

Caja flexible de CSS

Flexbox es un método de diseño potente y bien soportado que se introdujo con la última versión de CSS, CSS3. Con flexbox, es fácil centrar elementos en la página y crear interfaces de usuario dinámicas que se reducen y expanden automáticamente.

En este curso, aprenderá los fundamentos de flexbox y diseños dinámicos mediante la creación de una tarjeta de Twitter.

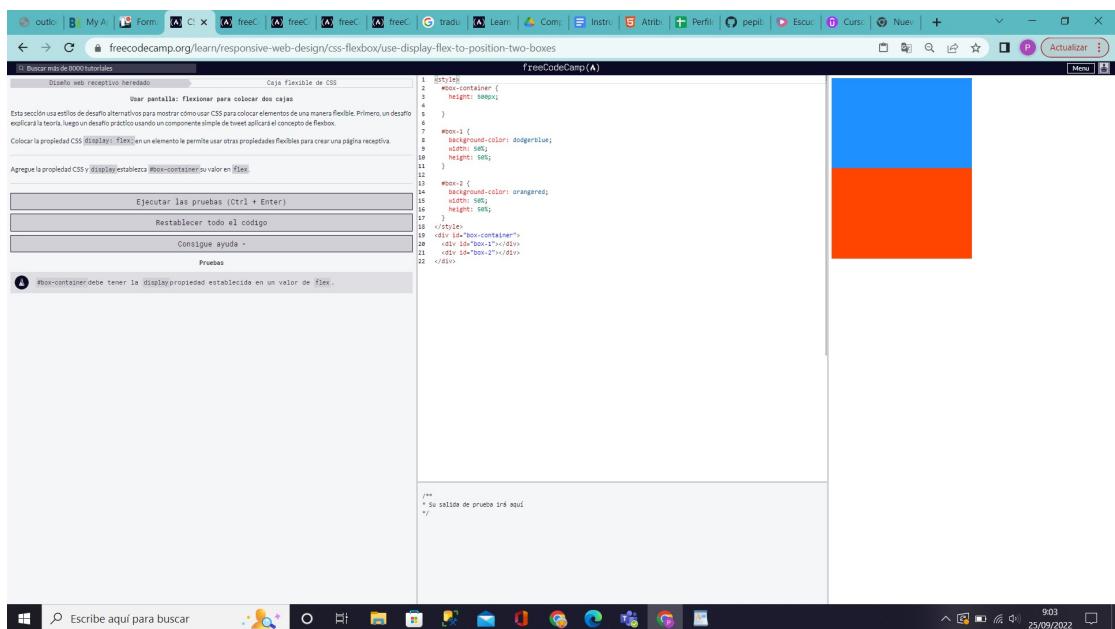
1. Usar pantalla: flexionar para colocar dos cajas
2. Agrega Flex Superpowers al Tweet Incrustado
3. Use la propiedad de dirección flexible para hacer una fila
4. Aplique la propiedad de dirección flexible para crear filas en el Tweet incrustado
5. Use la propiedad de dirección flexible para hacer una columna
6. Aplique la propiedad de dirección flexible para crear una columna en el Tweet incrustado
7. Alinear elementos usando la propiedad de justificar contenido
8. Use la propiedad de justificar el contenido en el Tweet incrustado
9. Alinear elementos usando la propiedad align-items
10. Use la propiedad align-items en el Tweet incrustado
11. Use la propiedad flex-wrap para ajustar una fila o columna
12. Utilice la propiedad flex-shrink para reducir elementos
13. Use la propiedad flex-grow para expandir elementos
14. Utilice la propiedad de base flexible para establecer el tamaño inicial de un elemento
15. Usar la propiedad abreviada flex
16. Use la propiedad order para reorganizar elementos
17. Utilice la propiedad align-self

1. Usar pantalla: flexionar para colocar dos cajas

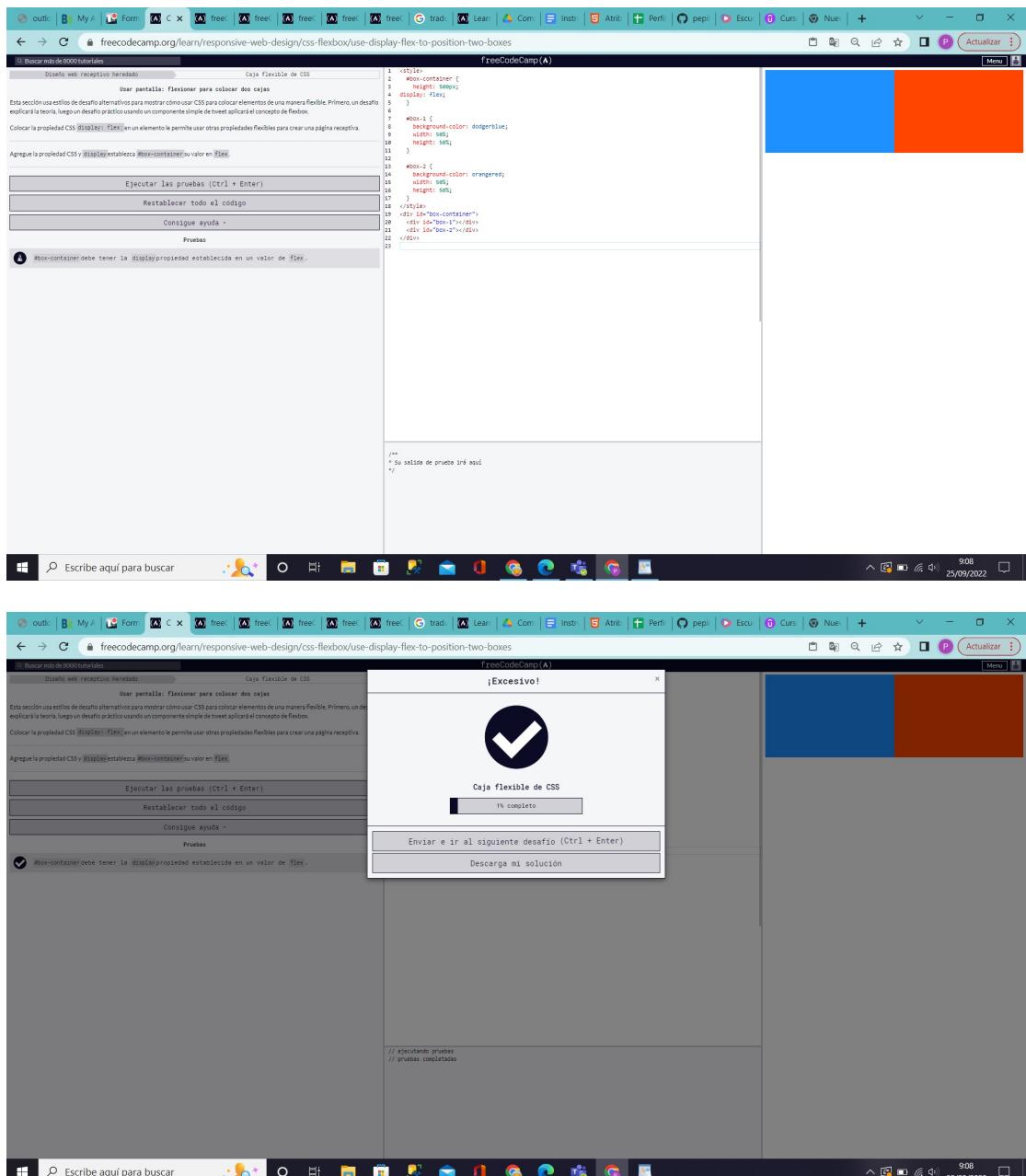
Esta sección usa estilos de desafío alternativos para mostrar cómo usar CSS para colocar elementos de una manera flexible. Primero, un desafío explicará la teoría, luego un desafío práctico usando un componente simple de tweet aplicará el concepto de flexbox.

