



**HTML**



**CSS**



## **HTML 5 y CSS 3. Nivel III**

noviembre, 2019



## 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 nuestro 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 2019. 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.

### 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. Recomendaciones Para Responsive Design

- 5.1.- Recomendaciones.
- 5.2.- Errores Frecuentes.

### Capítulo 6. Flexbox. Parte 1

- 6.1.- ¿qué es el Layout?.
- 6.2.- Introducción a Flexbox.
- 6.3.- Primeros Pasos Con Flexbox.

## Capítulo 7. Flexbox. Parte 2

- 7.1.- Propiedades Del Flex Container (padre).
- 7.2.- Propiedades de Los Flex Ítems (hijos).

## Capítulo 8. Flexbox. Parte 3

- 8.1.- Align-self.
- 8.2.- Flex-flow.
- 8.3.- Flex.
- 8.4.- Compatibilidad.

## Capítulo 9. Css Grid. Parte 1

- 9.1.- Definición.
- 9.2.- Elementos Del Grid.
- 9.3.- Espacio Entre Filas y Columnas.

## Capítulo 10. Css Grid. Parte 2

- 10.1.- Grid Items.
- 10.2.- Modificando Los Elementos Del Grid.

## Capítulo 11. Css Grid. Parte 3

- 11.1.- Grid Área.
- 11.2.- Valores Automáticos de Una Grilla.

## Capítulo 12. Css Grid. Parte 4

- 12.1.- Propiedades de Elemento de Grilla.
- 12.2.- Atajos de Css Grid.

## Capítulo 13. Estructuras de Diseño Reponsive

- 13.1.- Mostly Fluid.
- 13.2.- Column Drop.

13.3.- Layout Shifter.

13.4.- Estrategias Para Hacer Responsive Web Design.

#### Capítulo 14. Selectores de Contenido

14.1.- After y Before.

14.2.- First Line.

14.3.- First Letter.

#### Capítulo 15. Manejo de Eventos en Responsive

15.1.- Pointer Events.

15.2.- Touch Element.



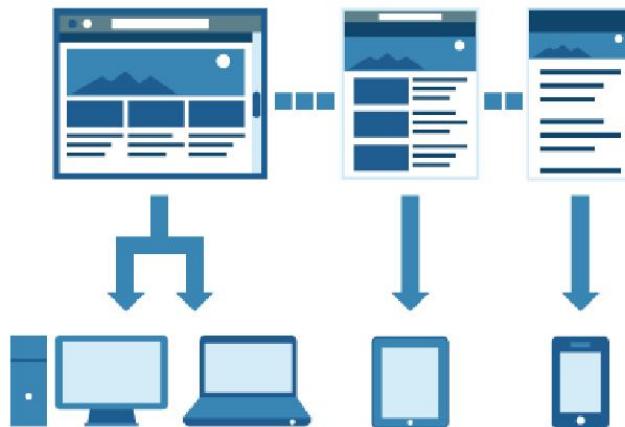
## Capítulo 1. RESPONSIVE WEB DESIGN

### 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, notebook 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). En dicha noticia se hace referencia al uso de Grillas Fluidas y Media Queries para crear un correcto diseño web responsive.



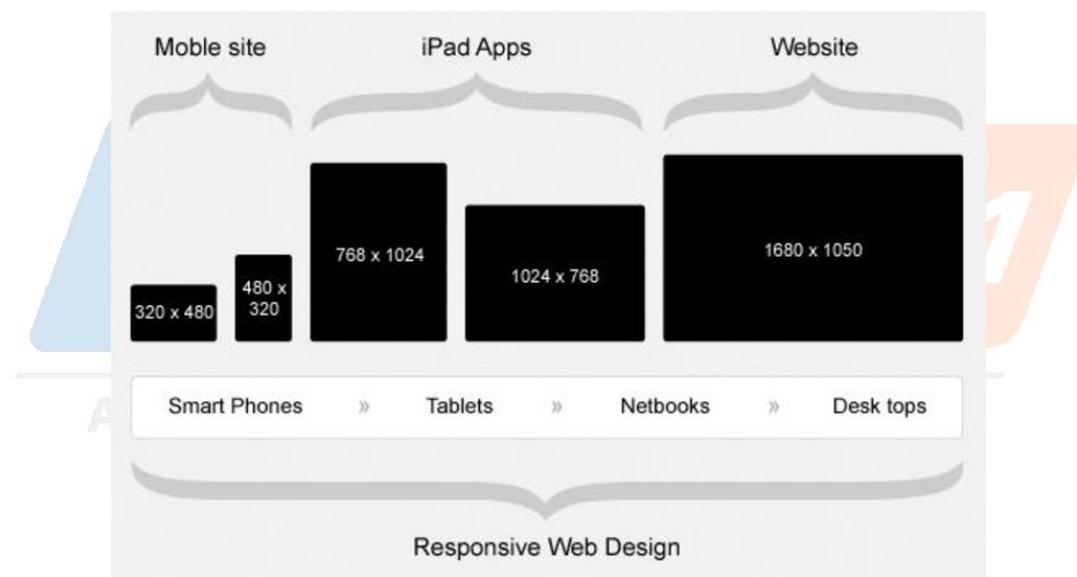
En un mundo donde incluso dos pantallas de las mismas dimensiones físicas pueden tener densidades de píxeles diferentes, las mediciones estáticas han llegado a ser innecesarias. Por lo tanto, Responsive Web Design utiliza una combinación de "Grillas Fluidas" y "Media Queries" para crear diseños que se 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.

## ACADEMIA DE SOFTWARE



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 ve 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:

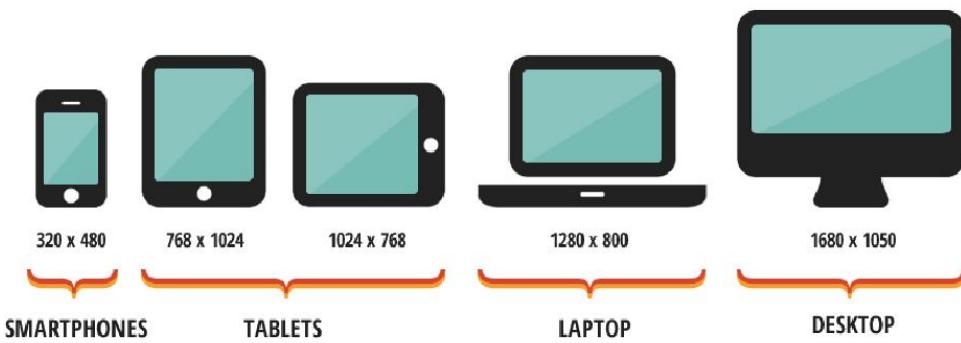
- Eliminar elementos de nuestra web que den problemas para que se puedan visualizar bien en todos los dispositivos.
- Falta de adaptación en dispositivos antiguos o muy nuevos.
- Mayor tiempo de carga, ya que enviamos más información de la necesaria.

### 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.



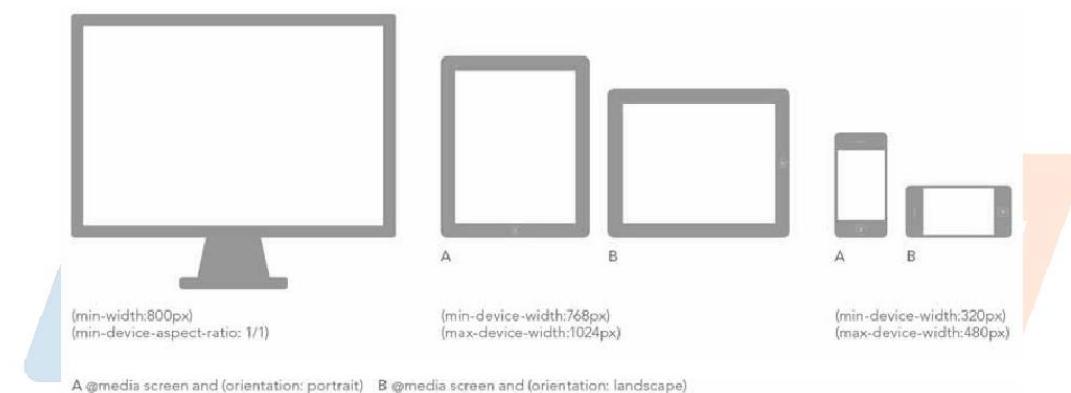
La siguiente imagen muestra algunos ejemplos de resoluciones de varios dispositivos:



## 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 verdadera 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 esta preguntando al dispositivo si su resolución horizontal (max-device-width) es igual o menor que 480px, si es correcto (en otras palabras, si esta 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.

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

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.

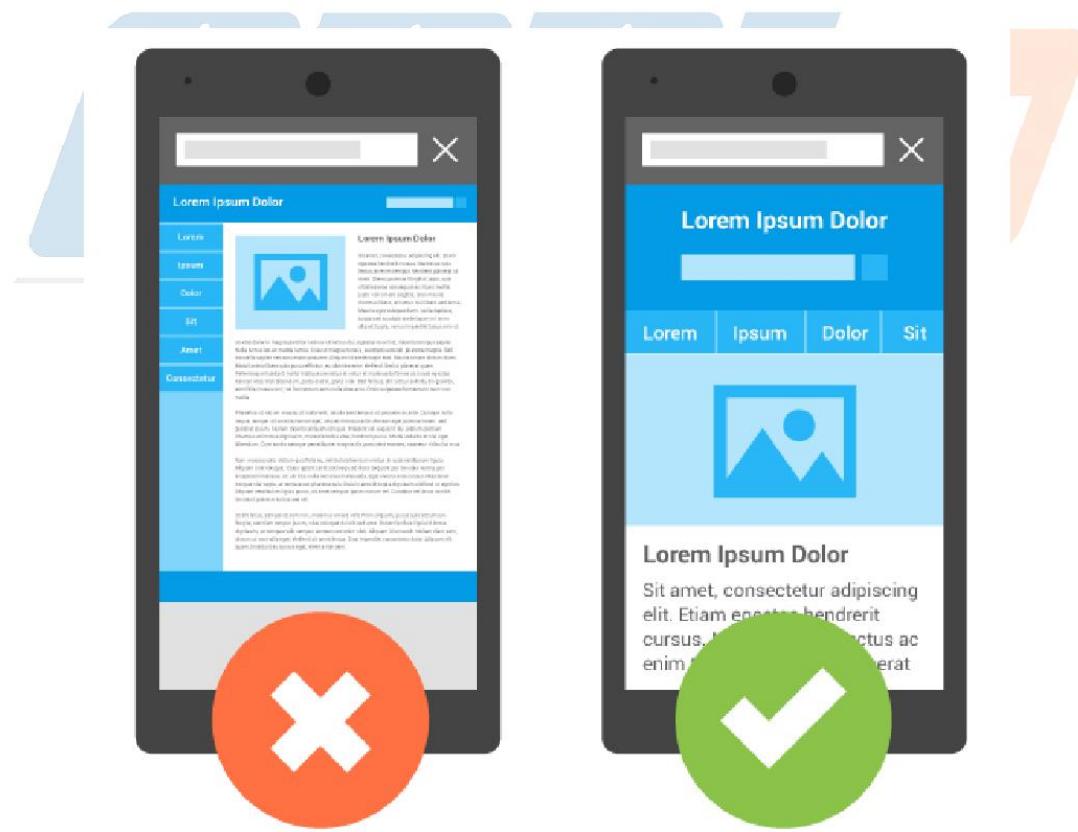
```
@media not screen and (color), print and (color)
```

