

## HTML 5 y CSS 3. Nivel III

julio, 2018



## Objetivos del nivel

- Conocer y aplicar el diseño para dispositivos móviles
- Patrones de diseños existen para el responsive design
- Uso de media queries gracias a CSS3
- Configurar la meta viewport y manipular la pantalla de los dispositivos
- Uso de preprocesadores en HTML y CSS

## Prerrequisitos del nivel

- HTML 5 y CSS 3 Nivel II

## Acerca de este manual

Este manual pertenece al Centro de Asesoramiento y Desarrollo Informático C.A. (CADIF1). Para obtener más información sobre este u otros cursos visite nuestra sitio Web [www.cadif1.com](http://www.cadif1.com), escribanos a la dirección de correo [cadi@cadif1.com](mailto:cadi@cadif1.com) o visítenos en nuestra sede ubicada en la Av. Pedro León Torres con calle 59, Centro Comercial Sotavento, piso 2 oficina 27, Barquisimeto estado Lara, Venezuela. Tlf. 0251-7179247, 0251-4410268.

Las marcas mencionadas en este manual son propiedad de sus respectivos dueños.  
Copyright 2018. Todos los derechos reservados.



## Contenido del nivel

### Capítulo 1. Responsive Web Design

- 1.1.- Responsive Web Design.
- 1.2.- Ventajas y Desventajas.
- 1.3.- Ajustar Resolución de la Pantalla.
- 1.4.- Estrategias Para Hacer Responsive Web Design.

### Capítulo 2. CSS Media queries

- 2.1.- CSS Media Queries.
- 2.2.- Sintaxis.
- 2.3.- Incluir Media Queries.

### Capítulo 3. Media queries - Media types y operadores lógicos

- 3.1.- Media types.
- 3.2.- Operadores Lógicos.

### Capítulo 4. Configuración del Viewport

- 4.1.- Sitios móviles: Viewport.
- 4.2.- Inserción.
- 4.3.- Usos.

### Capítulo 5. El Layout: Introducción a Flexbox

- 5.1.- ¿qué es el Layout?.
- 5.2.- Introducción a Flexbox.
- 5.3.- Primeros Pasos Con Flexbox.

### Capítulo 6. Flexbox y Sus Propiedades

- 6.1.- Propiedades Del Flex Container (padre).
- 6.2.- Propiedades de Los Flex Ítems (hijos).

## Capítulo 7. Css Grid. Parte 1

- 7.1.- Definicion.
- 7.2.- Elementos Del Grid.
- 7.3.- Espacio Entre Filas y Columnas.

## Capítulo 8. Css Grid. Parte 2

- 8.1.- Grid Items.
- 8.2.- Modificando Los Elementos Del Grid.

## Capítulo 9. Preprocesadores

- 9.1.- ¿Qué es un preprocesador?.
- 9.2.- Beneficios de usar un preprocesador.
- 9.3.- Tipos de preprocesadores.

## Capítulo 10. Preprocesador HTML: HAML

- 10.1.- ¿Qué es HAML?.
- 10.2.- Principios.
- 10.3.- Sintaxis.

## Capítulo 11. Maquetación con HAML

- 11.1.- Atributos.
- 11.2.- Clases e ID.
- 11.3.- DIV Implícito.

## Capítulo 12. Preprocesadores CSS: SASS

- 12.1.- Preprocesadores en CSS.
- 12.2.- Ventajas de Utilizar Preprocesadores.
- 12.3.- Sintaxis en Sass.

## Capítulo 13. Sass - Alcance, referencia y propiedades

- 13.1.- Alcance de un selector - Anidamiento.
- 13.2.- Referenciando a los selectores padre.
- 13.3.- Propiedades anidadas.

## Capítulo 14. Sass - Variables, operaciones y colores

- 14.1.- Uso de variable.
- 14.2.- Operaciones en el layout.
- 14.3.- Creación de colores.

## Capítulo 15. Sass - Importación, mixins y herencia

- 15.1.- Importar archivos.
- 15.2.- Mixins.
- 15.3.- Extender / Herencia.



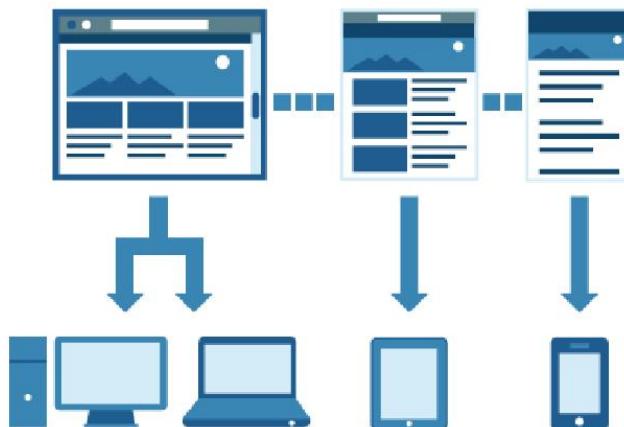
## Capítulo 1. RESPONSIVE WEB DESIGN

### 1.1.- Responsive Web Design

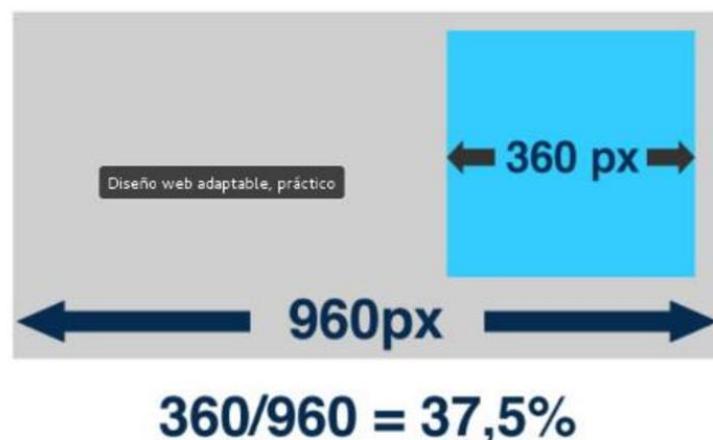
En el campo del diseño web y desarrollo se está llegando rápidamente al punto de ser incapaz de mantenerse al día con las nuevas resoluciones y dispositivos. Casi cada nuevo cliente en estos días quiere una versión móvil de su sitio web. Es prácticamente imprescindible, debido a que un diseño debe verse correctamente para el BlackBerry, el iPhone, Android, Windows Phone, el iPad, netbook y el Kindle, así como también, todas las resoluciones de pantalla deben ser compatibles.



El término Responsive Web Design (diseño de páginas web sensibles), fue acuñado originalmente por Ethan Marcotte en un artículo en A List Apart (Blog). Desde entonces, el concepto ha ganado gran popularidad.

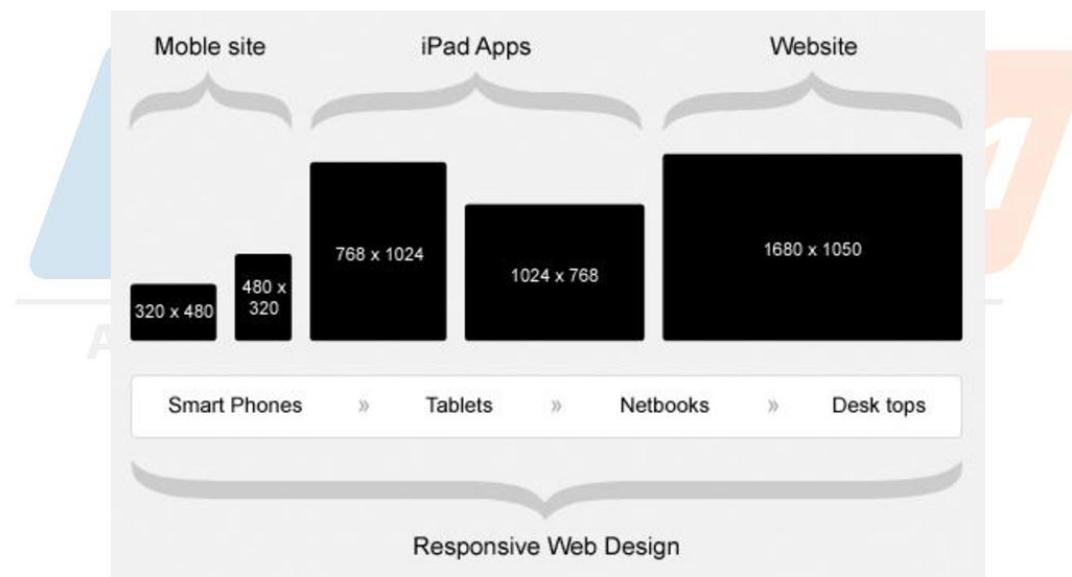


En un mundo donde incluso dos pantallas de las mismas dimensiones físicas pueden tener densidades de píxeles diferentes, las mediciones estáticas han llegado a ser innecesarias. Por lo tanto, Responsive Web Design utiliza una combinación de "Grillas Fluídicas" y "Media Queries" para crear diseños que se ajustan a todos los tamaños de pantalla disponibles sin dejar de mantener el sitio a la vez atractivo y utilizable. Esto difiere del método más tradicional de depender en las posiciones fijas y tamaños basadas en píxeles o elementos basados en mediciones estáticas.



Esta técnica de diseño web consiste en crear una estructura de una página web que según el tamaño de la pantalla (o ventana) en la que se visualice variará su contenido, para que siempre sea visible y cómodo de usar desde PC, tablets y smartphones.

Responsive Web Design es el enfoque que sugiere que el diseño y desarrollo deben responder al comportamiento del usuario y el medio ambiente basado en el tamaño de pantalla, la plataforma y la orientación. La práctica consiste en una combinación de grillas fluidas y diseños, imágenes y un uso inteligente de media queries. A medida que el usuario cambia de un dispositivo a otro, el sitio debería cambiar automáticamente para adaptarse a la resolución, el tamaño de la imagen y las habilidades de secuencias de comandos.



## 1.2.- Ventajas y Desventajas

Entre las ventajas de usar este tipo de diseño están:

- Se reducen costes de desarrollo.

Teniendo un solo diseño web optimizado para todos los dispositivos en vez de varios diseños independientes, uno para cada soporte.

- Eficiencia en la actualización.

El sitio solo se debe actualizar una vez y se ve reflejada en todas las plataformas. Cuando tenemos los portales independientes para Web y Mobile se debe realizar la actualización dos veces lo que crea la necesidad de mayor cantidad de recursos y posibilidad de error.

- Se mejoran los resultados de conversión.

Al tener una página que se vé de forma optimizada para móviles y tabletas, el porcentaje de usuarios que adquieren un producto o envían un formulario es mayor.

- Impacto en el visitante.

Genera impacto en las personas que la vean en acción, lo que permitirá asociar a la marca con creatividad e innovación.

- Baja el rebote de usuarios.

Una buena parte de los usuarios que abandonan una página web al entrar desde un dispositivo móvil es porque no pueden visualizar correctamente el contenido. Con el diseño responsive, el usuario disfrutará siempre de una buena experiencia de navegación.

- Permite desarrollar una estrategia de marketing.

Sobre la web unificada para todos los soportes, haciendo que esta sea más sólida y mejorando su efectividad.

Entre las desventajas están:

- Eliminar elementos de nuestra web que den problemas para que se puedan visualizar bien en todos los dispositivos.

- Falta de adaptación en dispositivos antiguos o muy nuevos.

