro de cajas y para trabajar con múltiples columnas. Con la entrada de Flexbox y Grids, Float ha vuelto a utilizarse para su propósito original más que para crear layouts de diseño de estructuras web.

2.1.2.1 Layout - Flexbox

Esta capacidad permite lograr presentaciones complejas que no resultan fáciles de manera “natural” a la hora de crear estructuras web. Lo que anteriormente se lograba con esfuerzo usando Floats hoy es más sencillo con Flexbox.

Como ejemplo, se tomarán los siguientes requisitos de una presentación para web:

1. Centrar a nivel vertical el contenido de un bloque dentro de su elemento padre.

-
- 
2. Conseguir que todos los elementos dentro de un elemento padre tomen la misma dimensión de ancho y alto, de acuerdo al espacio disponible.
 3. Establecer que todas las columnas dentro de múltiples columnas adopten la misma altura incluso cuando tengan distinto contenido.

Esas son características complejas que Flexbox lo hace más sencillo.

Teniendo un escenario como el siguiente, podemos ver como una propiedad “display” de presentación, con el valor “flex”, actúa sobre elementos HTML:

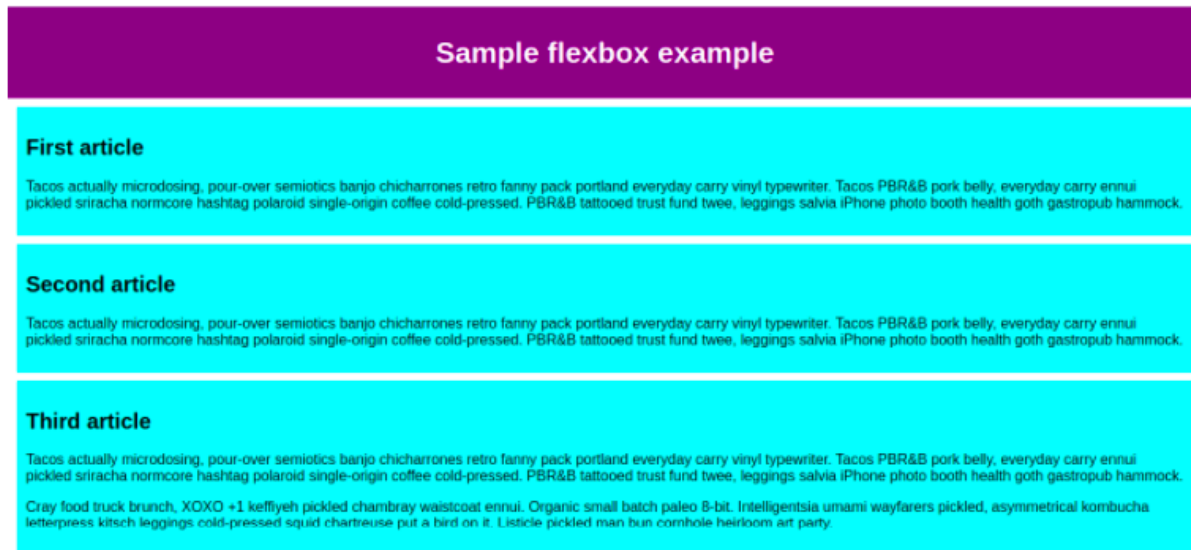


Figura Nº 8: Estructura HTML de filas sin Flex

(MDN Web Docs, 2021)

Si las tres filas que se ven corresponden a artículos (<article>) dentro de una sección (<section>), entonces podemos aplicar el valor “flex” a la sección completa de la siguiente manera:

```
section {  
  display: flex;  
}
```

Logrando el siguiente resultado:

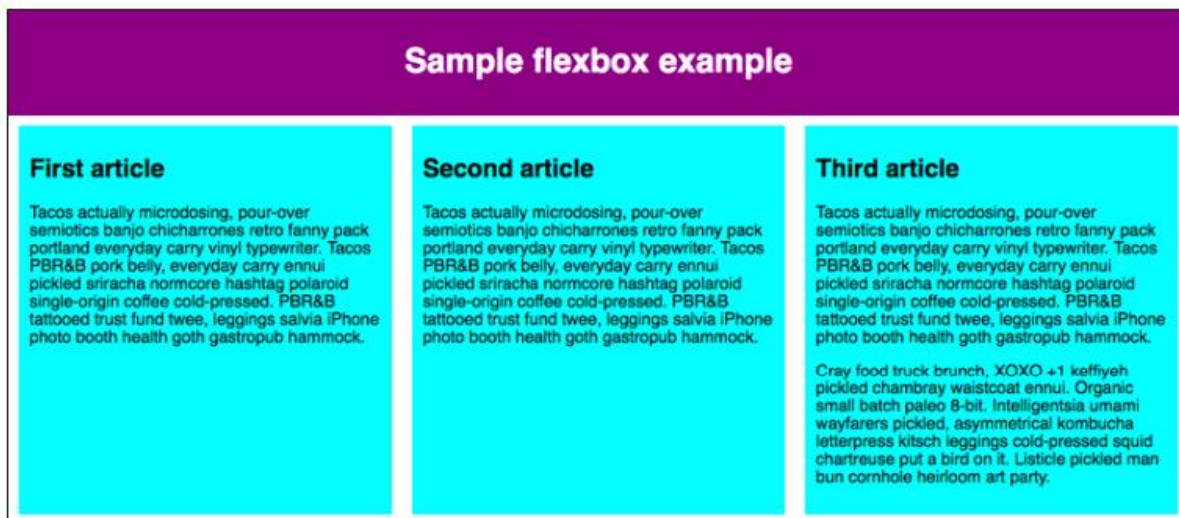


Figura Nº 9: Estructura HTML de filas con Flex

(MDN Web Docs, 2021)

Como se puede apreciar en la imagen anterior, se lograron muchas características utilizando una sola propiedad “display: flex”. Las tres quedaron como columnas, del mismo alto y ancho. Esto es porque el valor “flex” aplicó los ajustes a todo el contenido hijo del elemento en regla (section).

Para aclarar lo que ocurre se describe lo siguiente:

- El elemento al que se aplicó el valor de “flex” (en su propiedad “display”), está actuando como un elemento de “bloque”, en términos de cómo interactúa con el resto de la página.
- Los hijos de este elemento, vale decir los artículos, están actuando como ítems flexibles o de tipo flex.

Existen otros valores del tipo flex que se pueden aplicar para conseguir distintos resultados.

2.1.2.2 Layout - Grid

Podemos ver una grilla (o Grid) como una colección de líneas horizontales y verticales que crean un patrón de estructura para alinear elementos. Esto ayuda a crear estructuras donde los elementos no cambian de forma o tamaño, en el sentido de alterar el diseño que se quiere. Así, se logra un diseño con mucha mejor consistencia en su forma.

Comúnmente, una grilla tiene columnas, filas y espacios entre cada una. Espacios conocidos con el nombre de “glutter”.