## Capítulo 4. CONFIGURACIÓN DEL VIEWPORT

### 4.1.- Sitios móviles: Viewport

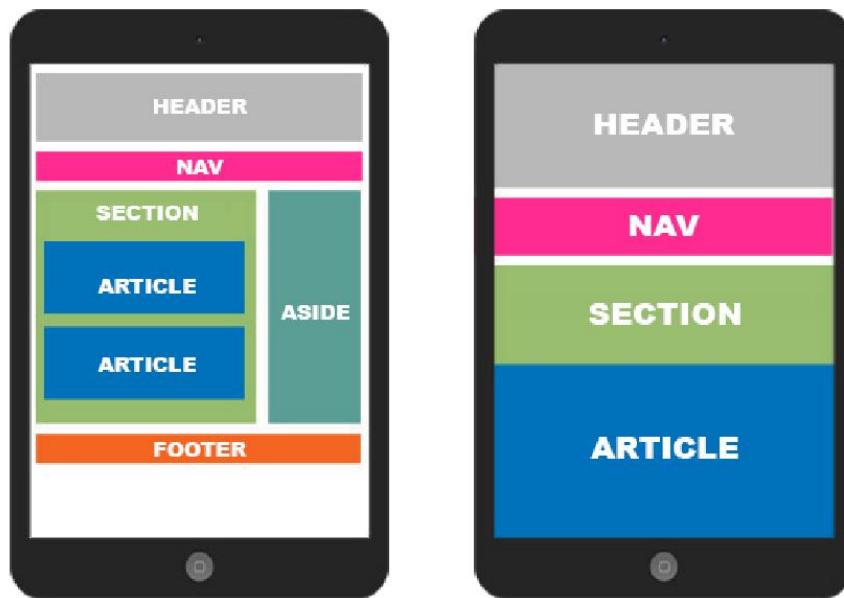
Prácticamente todos los navegadores de smartphones y tabletas al entrar a un sitio analizan el tamaño total, escalándolo para que se muestre completo en la pantalla.

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



**Sin el meta tag viewport**

**Con el meta tag viewport**

## 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 entero (px) o constante (device-width)                  | Define el ancho del viewport.                                      |
| <b>height</b>        | valor entero (px) o constante (device-height)                 | Define el height del viewport.                                     |
| <b>initial-scale</b> | Cualquier número real de 0.1 en adelante, o valor "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> | Cualquier número real de 0.1 en adelante, o valor "no escala" | Define la escala mínima del viewport.                              |
| <b>maximum-scale</b> | Cualquier número real de 0.1 en adelante, o valor "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. RECOMENDACIONES PARA RESPONSIVE DESIGN

### 5.1.- Recomendaciones

Mobile first: paradigma de diseño que consiste en maquetar la versión móvil de una página web. Esto permite comprobar cómo se verán las imágenes, los textos, los logos, y otros elementos en las pantallas pequeñas. Si se muestran sin problemas, entonces no debería haber ningún problema para adaptar posteriormente este diseño a las pantallas de mayor tamaño.

Navegación: la navegación es el aspecto más importante de una página web, por lo tanto, estos tienen que estar disponibles de una forma clara y ordenada para el usuario. Un método común para solucionar la falta de espacio en dispositivos más pequeños es el uso de un botón hamburguesa con una estructura off-canvas.

Botones: al diseñar para dispositivos de dimensiones más compactas se debe tener en cuenta que los botones que se van a usar tengan un tamaño apropiado para ser utilizados correctamente en dispositivos táctiles. Si los botones no tienen un tamaño apropiado estos serán más difíciles de cliquear.

Grillas fluidas: las dimensiones de los elementos de la maqueta deberán de ser definidas en unidades proporcionales o relativas, como lo son los porcentajes. Esto permite que la página mantenga su estructura intacta al cambiar sus proporciones, además permite a limitar la generación de posibles scrolls horizontales que se generan por el uso de unidades estativas.

Diseño minimalista: la popularidad del diseño minimalista ha ido en aumento durante los últimos años y por un buen motivo. Eliminan todo lo superfluo, e incitan a los usuarios a fijarse más en tu contenido, como consecuencia, se mejoran los ratios de conversión, y ayuda a que tu sitio web se cargue más rápido al usar menos elementos.

Optimiza la tipografía: en lo relativo al texto, asegúrate de que sea legible en las pantallas pequeñas. Un buen tamaño para el cuerpo del texto sería 16px o 1em y después ajusta en consecuencia el tamaño de los titulares. Al mismo tiempo, querrás ajustar la altura de línea de tu texto a 1.5em para asegurarte de que las líneas de los párrafos respiren lo suficiente.

## 5.2.- Errores Frecuentes

Diseño minimalista: la popularidad del diseño minimalista ha ido en aumento durante los últimos años y por un buen motivo. Eliminan todo lo superfluo, e incitan a los usuarios a fijarse más en tu contenido, como consecuencia, se mejoran los ratios de conversión, y ayuda a que tu sitio web se cargue más rápido al usar menos elementos.

Optimiza la tipografía: en lo relativo al texto, asegúrate de que sea legible en las pantallas pequeñas. Un buen tamaño para el cuerpo del texto sería 16px o 1em y después ajusta en consecuencia el tamaño de los titulares. Al mismo tiempo, querrás ajustar la altura de línea de tu texto a 1.5em para asegurarte de que las líneas de los párrafos respiren lo suficiente.



## Capítulo 6. FLEXBOX. PARTE 1

### 6.1.- ¿qué es el Layout?

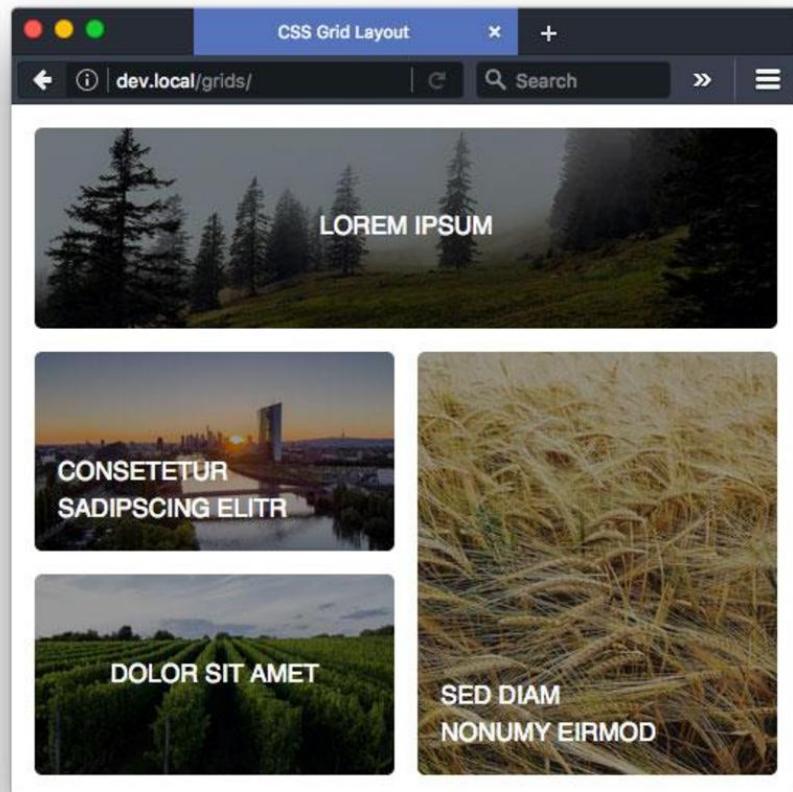
Es conocido como el boceto o plan de un proyecto, su finalidad es definir la estructura que tendrán los elementos dentro del diseño. En el ámbito de diseño web hace referencia a la maquetación, término nombrado a lo largo del curso pero que ahora se le dará un mayor enfoque.

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.

ACADEMIA DE SOFTWARE



## 6.2.- 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.



### 6.3.- 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 panels. The left panel is labeled "HTML" and contains the following code:

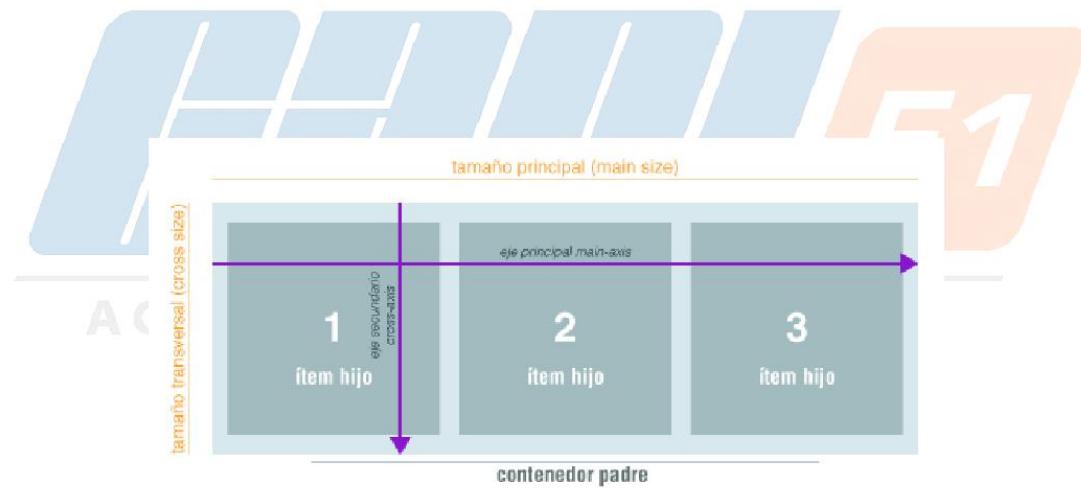
```
1 <html>
2 <head>
3   <title>Flexbox</title>
4 </head>
5 <body>
6   <div class="flex-container">
7     <div class="flex-item"></div>
8     <div class="flex-item"></div>
9     <div class="flex-item"></div>
10   </div>
11 </body>
12 </html>
```

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

```
.flex-container{
  display:flex;
}
```

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.



