



**Nombre: Rodrigo Mena Serna**

**Asignatura: Diseño de Interfaces Web**

**Segundo curso de Desarrollo de Aplicaciones Web**

**Nombre de la práctica: Documentación Flexbox W3Schools**

## Contenido

CSS Flexbox.....	3
Módulo de diseño CSS Flexbox .....	3
Elementos Flexbox .....	3
CSS Flex Container.....	3
Elemento padre (contenedor).....	3
La propiedad flex-direction .....	4
La propiedad flex-wrap .....	5
La propiedad flex-flow.....	6
La propiedad justify-content .....	7
La propiedad align-items.....	8
La propiedad align-content .....	10
Centrado perfecto .....	12
Las propiedades del contenedor CSS Flexbox.....	12
Elementos flexibles de CSS.....	13
Elementos secundarios (artículos) .....	13
La propiedad order.....	13
La propiedad flex-grow .....	14
La propiedad flex-shrink.....	14
La propiedad flex-basis.....	15
La propiedad flex.....	15
La propiedad align-self .....	16
Las propiedades de los elementos de CSS Flexbox .....	17
CSS Flex Responsive .....	17
Responsive Flexbox .....	17
Galería de imágenes receptivas usando Flexbox .....	19
Sitio web receptivo usando Flexbox.....	19

## CSS Flexbox



### Módulo de diseño CSS Flexbox

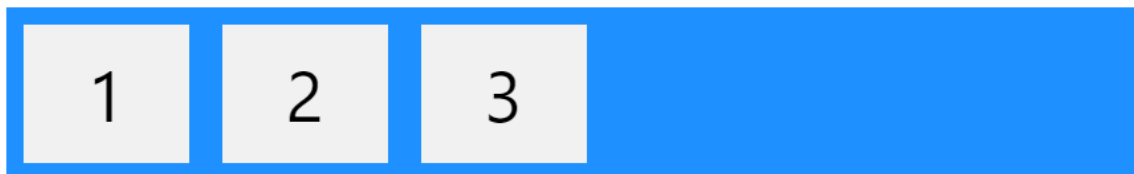
Antes del módulo Flexbox Layout, había cuatro modos de diseño:

- Bloque, para secciones en una página web
- En línea, para texto
- Tabla, para datos de tablas bidimensionales
- Posicionado, para la posición explícita de un elemento

El módulo de diseño de caja flexible facilita el diseño de una estructura de diseño receptiva flexible sin usar flotación o posicionamiento.

### Elementos Flexbox

Para comenzar a usar el modelo Flexbox, primero debe definir un contenedor flexible.



El elemento de arriba representa un contenedor flexible (el área azul) con tres elementos flexibles.

### Ejemplo

Un contenedor flexible con tres elementos flexibles:

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

### CSS Flex Container

Elemento padre (contenedor)

El contenedor flexible se vuelve flexible al establecer la propiedad display en flex:

### Ejemplo

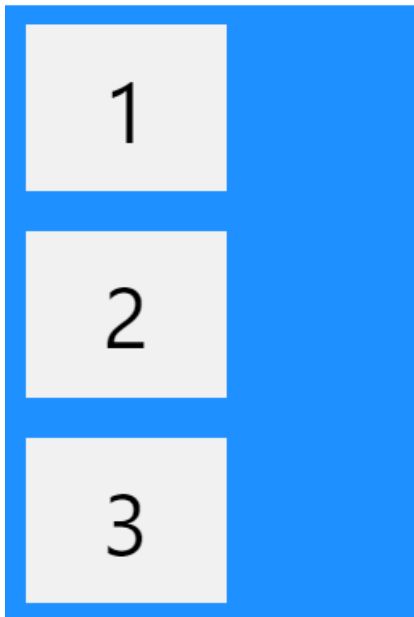
```
.flex-container {
  display: flex;
}
```

Las propiedades del contenedor flexible son:

- flex-direction.
- flex-wrap.
- flex-flow.
- justify-content.
- align-items.
- align-content.

La propiedad flex-direction

La propiedad flex-direction define en qué dirección el contenedor quiere apilar los elementos flexibles.