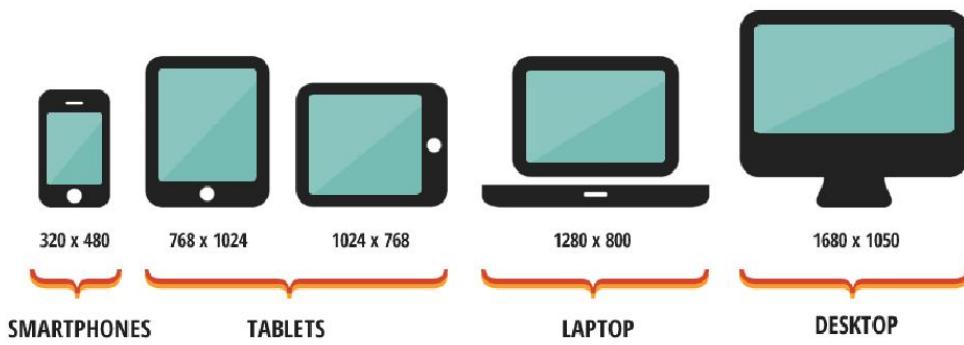
- Mayor tiempo de carga, ya que enviamos más información de la necesaria.

### 1.3.- Ajustar Resolución de la Pantalla

Cada vez vienen más dispositivos variando las resoluciones de pantalla, las definiciones y orientaciones. Nuevos dispositivos con nuevos tamaños de pantalla se están desarrollando todos los días, y cada uno de estos dispositivos pueden ser capaces de manejar variaciones en el tamaño, funcionalidad e incluso el color. Algunos están en Landscape (Horizontal), otros en Portrait (Vertical), otros incluso completamente cuadrados.



La siguiente imagen muestra algunos ejemplos de resoluciones de varios dispositivos:



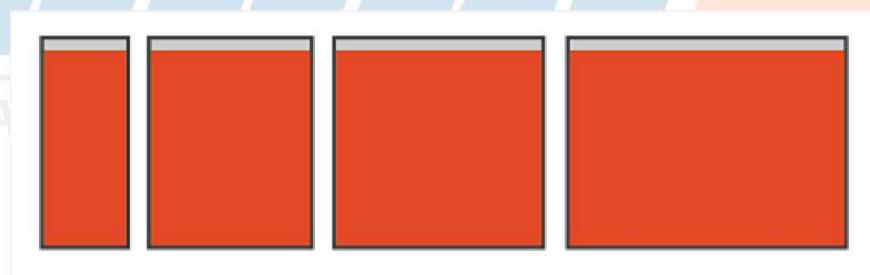
#### 1.4.- Estrategias Para Hacer Responsive Web Design

Responsive Design significa adaptar tu diseño al tamaño de pantalla del dispositivo. Puedes hacerlo como tu quieras, pero existen 5 patrones ya definidos que ayudarán en el diseño. Sus nombres son:

- Tiny Tweaks
- Mostly Fluid
- Column Drop
- Layout Shifter
- Off Canvas

Tiny Tweaks:

Es el patrón más simple y sencillo de implementar de todos. Se basa en una sola columna para el contenido. Sus cambios son básicamente que dependiendo del tamaño de pantalla, se amplían los espaciados y el tamaño de fuente. Es muy utilizado en sitios con mucho contenido escrito, así mejoran la experiencia de lectura.



Mostly Fluid:

Este patrón consiste en tener una grilla o Grid de tamaño flexible. Cuando se está en un smartphone todo forma una única columna y en varias filas quedan colocados los distintos bloques.

Según vaya creciendo la pantalla, los distintos bloques se agrupan ocupando toda la pantalla disponible. En pantallas más grandes, el diseño es el mismo pero queda

agrupado dentro de un contenedor que queda centrado en la página con un tamaño fijo de ancho.



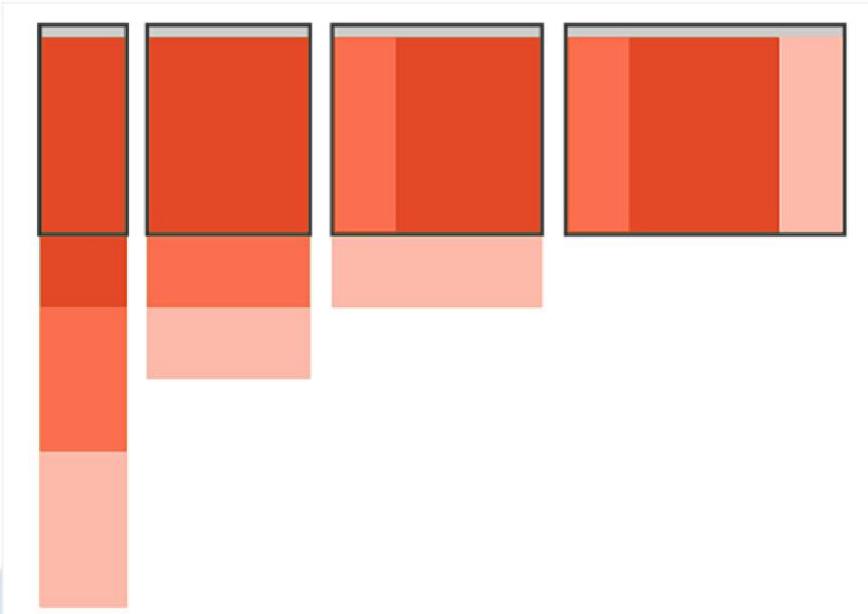
### ACADEMIA DE SOFTWARE

#### Column Drop:

Este patrón consiste en que cada bloque de contenido que en un smartphone vemos en filas, vaya formando columnas según vaya siendo más grande la pantalla del dispositivo.

Tendremos un primer breakpoint donde la segunda fila pasa a ser columna, pero ocupando la primera posición, habitualmente para un menú de navegación.

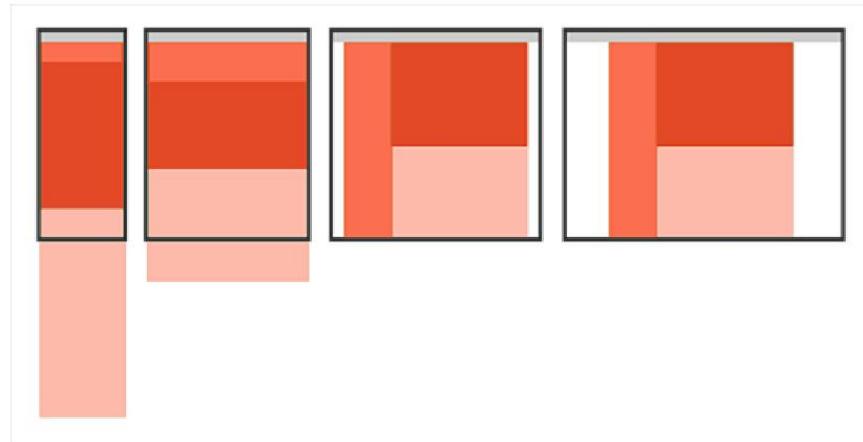
El contenido que aparecía en primer lugar en la versión móvil, pasa a ocupar la segunda columna en el primer corte. Tendremos un segundo punto de corte donde la última fila se convierte en columna también.



#### Layout Shifter:

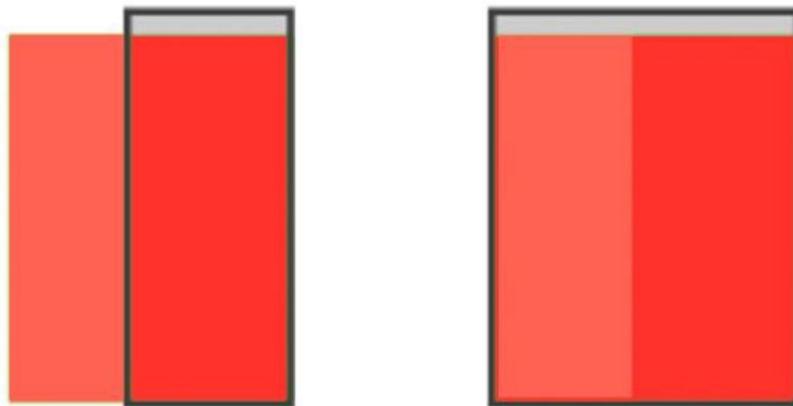
Este es uno de los patrones más complejos. Consiste en mover los bloques de contenido cambiando totalmente el Layout, de ahí el nombre del patrón.

Las columnas 2 y 3 estarán englobadas dentro de un bloque que llamaremos container-inner que nos permitirá hacer el cambio de layout.



#### Off Canvas:

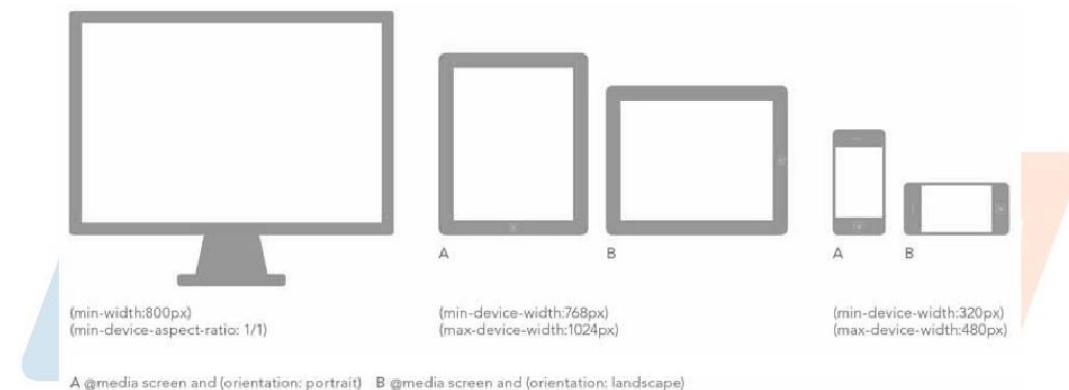
Es uno de los más utilizados, sobre todo en aplicaciones móviles. Este patrón esconde contenido en la pantalla y únicamente es visible si se realiza un determinado gesto. Este contenido oculto normalmente es un menú de navegación. Cuando la pantalla es más ancha, este contenido se hace visible.



## Capítulo 2. CSS MEDIA QUERIES

### 2.1.- CSS Media Queries

Una media query consiste en un tipo de medio y al menos una consulta que limita las hojas de estilo utilizando características del medio como ancho, alto y color. Añadido en CSS3, las media queries dejan que la presentación del contenido se adapte a un rango específico de dispositivos de salida sin tener que cambiar el contenido en sí.



## ACADEMIA DE SOFTWARE

Desde CSS 2.1, las hojas de estilos han disfrutado algo así como una conciencia del dispositivo a través de los media types. Si alguna vez has escrito una hoja de estilos para impresión, ya estás familiarizado con el concepto.

El W3C creó las media queries como parte de la especificación CSS3, mejorando la promesa de los media types. Una media query nos permite apuntar no sólo a ciertas clases de dispositivos, sino realmente inspeccionar las características físicas del dispositivo que está renderizando el trabajo.

```
<link rel="stylesheet" href="principal.css" media="screen" />
<link rel="stylesheet" href="impresion.css" media="print" />
```

## 2.2.- Sintaxis

Las media queries consisten de un media type y una o más expresiones, implicando características del medio, la cual se resuelve como verdadera o falsa. El resultado de la consulta es verdadero si el tipo de medio especificado en el media query concuerda con el tipo de dispositivo que está siendo mostrado y todas las expresiones en el media query son verdaderas.

```
@media(min-width:900px){p{color:red;}}
```

La query contiene dos componentes:

- un media type (screen).
- la consulta entre paréntesis, conteniendo una característica a inspeccionar seguida por el valor al que apuntamos.