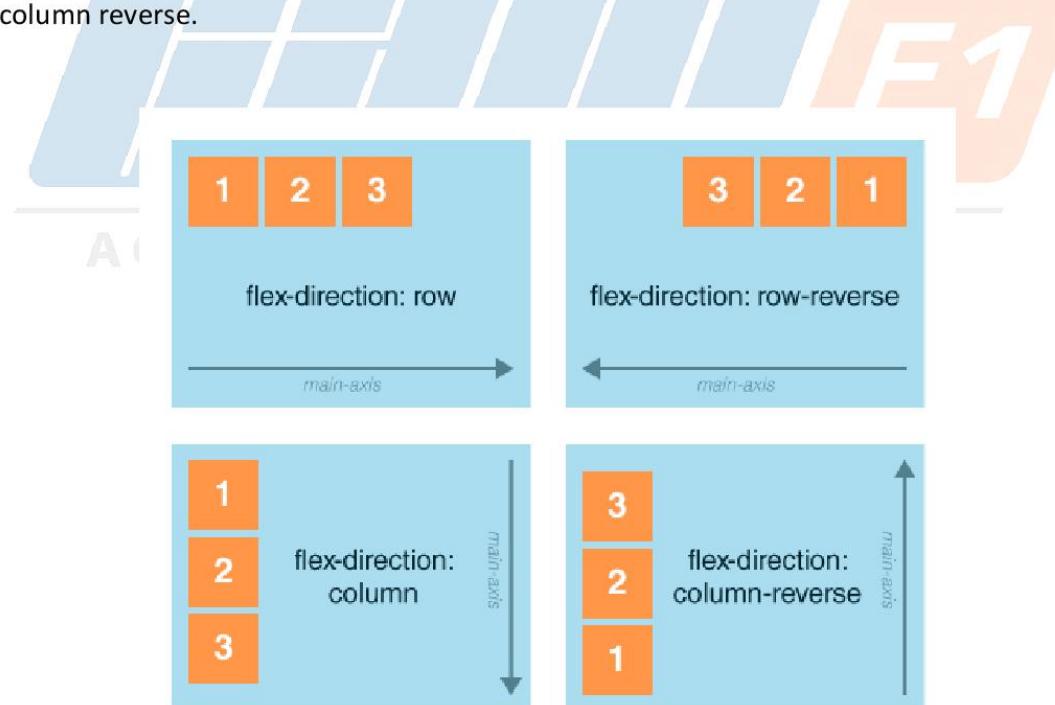
## Capítulo 7. FLEXBOX. PARTE 2

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

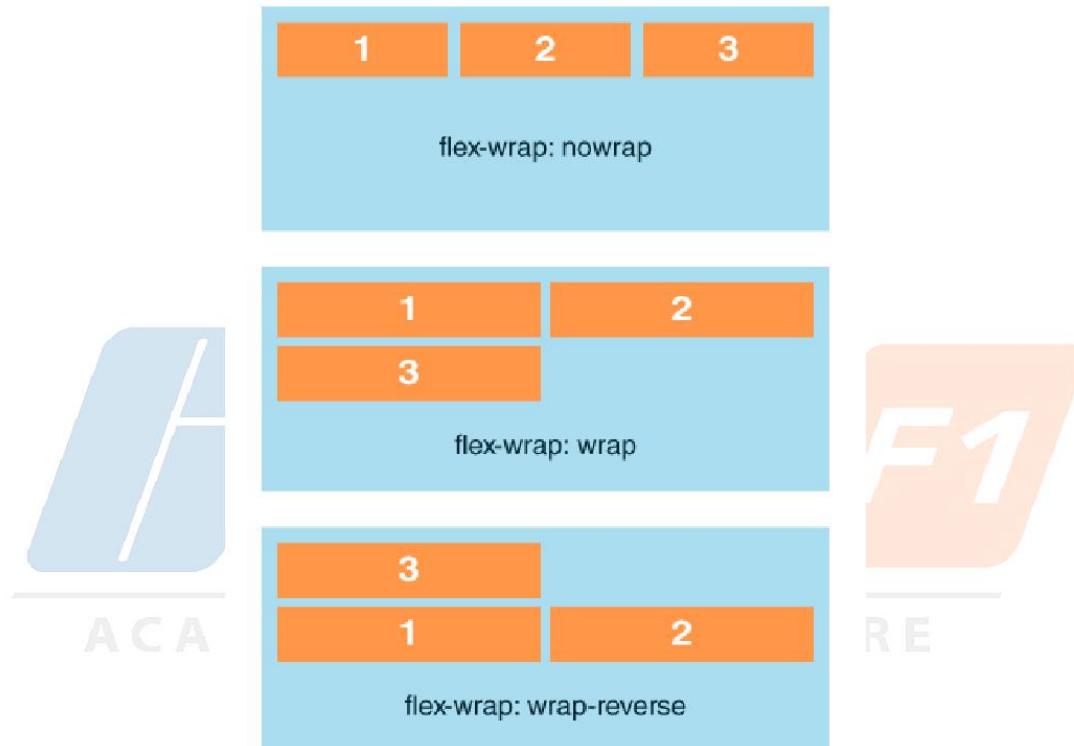
En el capítulo anterior se pudo apreciar el uso de una etiqueta padre y varias etiquetas hijas, este comportamiento cumple con la primera regla “Flexbox necesita un parente y por lo menos un hijo” dado que no hay forma de manipular directamente un elemento flexible sin crear antes su contexto. En otras 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:

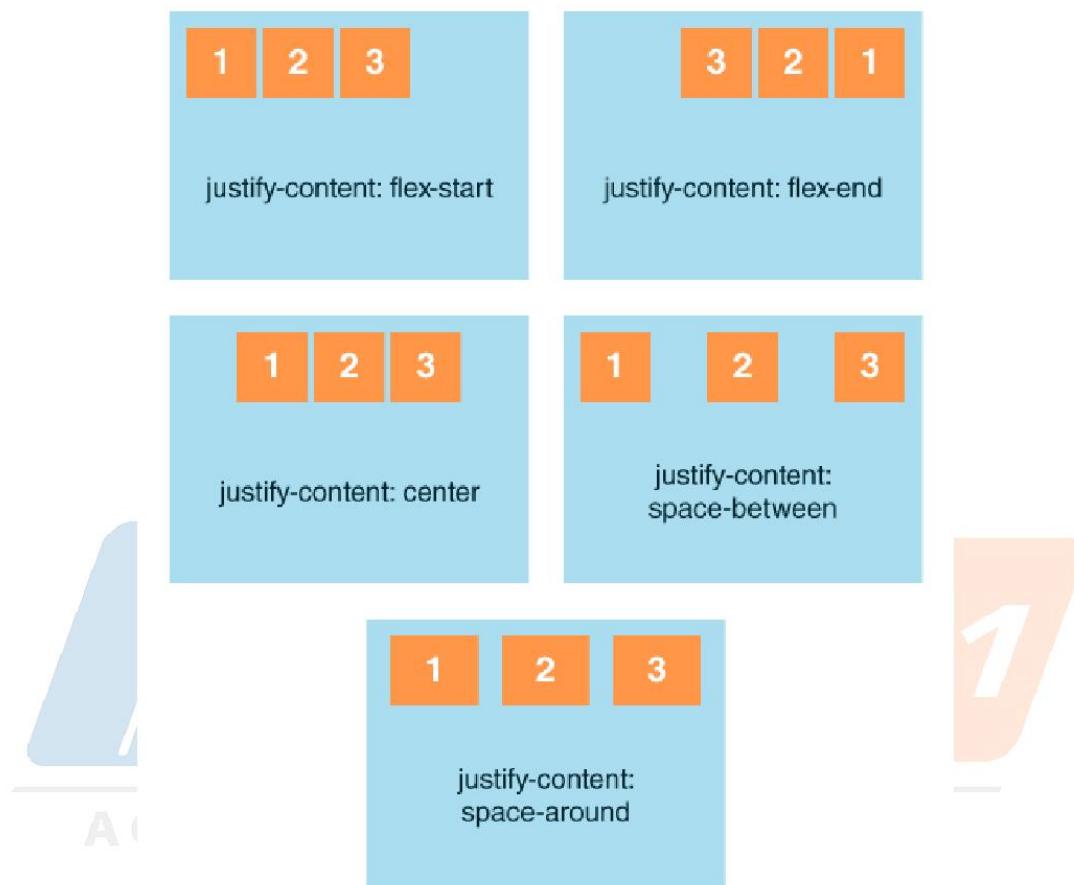
flex-direction: permite alterar la orientación de los elementos en el eje principal y secundario (horizontal y vertical). Sus valores pueden ser: row, row-reverse, column y column-reverse.



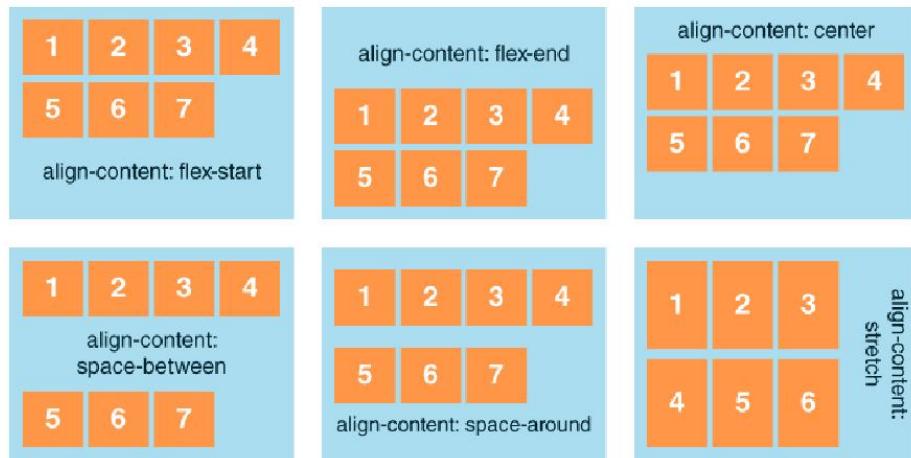
flex-wrap: controla la dirección de los elementos según su anchura y la del contenedor padre. Sus valores son: nowrap, wrap, wrap-reverse.



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.



`align-items`: funciona de manera similar a `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`.



## 7.2.- 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.

```
HTML
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>
```

```
CSS
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 }
```

4

2

1

3

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>
```

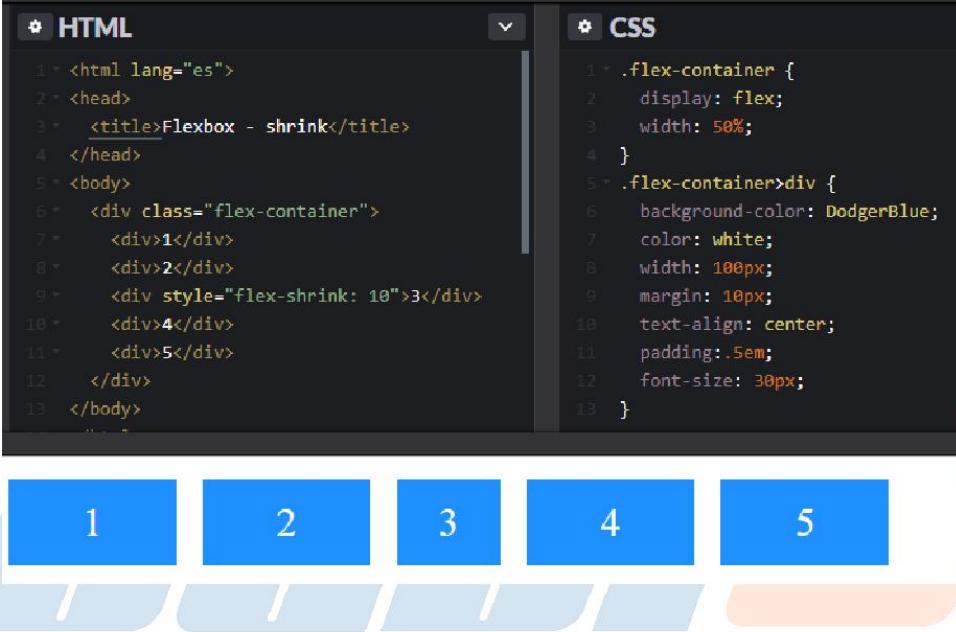
```
CSS
1 .flex-container {
2   display: flex;
3   width: 50%;
4 }
5 .flex-container>div {
6   background-color: DodgerBlue;
7   color: white;
8   width: 100px;
9   margin: 10px;
10  text-align: center;
11  padding:.5em;
12  font-size: 30px;
13 }
```

1

2

3

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



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