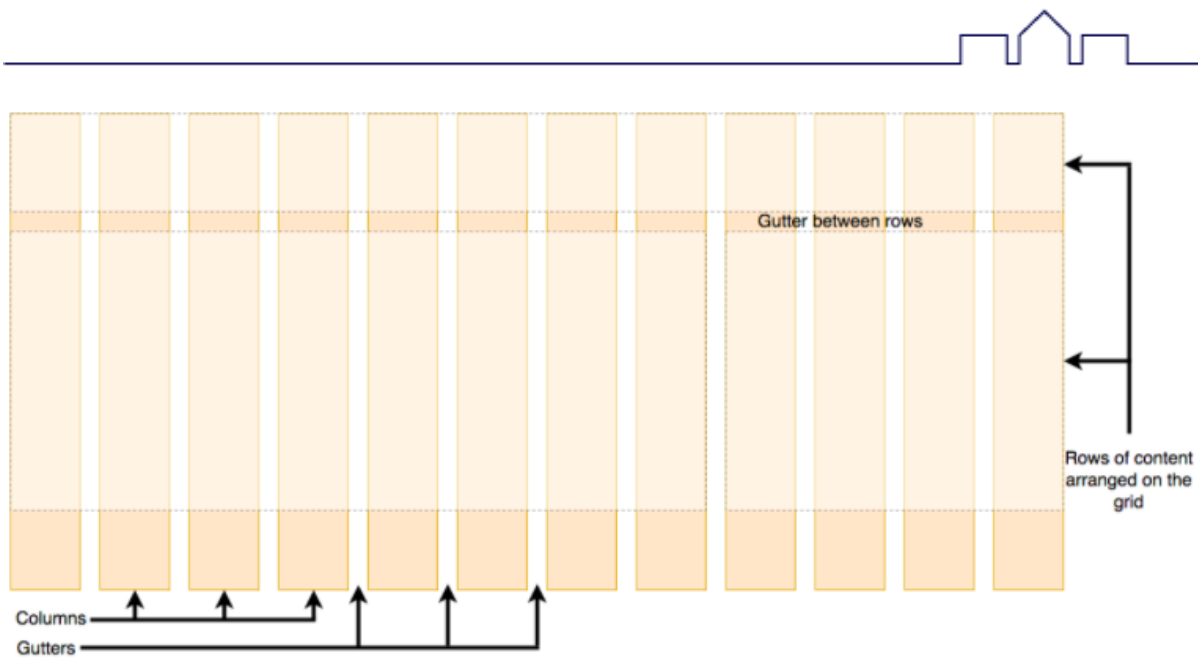


Figura Nº 10: Modelo Grid de Layout

(MDN Web Docs, 2021)

Como ejemplo, si tomamos un elemento contenedor de columnas, con una clase `.container`, podemos aplicar el valor “grid” a la propiedad “display”:

```
.container {
  display: grid;
}
```

Pero no veríamos nada distinto si no se establece las propiedades para las columnas que deba considerar:

```
.container {
  display: grid;
  grid-template-columns: 200px 200px 200px;
}
```

De esta manera, estamos indicando al contenedor que sus columnas tengan un ancho de 200px y deben ser solo 3 por fila:

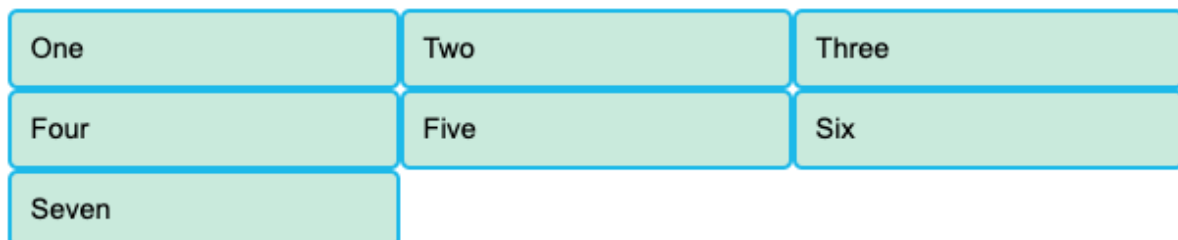


Figura N° 11: Columnas de un modelo Grid

(MDN Web Docs, 2021)

2.1.2.3 Layout - Float

Siendo una de las características en diseño más sencillas, vamos a demostrar su funcionamiento haciendo “float” un bloque de texto alrededor de otro elemento. Esto sería ideal para mostrar, por ejemplo, un texto alrededor de una imagen referencial.