## Ejemplo

El `column` valor apila los elementos flexibles verticalmente (de arriba a abajo):

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

## Ejemplo

El `column-reverse` valor apila los elementos flexibles verticalmente (pero de abajo hacia arriba):

```
.flex-container {  
  display: flex;  
  flex-direction: column-reverse;  
}
```

## Ejemplo

El **row** valor apila los elementos flexibles horizontalmente (de izquierda a derecha):

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

Inténtalo tú mismo "

## Ejemplo

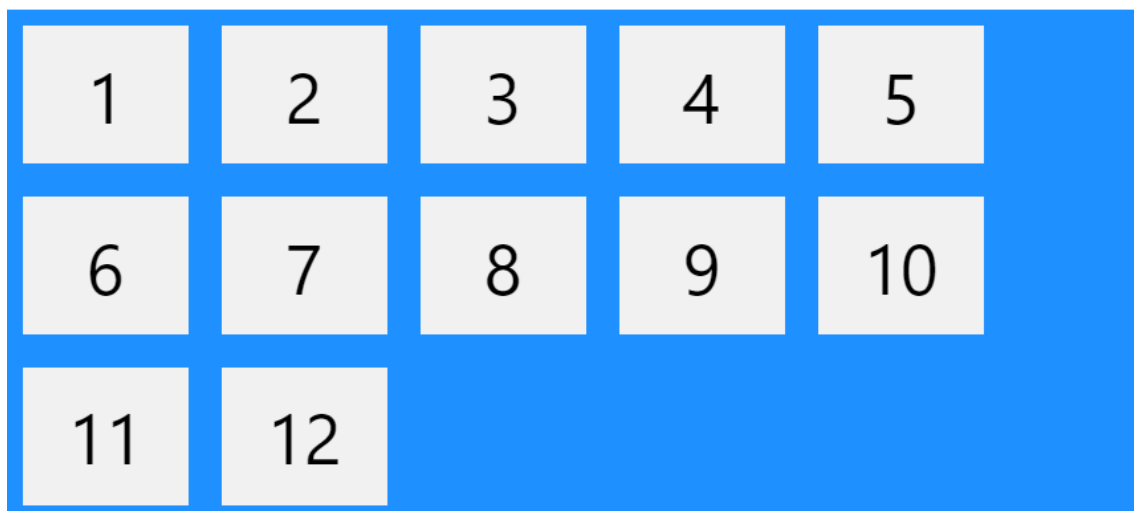
El **row-reverse** valor apila los elementos flexibles horizontalmente (pero de derecha a izquierda):

```
.flex-container {  
  display: flex;  
  flex-direction: row-reverse;  
}
```

La propiedad flex-wrap

La propiedad flex-wrap especifica si los elementos flexibles deben ajustarse o no.

Los ejemplos a continuación tienen 12 elementos flexibles para demostrar mejor la propiedad flex-wrap.



## Ejemplo

El **wrap** valor especifica que los elementos flexibles se ajustarán si es necesario:

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

## Ejemplo

El **nowrap** valor especifica que los elementos flexibles no se ajustarán (esto es predeterminado):

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

## Ejemplo

El **wrap-reverse** valor especifica que los elementos flexibles se ajustarán si es necesario, en orden inverso:

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

La propiedad flex-flow

La propiedad flex-flow es una propiedad abreviada para configurar las propiedades flex-direction y flex-wrap.

## Ejemplo

```
.flex-container {  
  display: flex;  
  flex-flow: row wrap;  
}
```

## La propiedad justify-content

La propiedad justify-content se utiliza para alinear los elementos flexibles:



### Ejemplo

El **center** valor alinea los elementos flexibles en el centro del contenedor:

```
.flex-container {  
  display: flex;  
  justify-content: center;  
}
```

### Ejemplo

El **flex-start** valor alinea los elementos flexibles al comienzo del contenedor (esto es predeterminado):

```
.flex-container {  
  display: flex;  
  justify-content: flex-start;  
}
```

### Ejemplo

El **flex-end** valor alinea los elementos flexibles al final del contenedor:

```
.flex-container {  
  display: flex;  
  justify-content: flex-end;  
}
```

### Ejemplo

El **space-around** valor muestra los elementos flexibles con espacio antes, entre y después de las líneas:

```
.flex-container {  
  display: flex;  
  justify-content: space-around;  
}
```

## Ejemplo

El **space-between** valor muestra los elementos flexibles con espacio entre líneas:

```
.flex-container {  
  display: flex;  
  justify-content: space-between;  
}
```

La propiedad align-items

La propiedad align-items se utiliza para alinear los elementos flexibles.