En el siguiente ejemplo se especifica la hoja de estilo "iphone.css" para el medio "screen" (la pantalla) y la resolución sea menor o igual que 480 px. En otras palabras, le está preguntando al dispositivo si su resolución horizontal (max-device-width) es igual o menor que 480px, si es correcto (en otras palabras, si está viendo la página en un dispositivo con una pantalla pequeña como el iPhone) entonces el dispositivo cargará "iphone.css", de lo contrario, el link es completamente ignorado.

```
<link rel="stylesheet" media="screen and (max-device-width: 480px)" href="iphone.css" />
```

## 2.3.- Incluir Media Queries

Existen varias formas de utilizar los media Queries:

- En el tag link:

```
<link rel="stylesheet" media="screen and (max-device-width: 480px)" href="ejemplo.css" />
```

- Como parte de una regla @media:

```
@media screen and (max-device-width: 480px){  
    .columna_derecha {  
        float: none;  
    }  
}
```

- O como una directiva @import:

```
@import url("ejemplo.css") screen and (max-device-width: 480px);
```

En cada caso el efecto es el mismo, si el dispositivo pasa el test planteado por la media query, el CSS que corresponda es aplicado al código. En resumen, las media queries son comentarios condicionales para todos. En lugar de apuntarle a una versión específica de un navegador, se pueden corregir problemas en el layout cuando se escala más allá de su resolución inicial e ideal.

## Capítulo 3. MEDIA QUERIES - MEDIA TYPES Y OPERADORES LÓGICOS

### 3.1.- Media types

Una de las características más importantes de las hojas de estilos CSS es que permiten definir diferentes estilos para diferentes medios o dispositivos: pantallas, impresoras, móviles, proyectores, etc.

Los medios más utilizados actualmente son screen (para definir el aspecto de la página en pantalla) y print (para definir el aspecto de la página cuando se imprime), seguidos de handheld (que define el aspecto de la página cuando se visualiza mediante un dispositivo móvil).

La siguiente tabla muestra la lista de los posibles medios a los que se puede hacer referencia:

Medio	Descripción
all	Todos los medios definidos
braille	Dispositivos táctiles que emplean el sistema braille
embossed	Impresoras braille
handheld	Dispositivos de mano: móviles, PDA, etc.
print	Impresoras y navegadores en el modo "Vista Previa para Imprimir"
projection	Proyectores y dispositivos para presentaciones
screen	Pantallas de ordenador
speech	Sintetizadores para navegadores de voz utilizados por personas discapacitadas
tty	Dispositivos textuales limitados como teletipos y terminales de texto
tv	Televisores y dispositivos con resolución baja

Las reglas @media son un tipo especial de regla CSS que permiten indicar de forma directa el medio o medios en los que se aplicarán los estilos incluidos en la regla. Para especificar el medio en el que se aplican los estilos, se incluye su nombre después de @media. Si los estilos se aplican a varios medios, se incluyen los nombres de todos los medios separados por comas.

```
@media print {  
    body { font-size: 10pt }  
}  
  
@media screen {  
    body { font-size: 13px }  
}  
  
@media screen, print {  
    body { line-height: 1.2 }  
}
```

### 3.2.- Operadores Lógicos

Se pueden crear Media Queries complejos utilizando operadores lógicos, incluyendo not, and y only. El operador and es usado para combinar múltiples medias en un sólo Media Query, requiriendo que cada función devuelva true para que el Query se aplique. El operador not se utiliza para negar un Media Query completo y el operador only se usa para aplicar un estilo sólo si el Query completo es correcto.

Además, se pueden combinar múltiples Media Queries separados por comas en una lista; si alguno de los Queries devuelve true, todo el media devolverá true. Esto es equivalente a un operador or.

```
@media (min-width: 700px){  
    .columna_derecha  
    {  
        float: none;  
    }  
}
```

and

El keyword and se usa para combinar múltiples media features, así como combinar éstos con media types.

Coma

Cuando se utilizan las listas separadas por comas en los Media Queries, si algunas de las Media Queries devuelven true, los estilos se aplican. Cada Media Query separado por comas en la lista se trata como un Query individual y cualquier operador aplicado a un Media Query no afecta a los demás. Esto significa que los Media Queries separados por comas puede dirigirse a diferentes media features, types o states.

```
@media (min-width: 700px), handheld and (orientation: landscape)  
{ ... }
```

not

La keyword not se aplica al Media Query en su totalidad y devuelve true si el Media Query devuelve false. Este keyword no se puede utilizar para negar un query individual, solamente un query entero.

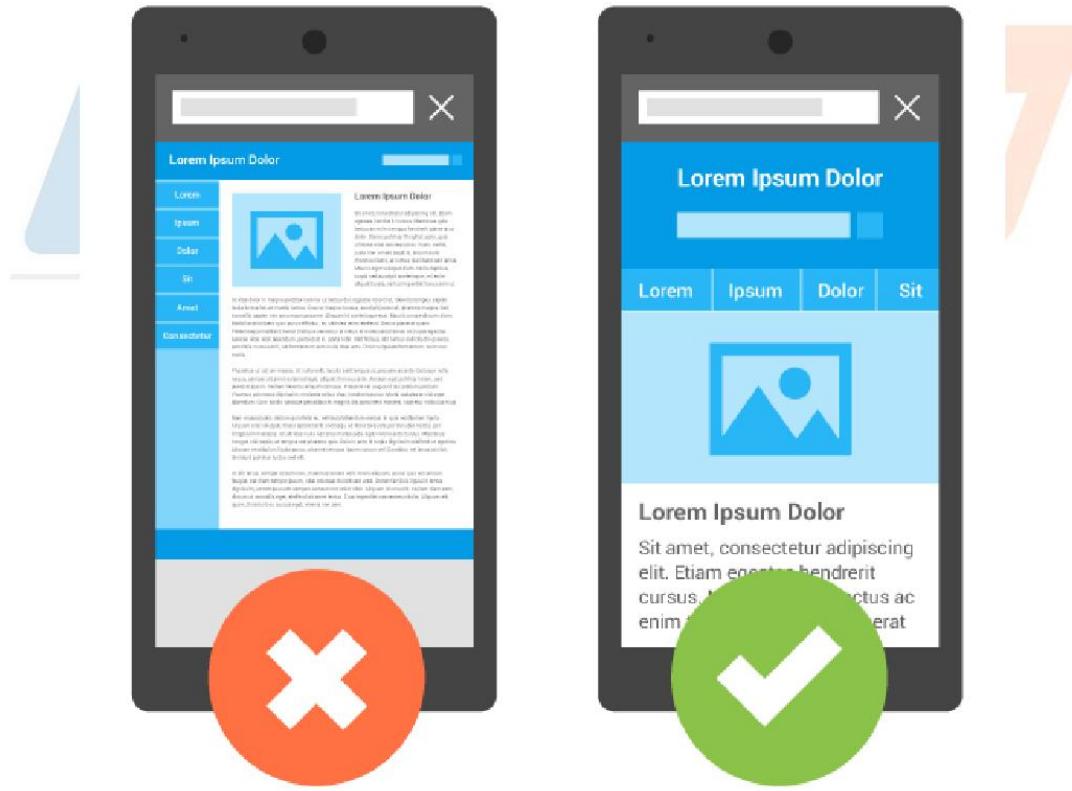
```
@media not screen and (color), print and (color)
```

## **Capítulo 4. CONFIGURACIÓN DEL VIEWPORT**

## 4.1.- Sitios móviles: Viewport

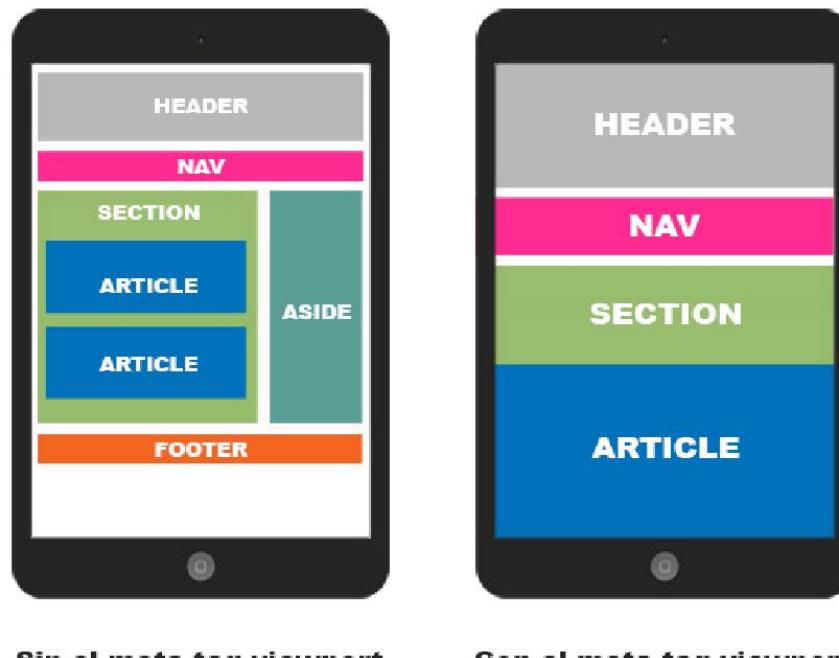
Prácticamente todos los navegadores de smartphones y tabletas al entrar a un sitio analizan el tamaño total, escalándolo para que se muestre completo en la pantalla.

Esta es una conducta normal del navegador, pero cuando se está construyendo una versión móvil del sitio puede generar problemas para diagramar y que se vea consistente en todas las plataformas. La escala automática se puede prevenir y controlar de forma muy sencilla utilizando el atributo `viewport`.



Inicialmente creada por Apple para definir diversas directrices sobre cómo el iPhone debe renderizar un documento web, el viewport es una etiqueta META que se ha convertido en un estándar en el momento actual. La mayoría de dispositivos móviles la soportan en la mayor gama de sistemas operativos, lo que la convierte en un integrante imprescindible para cualquier página que esté pensada para verse también en dispositivos en movilidad.

Aunque la hoja de estilos este vinculada con la página utilizando Media Queries, el dispositivo móvil ignora los estilos y despliega la versión de escritorio. El viewport cuando se esta hablando de dispositivos móviles, no corresponde al tamaño real de la pantalla en píxeles, sino al espacio que la pantalla está emulando que tiene.



## 4.2.- Inserción

El viewport es un atributo del tag que debe incluirse entre las etiquetas de un documento HTML, así como se muestra en el siguiente ejemplo:

```
<!doctype html>
<html lang="es">
  <head>
    <title>Viewport</title>
    <meta name="viewport" />
  </head>
  <body>
    </body>
  </html>
```

El viewport es el área visual en la cual mostramos un documento HTML y podemos manipular esta característica para controlar como mostrarlo en la pantalla de un móvil. Utilizando sus atributos podemos configurar elementos como el ancho, alto, tamaño y escala.