Colocar la propiedad CSS `display: flex;` en un elemento le permite usar otras propiedades flexibles para crear una página receptiva.

Agregue la propiedad CSS y establezca `#box-container` su valor en `flex`.



```
1 0ptile
2 #box-container {
3   height: 50px;
4 }
5
6 #box-1 {
7   background-color: dodgerblue;
8   width: 50px;
9   height: 50px;
10 }
11
12 #box-2 {
13   background-color: orangered;
14   width: 50px;
15   height: 50px;
16 }
17
18 /*style*/
19 #box-container{
20   display:flex;
21   <div id="box-1"></div>
22   <div id="box-2"></div>
23 }
```



<style>

```
#box-container {
```

```
height: 500px;
```

display: flex;

}

```
#box-1 {
```

```

background-color: dodgerblue;
width: 50%;

height: 50%;

}

#box-2 {

background-color: orangered;
width: 50%;

height: 50%;

}

</style>

<div id="box-container">
<div id="box-1"></div>
<div id="box-2"></div>
</div>

```

2. Agrega Flex Superpowers (superpoderes) al Tweet Incrustado

A la derecha está el tweet incrustado que se usará como ejemplo práctico. Algunos de los elementos se verían mejor con un diseño diferente. El último desafío lo demostró display: flex. Aquí lo agregará a varios componentes en el tweet incrustado para comenzar a ajustar su posicionamiento.

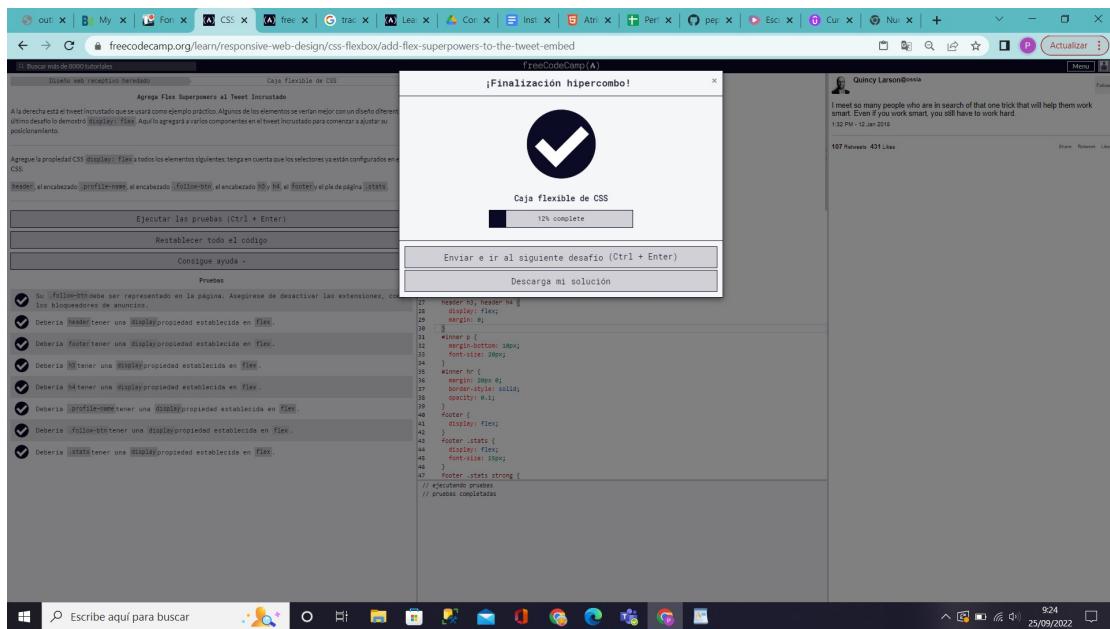
Agregue la propiedad CSS display: flex a todos los elementos siguientes; tenga en cuenta que los selectores ya están configurados en el CSS:

header, el encabezado .profile-name, el encabezado .follow-btn, el encabezado h3y h4, el footer y el pie de página .stats.

The screenshot shows a browser window with the URL freeCodeCamp.org/learn/responsive-web-design/css-flexbox/add-flex-superpowers-to-the-tweet-embed. The main content area contains a code editor with the following CSS code:

```
1  @charset "utf-8";
2  body {
3      font-family: Arial, sans-serif;
4  }
5  header {
6      display: flex;
7      align-items: center;
8  }
9  header .profile-thumbail {
10     width: 50px;
11     height: 50px;
12     border-radius: 50px;
13     margin-right: 10px;
14 }
15 header .profile-name {
16     margin-left: 10px;
17 }
18 header .follow-btn {
19     margin: 0 auto;
20 }
21 header .follow-btn button {
22     border: 1px solid #ccc;
23     padding: 5px;
24 }
25 }
26 header h3, header h4 {
27     margin: 0;
28 }
29 header p {
30     margin-top: 20px;
31     font-size: 20px;
32 }
33 header hr {
34     margin: 10px 0;
35     border-top: 1px solid #ccc;
36     opacity: 0.1;
37 }
38 footer {
39 }
40 footer .stats {
41 }
42 footer .stats strong {
43     font-size: 15px;
44 }
45 footer .stats strong {
46     font-size: 10px;
47 }
48 */
49 /* Su salida de prueba irá aquí
50 */
```

This screenshot shows the same browser window at a later time, indicated by the timestamp 9:11 25/09/2022 in the bottom right corner. The right sidebar now displays a tweet from Quincy Larson dated 1:32 PM - 12 Jan 2018.