En estos ejemplos, usamos un contenedor de 200 píxeles de alto para demostrar mejor la propiedad align-items.

## Ejemplo

El **center** valor alinea los elementos flexibles en el centro del contenedor:

```
.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: center;  
}
```

## Ejemplo

El **flex-start** valor alinea los elementos flexibles en la parte superior del contenedor:

```
.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: flex-start;  
}
```



## Ejemplo

El **flex-end** valor alinea los elementos flexibles en la parte inferior del contenedor:

```
.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: flex-end;  
}
```

## Ejemplo

El **stretch** valor estira los elementos flexibles para llenar el contenedor (esto es predeterminado):

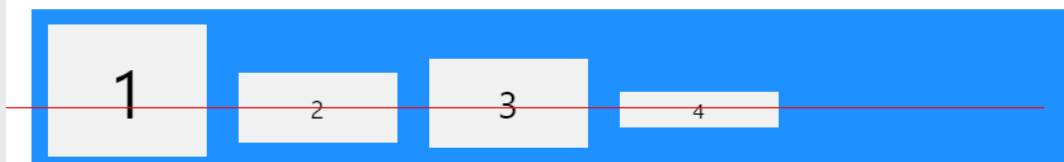
```
.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: stretch;  
}
```

## Ejemplo

El **baseline** valor alinea los elementos flexibles como se alinean sus líneas base:

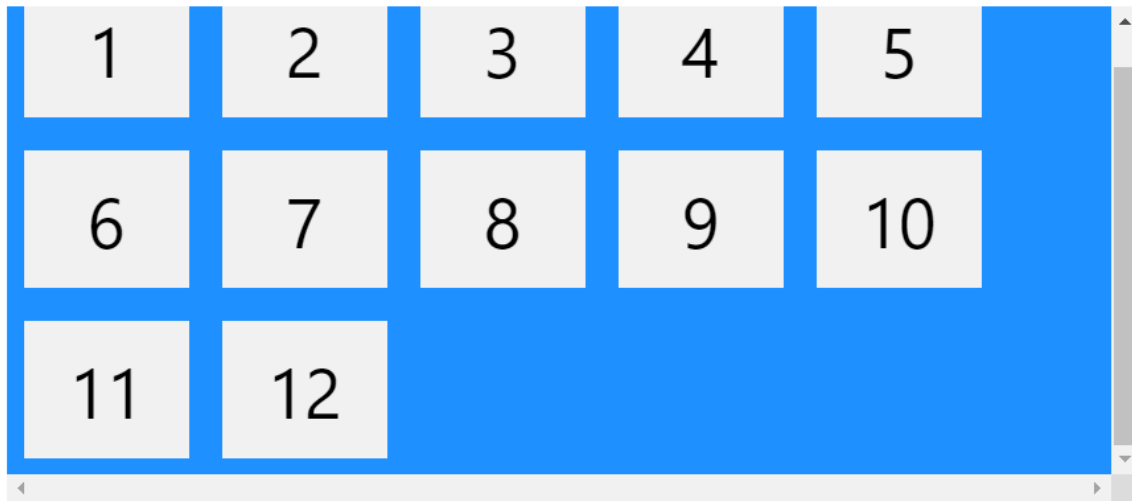
```
.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: baseline;  
}
```

**Nota:** el ejemplo utiliza un tamaño de fuente diferente para demostrar que los elementos se alinean con la línea de base del texto:



### La propiedad align-content

La propiedad align-content se utiliza para alinear las líneas flexibles.



En estos ejemplos, usamos un contenedor de 600 píxeles de alto, con la propiedad flex-wrap establecida en wrap, para demostrar mejor la propiedad align-content.