El resultado sería algo como el siguiente:

Simple float example

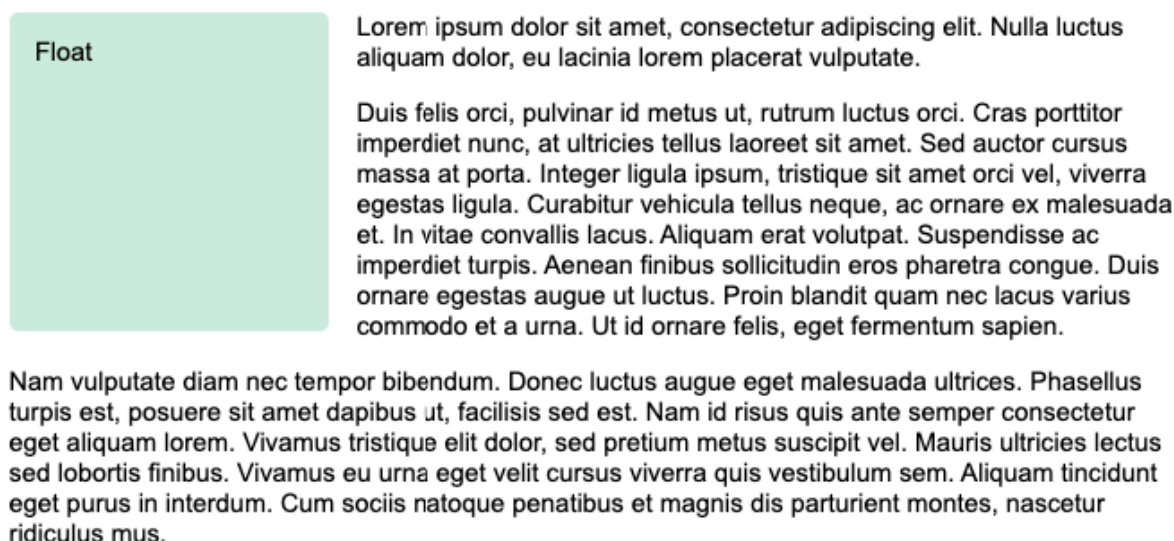


Figura N° 12: Elemento Flotante

(MDN Web Docs, 2021)

El efecto anterior se logra aplicando la propiedad “float” al elemento de color verde, para que se apegue a su elemento padre, en este caso a la izquierda usando una regla como: “float: left”. Así, todo el resto del contenido que venga a continuación, dentro del mismo elemento padre, se agregará directamente a continuación, como envolviendo al elemento flotante.

Si ese elemento flotante tuviera una clase de nombre .box se lograría el resultado anterior de la siguiente manera:

```
.box {  
  float: left;  
  margin-right: 15px;  
  width: 150px;  
  height: 100px;
```

```
border-radius: 5px;
background-color: rgb(207,232,220);
padding: 1em;
}
```

2.2 Ejemplos/Casos de éxito

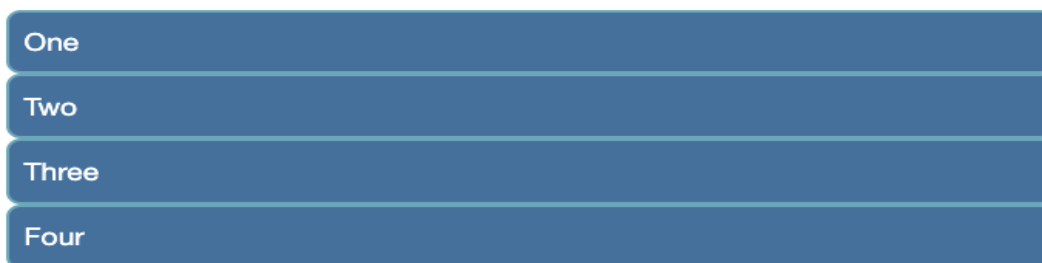
Se ha solicitado migrar una tabla HTML a un modelo en grilla, siguiendo los siguientes requisitos de diseño:

1. Generar una grilla de 4 cajas (o elementos).
2. Cada caja debe tener un ancho de 250 pixeles.
3. La grilla debe ser un máximo de 3 columnas, manteniendo un espacio fijo entre ellas de 20 pixeles.