Su **.follow-btn** debe ser representado en la página. Asegúrese de desactivar las extensiones, como los bloqueadores de anuncios.

Debería **header** tener una displaypropiedad establecida en **flex**.

Debería **footer** tener una displaypropiedad establecida en **flex**.

Debería **h3** tener una displaypropiedad establecida en **flex**.

Debería **h4** tener una displaypropiedad establecida en **flex**.

Debería **.profile-name** tener una displaypropiedad establecida en **flex**.

Debería **.follow-btn** tener una displaypropiedad establecida en **flex**.

Debería **.stats** tener una displaypropiedad establecida en **flex**.

Sí:

```
<style>
body {
    font-family: Arial, sans-serif;
}
header {
    display: flex;
```

```
header .profile-thumbnail {  
    width: 50px;  
    height: 50px;  
    border-radius: 4px;  
}  
  
header .profile-name {  
    display: flex;  
    margin-left: 10px;  
}  
  
header .follow-btn {  
    display: flex;  
    margin: 0 0 0 auto;  
}  
  
header .follow-btn button {  
    border: 0;  
    border-radius: 3px;  
    padding: 5px;  
}  
  
header h3, header h4 {  
    display: flex;  
    margin: 0;  
}  
  
#inner p {  
    margin-bottom: 10px;  
    font-size: 20px;  
}
```

```
#inner hr {  
    margin: 20px 0;  
    border-style: solid;  
    opacity: 0.1;  
}  
  
footer {  
    display: flex;  
}  
  
footer .stats {  
    display: flex;  
    font-size: 15px;  
}  
  
footer .stats strong {  
    font-size: 18px;  
}  
  
footer .stats .likes {  
    margin-left: 10px;  
}  
  
footer .cta {  
    margin-left: auto;  
}  
  
footer .cta button {  
    border: 0;  
    background: transparent;  
}  
  
</style>
```

```
<header>



<div class="profile-name">
  <h3>Quincy Larson</h3>
  <h4>@ossia</h4>
</div>

<div class="follow-btn">
  <button>Follow</button>
</div>

</header>

<div id="inner">

  <p>I meet so many people who are in search of that one trick that will help them work smart. Even if you work smart, you still have to work hard.</p>

  <span class="date">1:32 PM - 12 Jan 2018</span>

  <hr>

</div>

<footer>

  <div class="stats">

    <div class="Retweets">
      <strong>107</strong> Retweets
    </div>

    <div class="likes">
      <strong>431</strong> Likes
    </div>
  </div>

  <div class="cta">
```

```

<button class="share-btn">Share</button>

<button class="retweet-btn">Retweet</button>

<button class="like-btn">Like</button>

</div>

</footer>

```

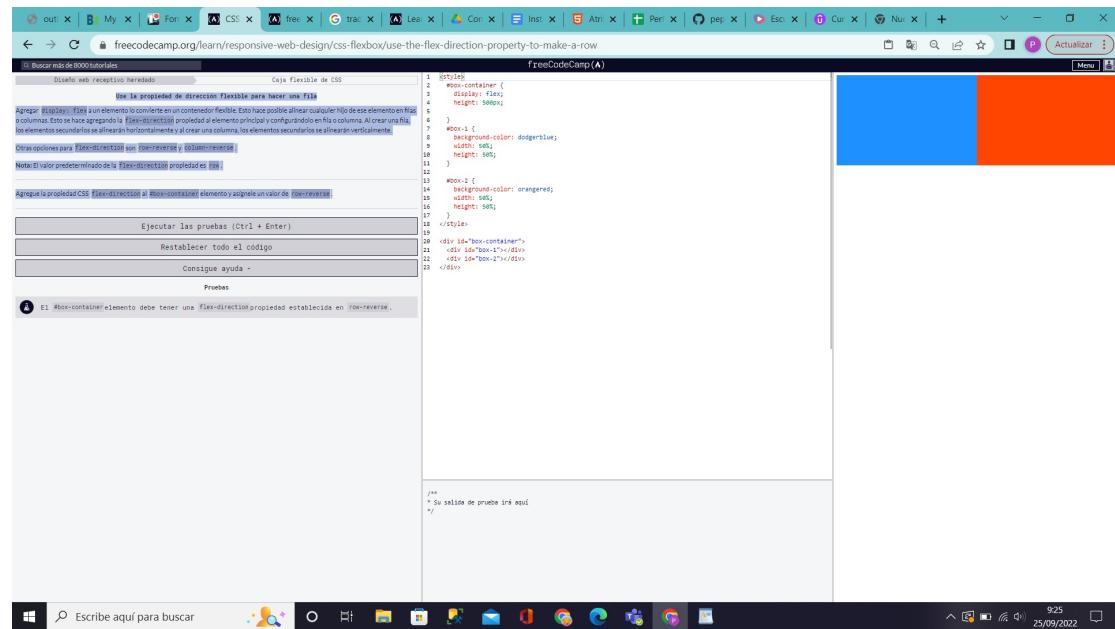
3. Use la propiedad de dirección flexible para hacer una fila

Agregar display: flex a un elemento lo convierte en un contenedor flexible. Esto hace posible alinear cualquier hijo de ese elemento en filas o columnas. Esto se hace agregando la flex-directionpropiedad al elemento principal y configurándolo en fila o columna. Al crear una fila, los elementos secundarios se alinearán horizontalmente y al crear una columna, los elementos secundarios se alinearán verticalmente.

Otras opciones para flex-directionson row-reversey column-reverse.

Nota: El valor predeterminado de la flex-directionpropiedad es row.

Agregue la propiedad CSS flex-directional #box-containerelemento y asígnele un valor de row-reverse.



The screenshot shows a browser window with the URL freeCodeCamp.org/learn/responsive-web-design/css-flexbox/use-the-flex-direction-property-to-make-a-row. The page content is in Spanish. It includes a code editor with the following CSS:

```

1  /*styles*/
2  #box-container {
3      display: flex;
4      height: 80px;
5      flex-direction: row-reverse;
6  }
7  #box-1 {
8      background-color: dodgerblue;
9      width: 50px;
10     height: 50px;
11 }
12 #box-2 {
13     background-color: orange;
14     width: 50px;
15     height: 50px;
16 }
17 
```

Below the code editor, there's a note: "El #box-container elemento debe tener una flex-direction propiedad establecida en row-reverse." At the bottom of the code editor, there's a section for "Pruebas" (Tests) with a note: "/* Salida de prueba irá aquí */".