### Ejemplo

El `space-between` valor muestra las líneas flexibles con el mismo espacio entre ellas:

```
.flex-container {  
  display: flex;  
  height: 600px;  
  flex-wrap: wrap;  
  align-content: space-between;  
}
```

### Ejemplo

El `space-around` valor muestra las líneas flexibles con espacio antes, entre y después de ellas:

```
.flex-container {  
  display: flex;  
  height: 600px;  
  flex-wrap: wrap;  
  align-content: space-around;  
}
```

## Ejemplo

El **stretch** valor estira las líneas flexibles para ocupar el espacio restante (esto es predeterminado):

```
.flex-container {  
  display: flex;  
  height: 600px;  
  flex-wrap: wrap;  
  align-content: stretch;  
}
```

## Ejemplo

Las **center** pantallas de valor muestran las líneas flexibles en el medio del contenedor:

```
.flex-container {  
  display: flex;  
  height: 600px;  
  flex-wrap: wrap;  
  align-content: center;  
}
```

## Ejemplo

El **flex-start** valor muestra las líneas flexibles al comienzo del contenedor:

```
.flex-container {  
  display: flex;  
  height: 600px;  
  flex-wrap: wrap;  
  align-content: flex-start;  
}
```

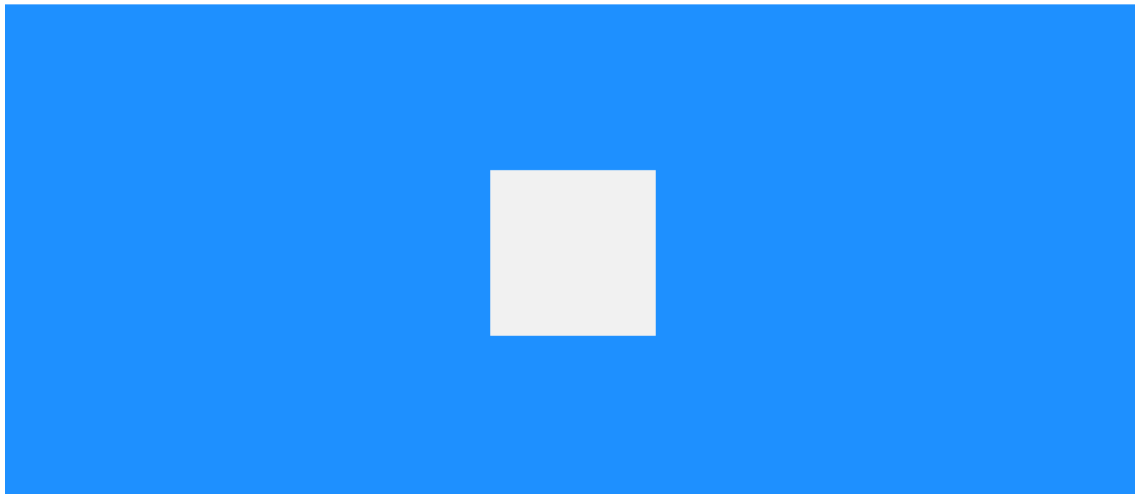
## Ejemplo

El **flex-end** valor muestra las líneas flexibles al final del contenedor:

```
.flex-container {  
  display: flex;  
  height: 600px;  
  flex-wrap: wrap;  
  align-content: flex-end;  
}
```

### Centrado perfecto

En el siguiente ejemplo resolveremos un problema de estilo muy común: el centrado perfecto.



**SOLUCIÓN:** Establezca las propiedades `justify-content` y `align-items` en `center`, y el elemento flexible quedará perfectamente centrado.

### Ejemplo

```
.flex-container {  
  display: flex;  
  height: 300px;  
  justify-content: center;  
  align-items: center;  
}
```

### Las propiedades del contenedor CSS Flexbox

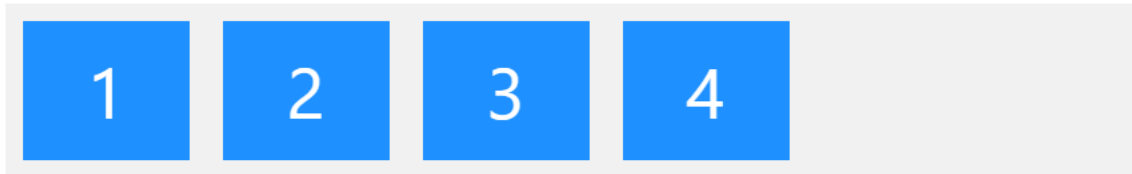
La siguiente tabla enumera todas las propiedades de CSS Flexbox Container:

Property	Description
<a href="#"><code>align-content</code></a>	Modifies the behavior of the <code>flex-wrap</code> property. It is similar to <code>align-items</code> , but instead of aligning flex items, it aligns flex lines
<a href="#"><code>align-items</code></a>	Vertically aligns the flex items when the items do not use all available space on the cross-axis
<a href="#"><code>display</code></a>	Specifies the type of box used for an HTML element
<a href="#"><code>flex-direction</code></a>	Specifies the direction of the flexible items inside a flex container
<a href="#"><code>flex-flow</code></a>	A shorthand property for <code>flex-direction</code> and <code>flex-wrap</code>
<a href="#"><code>flex-wrap</code></a>	Specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line
<a href="#"><code>justify-content</code></a>	Horizontally aligns the flex items when the items do not use all available space on the main-axis

## Elementos flexibles de CSS

### Elementos secundarios (artículos)

Los elementos secundarios directos de un contenedor flexible se convierten automáticamente en elementos flexibles (flex).



El elemento de arriba representa cuatro elementos flexibles azules dentro de un contenedor flexible gris.

### Ejemplo

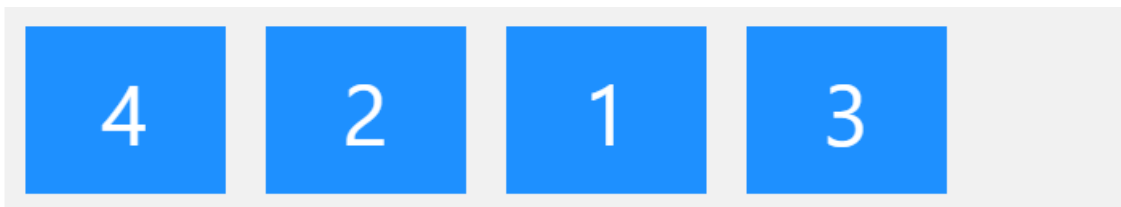
```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
</div>
```

Las propiedades del elemento flexible son:

- order.
- flex-grow.
- flex-shrink.
- flex-basis.
- flex.
- align-self.

La propiedad order

La propiedad order especifica el orden de los elementos flexibles.



El primer elemento flexible del código no tiene que aparecer como el primer elemento del diseño.

El valor del pedido debe ser un número, el valor predeterminado es 0.

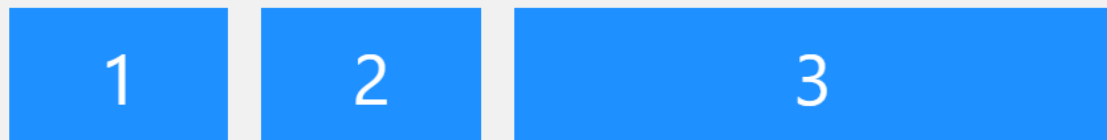
## Ejemplo

La propiedad *order* puede cambiar el orden de los elementos flexibles:

```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```

La propiedad *flex-grow*

La propiedad *flex-grow* especifica cuánto crecerá un elemento flexible en relación con el resto de los elementos flexibles.



El valor debe ser un número, el valor predeterminado es 0.

## Ejemplo

Haz que el tercer elemento flexible crezca ocho veces más rápido que los otros elementos flexibles:

```
<div class="flex-container">
  <div style="flex-grow: 1">1</div>
  <div style="flex-grow: 1">2</div>
  <div style="flex-grow: 8">3</div>
</div>
```

La propiedad *flex-shrink*

La propiedad *flex-shrink* especifica cuánto se encogerá un elemento flexible en relación con el resto de los elementos flexibles.



El valor debe ser un número, el valor predeterminado es 1.

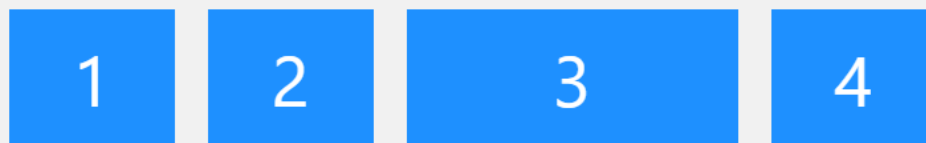
## Ejemplo

No permita que el tercer elemento flexible se encoja tanto como los otros elementos flexibles:

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-shrink: 0">3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
</div>
```

La propiedad flex-basis

La propiedad flex-basis especifica la longitud inicial de un elemento flexible.