Atributo	Valores	Descripción
<b>width</b>	valor en integral (pixels) o constante (device-width)	Define el ancho del viewport
<b>height</b>	valor en integral (pixels) o constantes (device-height)	Define el height del viewport
<b>initial-scale</b>	Con cualquier número real de 0.1 en adelante, el valor 1 no escala	Define la escala inicial del viewport
<b>user-scale</b>	"yes" "no"	Define los permisos para que el usuario pueda escalar el viewport
<b>minimum-scale</b>	Con cualquier número real de 0.1 en adelante, el valor 1 no escala	Define la escala mínima del viewport
<b>maximum-scale</b>	Con cualquier número real de 0.1 en adelante, el valor 1 no escala	Define la escala máxima del viewport

## 4.3.- Usos

Dentro de content="" puedes ingresar una gran cantidad de valores delimitados por comas, pero vamos a concentrarnos en los fundamentales por ahora.

```
<meta name="viewport" content="">
```

Por ejemplo, si tu diseño móvil está hecho intencionalmente a 320px, se puede especificar el ancho del viewport:

```
<meta name="viewport" content="width=320">
```

Para layouts flexibles es más práctico basar el ancho del viewport en el dispositivo en cuestión, así que para empatar el ancho del layout con el del dispositivo se coloca:

```
<meta name="viewport" content="width=device-width">
```

Para estar extra seguro de que el layout se mostrará como se planea, puede también fijar el nivel del zoom. Asegurará que al abrir, tu layout se mostrará correctamente a una escala de 1:1.

```
<meta name="viewport" content="initial-scale=1">
```

El atributo viewport permite muchas configuraciones, pero para asegurar compatibilidad con la mayor cantidad de pantallas, navegadores móviles y resoluciones se recomienda utilizar el viewport optimizado para móviles:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

## Capítulo 5. EL LAYOUT: INTRODUCCIÓN A FLEXBOX

### 5.1.- ¿qué es el Layout?

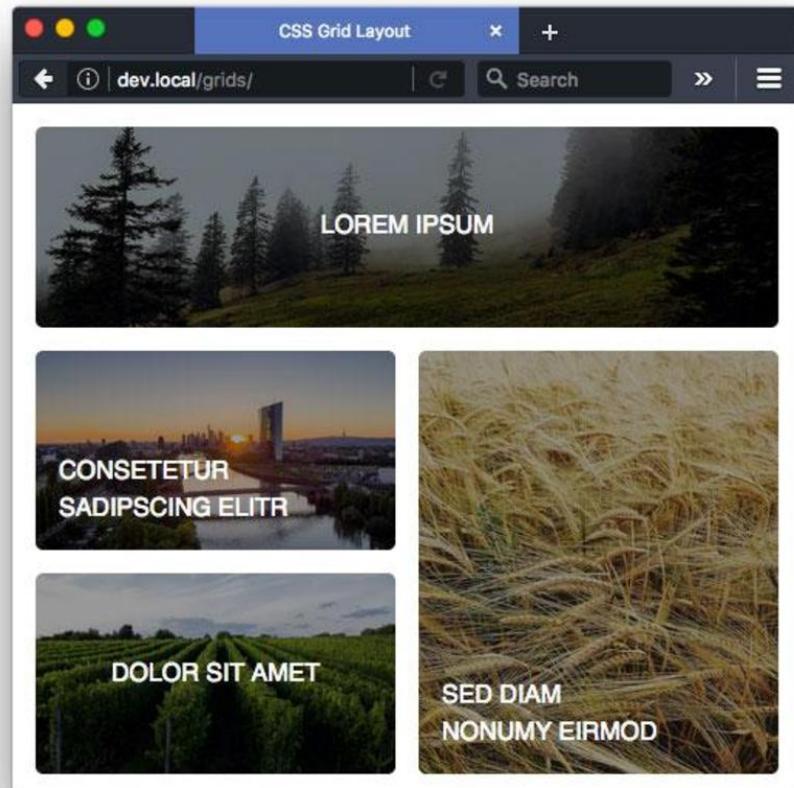
Es conocido como el boceto o plan de un proyecto, su finalidad es definir la estructura que tendrán los elementos dentro del diseño. En el ámbito de diseño web hace referencia a la maquetación, término nombrado a lo largo del curso pero que ahora se le dará un mayor enfoque.

El diseño de una página web puede tener varios tipos de modelos de layout:

- Block layout
- Inline layout
- Table layout
- Flexbox layout
- Grid layout

Cada uno de estos modelos se puede definir con la propiedad display. Anteriormente se trabajaron algunos de ellos como el modelo de bloque y en línea, sin embargo, se ven nuevos términos como Flexbox y Grid layout.



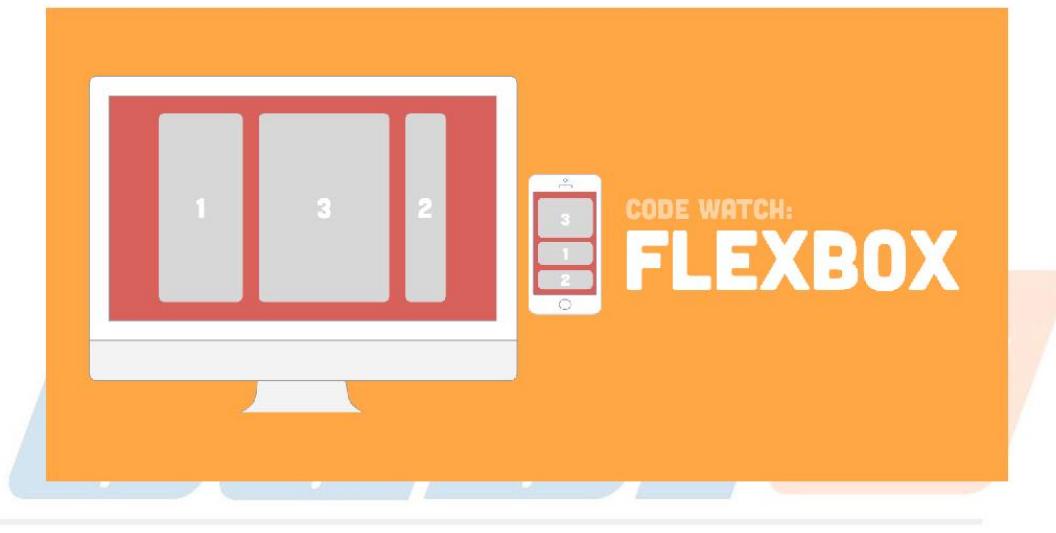


## 5.2.- Introducción a Flexbox

Flexbox es un nuevo modelo de layout, denominado diseño de caja flexible, que vino con el estándar CSS3. Se trata de un conjunto de propiedades pensadas para facilitar el trabajo del diseñador web, distribuyendo los elementos de una interfaz con mayor precisión en los ejes X y Y, ahorrando tiempo y líneas de código. Su creación data del año 2009 pero es en el 2014 que se vuelve parte de la especificación con la sintaxis actual.

Entre las ventajas de usar flexbox se encuentran:

- Se pueden crear diseños responsive con menos líneas de código.
- Permite la alineación horizontal y vertical de los elementos.
- Re-ordena el contenido sin manipular el HTML.



### 5.3.- Primeros Pasos Con Flexbox

Para empezar a usar flexbox se define en el css, la propiedad display con el valor flex, tal como se muestra a continuación:

The screenshot shows a code editor with two tabs: 'HTML' and 'CSS'. The 'HTML' tab contains the following code:

```

1 <html>
2 <head>
3   <title>Flexbox</title>
4 </head>
5 <body>
6   <div class="flex-container">
7     <div class="flex-item"></div>
8     <div class="flex-item"></div>
9     <div class="flex-item"></div>
10   </div>
11 </body>
12 </html>

```

The 'CSS' tab contains the following code:

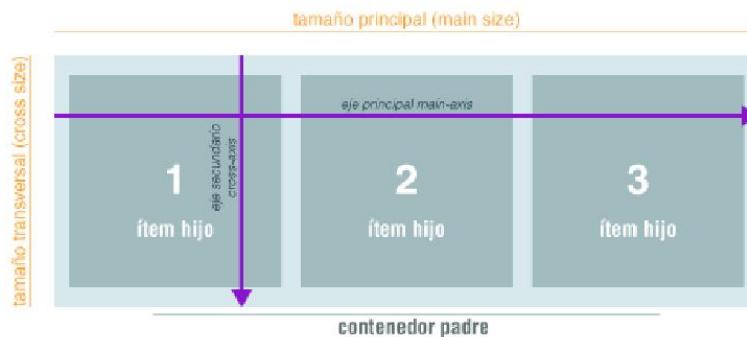
```

1 .flex-container{
2   display:flex;
3 }

```

En este sentido, si se desea comprender cómo funciona Flexbox, se deben conocer los siguientes conceptos:

- Contenedor: elemento padre que poseerá cada uno de los elementos flexibles y adaptables.
- Ítem: elementos hijos del contenedor que tendrán un comportamiento flexible según el elemento padre.
- Main Axis: orientación principal del contenedor flexible, por defecto corresponde al eje horizontal.
- Cross Axis: orientación secundaria del contenedor flexible y está determinada por el eje vertical.



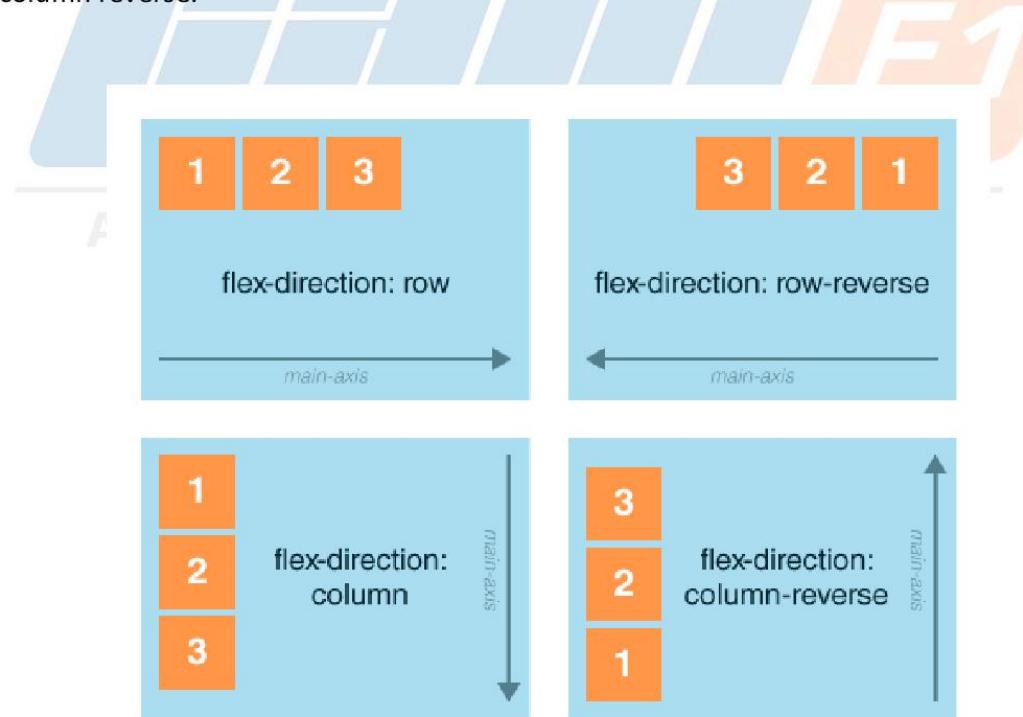
## Capítulo 6. FLEXBOX Y SUS PROPIEDADES

### 6.1.- Propiedades Del Flex Container (padre)

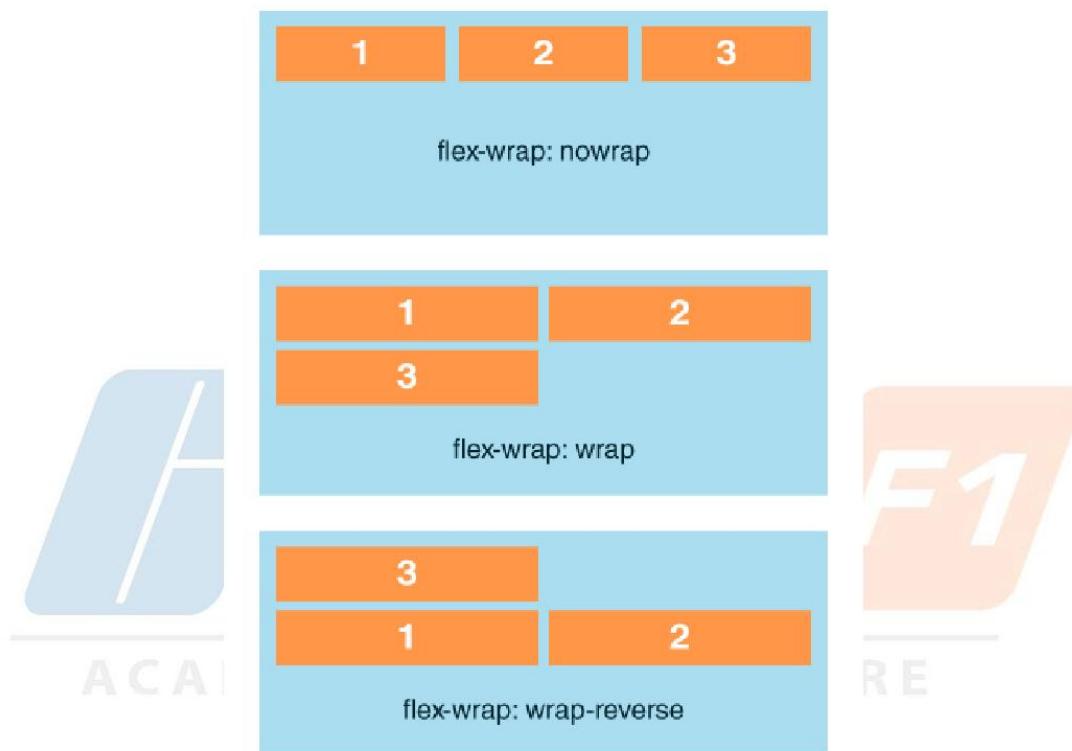
En el capítulo anterior se pudo apreciar el uso de una etiqueta padre y varias etiquetas hijas, este comportamiento cumple con la primera regla “Flexbox necesita un parente y por lo menos un hijo” dado que no hay forma de manipular directamente un elemento flexible sin crear antes su contexto. En otras palabras, al definir un `display:flex;` en un elemento html realmente se le indica al navegador que se trabajará con un contenedor parente y dentro de él estarán los ítems flexibles.