The browser taskbar at the bottom shows various icons and the date/time: 9:34 25/09/2022.

Una vez que tenga un contenedor flexible agregando `display: flex;` al contenedor principal, puede especificar si desea que los elementos secundarios se apilen en una fila agregando lo siguiente:

```

#box-container {

    display: flex;

    /* This makes the flex container */

    /* Esto hace que el contenedor flexible */

```

```
height: 500px;
flex-direction: row-reverse;

/* This makes the direction be a row with reversed elements */

/* Esto hace que la dirección sea una fila con elementos invertidos */

}
```

Notarás cómo los colores cambian de posición, ya que la dirección predeterminada de los contenedores flexibles son filas, como habrás notado en el ejemplo del tweet .

```
<style>

#box-container {
    display: flex;
    height: 500px;
flex-direction: row-reverse;

}

#box-1 {
    background-color: dodgerblue;
    width: 50%;
    height: 50%;
}

#box-2 {
    background-color: orangered;
    width: 50%;
    height: 50%;
}

</style>

<div id="box-container">
```

```

<div id="box-1"></div>

<div id="box-2"></div>

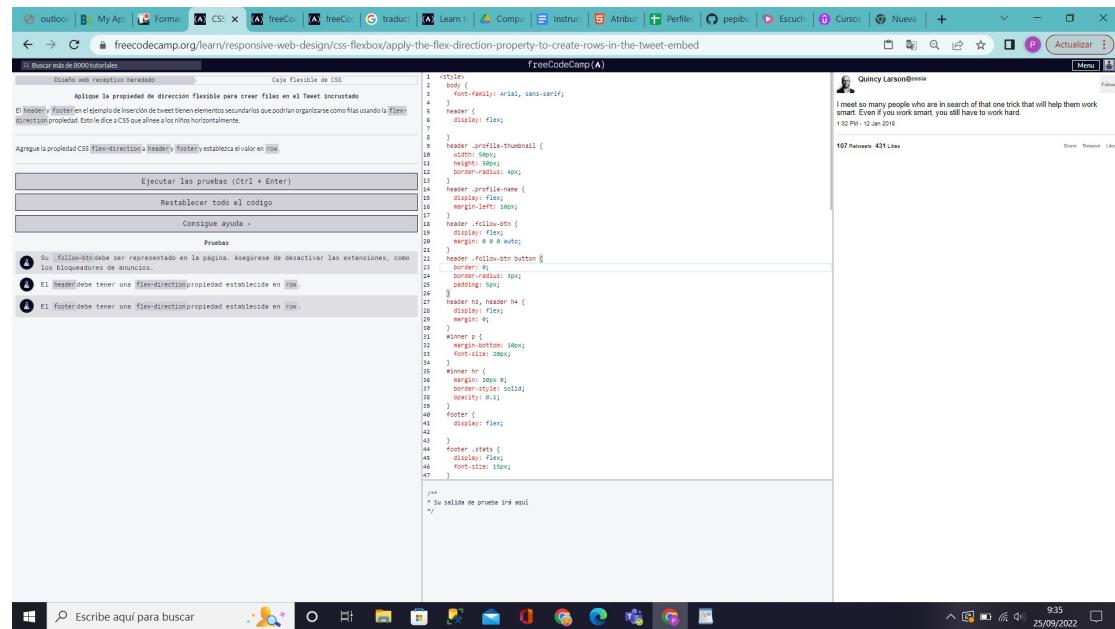
</div>

```

4. Aplique la propiedad de dirección flexible para crear filas en el Tweet incrustado

El headery footeren el ejemplo de inserción de tweet tienen elementos secundarios que podrían organizarse como filas usando la flex-directionpropiedad. Esto le dice a CSS que alinee a los niños horizontalmente.

Agregue la propiedad CSS flex-directiona headery footer establezca el valor en row.



outlook | B My API | Formas | CSS x freeCo traduc... Learn | Comp | Instruc | Atribu | Perfil | pepi... Escuch | Cursos | Nueva | +

freeCodeCamp(A)

Quincy Larson@ossea

I need so many people who are in search of that one trick that will help them work smart. Even if you work smart, you still have to work hard.

1:32 PM - 12 Jan 2016

107 Retweets 431 Likes

Ejecutar las pruebas (Ctrl + Enter)

Restablecer todo el código

Consigue ayuda -

Pruebas

```

1  /*styles
2   body {
3     font-family: Arial, sans-serif;
4   }
5   header {
6     display: flex;
7     flex-direction: row;
8   }
9   header .profile-thumb {
10    width: 40px;
11    height: 40px;
12    border-radius: 4px;
13  }
14  header .profile-name {
15    display: flex;
16    margin-left: 10px;
17  }
18  header .follow-btn {
19    display: flex;
20    margin: 0 auto;
21  }
22  header .follow-btn button {
23    width: 100px;
24    border-radius: 1px;
25    padding: 5px;
26  }
27  header h3, header h4 {
28    display: flex;
29    margin: 0;
30  }
31  header x {
32    margin-bottom: 10px;
33    font-size: 18px;
34  }
35  header hr {
36    margin: 20px 0;
37    border-style: solid;
38    opacity: 0.1;
39  }
40  footer {
41    display: flex;
42    flex-direction: column;
43  }
44  footer .stats {
45    display: flex;
46    font-size: 15px;
47  }

/*
// Salida de prueba irá aquí
*/

```

Escribe aquí para buscar

outlook | B My API | Formas | CSS x freeCo traduc... Learn | Comp | Instruc | Atribu | Perfil | pepi... Escuch | Cursos | Nueva | +

freeCodeCamp(A)

Quincy Larson@ossea

I need so many people who are in search of that one trick that will help them work smart. Even if you work smart, you still have to work hard.

1:32 PM - 12 Jan 2016

107 Retweets 431 Likes

Ejecutar las pruebas (Ctrl + Enter)

Restablecer todo el código

Consigue ayuda -

Pruebas

Su `.follow-btn` debe ser representado en la página. Asegúrese de desactivar las extensiones, como los bloqueadores de anuncios.

El `header` debe tener una `flex-direction` propiedad establecida en `row`.

El `footer` debe tener una `flex-direction` propiedad establecida en `row`.