```
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>
```

The 'CSS' tab 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 five blue rectangular boxes labeled 1 through 5. Box 3 is significantly larger than the others, while boxes 4 and 5 are smaller, demonstrating how the 'flex-shrink' property affects item size in a flex container.

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 tabs: 'HTML' and 'CSS'. The 'HTML' tab 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 'CSS' tab 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 }
```

1

2

3

4

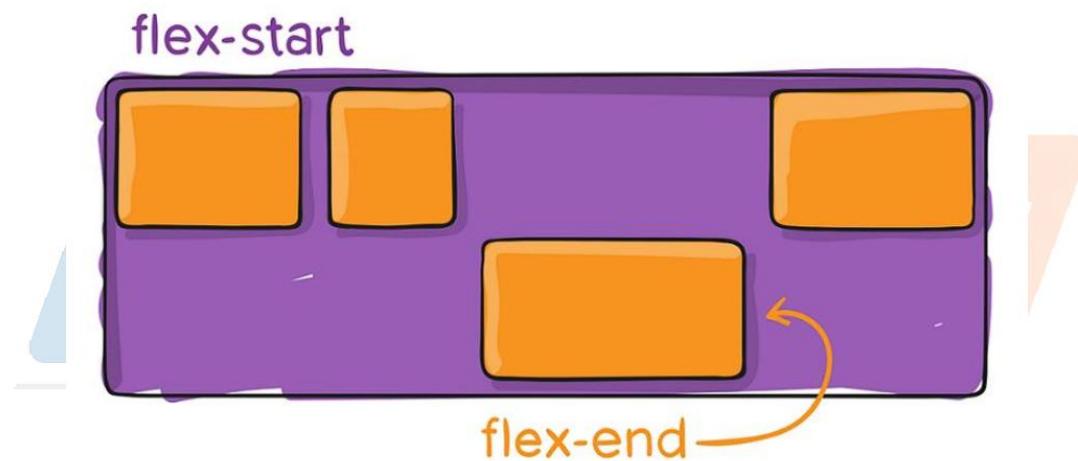


ACADEMIA DE SOFTWARE

## Capítulo 8. FLEXBOX. PARTE 3

### 8.1.- Align-self

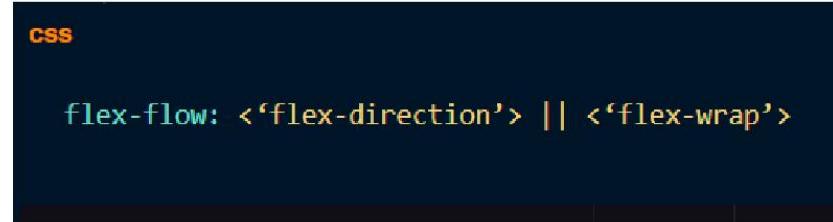
Esta propiedad permite definir la alineación que tenga un elemento hijo, ya que sobrescribe el efecto que tenía sobre este la propiedad align-items de su contenedor padre. Nótese que las propiedades float, clear y vertical-align no tienen efecto en elemento flexible.



### 8.2.- Flex-flow

Esta propiedad es un atajo para las propiedades flex-direction y flex-wrap en una sola línea de código, y estas propiedades juntas permiten definir rápidamente el eje principal y el eje secundario.

El valor por defecto de esta propiedad es row no-wrap.



```
css
flex-flow: <'flex-direction'> || <'flex-wrap'>
```

### 8.3.- Flex

Esta propiedad es un atajo para las propiedades flex-grow, flex-shrink y flex-basis en una sola línea de código, y los parámetros de la propiedad se escribirán en ese orden. El segundo parámetro y el tercer parámetro son opcionales.

El valor por defecto de esta propiedad es 0 1 auto.



```
css
.item {
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

### 8.4.- Compatibilidad

Dependiendo de la versión del navegador, puede haber conflictos de compatibilidad al momento de asignar a un elemento la propiedad display con el valor flex, esto se debe a los cambios de nombre que este valor ha tenido con el paso del tiempo. Estos nombres serían:

- Flex: Nombre actual del valor de la propiedad display para definir a un elemento como un contenedor flexible.
- Flexbox: Sintaxis no oficial utilizada en algunos navegadores en el 2011.

- Box: Sintaxis vieja que se utilizaba en 2009.

|  |  |  |  |
|---|---|---|---|
| Chrome  | Safari  | Firefox   | Opera   |
| 20- (old)<br>21+ (new)  | 3.1+ (old)<br>6.1+ (new)  | 2-21 (old)<br>22+ (new)   | 12.1+ (new)   |
|  |  |  |  |
| IE<br>10 (tweener)<br>11+ (new)   | Edge<br>17+ (new)   | Android<br>2.1+ (old)<br>4.4+ (new)   | iOS<br>3.2+ (old)<br>7.1+ (new)   |

ACADEMIA DE SOFTWARE

## Capítulo 9. CSS GRID. PARTE 1

### 9.1.- Definición

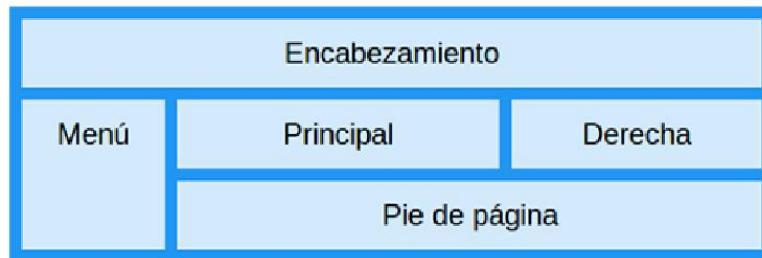
Uno de los procesos más problemáticos y frustrantes de CSS, sobre todo para novatos o principiantes, es el proceso de colocar y distribuir los elementos a lo largo de una página. Mecanismos como posicionamiento, floats o elementos en bloque o en línea, suelen ser insuficientes (o muy complejos) para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

ACADEMIA DE SOFTWARE



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 |

## 9.2.- 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.

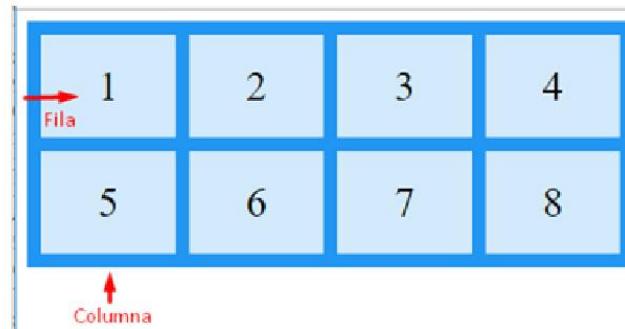
Inline-grid: establece que los ítems de la grilla se comporten como elementos de línea, ajustando sus anchos a su propio contenido.

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 cuadricula. 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>
```

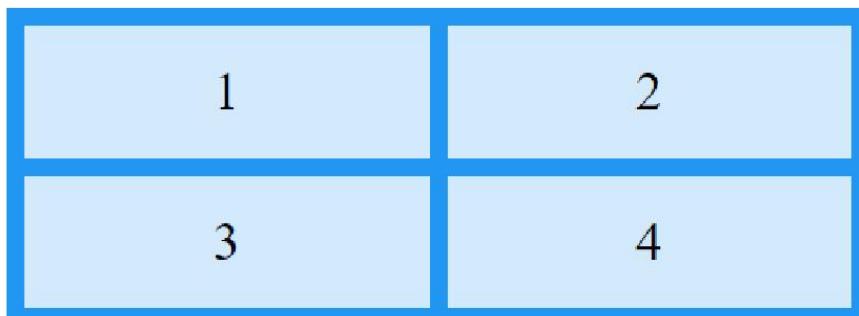
La visualización del archivo html de la lámina anterior, se muestra en la siguiente imagen. Esta es una rejilla de 4 columnas con anchos automáticos.



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   | CSS   |
| <pre>&lt;div class="grid-container"&gt;   &lt;div&gt;1&lt;/div&gt;   &lt;div&gt;2&lt;/div&gt;   &lt;div&gt;3&lt;/div&gt;   &lt;div&gt;4&lt;/div&gt; &lt;/div&gt;</pre> | <pre>.grid-container {   display: grid;   grid-template-columns: auto auto;</pre> |

La propiedad “grid-template-rows” hará exactamente lo mismo que la propiedad anterior, pero esta vez haciendo referencia a 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;
}
```

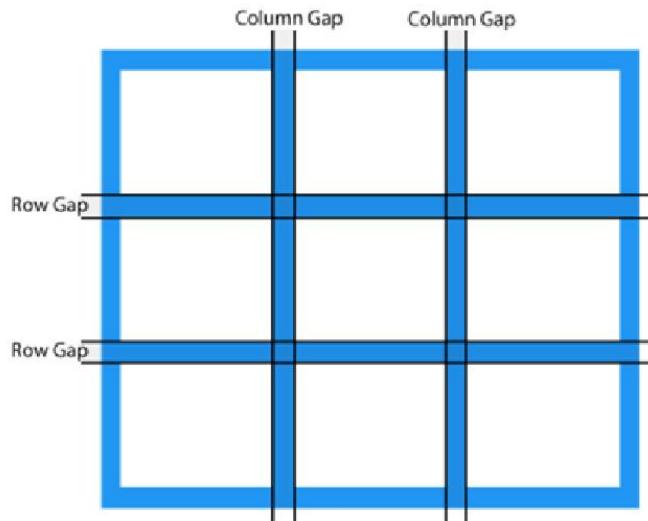
### 9.3.- Espacio Entre Filas y Columnas

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;  
}
```

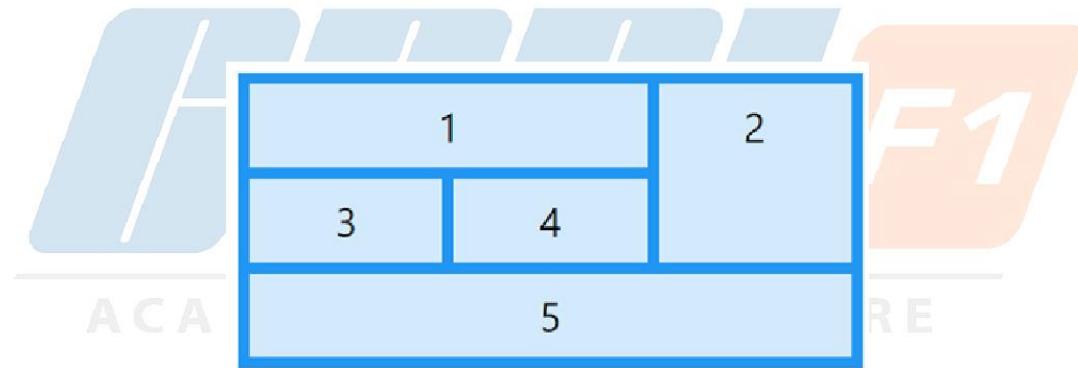
## Capítulo 10. CSS GRID. PARTE 2

### 10.1.- 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.



### 10.2.- Modificando Los Elementos Del Grid

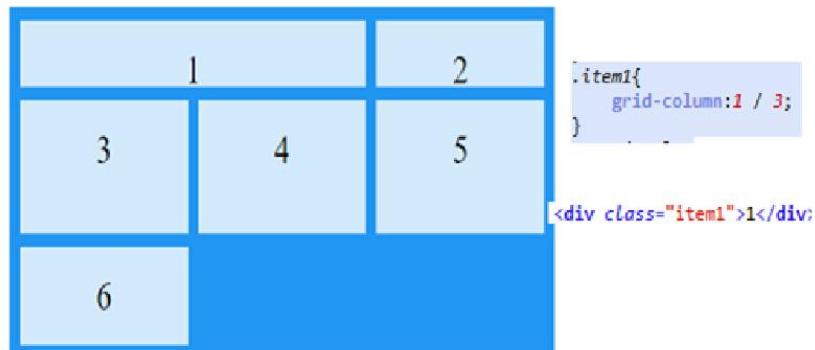
La propiedad grid-column define en el que la columna (s) para colocar un elemento. La propiedad define en que columna comenzará y en que columna terminará el artículo.