Las propiedades usadas para la manipulación del flex-container son:

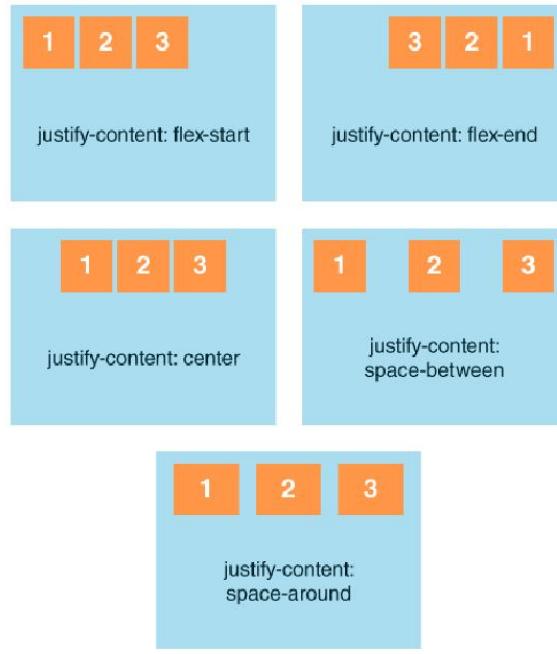
`flex-direction:` permite alterar la orientación de los elementos en el eje principal y secundario (horizontal y vertical). Sus valores pueden ser: `row`, `row-reverse`, `column` y `column-reverse`.



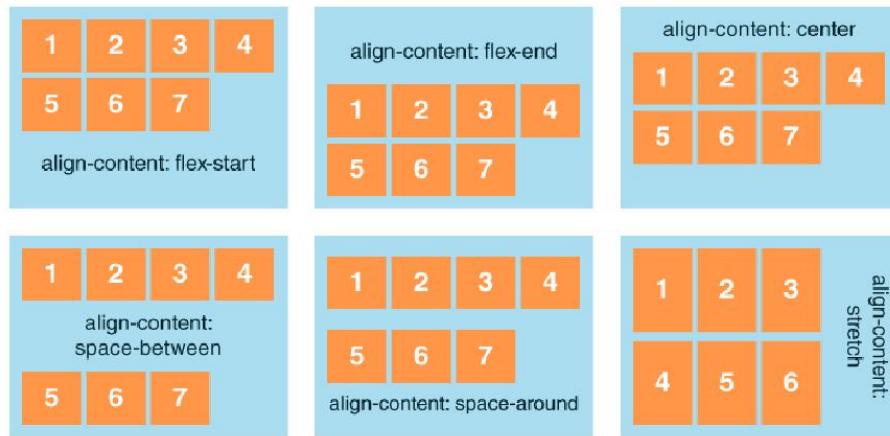
flex-wrap: controla la dirección de los elementos según su anchura y la del contenedor padre. Sus valores son: nowrap, wrap, wrap-reverse.



justify-content: permite la alineación y distribución de los elementos de manera horizontal por defecto, los espacios son tratados de forma equitativa. Sus valores son: flex-start, flex-end, center, space-between, space-around.



justify-content, sin embargo, su alineación es hacia el eje vertical por defecto. Los valores que puede adquirir esta propiedad son: flex-start, flex-end, center, baseline, stretch.



## 6.2.- Propiedades de Los Flex Ítems (hijos)

Por otro lado, aunque se han ido manejando los elementos de la caja padre, se pueden controlar comportamientos específicos de cada uno de estos ítems con las siguientes propiedades:

order: ordena los elementos del HTML de forma visual, estableciendo una posición con un número, por defecto todos los ítems tienen un orden 0 y se muestran según su estructura.

The screenshot shows a code editor with two panes. The left pane is labeled "HTML" and contains the following code:

```
1 <html lang="es">
2   <head>
3     <title>Flexbox - order</title>
4   </head>
5   <body>
6     <div class="flex-container">
7       <div style="order: 3">1</div>
8       <div style="order: 2">2</div>
9       <div style="order: 4">3</div>
10      <div style="order: 1">4</div>
11    </div>
12  </body>
13 </html>
```

The right pane is labeled "CSS" and contains the following code:

```
1 .flex-container {
2   display: flex;
3 }
4 .flex-container>div {
5   background-color: DodgerBlue;
6   color: white;
7   width: 50px;
8   margin: 10px;
9   text-align: center;
10  padding: .5em;
11  font-size: 30px;
12 }
```

Below the CSS pane, there are four blue rectangular boxes arranged horizontally, each containing a number: 4, 2, 1, and 3. These numbers correspond to the order properties defined in the CSS code.

flex-grow: define el tamaño de crecimiento del elemento ítem y éste crecerá en relación a sus hermanos dentro del contenedor flexible. Su valor, al igual que la propiedad order, se establece con un número.



```

HTML
1 <html lang="es">
2 <head>
3   <title>Flexbox - flex-grow</title>
4 </head>
5 <body>
6   <div class="flex-container">
7     <div style="flex-grow: 3">1</div>
8     <div style="flex-grow: 1">2</div>
9     <div style="flex-grow: 8">3</div>
10   </div>
11 </body>
12 </html>

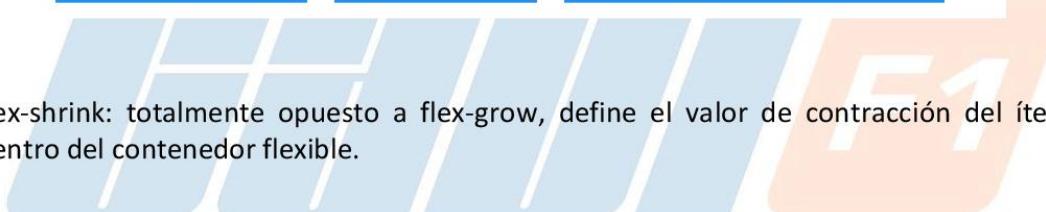
```

```

CSS
1 .flex-container {
2   display: flex;
3   width: 50%;
4 }
5 .flex-container>div {
6   background-color: DodgerBlue;
7   color: white;
8   width: 100px;
9   margin: 10px;
10  text-align: center;
11  padding:.5em;
12  font-size: 30px;
13 }

```

**flex-shrink:** totalmente opuesto a flex-grow, define el valor de contracción del ítem dentro del contenedor flexible.




```

HTML
1 <html lang="es">
2 <head>
3   <title>Flexbox - shrink</title>
4 </head>
5 <body>
6   <div class="flex-container">
7     <div>1</div>
8     <div>2</div>
9     <div style="flex-shrink: 10">3</div>
10    <div>4</div>
11    <div>5</div>
12   </div>
13 </body>

```

```

CSS
1 .flex-container {
2   display: flex;
3   width: 50%;
4 }
5 .flex-container>div {
6   background-color: DodgerBlue;
7   color: white;
8   width: 100px;
9   margin: 10px;
10  text-align: center;
11  padding:.5em;
12  font-size: 30px;
13 }

```

flex-basis: define el tamaño de un elemento antes de que se distribuya el espacio restante. Su comportamiento es similar a width y su valor puede ser en píxeles, porcentaje, etc.

The screenshot shows a code editor with two panes. The left pane is labeled "HTML" and contains the following code:

```
1 <html lang="es">
2   <head>
3     <title>Flexbox - basis</title>
4   </head>
5   <body>
6     <div class="flex-container">
7       <div>1</div>
8       <div>2</div>
9       <div style="flex: 0 0 300px">3</div>
10      <div>4</div>
11    </div>
12  </body>
13 </html>
```

The right pane is labeled "CSS" and contains the following code:

```
1 .flex-container {
2   display: flex;
3   width: 50%;
4 }
5 .flex-container>div {
6   background-color: DodgerBlue;
7   color: white;
8   width: 100px;
9   margin: 10px;
10  text-align: center;
11  padding: .5em;
12  font-size: 30px;
13 }
```

Below the code editor, there is a visual representation of four blue rectangular boxes labeled 1, 2, 3, and 4 from left to right. Box 3 is wider than boxes 1, 2, and 4, illustrating its larger flex-basis value.

ACADEMIA DE SOFTWARE

## Capítulo 7. CSS GRID. PARTE 1

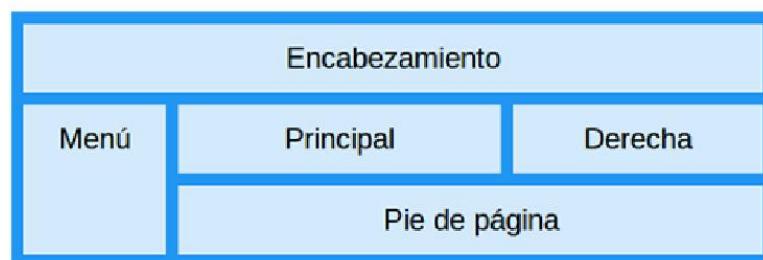
### 7.1.- Definición

Uno de los procesos más problemáticos y frustrantes de CSS, sobre todo para novatos o principiantes, es el proceso de colocar y distribuir los elementos a lo largo de una página. Mecanismos como posicionamiento, floats o elementos en bloque o en línea, suelen ser insuficientes (o muy complejos) para crear un layout o estructuras para páginas web actuales.

El sistema flexbox es una gran mejora, sin embargo, está orientado a estructuras de una sola dimensión, por lo que aún se necesita algo más potente para estructuras web. Con el paso del tiempo, muchos frameworks y librerías utilizan un sistema grid donde definen una cuadrícula determinada, y modificando los nombres de las clases de los elementos HTML, podemos darle tamaño, posición o colocación.

El Diseño de Grid CSS ofrece un sistema basado en el diseño de la red, con filas y columnas, por lo que es más fácil diseñar páginas web sin tener que utilizar flotadores y posicionamiento.

En la imagen se puede observar lo que podría lograrse al dividir el layout en filas y columnas usando la Grilla de CSS.



Respecto al tema de la compatibilidad con los navegadores tenemos que las propiedades de la cuadrícula son compatibles con todos los navegadores modernos.

57.0	16.0	52.0	10	44

## 7.2.- Elementos Del Grid

Es necesario entender que una grilla css está comprendida por un contenedor y una serie de ítems dentro de este. Para hacer que un elemento HTML se comporte como un contenedor de la grilla, tiene que establecer la propiedad display con el valor grid o inline-grid

Grid: establece que los ítems de la grilla tengan propiedades de bloque y sus anchos se ajusten al espacio disponible.