Caja flexible de CSS

24h complete

Enviar e ir al siguiente desafío (Ctrl + Enter)

Descarga mi solución

```

17  header h3, header h4 {
18    display: flex;
19    margin: 0;
20  }
21  header p {
22    margin-bottom: 10px;
23    font-size: 18px;
24  }
25  header hr {
26    margin: 20px 0;
27    border-style: solid;
28    opacity: 0.1;
29  }
30  footer {
31    display: flex;
32    flex-direction: column;
33  }
34  footer .stats {
35    display: flex;
36    font-size: 15px;
37  }

/*
// Ejecutando pruebas
// Pruebas completadas

```

Escribe aquí para buscar

```

<style>

body {
  font-family: Arial, sans-serif;
}

header {
  display: flex;

```

```
flex-direction: row;
}

header .profile-thumbnail {
    width: 50px;
    height: 50px;
    border-radius: 4px;
}

header .profile-name {
    display: flex;
    margin-left: 10px;
}

header .follow-btn {
    display: flex;
    margin: 0 0 0 auto;
}

header .follow-btn button {
    border: 0;
    border-radius: 3px;
    padding: 5px;
}

header h3, header h4 {
    display: flex;
    margin: 0;
}

#inner p {
    margin-bottom: 10px;
    font-size: 20px;
}
```

```
#inner hr {  
    margin: 20px 0;  
    border-style: solid;  
    opacity: 0.1;  
}  
  
footer {  
    display: flex;  
  
flex-direction: row;  
}  
  
footer .stats {  
    display: flex;  
    font-size: 15px;  
}  
  
footer .stats strong {  
    font-size: 18px;  
}  
  
footer .stats .likes {  
    margin-left: 10px;  
}  
  
footer .cta {  
    margin-left: auto;  
}  
  
footer .cta button {  
    border: 0;  
    background: transparent;  
}  
  
</style>  
  
<header>  
  


<div class="profile-name">
  <h3>Quincy Larson</h3>
  <h4>@ossia</h4>
</div>

<div class="follow-btn">
  <button>Follow</button>
</div>

</header>

<div id="inner">
  <p>I meet so many people who are in search of that one trick that will help them work smart. Even if you work smart, you still have to work hard.</p>
  <span class="date">1:32 PM - 12 Jan 2018</span>
  <hr>
</div>

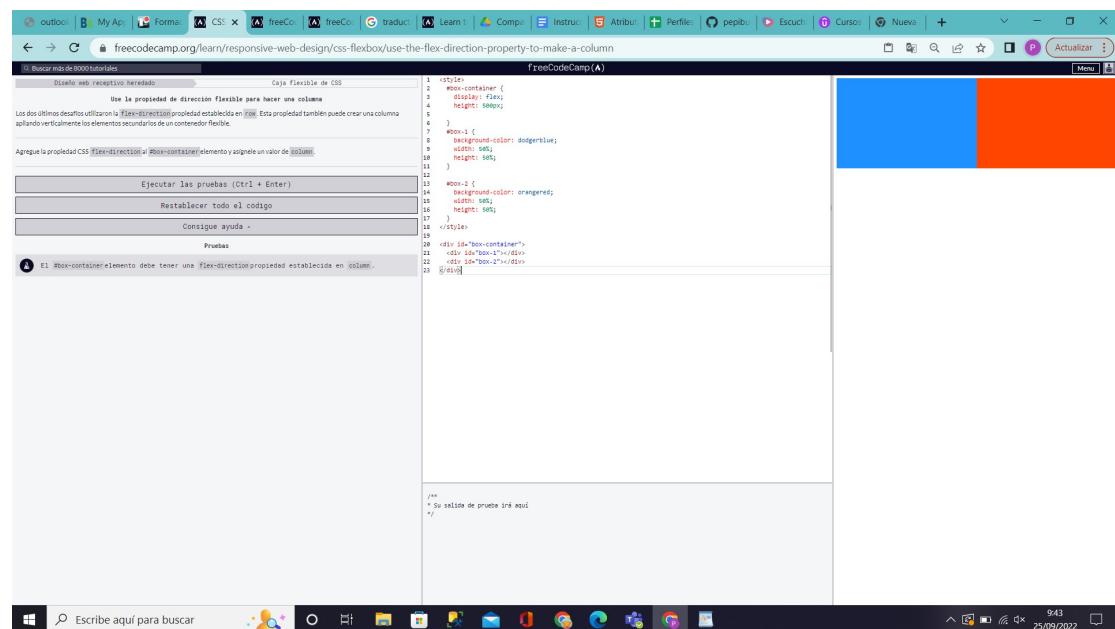
<footer>
  <div class="stats">
    <div class="Retweets">
      <strong>107</strong> Retweets
    </div>
    <div class="likes">
      <strong>431</strong> Likes
    </div>
  </div>
  <div class="cta">
    <button class="share-btn">Share</button>
    <button class="retweet-btn">Retweet</button>
    <button class="like-btn">Like</button>
  </div>
</div>
```

</footer>

5. Use la propiedad de dirección flexible para hacer una columna

Los dos últimos desafíos utilizaron la `flex-direction` propiedad establecida en `row`. Esta propiedad también puede crear una columna apilando verticalmente los elementos secundarios de un contenedor flexible.

Agregue la propiedad CSS `flex-direction` al `#box-container` elemento y asignele un valor de `column`.



The screenshot shows a browser window with the URL freeCodeCamp.org/learn/responsive-web-design/css-flexbox/use-the-flex-direction-property-to-make-a-column. The page title is "Caja flexible de CSS". The main content area contains the following CSS code:

```
1  #box-container {  
2     display: flex;  
3     height: 300px;  
4  }  
5  #box-1 {  
6     background-color: dodgerblue;  
7     width: 50%;  
8     height: 50%;  
9  }  
10 #box-2 {  
11     background-color: orangered;  
12     width: 50%;  
13     height: 50%;  
14  }  
15  /**/  
16  /* Su salida de prueba irá aquí */  
17  /**/
```

Below the code editor is a preview area showing two colored boxes: a blue one on top and an orange one below it, representing a column layout. The browser interface includes a navigation bar at the top and a taskbar at the bottom.

outlook My Au Forma CS treeC freeC freeC freeC tradu Learn Comp Instru Atribu Perfil pepi Escuelo Curso Nuevo + Actualizar

Buscar más de 8000 tutoriales

Diseño web: **receptivo heredado**

Caja flexible de CSS

Use la propiedad de dirección flexible para hacer una columna

Los dos últimos desplazos utilizan la `flex-direction` propiedad establecida en `column`. Esta propiedad también puede crear una columna apilando verticalmente los elementos secundarios de un contenedor flexible.