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: columnalInicial / 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.

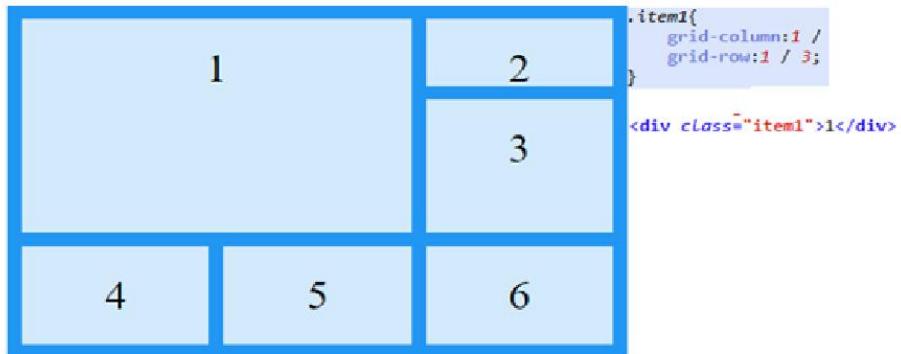


La propiedad grid-row define en qué fila para colocar un elemento.

Se define en que 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 11. CSS GRID. PARTE 3

### 11.1.- Grid Area

Grid-area:

Le da a un elemento un nombre que le permite asociarse con la grilla definida con la propiedad grid-template-areas en su elemento padre.

Alternativamente, esta propiedad se puede usar como atajo para las propiedades grid-row-start, grid-column-start , grid-row-end y grid-column-end.

```
.container {  
    grid-template-areas:  
        "<grid-area-name> | . | none | ..."   
        "...";  
}
```

Grid-template-areas:

Define la estructura de una grilla haciendo referencia a los nombres de las diferentes áreas de la grillas, que luego son especificadas con la propiedad grid-area. Repetir el nombre de una de estas áreas hace que el contenido afecto se extienda en todas estas celdas.

Un punto significa una casilla vacía.

La sintaxis misma permite una visualización de la estructura general de la grilla.

Se puede usar el valor none para dejar la estructura de la grilla sin definir.

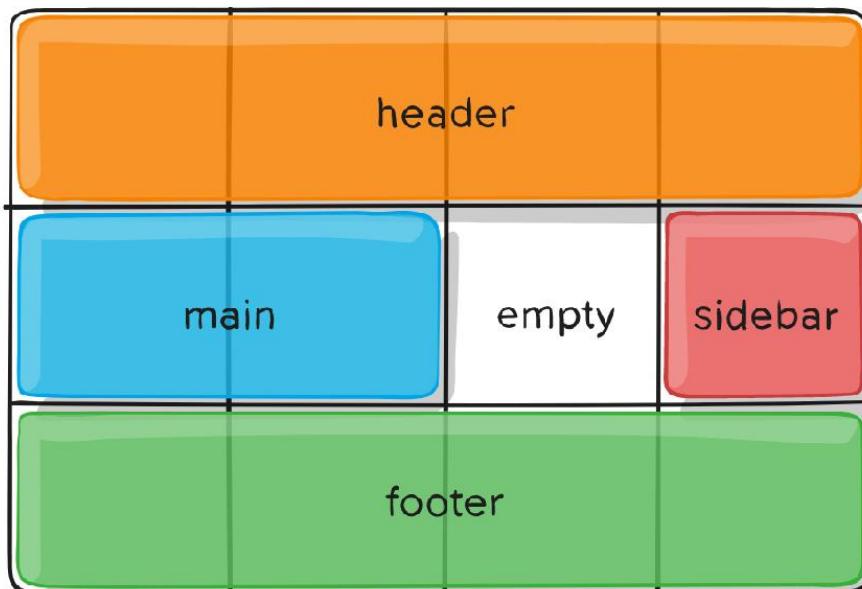
```
.container {  
    display: grid;  
    grid-template-columns: 50px 50px 50px 50px;  
    grid-template-rows: auto;  
    grid-template-areas:  
        "header header header header"  
        "main main . sidebar"  
        "footer footer footer footer";  
}
```

Grid-area:

Le da a un elemento un nombre que le permite asociarse con la grilla definida con la propiedad grid-template-areas en su elemento padre.

Alternativamente, esta propiedad se puede usar como atajo para las propiedades grid-row-start, grid-column-start , grid-row-end y grid-column-end.

```
.item-a {  
    grid-area: header;  
}  
  
.item-b {  
    grid-area: main;  
}  
  
.item-c {  
    grid-area: sidebar;  
}  
  
.item-d {  
    grid-area: footer;  
}
```

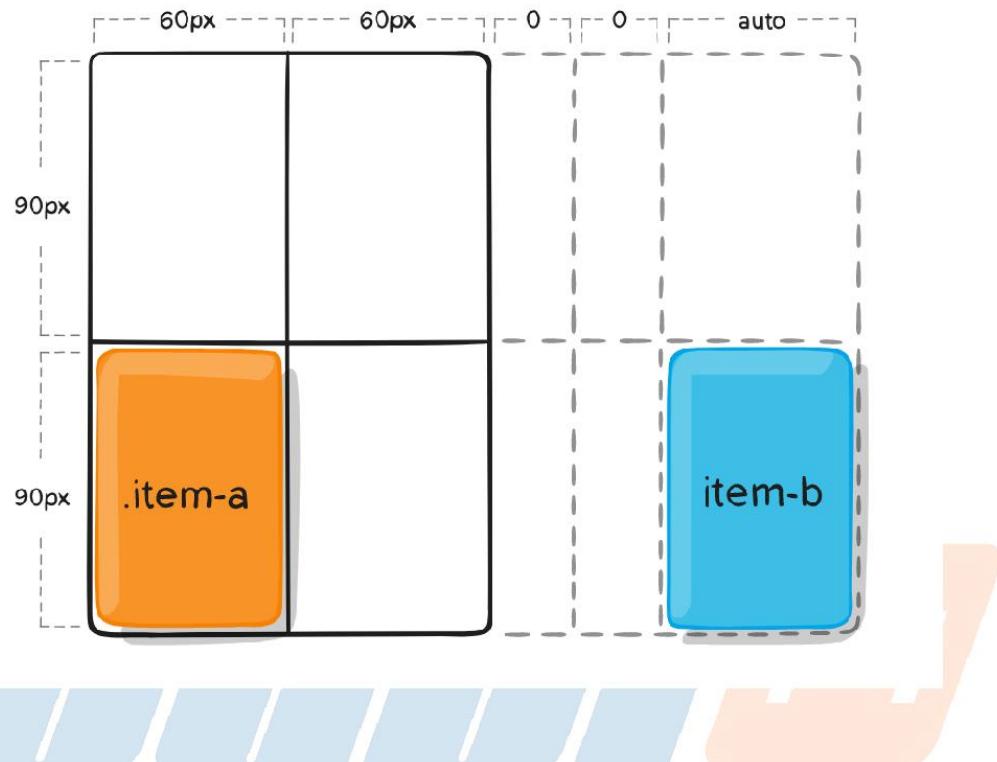


### 11.2.- Valores Automáticos de Una Grilla

grid-auto-columns y grid-auto-rows:

Estas propiedades especifican el tamaño de las celdas generadas automáticamente en la grilla. Estas celdas autogeneradas, o implícitas, son generadas cuando hay más elementos de grilla que celdas definidas o cuando un elemento de grilla se coloca fuera de los de las celdas definidas en el elemento padre.

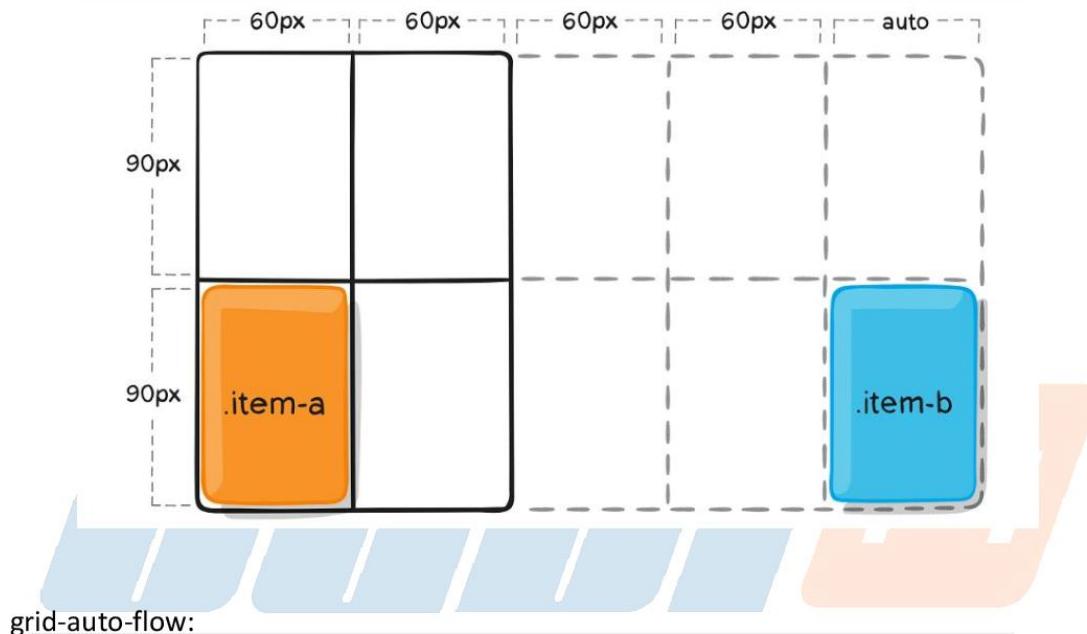
Por defecto, el valor de estas celdas automáticas es 0.



Pero usando esta propiedad ese valor puede ser manipulado, cambiando el tamaño de estas celdas automáticas.