Se tiene la siguiente estructura HTML, que utiliza elementos de tipo “<div>” tanto para el elemento padre como para sus elementos hijos o mejor dicho las columnas:

```
<div class="grid">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
</div>
```

La vista actual de esta estructura es la siguiente:



Para lograr la apariencia de una grilla con 3 columnas se aplica el siguiente CSS:

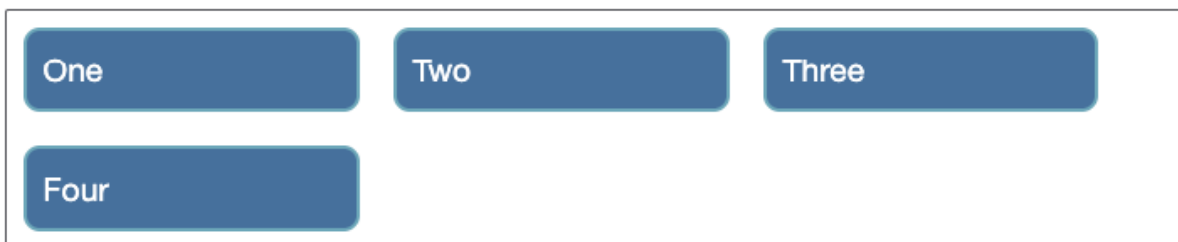
```
.grid {
  display: grid;
  grid-template-columns: 200px 200px 200px;
  grid-gap: 20px;
}
```



Lo anterior corresponde a la aplicación de las tres siguientes propiedades:

- display: se aplica el modelo de presentación “grid”.
- grid-template-columns: se aplica la plantilla para indicar cuántas columnas se deben mostrar y de que ancho.
- grid-gap: aplicar un espacio de 20 pixeles.

Finalmente se logra el siguiente resultado:



2.3 Links de interés/material complementario

Ejemplos y otros atributos para @media: <https://developer.mozilla.org/en-US/docs/Web/CSS/@media>

Detalles y ejemplos del modelo Flexbox: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox

Detalles y ejemplos del modelo Grid: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Grids

Detalles y ejemplos con Float: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Floats

IDEAS CLAVE

Por medio del siguiente mapa conceptual, se destacan las ideas clave de esta semana:

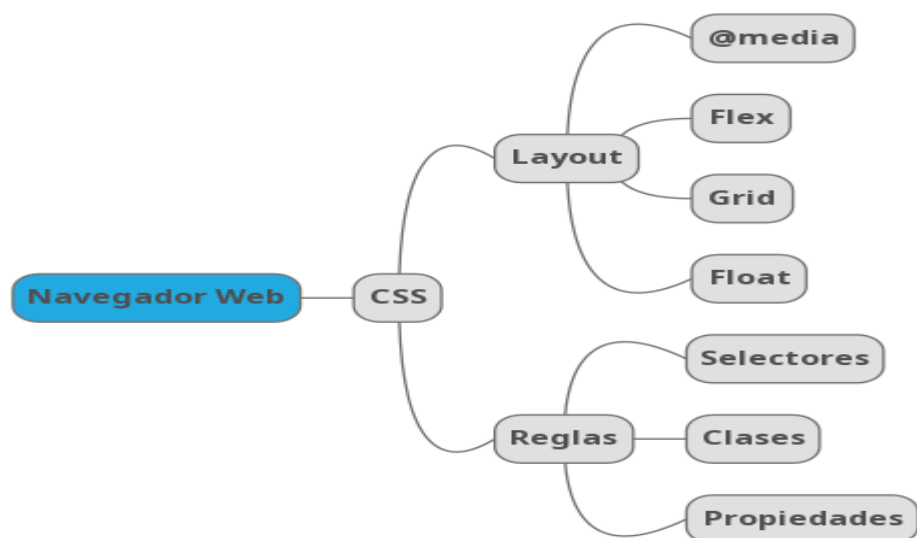


Figura Nº 13: Mapa Mental

REFERENCIAS BIBLIOGRÁFICAS

- What is CSS? (s/f). Recuperado el 18 de marzo de 2021, de Mozilla.org website: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS
- Media queries. (s/f). Recuperado el 22 de marzo de 2021, de Mozilla.org website: https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries
- Test your skills: Grid Layout. (s/f). Recuperado el 23 de marzo de 2021, de Mozilla.org website: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Grid_skills