Agregue la propiedad CSS `flex-direction: #box-container;elemento` y asignele un valor de `column`.

Ejecutar las pruebas (Ctrl + Enter)

Restablecer todo el código

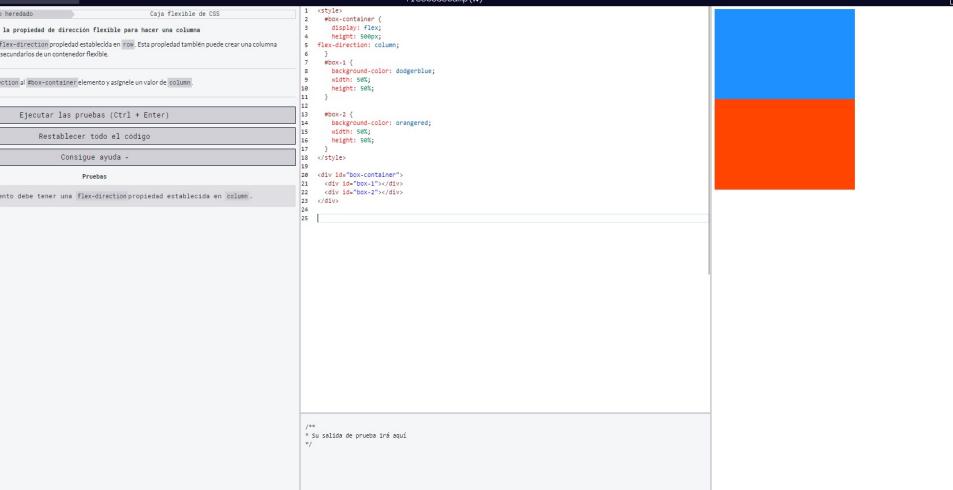
Consigna ayuda -

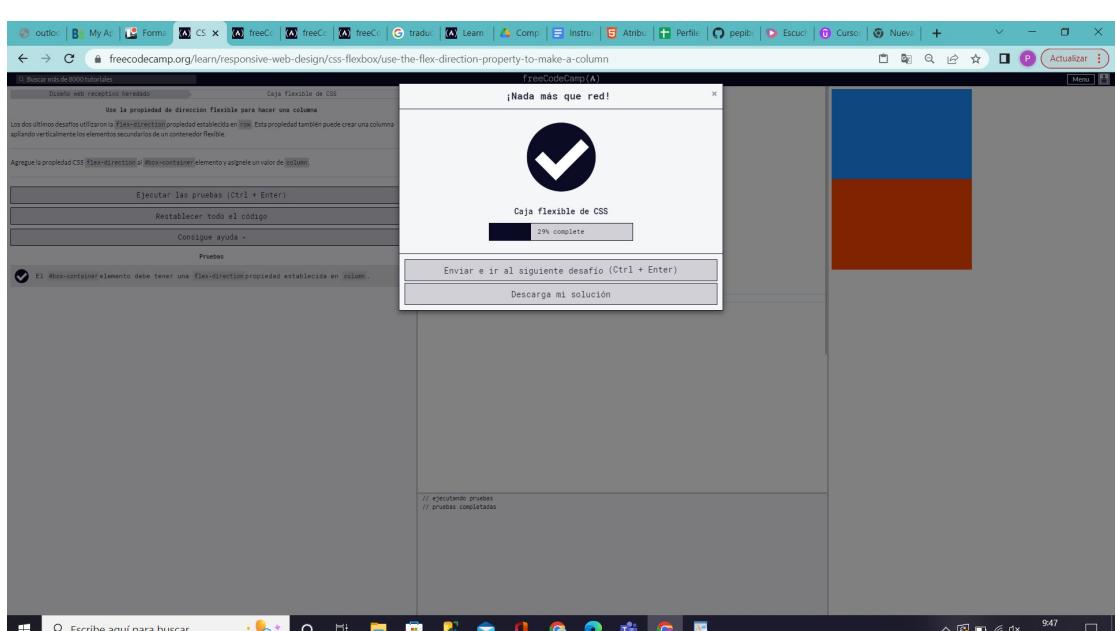
Pruebas

E1 `#box-container;elemento` debe tener una `flex-direction` propiedad establecida en `column`.

```
1 <style>
2   #box-container {
3     display: flex;
4     height: 80px;
5     flex-direction: column;
6   }
7   #box-1 {
8     background-color: dodgerblue;
9     width: 50px;
10    height: 50px;
11  }
12  #box-2 {
13    background-color: orange;
14    width: 50px;
15    height: 50px;
16  }
17  #box-3 {
18    background-color: orangered;
19    width: 50px;
20    height: 50px;
21  }
22  #box-4 {
23    background-color: red;
24    width: 50px;
25    height: 50px;
26  }
27</style>
28
29 <div id="box-container">
30   <div id="box-1"></div>
31   <div id="box-2"></div>
32   <div id="box-3"></div>
33   <div id="box-4"></div>
34 </div>
35
36 /**
37 * Su salón de prueba irá aquí
38 */
```

Escribe aquí para buscar





```
<style>

#box-container {
    display: flex;
    height: 500px;
}

flex-direction: column;

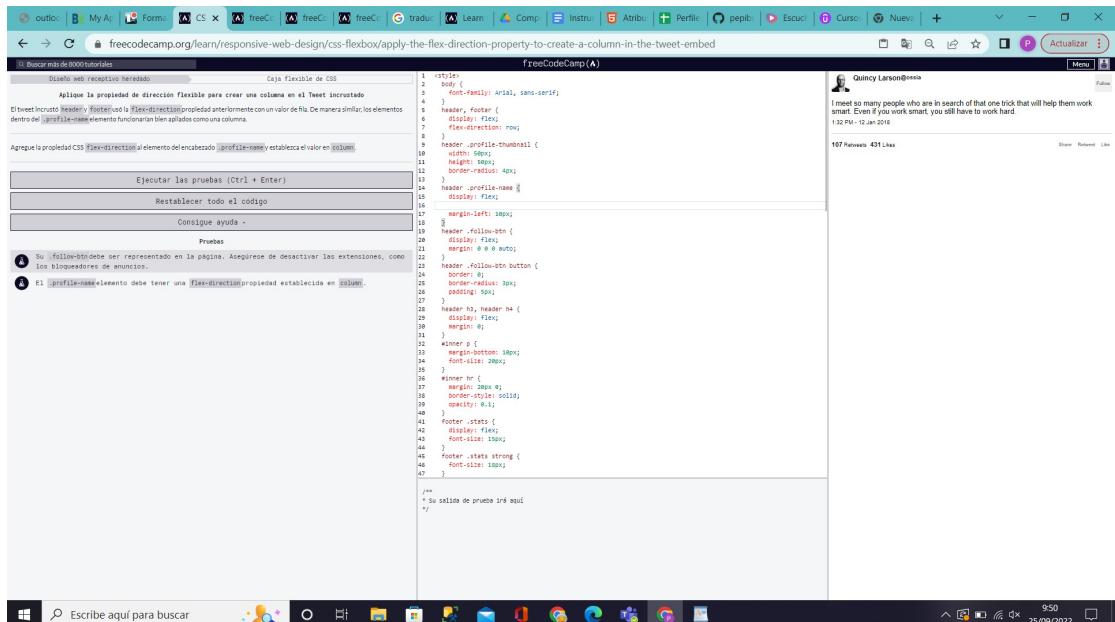
}
```

```
#box-1 {  
background-color: dodgerblue;  
width: 50%;  
height: 50%;  
}  
  
#box-2 {  
background-color: orangered;  
width: 50%;  
height: 50%;  
}  
  
</style>  
  
<div id="box-container">  
  <div id="box-1"></div>  
  <div id="box-2"></div>  
</div>
```