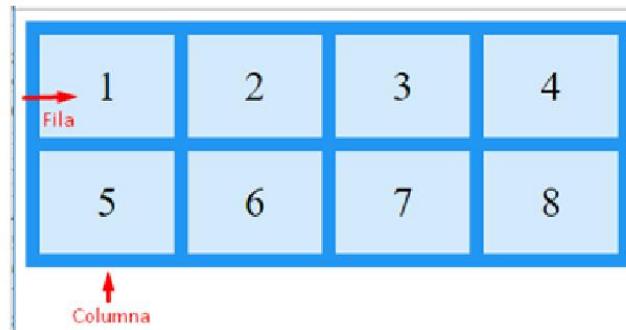
Inline-grid: establece que los ítems de la grilla se comporten como elementos de línea, ajustando sus anchos a su propio contenido.

Contenedores de grid consisten en elementos de la cuadrícula, colocados dentro de las columnas y filas.

En la siguiente imagen se puede observar el uso del valor “grid” de la propiedad display para lograr el diseño de la cuadrícula. Hay algunas otras propiedades involucradas que se explicaran en breve.

```
<style type="text/css">
.grid-container {
    display: grid;
    grid-template-columns: auto auto auto auto;
    grid-gap: 10px;
    background-color: #2196F3;
    padding: 10px;
}
.grid-container > div {
    background-color: rgba(255, 255, 255, 0.8);
    text-align: center;
    padding: 20px 0;
    font-size: 30px;
}
</style>
</head>
<body>
<div class="grid-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
    <div>7</div>
    <div>8</div>
</div>
</body>
</html>
```

La visualización del archivo html de la lámina anterior, se muestra en la siguiente imagen. Esta es una rejilla de 4 columnas con anchos automáticos.

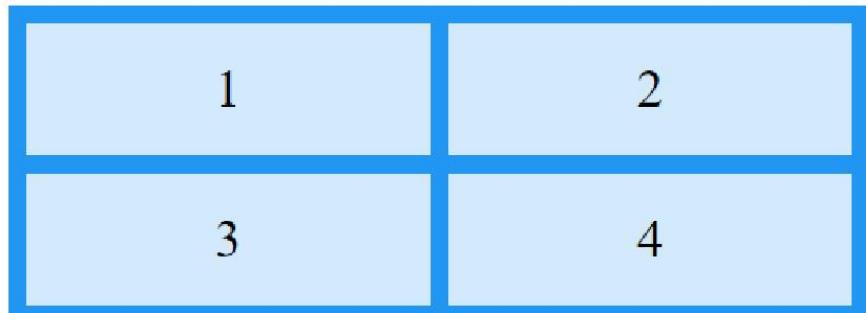


La propiedad “grid-template-columns” define el número de columnas en su diseño de cuadrícula, y se puede definir la anchura de cada columna.

El valor es una lista separada por espacios, donde cada valor define la longitud de la columna respectiva.

Si desea que su diseño de cuadrícula que contiene 4 columnas, especifique el ancho de las columnas 4(px, em, % o cualquier otra unidad), o "auto" si todas las columnas deben tener la misma anchura.

Si se define la propiedad `grid-template-columns: auto auto` para un `grid-container` con 4 items se tendría lo siguiente:



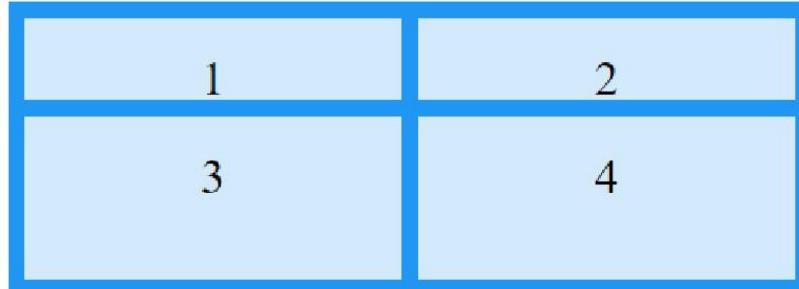
HTML

```
<div class="grid-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
</div>
```

CSS

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto;
```

La propiedad “grid-template-rows” hará exactamente lo mismo que la propiedad anterior, pero esta vez haciendo referencia las filas de la cuadricula. Si para el ejemplo anterior se utiliza esta propiedad para modificar el tamaño de las filas, se tendría lo siguiente:



HTML

```
<div class="grid-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
</div>
```

CSS

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto;
  grid-template-rows: 50px 100px;
}
```

### 7.3.- Espacio Entre Filas y Columnas

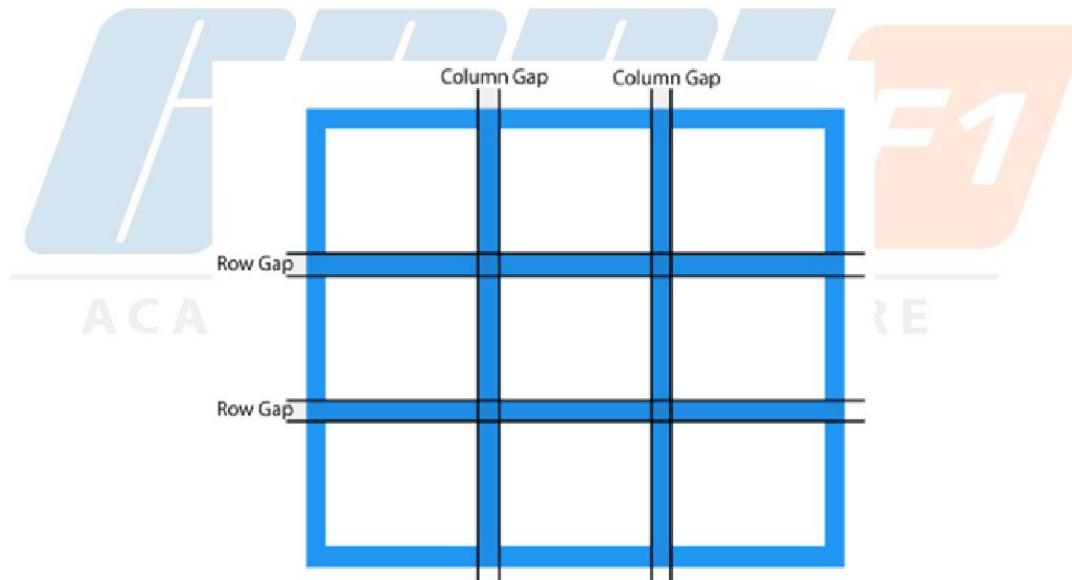
Las grillas usualmente se ven acompañadas de propiedades que establecen espacios entre sus filas y sus columnas, de manera de lograr un diseño un poco más ordenado y agradable a la vista. Existen espacios entre filas y espacios entre columnas de la grilla, donde la combinación de ellas logrará el efecto deseado del diseño.

Es por esto que para CSS Grid se tienen las siguientes 2 propiedades:

grid-column-gap: Espacio entre columnas

grid-row-gap: Espacio entre filas.

La imagen muestra gráficamente los espacios anteriormente mencionados



Los valores para las propiedades grid-column-gap y grid-row-gap pueden estar definidas en múltiples unidades (% , em, px, cm, mm entre otros). Si se desea establecer espaciado idéntico para filas y columnas, se puede usar la propiedad grid-gap.

```
.grid-container {  
    display: grid;  
    grid-template-columns: auto auto;  
    grid-template-rows: 50px 100px;  
    grid-gap: 10px;  
    background-color: #2196F3;  
    padding: 10px;  
}
```



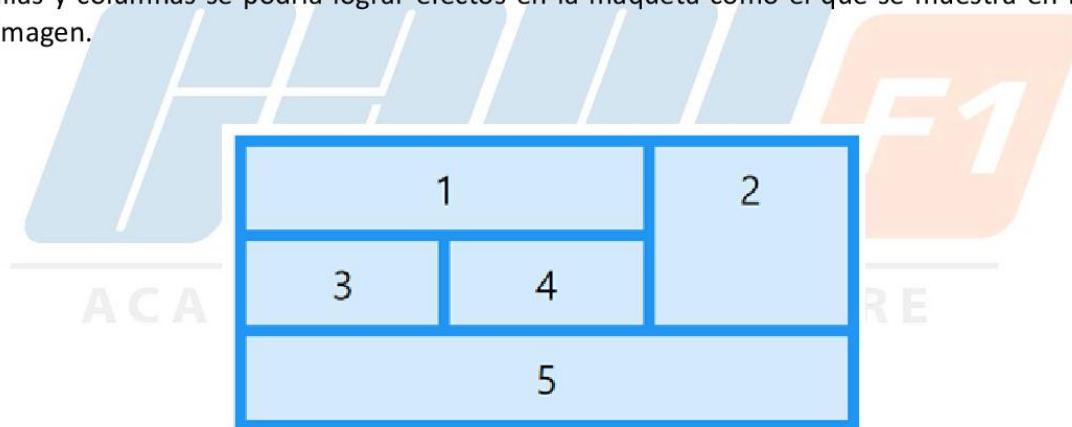
## Capítulo 8. CSS GRID. PARTE 2

### 8.1.- Grid Items

Una rejilla recipiente contiene rejilla artículos que suelen llamarse Grid Items. Hasta ahora se han realizado grillas cuadradas, donde cada elemento tienen la misma proporción.

Por defecto, un contenedor tiene un ítem para cada columna, en cada fila, pero puede estilizar los elementos de la cuadrícula para que puedan abarcar varias columnas y / o filas.

Modificando el ancho de los ítems del grid para que ocupen diferentes cantidades de filas y columnas se podría lograr efectos en la maqueta como el que se muestra en la imagen.



### 8.2.- Modificando Los Elementos Del Grid

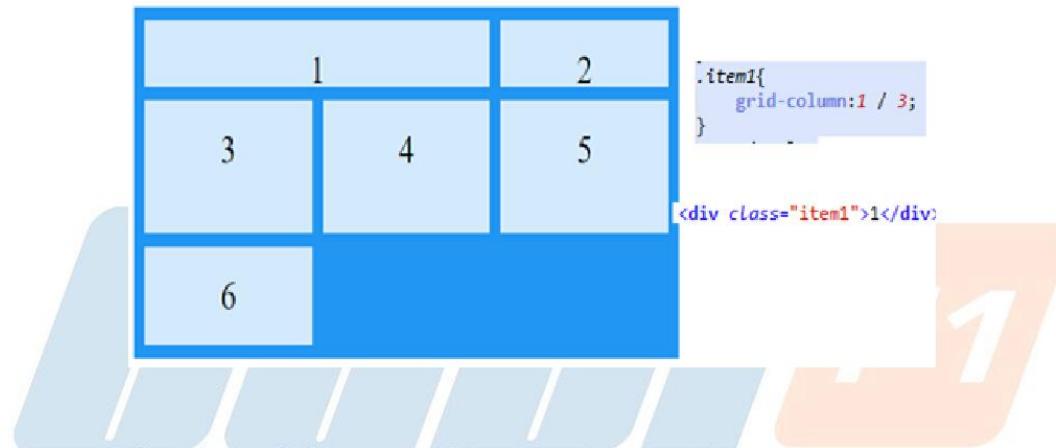
La propiedad `grid-column` define en el que la columna (s) para colocar un elemento. La propiedad define en que columna comenzará y en que columna terminará el artículo.

La propiedad `grid-column` define en el que la columna (s) para colocar un elemento.

La propiedad define en que columna comenzará y en que columna terminará el artículo.

Las sintaxis para el valor de esta propiedad es la siguiente:  
grid-column: columnalnicial / columnaFinal

En la imagen se muestra un Grid de 3 columnas y 2 filas, donde al primer elemento se le asigna la propiedad grid-column: 1 / 3. El primer elemento del grid comenzará en la primera columna y terminará donde comienza la tercera.