## Ejemplo

Establezca la longitud inicial del tercer elemento flexible en 200 píxeles:

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-basis: 200px">3</div>
  <div>4</div>
</div>
```

La propiedad flex

La propiedad flex es una propiedad abreviada para las propiedades flex-grow, flex-shrinky flex-basis.

## Ejemplo

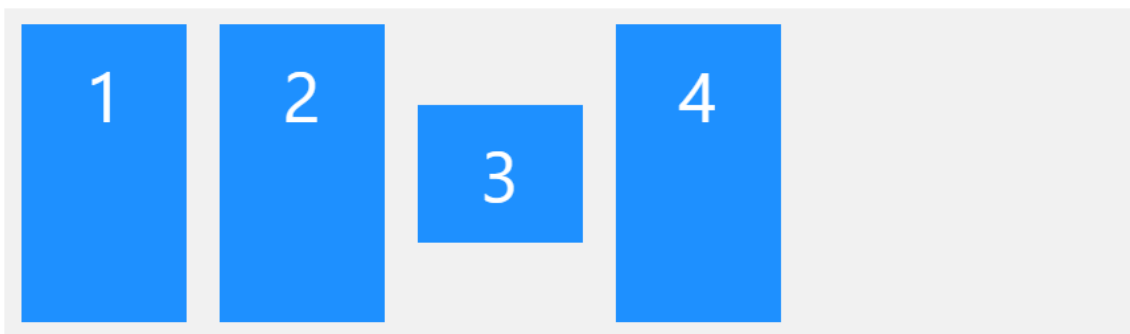
Haga que el tercer elemento flexible no se pueda crecer (0), no se pueda encoger (0) y con una longitud inicial de 200 píxeles:

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex: 0 0 200px">3</div>
  <div>4</div>
</div>
```

La propiedad align-self

La propiedad align-self especifica la alineación del elemento seleccionado dentro del contenedor flexible.

La propiedad align-self anula la alineación predeterminada establecida por la propiedad align-items del contenedor.



En estos ejemplos, usamos un contenedor de 200 píxeles de alto para demostrar mejor la propiedad align-self:

## Ejemplo

Alinee el tercer elemento flexible en el medio del contenedor:

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="align-self: center">3</div>
  <div>4</div>
</div>
```



## Ejemplo

Alinee el segundo elemento flexible en la parte superior del contenedor y el tercer elemento flexible en la parte inferior del contenedor:

```
<div class="flex-container">
  <div>1</div>
  <div style="align-self: flex-start">2</div>
  <div style="align-self: flex-end">3</div>
  <div>4</div>
</div>
```

Las propiedades de los elementos de CSS Flexbox

La siguiente tabla enumera todas las propiedades de los elementos CSS Flexbox:

Property	Description
<u>align-self</u>	Specifies the alignment for a flex item (overrides the flex container's align-items property)
<u>flex</u>	A shorthand property for the flex-grow, flex-shrink, and the flex-basis properties
<u>flex-basis</u>	Specifies the initial length of a flex item
<u>flex-grow</u>	Specifies how much a flex item will grow relative to the rest of the flex items inside the same container
<u>flex-shrink</u>	Specifies how much a flex item will shrink relative to the rest of the flex items inside the same container
<u>order</u>	Specifies the order of the flex items inside the same container

## CSS Flex Responsive

### Responsive Flexbox

Aprendió del capítulo CSS Media Queries que puede usar consultas de medios para crear diferentes diseños para diferentes dispositivos y tamaños de pantalla.

Computadoras portátiles y de escritorio:

1	2	3
---	---	---

Móviles y Tablets :

1
2
3

Por ejemplo, si desea crear un diseño de dos columnas para la mayoría de los tamaños de pantalla y un diseño de una columna para pantallas pequeñas (como teléfonos y tabletas), puede cambiar el flex-direction de row a column en un punto de interrupción específico (800 px en el ejemplo a continuación):

### Ejemplo