6. Aplique la propiedad de dirección flexible para crear una columna en el Tweet incrustado

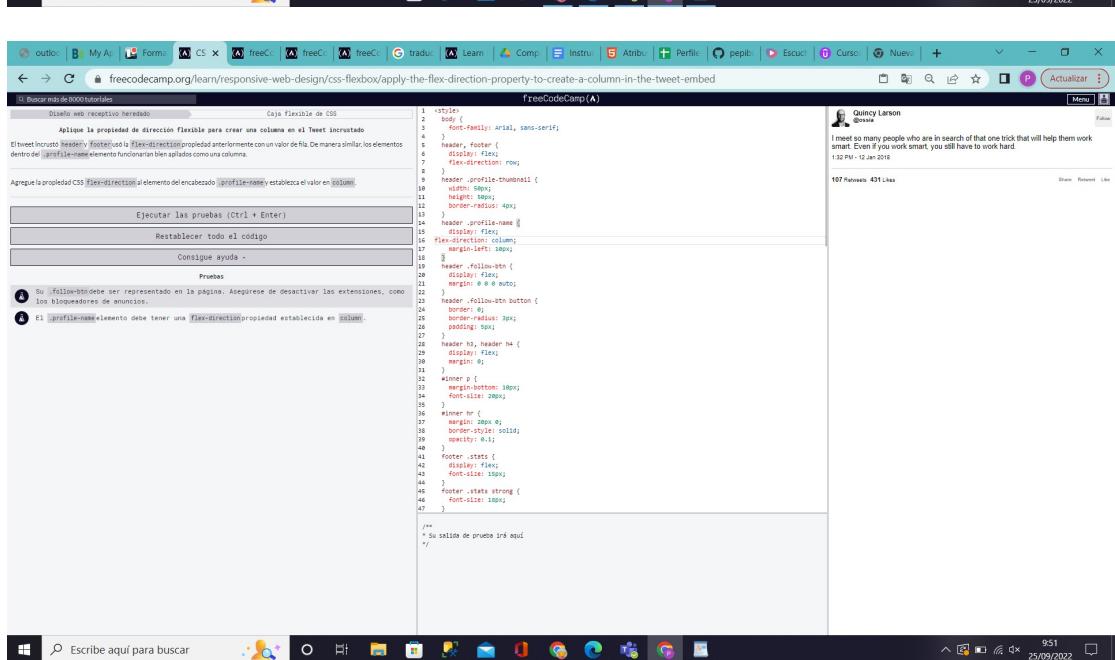
El tweet incrustó headery footerusó la flex-directionpropiedad anteriormente con un valor de fila. De manera similar, los elementos dentro del .profile-nameelemento funcionarían bien apilados como una columna.

Agregue la propiedad CSS flex-directional elemento del encabezado .profile-namey establezca el valor en column.



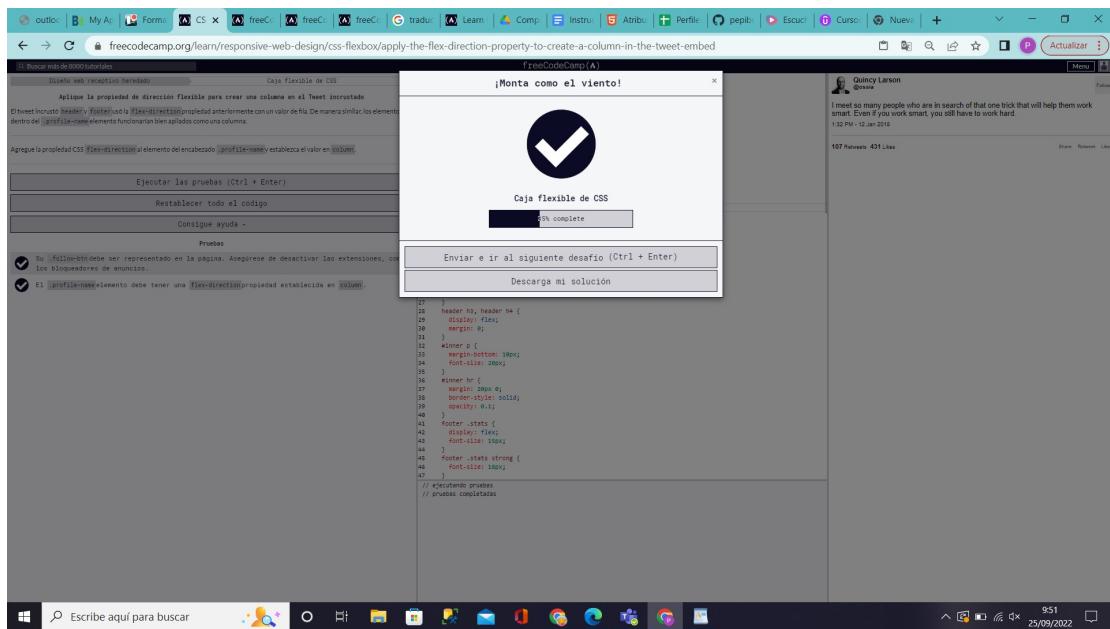
Caja flexible de CSS

```
1  style
2   body {
3     font-family: Arial, sans-serif;
4   }
5   header, footer {
6     display: flex;
7     flex-direction: row;
8   }
9   header .profile-thumball {
10    margin-left: 10px;
11    height: 40px;
12    border-radius: 4px;
13  }
14  header .profile-name {
15    display: flex;
16  }
17  header .profile-name {
18    margin-left: 10px;
19  }
20  header .follow-btn {
21    display: flex;
22    justify-content: space-between;
23    margin: 0;
24  }
25  header h2, header h4 {
26    display: flex;
27    justify-content: space-between;
28    margin: 0;
29  }
30  header p {
31    margin-bottom: 10px;
32  }
33  header hr {
34    margin-top: 20px;
35    border-style: solid;
36    border-width: 0.5px;
37  }
38  footer .stats {
39    display: flex;
40    justify-content: space-between;
41    font-size: 15px;
42  }
43  footer .stats strong {
44    font-size: 18px;
45  }
46
47
48  /* Su salida de prueba irá aquí
49 */
50
```