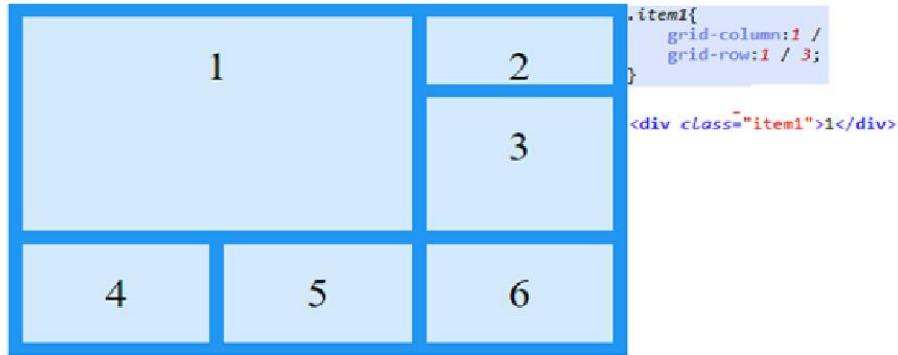


La propiedad grid-row define en qué fila para colocar un elemento.

Se define en qué fila el elemento empezará, y donde va a terminar el artículo.

La propiedad define en que columna comenzará y en que columna terminará el artículo. SU sintaxis es exactamente igual a la propiedad grid-column.

Si para el mismo ejemplo anterior se agrega al primer elemento un grid-row: 1 / 3 , es decir, se quiere que el elemento comience en la primera fila y termine al comienzo de la tercera, se tendría la siguiente salida:



justify-content: permite la alineación y distribución de los elementos de manera horizontal por defecto, los espacios son tratados de forma equitativa. Sus valores son: flex-start, flex-end, center, space-between, space-around.



## Capítulo 9. PREPROCESADORES

### 9.1.- ¿Qué es un preprocesador?

Un preprocesador es una herramienta que permite escribir pseudo-código que luego será convertido a un archivo (formato) real. Ese pseudo-código se conforma de variables, condiciones, bucles o funciones. Se podría decir que tiene un lenguaje de programación que genera HTML y CSS. El objetivo de estos preprocesadores es tener un código más sencillo de mantener y editar.



### 9.2.- Beneficios de usar un preprocesador

Organizar mejor el código.

Los pre-procesadores de CSS permiten anidar selectores, por ejemplo, agrupar los distintos estados de los enlaces.

```
a
  text-decoration: none
  &:hover
    text-decoration: underline
```

Usar variables.

En vez “buscar y reemplazar” para cambiar por ej. un color, se puede definir como variable.

```
$color-principal: #009B77
h1
  border: 5px solid $color-principal
```

Reutilizar código.

Los mixins te permiten re-utilizar bloques de código.

```
@mixin transicion($propiedad, $duracion:0.3s, $tweening:ease-out)
  -webkit-transition: $propiedad $duracion $tweening;
  -moz-transition: $propiedad $duracion $tweening;
  -o-transition: $propiedad $duracion $tweening;
  transition: $propiedad $duracion $tweening;

a
  @include transicion(color, 0.5s);
```

Hacer operaciones.

En vez de usar la calculadora y explicar en un comentario de dónde sacaste ese número, se puede hacer la cuenta directamente.

```
$margenes: 0.5em;  
  
h1  
    margin-bottom: $margenes * 4  
  
h2  
    padding-bottom: $margenes * 2
```

Usar funciones.

Por ej. para un degradé o para diferentes estados de un botón o enlace, se necesitan variantes de un color, hexadecimal no es un sistema particularmente intuitivo, así que en vez de usar una aplicación para elegir los colores, se puede pedirle al pre-procesador un color más claro, oscuro, transparente o saturado.

ACADEMIA DE SOFTWARE

```
$negro: #2C3E50;  
  
.navbar  
    background-image: linear-gradient(to bottom, $negro,  
    ligthen($negro, 10%));
```

Comentando en el código.

Con los pre-procesadores, hay 2 tipos de comentarios, los clásicos que se incluyen en la compilación a CSS y los estilo JavaScript que quedan sólo en el archivo original, más privados, para ti mismo y tus compañeros de trabajo.

```
.btn  
  // Falta el resto de los prefijos. Usar un mixin!  
  background-image: linear-gradient(to bottom, $colorEnlace,  
  ligthen($colorEnlace, 10%));
```

### Incluir archivos.

Puedes tener unas reglas básicas en un archivo maestro e incluirlo en todos tus proyectos, para no repetirte. Reduces las peticiones al servidor ya que todo va a estar incluido en un sólo archivo.

```
@import "normalize"  
@import "css3-mixins"
```

### 9.3.- Tipos de preprocesadores

Actualmente existen muchos preprocesadores, aunque las características principales son comunes en casi todos. Los más populares son SASS, Less y Stylus para código CSS, y HAML y Jade para HTML.



## Capítulo 10. PREPROCESADOR HTML: HAML

### 10.1.- ¿Qué es HAML?

Es un lenguaje de marcado ligero que se usa para describir el HTML de un documento web sin emplear el código empotrado tradicional. Está diseñado para solucionar varios problemas de los motores de plantillas tradicionales y también para ser un lenguaje de marcado tan elegante como sea posible.

Sirviendo como su propio lenguaje de marcado, el código escrito en Haml es posteriormente procesada a HTML. Haml promueve marcado seco y bien estructurado, que proporciona una experiencia agradable para cualquier persona que tenga que escribir o leer código.



En Ruby se usa mucho ERB como sistema de plantillas para crear archivos HTML con código Ruby empotrado. Ahí entra HAML, un lenguaje de marcado ligero con el cual se puede generar HTML a partir de un sencillo archivo. Haml busca embellecer las plantillas

y deshacerse de los lenguajes "feos" que viene utilizando en las plantillas HTML. Resulta sumamente útil para desarrollo web en general, no solo para usar con Ruby.

Haml puede ser usado dentro de una aplicación web o por su cuenta.

### 10.2.- Principios

Al usar pre procesadores, se deberían tomar en cuenta algunas buenas prácticas:

- El marcado debería ser bonito. El marcado no debería usarse solamente como una herramienta para lograr que los navegadores muestren una página como el autor lo deseé. El renderizado no es lo único que la gente tiene que ver; también se tiene que ver, modificar y entender el marcado. Por tanto, este debería ser tan amigable y placentero como el resultado del renderizado.
- El marcado no debería repetirse. Haml evita todo esto recurriendo al sangrado en vez del texto para determinar dónde empiezan y terminan los elementos y los bloques de código. Esto no solo da como resultado plantillas más compactas sino que además hace el código mucho más limpio a la vista.
- El marcado debería tener buen sangrado. Haml formatea las etiquetas de manera que todas estén bien sangradas y reflejen la estructura subyacente del documento.
- La estructura de HTML debería ser clara. Debido a que la lógica de Haml se basa en el sangrado de elementos hijos, esta estructura se preserva naturalmente, lo que hace al documento mucho más fácil y lógico de leer por simples humanos.

El siguiente es un ejemplo del uso de etiquetas Haml y el código HTML resultante:

```
<!DOCTYPE html>
<html>
<head>
  <title>CADI F1</title>
</head>
<body>
  <h1>Nivel III de HTML5 y CSS3</h1>
  <p>Uso de preprocesadores</p>
  <ul class='menu'>
    <li>HTML5</li>
    <li>CSS3</li>
  </ul>
</body>
</html>
```

### 10.3.- Sintaxis

El carácter de porcentaje (%) se debe colocar al principio de una línea. Seguido inmediatamente por el nombre de un elemento (etiqueta HTML), a continuación, opcionalmente mediante modificadores (atributos), un espacio o un salto de línea, y el texto a ser representado dentro del elemento. Cualquier cadena es un nombre de elemento válido; HAML generará automáticamente la apertura y cierre de etiquetas para cualquier elemento.

```
1 %h1 Preprocesador HAML
2
3 %h1
4 | Preprocesador HAML
```

## Capítulo 11. MAQUETACIÓN CON HAML

### 11.1.- Atributos

Entre paréntesis se representan los atributos de un elemento. Se colocará dos puntos y luego el nombre del atributo, posterior el signo => y entre comillas el valor del atributo.

```
%html { :lang => "es"} 1 <html lang="es">
```

Los atributos también se pueden colocar en varias líneas para dar cabida a muchos atributos. Sin embargo, las nuevas líneas sólo pueden ser colocados inmediatamente después de las comas.

```
%img{ :src => 'logo.png',  
      alt=>'Logo de la empresa'}
```

HAML también soporta otra forma de colocar atributos. Estos se utilizan con paréntesis en lugar de corchetes.

```
%html (lang="es")  
%img (src='logo.png')
```

### 11.2.- Clases e ID

El punto (.) y el numeral (#) son tomados de CSS. Se utilizan como atajos para especificar los atributos class e id de un elemento, respectivamente. Los nombres de clases

múltiples se pueden especificar de una manera similar a CSS, por el encadenamiento de los nombres de las clases. Se colocan inmediatamente después de la etiqueta.

```
%div#curso
  %span.articulo
    %h1.sass
      SASS
    %h2.subtitleo.primario
      Introducción
```

```
1 <div id='curso'>
2   <span class='articulo'></span>
3   <h1 class='sass'>
4     SASS
5   </h1>
6   <h2 class='subtitleo primario'>
7     Introducción
8   </h2>
9 </div>
```

### 11.3.- DIV Implícito

Debido a que los divs se utilizan tan a menudo, son los elementos predeterminados. Si sólo se define una clase o un id usando . o # respectivamente, un div se colocará automáticamente.

```
#contenido
  .articulo
    %h3 SASS
```

Es lo mismo:

```
div#contenido
  div.articulo
    %h3 SASS
```

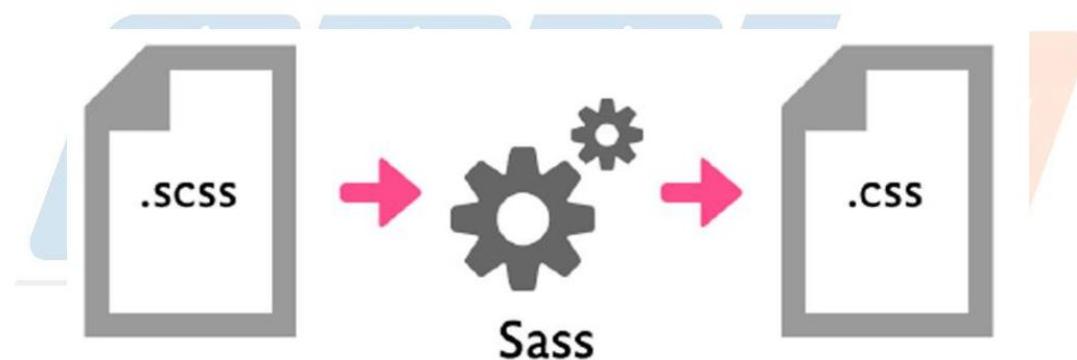
```
1 <div id='contenido'>
2   <div class='articulo'>
3     <h3>SASS</h3>
4   </div>
5 </div>
```

## Capítulo 12. PREPROCESADORES CSS: SASS

### 12.1.- Preprocesadores en CSS

Un preprocesador de CSS es una herramienta que permite escribir pseudo-código CSS que luego será convertido a CSS real. Ese pseudo-código se conforma de variables, condiciones, bucles o funciones. Podríamos decir que tenemos un lenguaje de programación que genera CSS.

El objetivo de estos preprocesadores es tener un código más sencillo de mantener y editar. Los preprocesadores incluyen características tales como variables, funciones, mixins, anidación o modularidad.