```
.flex-container {  
  display: flex;  
  flex-direction: row;  
}  
  
/* Responsive layout - makes a one column layout instead of a two-column layout  
*/  
@media (max-width: 800px) {  
  .flex-container {  
    flex-direction: column;  
  }  
}
```

Otra forma es cambiar el porcentaje de la propiedad flex de los elementos flexibles para crear diferentes diseños para diferentes tamaños de pantalla. Tenga en cuenta que también tenemos que incluir flex-wrap: wrap; en el contenedor flexible para que este ejemplo funcione:

### Ejemplo

```
.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
}  
  
.flex-item-left {  
  flex: 50%;  
}  
  
.flex-item-right {  
  flex: 50%;  
}  
  
/* Responsive layout - makes a one column layout instead of a two-column layout  
*/  
@media (max-width: 800px) {  
  .flex-item-right, .flex-item-left {  
    flex: 100%;  
  }  
}
```

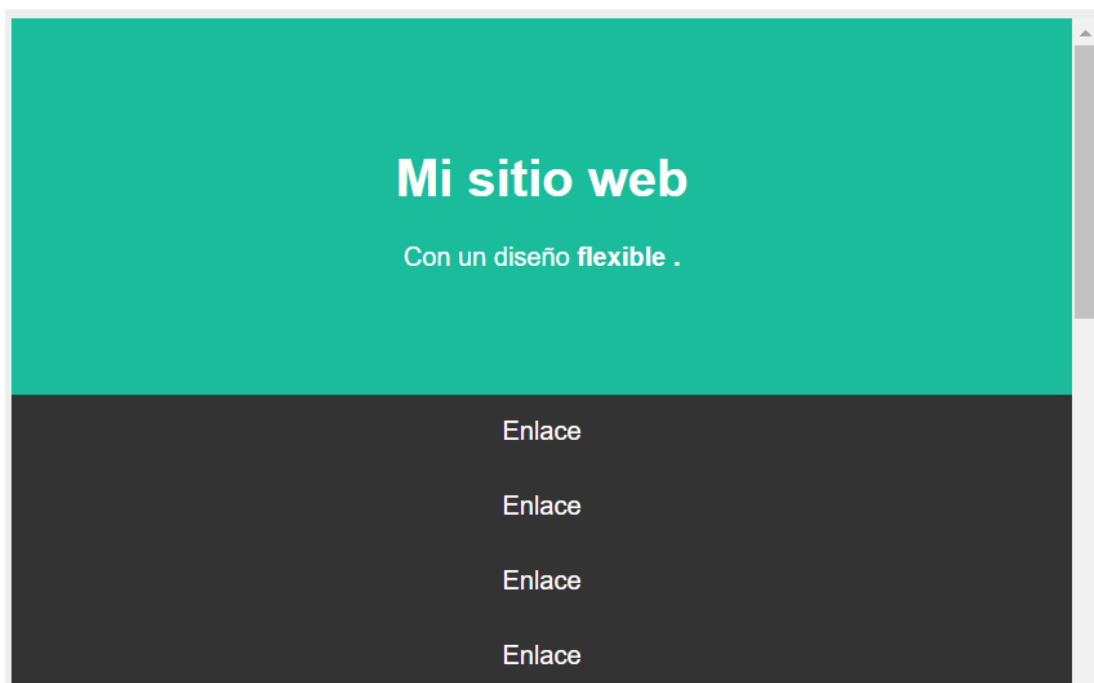
### Galería de imágenes receptivas usando Flexbox

Use flexbox para crear una galería de imágenes receptiva que varía entre cuatro, dos o imágenes de ancho completo, según el tamaño de la pantalla:



### Sitio web receptivo usando Flexbox

Use flexbox para crear un sitio web receptivo, que contenga una barra de navegación flexible y contenido flexible:



## Acerca de mí

Foto de mí:

Imagen

Algún texto sobre mí en culpa qui officia deserunt mollit anim..

### Más texto

Lorem ipsum dolor siéntate ame.

Imagen

### Más texto

Lorem ipsum dolor siéntate ame.

Imagen

Imagen

Imagen

## TÍTULO ENCABEZAMIENTO


Descripción del título, 7 de diciembre de 2017

Imagen

## TÍTULO ENCABEZAMIENTO

Descripción del título, 7 de diciembre de 2017

Imagen




Algún texto..

Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco.

## TÍTULO ENCABEZAMIENTO

Descripción del título, 2 de septiembre de 2017

Imagen



Algún texto..

Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco.

**Pie de página**

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>Page Title</title>
5  <meta charset="UTF-8">
6  <meta name="viewport" content="width=device-width, initial-scale=1">
7  <style>
8  * {
9  |   box-sizing: border-box;
10 }
11
12 /* Style the body */
13 body {
14 |   font-family: Arial;
15 |   margin: 0;
16 }
17
18 /* Header/logo Title */
19 .header {
20 |   padding: 60px;
21 |   text-align: center;
22 |   background: #1abc9c;
23 |   color: white;
24 }
25
26 /* Style the top navigation bar */
27 .navbar {
28 |   display: flex;
29 |   background-color: #333;
30 }
31
32 /* Style the navigation bar links */