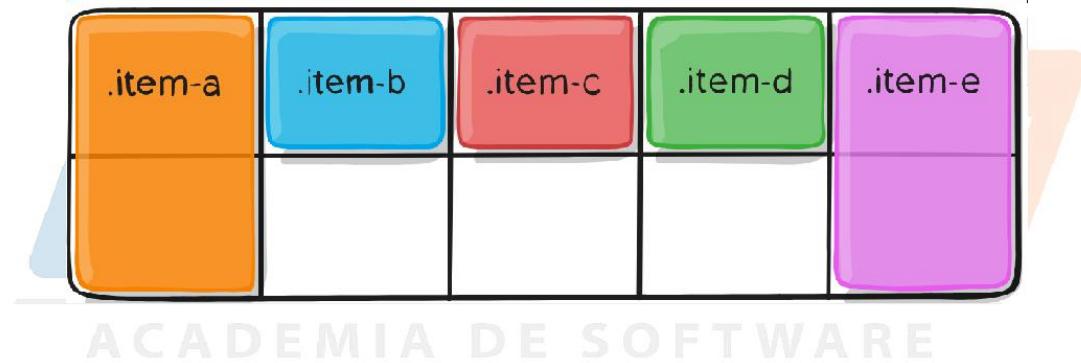
ACADEMIA DE SOFTWARE

```
.container {  
    grid-auto-columns: 60px;  
}
```



Si se tienen elementos de grilla a los cuales no se le han definido una posición explícita en la grilla, estos serán posicionados en función del algoritmo de auto posicionamiento de la grilla. Esta propiedad permite controlar la manera en la que funciona este algoritmo.

```
.item-a {  
    grid-column: 1;  
    grid-row: 1 / 3;  
}  
.item-e {  
    grid-column: 5;  
    grid-row: 1 / 3;  
}
```

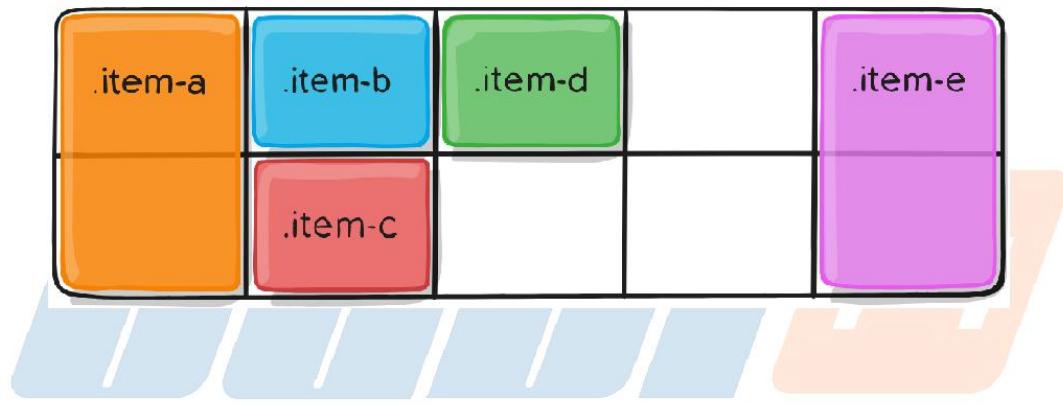


## ACADEMIA DE SOFTWARE

Valores de grid-auto-flow:

- row: Le indica al algoritmo que posicione los elementos con prioridad en llenar las filas, y añade filas nuevas si es necesario. Valor por defecto.
- column: Le indica al algoritmo que posicione los elementos con prioridad en llenar las columnas, y añade columnas nuevas si es necesario.
- dense: Le indica al algoritmo que posicione los elementos con prioridad en no dejar espacios vacíos, utilizando los elementos más pequeños para esto.

```
.container {  
    display: grid;  
    grid-template-columns: 60px 60px 60px 60px 60px;  
    grid-template-rows: 30px 30px;  
    grid-auto-flow: column;  
}
```



ACADEMIA DE SOFTWARE

## Capítulo 12. CSS GRID. PARTE 4

### 12.1.- Propiedades de Elemento de Grilla

Justify-self:

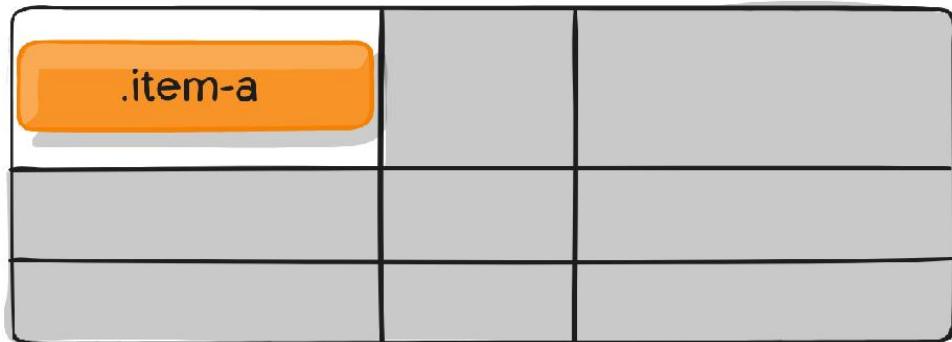
Esta propiedad alinea al elemento de grilla en función del eje x, o el eje de las filas. Esta propiedad solo afecta a un solo elemento dentro de una celda de una grilla.

```
.item-a {  
    justify-self: center;  
}
```

Align-self:

Esta propiedad alinea al elemento de grilla en función del eje y, o el eje de las columnas. Esta propiedad solo afecta a un solo elemento dentro de una celda de una grilla.

```
.item-a {  
    align-self: center;  
}
```

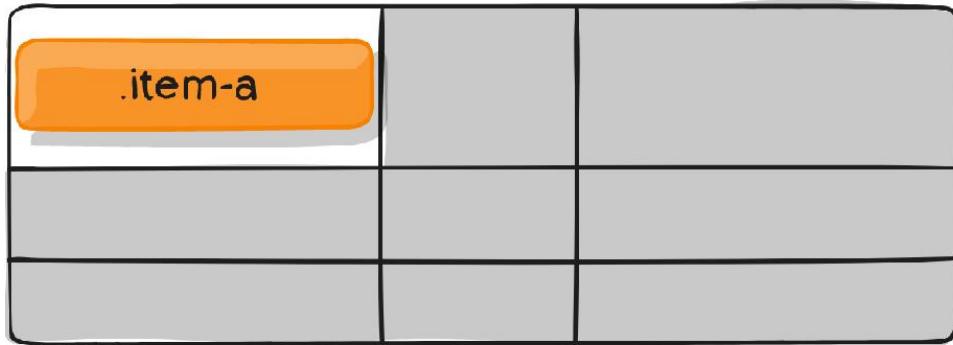


Place-self:

Es la combinación de las propiedades justify-self y align-self en una sola línea de código.  
Los posibles valores son align-self y luego justify-self

ACADEMIA DE SOFTWARE

```
.item-a {  
    place-self: center stretch;  
}
```



## 12.2.- Atajos de Css Grid

Grid-template:

Este atajo nos permite hacer uso de las propiedades grid-template-row, grid-template-columns y grid-template-areas en una sola declaración.

Grid-template:

Este atajo nos permite hacer uso de las propiedades grid-template-row, grid-template-columns y grid-template-areas en una sola declaración.

```
.container {  
    grid-template:  
        [row1-start] "header header header" 25px [row1-end]  
        [row2-start] "footer footer footer" 25px [row2-end]  
        / auto 50px auto;  
}
```

Grid:

Este atajo permite definir la mayoría de las propiedades de una grilla en sola definición, permite definir las propiedades grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns, y grid-auto-flow.

Nótese que solo puede definir las propiedades explícitas o las propiedades implícitas en sola declaración de grilla.

```
css  
  
.container {  
    grid: 100px 300px / auto-flow 200px;  
}
```

```
css  
  
.container {  
    grid-template-rows: 100px 300px;  
    grid-auto-flow: column;  
    grid-auto-columns: 200px;  
}
```

## Capítulo 13. ESTRUCTURAS DE DISEÑO RESPONSIVE

### 13.1.- Mostly Fluid

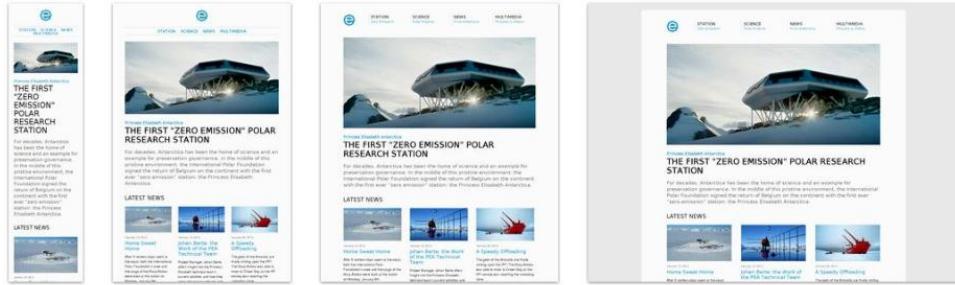
Esta técnica es el patrón más popular y dentro de la comunidad responsive uno de los mayormente empleados en los distintos diseños. En dimensiones de smartphone se apilan todas las columnas y según aumentan estas dimensiones se voltean en un desarrollo multicolumna tanto como el ancho del navegador deje acomodarlas. Estos elementos se vuelven fluidos y se expanden hasta los diferentes breakpoints. Además se puede combinar con la técnica de Column Drop.



### ACADEMIA DE SOFTWARE

Otro concepto que este patrón emplea es definir un ancho máximo para el wrapper que contiene las capas pudiendo crecer el diseño y expandirse a cualquier resolución del navegador quedando un espacio en blanco a los lados del contenido. A estas dimensiones el diseño se basa en los layouts de ancho fijo y centrados típicos de los últimos años.

El origen del nombre “mayormente líquido” es debido a que la estructura principal de los layouts no cambia realmente hasta dimensiones mobile donde se aplica la caída de columna. Como hemos comprobado las columnas se vuelven líquidas con su contenido hasta alcanzar el ancho máximo.



### 13.2.- Column Drop

Es un patrón popular que comienza con un layout multicolumna y termina con un layout sencillo de una columna, dejando caer las columnas a medida que la dimensión de la pantalla se va estrechando.

A diferencia del patrón Mostly Fluid, el tamaño total de los elementos en esta disposición tiende a permanecer constante. En este diseño la adaptación a distintos tamaños de pantalla se basa en apilar las columnas como en el siguiente wireframe:

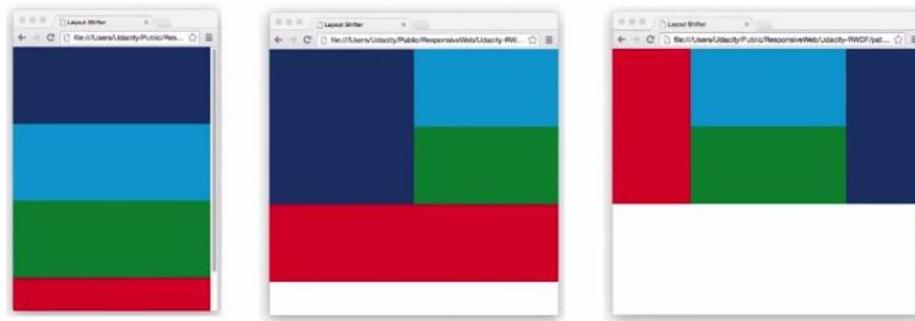
Saber el cuándo y cómo cada columna se debe de apilar apilan en los breakpoints de resolución no es un dogma, debe ser es diferente para cada diseño.