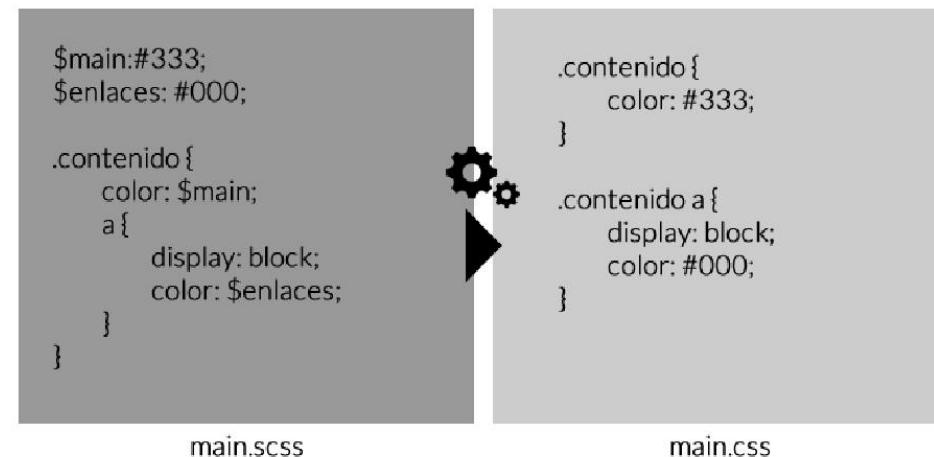
Actualmente existen muchos preprocesadores, aunque las características principales son comunes en casi todos. Los más populares son Less, Sass y Stylus. En el curso se utilizará SASS ya que tiene una comunidad y referencia mucho más extensa.



## 12.2.- Ventajas de Utilizar Preprocesadores

- Utilización de los CSS con comportamientos dinámicos (variables, clases, operaciones, sentencias, métodos y funciones).
- Capa de abstracción, donde no se trabaja directamente sobre los archivos css.
- Separación absoluta de desarrollo y producción.
- Reducción considerable del tiempo de desarrollo, ya que puede escribir menos código y más limpio.
- Cambios menos costosos, ya que con sustituir el valor de una variable puede ahorrar mucho tiempo y trabajo.
- Reutilización de bloques de código.

### ACADEMIA DE SOFTWARE

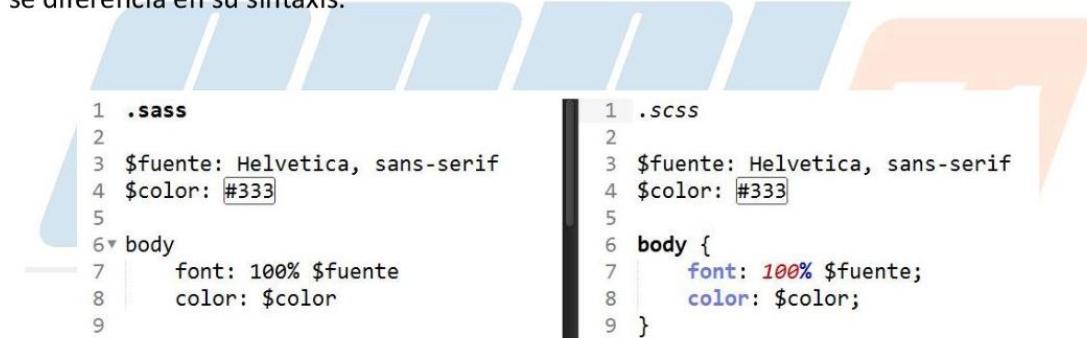


### 12.3.- Sintaxis en Sass

Sass permite el uso de dos sintaxis diferentes para crear sus archivos. La primera, conocida como SCSS (del inglés, Sassy CSS) y es una extensión de la sintaxis de CSS3. Esto significa que cualquier hoja de estilos CSS3 válida también es un archivo SCSS válido.

La segunda sintaxis, conocida como "sintaxis identada" o simplemente "sintaxis sass" permite escribir los estilos CSS de manera más concisa. En este caso, el anidamiento de selectores se indica con tabulaciones en vez de con llaves y las propiedades se separan con saltos de línea en vez de con puntos y coma. Los archivos creados con esta segunda sintaxis utilizan la extensión .sass.

En cualquier caso, las dos sintaxis tienen exactamente las mismas funcionalidades y sólo se diferencia en su sintaxis.



```
1 .sass
2
3 $fuente: Helvetica, sans-serif
4 $color: #333
5
6 ▾ body
7   font: 100% $fuente
8   color: $color
9

1 .scss
2
3 $fuente: Helvetica, sans-serif
4 $color: #333
5
6 body {
7   font: 100% $fuente;
8   color: $color;
9 }
```

## Capítulo 13. SASS - ALCANCE, REFERENCIA Y PROPIEDADES

### 13.1.- Alcance de un selector - Anidamiento

Sass permite anidar las reglas CSS para que las hojas de estilos sean más concisas y fáciles de escribir. A los selectores anidados se les prefija automáticamente todos los selectores de los niveles superiores.

Gracias a las reglas anidadas, se evita tener que repetir una y otra vez los mismos selectores y se simplifica enormemente la creación de hojas de estilos complejas.

El siguiente es un ejemplo de un archivo pre procesado y el archivo css resultante:

```

1 .recuadro {
2   width: 500px;
3   .mensaje {
4     border: 1px solid red;
5     .titulo {
6       color: red;
7     }
8     .usuario {
9       border: 2px solid black;
10    .titulo {
11      color: black;
12    }
13  }
14}
15 }
```

### 13.2.- Referenciando a los selectores padre

En ocasiones es necesario modificar el comportamiento por defecto de los selectores anidados. Imagina que quieres aplicar estilos especiales en el estado hover del selector o cuando el elemento body de la página tiene una determinada clase.

En estos casos, puedes utilizar el carácter (&) para hacer referencia al selector padre dentro del cual se encuentra la regla anidada.

```

1 a {
2   font-weight: bold;
3   text-decoration: none;
4   &:hover
5     text-decoration: underline
6   body.firefox &
7     font-weight: normal
8
9
10
11
12
1 a {
2   font-weight: bold;
3   text-decoration: none;
4 }
5
6 a:hover {
7   text-decoration: underline;
8 }
9
10 body.firefox a {
11   font-weight: normal;
12 }

```

### 13.3.- Propiedades anidadas

CSS define varias propiedades cuyos nombres parecen estar agrupados de forma lógica. Así por ejemplo, las propiedades font-family, font-size y font-weight están todas relacionadas con el grupo font. En CSS es obligatorio escribir el nombre completo de todas estas propiedades. Sass permite utilizar atajo para definir las propiedades relacionadas.

```

1 .header
2   font:
3     family: Arial
4     size: 20em
5     weight: bold
6
1 .header {
2   font-family: Arial;
3   font-size: 20em;
4   font-weight: bold;
5 }
6

```

## Capítulo 14. SASS - VARIABLES, OPERACIONES Y COLORES

### 14.1.- Uso de variable

La funcionalidad básica de Sass es el uso de variables para almacenar valores que utiliza una y otra vez en las hojas de estilos. Para ello, utiliza cualquier palabra como nombre de la variable, se le añade el símbolo \$ por delante y establece su valor como si fuera una propiedad CSS normal. Puede utilizar variables para representar los colores, tamaños, porcentajes, etc. El siguiente es un ejemplo de uso de variables:

```

1 $fuente-principal: Helvetica, sans-serif
2 $color-primario: #333
3
4 body
5   font: 100% $fuente-principal
6   color: $color-primario

```

```

1 body {
2   font: 100% Helvetica, sans-serif;
3   color: #333;
4 }
5

```

### 14.2.- Operaciones en el layout

Una vez que ha definido una variable, Sass permite realizar operaciones básicas en esa variable con operadores como sumar, restar, multiplicar y dividir (+, -, \* y /). Por ejemplo:

```

1 .contendor
2   width: 100%
3
4▼ article[role="main"]
5   float: left
6   width: 600px / 960px * 100%
7
8▼ aside[role="complementary"]
9   float: right
10  width: 300px / 960px * 100%
11
12
13

```

```

1 .contendor {
2   width: 100%;
3 }
4
5▼ article[role="main"] {
6   float: left;
7   width: 62.5%;
8 }
9
10▼ aside[role="complementary"] {
11   float: right;
12   width: 31.25%;
13 }

```

### 14.3.- Creación de colores

En el uso de color, Sass permiten manipular el color y hacer muchas conversiones muy fáciles gracias a las funciones que trae integradas como aclarar y oscurecer, saturar y desaturar, entre otras más.

```
1 $color-normal: #333
2 body
3   background: $color-normal
4
5 $color-claro: #555
6 $color-oscuro: #222
7 .body
8   background: lighten($color-claro, 52%)
9   background: darken($color-oscuro, 52%)
10
11 $color-saturado: #af3df1
12 #body
13   background: saturate($color-saturado,
14     15%)
15 #header
16   background: mix(#222, #993233)
17
```

```
1 body {
2   background: #333;
3 }
4
5 .body {
6   background: #dadada;
7   background: black;
8 }
9
10 #body {
11   background: #b32fff;
12 }
13
14 #header {
15   background: #5e2a2b;
16 }
17
```

ACADEMIA DE SOFTWARE

## Capítulo 15. SASS - IMPORTACIÓN, MIXINS Y HERENCIA

### 15.1.- Importar archivos

CSS tiene una opción de importación que le permite dividir el archivo CSS en porciones más pequeñas, más fáciles de mantener. El único inconveniente es que cada vez que utilice @import en CSS se crea otra petición HTTP en el servidor. Sass construye en la parte superior del archivo CSS (@import), pero en lugar de requerir una solicitud HTTP, Sass tomará el archivo que desea importar y lo combina con el archivo que está importando para que pueda usar un solo archivo CSS para el navegador.

```

1 html,
2 body,
3 ul,
4 ol
5 margin: 0
6 padding: 0
7
8
9
10
11
12
13
1 @import fuente
2
3 body
4 font: 100% Helvetica, sans-serif
5 background-color: #efefef
6
7
8
9
10
11
12
13
1 html,
2 body,
3 ul,
4 ol {
5 margin: 0;
6 padding: 0;
7 }
8
9 body {
10 font: 100% Helvetica, sans-serif;
11 background-color: #efefef;
12 }
13

```

### 15.2.- Mixins

Algunas cosas en CSS son tediosas de escribir, especialmente con CSS3 y los muchos prefijos de proveedores existen. Un mixin le permite hacer grupos de declaraciones CSS que luego puede volver a utilizar. Por ejemplo:

```

1 ▶ =border-radius($radius)
2   -webkit-border-radius: $radius
3   -moz-border-radius: $radius
4   -ms-border-radius: $radius
5   border-radius: $radius
6
7 .box
8   +border-radius(10px)
9
1 .box {
2   -webkit-border-radius: 10px;
3   -moz-border-radius: 10px;
4   -ms-border-radius: 10px;
5   border-radius: 10px;
6 }
7

```

### 15.3.- Extender / Herencia

Esta es una de las características más útiles de Sass. Usando @extend le permite compartir un conjunto de propiedades CSS de un selector a otro.

```
1*.mensaje  
2  border: 1px solid #ccc  
3  padding: 10px  
4  color: #333  
5*.correcto  
6  @extend .mensaje  
7  border-color: green  
8*.error  
9  @extend .mensaje  
10 border-color: red  
11*.precaucion  
12 @extend .mensaje  
13 border-color: yellow  
14  
15  
16  
17  
18
```

```
1 .mensaje, .correcto, .error, .precaucion {  
2  border: 1px solid #ccc;  
3  padding: 10px;  
4  color: #333;  
5 }  
6  
7 .correcto {  
8  border-color: green;  
9 }  
10  
11 .error {  
12  border-color: red;  
13 }  
14  
15 .precaucion {  
16  border-color: yellow;  
17 }  
18
```

ACADEMIA DE SOFTWARE