33 .navbar a {
34 |   color: white;
35 |   padding: 14px 20px;
36 |   text-decoration: none;
37 |   text-align: center;
38 }
39
40 /* Change color on hover */
41 .navbar a:hover {
42 |   background-color: #ddd;
43 |   color: black;
44 }
```

```
/* Column container */
.row {
  display: flex;
  flex-wrap: wrap;
}

/* Create two unequal columns that sits next to each other */
/* Sidebar/left column */
.side {
  flex: 30%;
  background-color: #f1f1f1;
  padding: 20px;
}

/* Main column */
.main {
  flex: 70%;
  background-color: white;
  padding: 20px;
}

/* Fake image, just for this example */
.fakeimg {
  background-color: #aaa;
  width: 100%;
  padding: 20px;
}

/* Footer */
.footer {
  padding: 20px;
  text-align: center;
  background: #ddd;
}

/* Responsive layout - when the screen is less than 700px wide, make the two columns stack
@media screen and (max-width: 700px) {
  .row, .navbar {
    flex-direction: column;
  }
}
```

```

    }
  }
</style>
</head>
<body>

<!-- Note -->
<div style="background: yellow;padding:5px">
  <h4 style="text-align:center">Resize the browser window to see the responsive effect.</h4>
</div>

<!-- Header -->
<div class="header">
  <h1>My Website</h1>
  <p>With a <b>flexible</b> layout.</p>
</div>

<!-- Navigation Bar -->
<div class="navbar">
  <a href="#">Link</a>
  <a href="#">Link</a>
  <a href="#">Link</a>
  <a href="#">Link</a>
</div>

<!-- The flexible grid (content) -->
<div class="row">
  <div class="side">
    <h2>About Me</h2>
    <h5>Photo of me:</h5>
    <div class="fakeimg" style="height:200px;">Image</div>
    <p>Some text about me in culpa qui officia deserunt mollit anim..</p>
    <h3>More Text</h3>
    <p>Lorem ipsum dolor sit ame.</p>
    <div class="fakeimg" style="height:60px;">Image</div><br>
    <div class="fakeimg" style="height:60px;">Image</div><br>
    <div class="fakeimg" style="height:60px;">Image</div>
  </div>
  <div class="main">
    <h2>TITLE HEADING</h2>
    <h5>Title description, Dec 7, 2017</h5>
    <div class="fakeimg" style="height:200px;">Image</div>
    <p>Some text..</p>
  </div>
</div>

```



```
<div class="fakeimg" style="height:200px;">Image</div>
<p>Some text..</p>
<p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing
<br>
<h2>TITLE HEADING</h2>
<h5>Title description, Sep 2, 2017</h5>
<div class="fakeimg" style="height:200px;">Image</div>
<p>Some text..</p>
<p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing
</div>
</div>

<!-- Footer -->
<div class="footer">
  <h2>Footer</h2>
</div>

</body>
</html>
```