ACADEMIA DE SOFTWARE



### 13.3.- Layout Shifter

Este modelo de diseño adaptativo es creación de Luke Wroblewski, este modelo no es tan popular como los que hemos mostrado en anteriores ocasiones pero es un diseño diferente y muy fresco. La idea principal de este modelo es tener una página dividida en varias secciones que pueden reorganizarse de diferentes maneras a lo largo de nuestro diseño responsive.



Layout shifter tiene la distinción de ser el modelo diseño responsive que mejor ilustra esta filosofía de diseño, debido a que la estructura de la página puede verse reorganizada de diferentes maneras.

El factor más importante a tomar en cuenta es la manera en la que el contenido se mueve y organiza, sin tener la necesidad de hacer un diseño fluido o de colocar columnas unas encima de otras.

ACADEMIA DE SOFTWARE



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

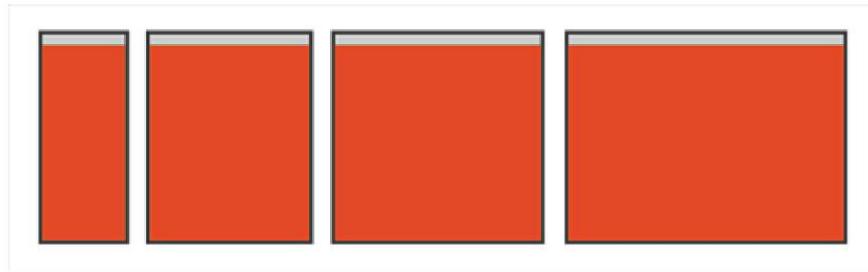
Responsive Design significa adaptar tu diseño al tamaño de pantalla del dispositivo. Puedes hacerlo como tu quieras, pero existen 5 patrones ya definidos que ayudarán en el diseño. Sus nombre son:

- 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.

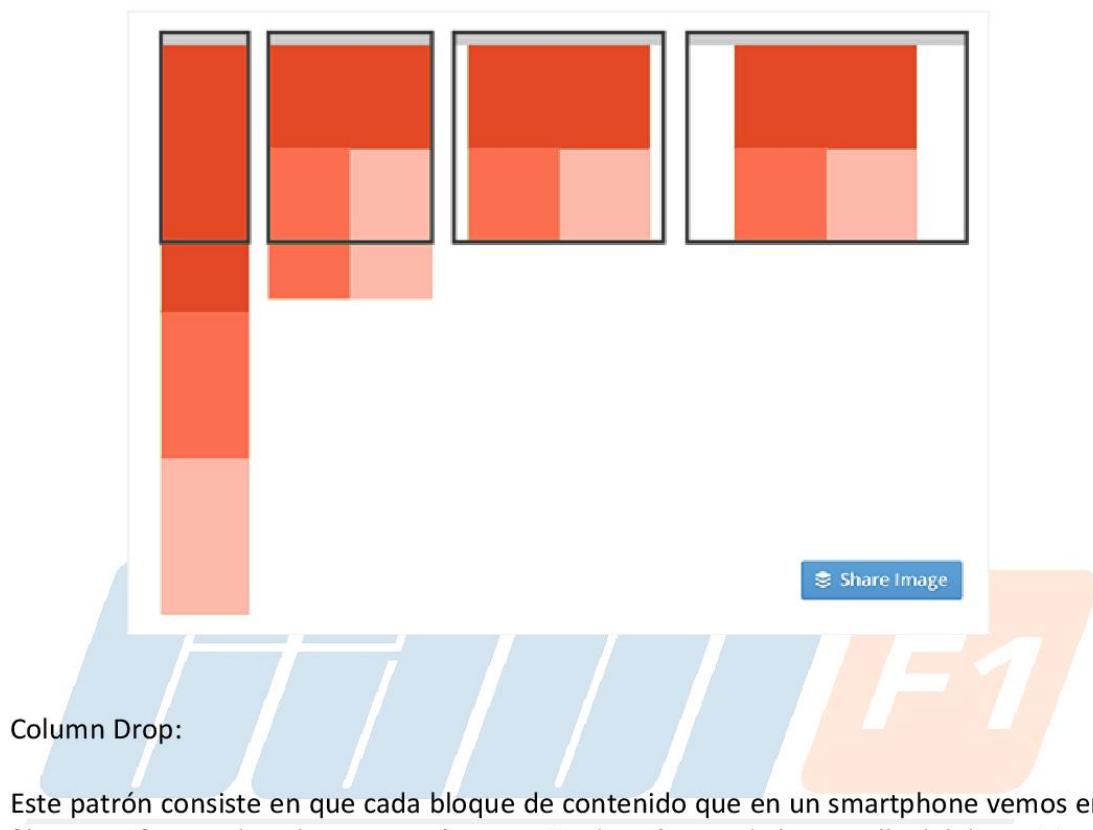


#### Mostly Fluid:

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

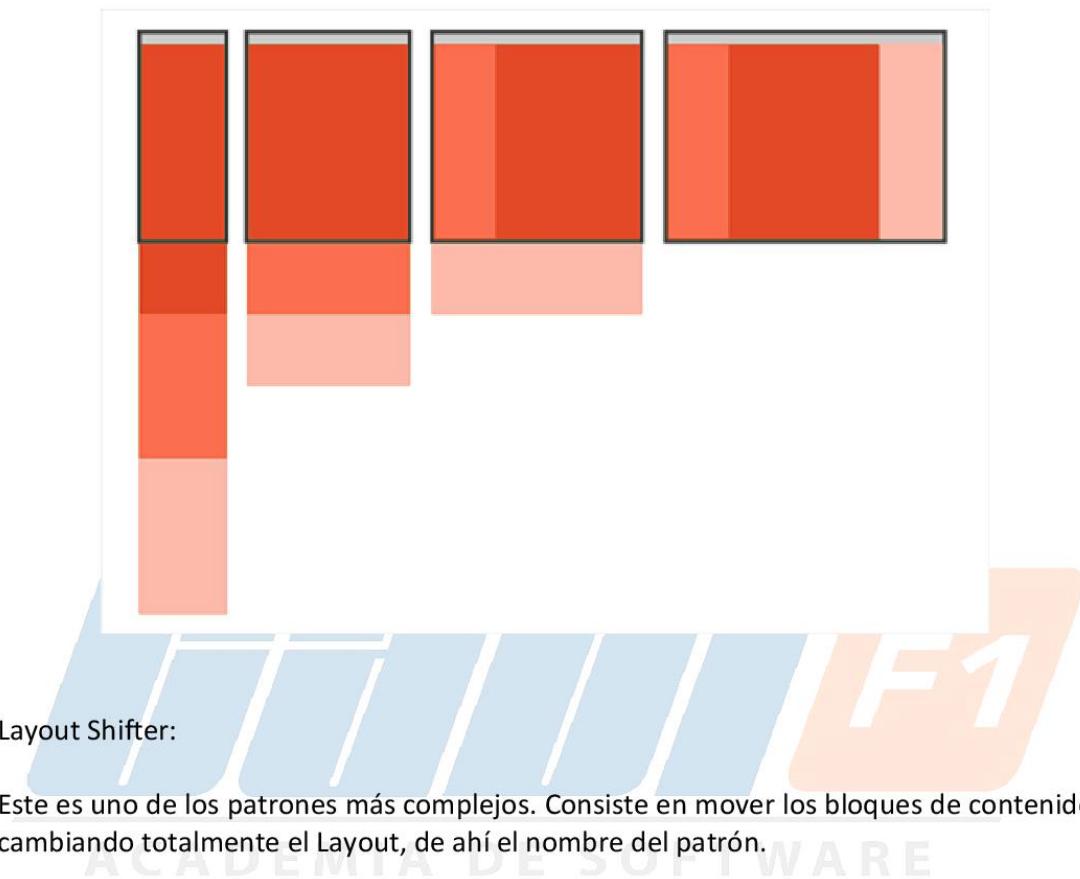


#### Column Drop:

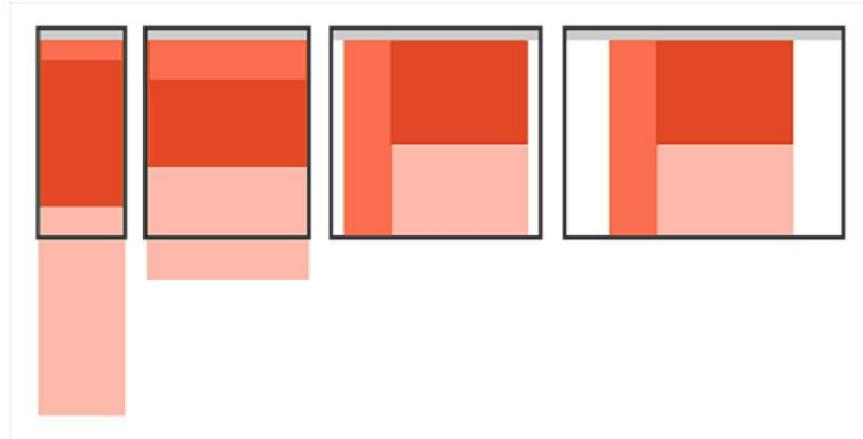
Este patrón consiste en que cada bloque de contenido que en un smartphone vemos en filas, vaya formando columnas según vaya siendo más grande la pantalla del dispositivo.

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.

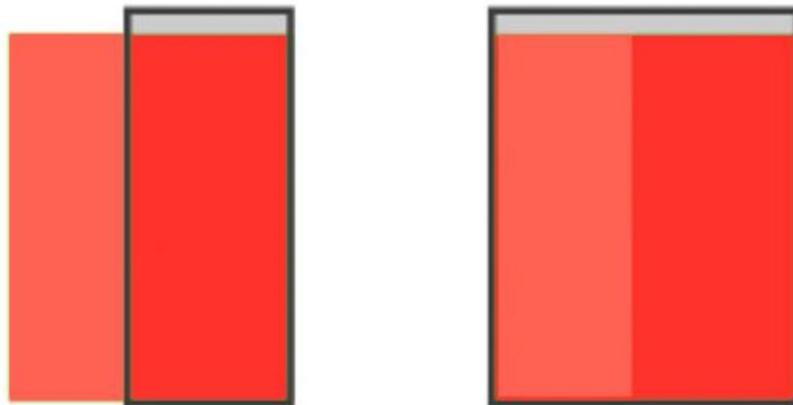


Las columnas 2 y 3 estarán englobadas dentro de un bloque que llamaremos container-inner que nos permitirá hacer el cambio de layout.



#### 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 14. SELECTORES DE CONTENIDO

### 14.1.- After y Before

::after

Este es un pseudoelemento que te permite insertar contenido dentro de una desde el css, sin tener que este en el html. Mientras que el resultado final no puede apreciarse en el código fuente, aparece correctamente en la página.

El contenido generado se colocara dentro del elemento seleccionado, despues del contenido del mismo

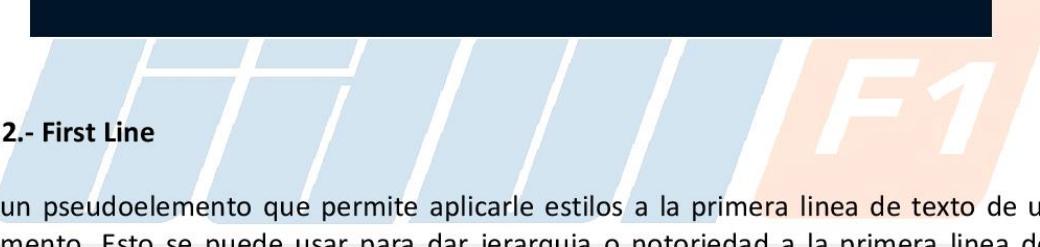
::before

Es exactamente los mismo que ::after, solamente que coloca el contenido se colocara antes del contenido del elemento seleccionado.

Esta propiedad va a manejarse principalmente por la propiedad content, la cual puede recibir diferentes valores e función de que se desea mostrar.

Valores:

- Un texto,"Un texto";
- Una imagen, url(/path/to/image.jpg);
- Nada: Un string vacio;
- Un contador: counter(li);



```
css
div::after {
    content: "hi";
}

html
<div>
    <!-- Rest of stuff inside the div -->
    hi
</div>
```

#### 14.2.- First Line

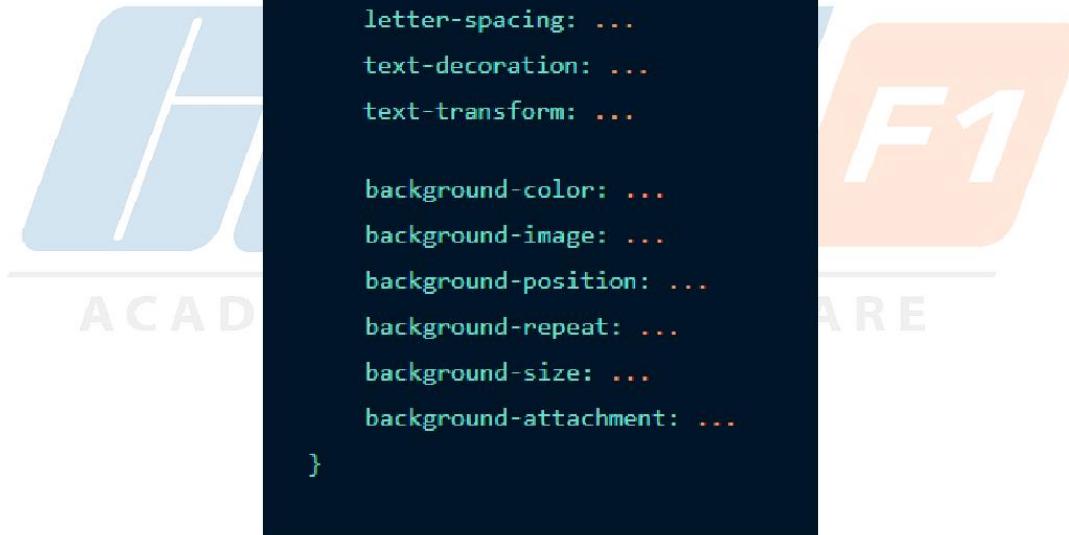
Es un pseudoelemento que permite aplicarle estilos a la primera linea de texto de un elemento. Esto se puede usar para dar jerarquia o notoriedad a la primera linea del elemento. La longitud de esta primera linea dependera de varios factores como la altura de la linea, el ancho del viewport, tamaño de fuente, distancia entre las letras o las palabras. Despues del primer salto de linea los estilos ya no entraran en efecto.

Este pseudoelemento solo funciona en elementos de bloque, por lo que no funcionara en elementos en linea, aunque estos tengan un salto de linea.

Los estilos asignados en este pesudo elemento sobreescreiben los otros estilos asignados al elemento, aun aquellos son declarados importantes.

**css**

```
.element::first-line {  
    font-style: ...  
    font-variant: ...  
    font-weight: ...  
    font-size: ...  
    font-family: ...  
  
    line-height: ...  
    color: ...  
    word-spacing: ...  
    letter-spacing: ...  
    text-decoration: ...  
    text-transform: ...  
  
    background-color: ...  
    background-image: ...  
    background-position: ...  
    background-repeat: ...  
    background-size: ...  
    background-attachment: ...  
}
```



```
article {  
    padding: 20px;  
}  
  
p {  
    color: #444;  
}  
  
p::first-line {  
    color: deepskyblue;  
}
```

### ::first-line vs. tag selector

This paragraph color is set to #444 with p { color: #444; }. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

### 14.3.- First Letter

Este pseudoelemento permite aplicar estilos a la primera letra del texto de un elemento sin necesitar el uso de etiquetas como span en el html. Mientras que en no se añaden etiquetas en el código fuente, el comportamiento es similar a el uso de la etiqueta firstletter. Se le pueden asignar estilos a la primera letra, tal como se haría con elemento cualquiera.

Este pseudoelemento solo funciona en elementos de bloque, por lo que no funcionará en elementos en línea.

Si se tiene first-letter y first-line en el mismo elemento, los estilos de first-letter tienen mayor prioridad, por lo que sobreescribe otros estilos.

ACADEMIA DE SOFTWARE

**CSS**

```
p::first-letter {  
    font-weight: bold;  
    color: red;  
}
```

**HTML**

```
<p>  
    The first letter is bold and red  
</p>
```

The first letter is bold and red

---

ACADEMIA DE SOFTWARE

## Capítulo 15. MANEJO DE EVENTOS EN RESPONSIVE

### 15.1.- Pointer Events

La propiedad pointer-events permite controlar como responden los elementos html a los eventos de generados por el mouse o touch. Esto incluye los estados hover y active de css, los eventos click y tap de javascript y si el cursos es visible o no.



Esta propiedad tiene varios valores posibles, pero solo podremos usar tres para los elementos html, estos son:

- none: Previene todos los eventos de click, los estados y las opciones del cursos especificadas en el elemento.
- auto: Retorna la funcionalidad por defecto al elemento.
- inherit: Usa el valor de la propiedad del elemento padre

try selecting text  
through me

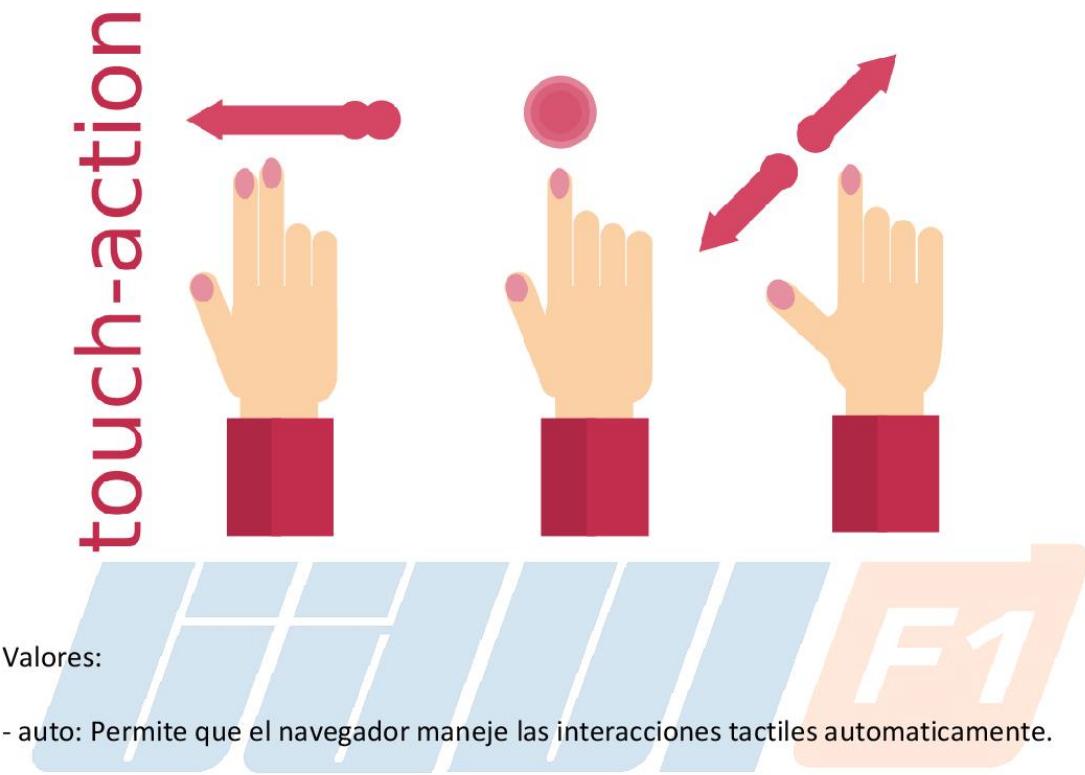
This is some basic flow content. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Magni eos ipsum sunt repellat nisi modi voluptatum ipsa eligendi minima cumque. Accusantium laudantium autem quae earum eaque expedita quia molestiae in.

### 15.2.- Touch Element

Esta propiedad permite controlar el efecto de las interacciones touchscreen de un elemento, y es similar a pointer-events.

Esta propiedad es utilizada principalmente para elementos interactivos de la interfaz de usuario que necesitan diferentes comportamientos dependiendo del dispositivo usado. Cuando se espera que un elemento se comporte de una manera con el mouse, pero de otra en una pantalla tactil, se utiliza touch-action.

Por defecto, el navegador maneja las interacciones tátiles automáticamente.



Valores:

- auto: Permite que el navegador maneje las interacciones tactiles automáticamente.
- none: Desabilita la capacidad del navegador de manejar automáticamente las interacciones. Esta opción abre la posibilidad a definir estas interacciones desde JavaScript.
- pan-left: Permite la interacción con un solo dedo donde el usuario mueve su dedo a la derecha, lo que causa que la pantalla se mueva hacia la izquierda.
- pan-right: Permite la interacción con un solo dedo donde el usuario mueve su dedo a la izquierda, lo que causa que la pantalla se mueva hacia la derecha.
- pan-down: Permite la interacción con un solo dedo donde el usuario mueve su dedo hacia arriba, lo que causa que la pantalla se mueva hacia abajo.
- pan-up: Permite la interacción con un solo dedo donde el usuario mueve su dedo hacia abajo, lo que causa que la pantalla se mueva hacia arriba.

- pinch-zoom: Permite la interacción con múltiples dedos. Puede ser combinada con otros valores pan-.
- pan-x: Permite el movimiento horizontal de la pantalla con un solo dedo. Es un atajo para los valores pan-left y pan-right.
- pan-y: Permite el movimiento vertical de la pantalla con un solo dedo. Es un atajo para los valores pan-down y pan-up.
- manipulation: Permite las interacciones de zoom y de pinchar, pero desabilia otras interacciones como doble-tap para hacer zoom.  
Es un atajo para las propiedades pan-x, pan-y y pinch-zoom.