Caja flexible de CSS

```
1  style
2   body {
3     font-family: Arial, sans-serif;
4   }
5   header, footer {
6     display: flex;
7     flex-direction: row;
8   }
9   header .profile-thumball {
10    margin-left: 10px;
11    height: 40px;
12    border-radius: 4px;
13  }
14  header .profile-name {
15    display: flex;
16    flex-direction: column;
17    margin-left: 10px;
18  }
19  header .follow-btn {
20    display: flex;
21    justify-content: space-between;
22    margin: 0;
23  }
24  header h2, header h4 {
25    display: flex;
26    justify-content: space-between;
27    margin: 0;
28  }
29  header p {
30    margin-bottom: 10px;
31  }
32  header hr {
33    margin-top: 20px;
34    border-style: solid;
35    border-width: 0.5px;
36  }
37  footer .stats {
38    display: flex;
39    justify-content: space-between;
40    font-size: 15px;
41  }
42  footer .stats strong {
43    font-size: 18px;
44  }
45
46
47
48  /* Su salida de prueba irá aquí
49 */
50
```



```
<style>

body {
    font-family: Arial, sans-serif;
}

header, footer {
    display: flex;
    flex-direction: row;
}

header .profile-thumbnail {
    width: 50px;
    height: 50px;
    border-radius: 4px;
}

header .profile-name {
    display: flex;
}
```

```
flex-direction: column;
```

```
margin-left: 10px;
```

```
}
```

```
header .follow-btn {
```

```
display: flex;
```

```
margin: 0 0 0 auto;
```

```
}
```

```
header .follow-btn button {
```

```
border: 0;
```

```
border-radius: 3px;
```

```
padding: 5px;
```

```
}
```

```
header h3, header h4 {
```

```
display: flex;
```

```
margin: 0;
```

```
}
```

```
#inner p {
```

```
margin-bottom: 10px;
```

```
font-size: 20px;
```

```
}
```

```
#inner hr {
```

```
margin: 20px 0;
```

```
border-style: solid;
```

```
opacity: 0.1;
```

```
}
```

```
footer .stats {
```

```
display: flex;  
font-size: 15px;  
}  
  
footer .stats strong {  
font-size: 18px;  
}  
  
footer .stats .likes {  
margin-left: 10px;  
}  
  
footer .cta {  
margin-left: auto;  
}  
  
footer .cta button {  
border: 0;  
background: transparent;  
}  
  
</style>
```

7. Alinear elementos usando la propiedad de justificar contenido

A veces, los artículos flexibles dentro de un contenedor flexible no llenan todo el espacio del contenedor. Es común querer decirle a CSS cómo alinear y espaciar los elementos flexibles de cierta manera. Afortunadamente, la `justify-content` propiedad tiene varias opciones para hacer esto. Pero primero, hay una terminología importante que debe comprender antes de revisar esas opciones.

Para obtener más información sobre las propiedades de la caja flexible, lea aquí Recuerde que configurar un contenedor flexible como una fila coloca los elementos flexibles

uno al lado del otro de izquierda a derecha. Un contenedor flexible configurado como una columna coloca los elementos flexibles en una pila vertical de arriba a abajo. Para cada uno, la dirección en la que se organizan los elementos flexibles se denomina eje principal . Para una fila, esta es una línea horizontal que atraviesa cada elemento. Y para una columna, el eje principal es una línea vertical a través de los elementos.

Hay varias opciones sobre cómo espaciar los elementos flexibles a lo largo de la línea que es el eje principal. Uno de los más utilizados es `justify-content: center;`, que alinea todos los elementos flexibles al centro dentro del contenedor flexible. Otras opciones incluyen:

- **flex-start:** alinea los elementos al inicio del contenedor flexible. Para una fila, esto empuja los elementos a la izquierda del contenedor. Para una columna, esto empuja los elementos a la parte superior del contenedor. Esta es la alineación predeterminada si no se especifica `justify-content` .
- **flex-end:** alinea los elementos al final del contenedor flexible. Para una fila, esto empuja los artículos a la derecha del contenedor. Para una columna, esto empuja los artículos al fondo del contenedor.
- **space-between:** alinea los elementos en el centro del eje principal, con espacio adicional colocado entre los elementos. El primer y el último elemento se empujan hasta el borde del contenedor flexible. Por ejemplo, en una fila, el primer elemento está contra el lado izquierdo del contenedor, el último elemento está contra el lado derecho del contenedor, luego el espacio restante se distribuye uniformemente entre los demás elementos.
- **space-around:** similar `space-between` pero el primer y el último elemento no están bloqueados en los bordes del contenedor, el espacio se distribuye alrededor de todos los elementos con medio espacio en cada extremo del contenedor flexible.
- **space-evenly:** Distribuye el espacio uniformemente entre los elementos flexibles con un espacio completo en cada extremo del contenedor flexible.

Un ejemplo ayuda a mostrar esta propiedad en acción. Agregue la propiedad CSS `justify-content` a un elemento `#box` y asignele un valor de `center`.

Bonificación

Pruebe las otras opciones para la `justify-content` propiedad en el editor de código para ver sus diferencias. Pero tenga en cuenta que un valor de `center` es el único que superará este desafío.

[outlook](#) [B My A!](#) [Forma](#) [CS](#) [freeCodeCamp](#) [freeCodeCamp](#) [freeCodeCamp](#) [traduc](#) [Learn](#) [Comp](#) [Instru](#) [Atribu](#) [Perfil](#) [pepit](#) [Escuci](#) [i Curso](#) [Nuevo](#) + Actualizar

Buscador: Buscar más de 8000 tutoriales freeCodeCamp(A)

Diseño web: Recipientes heredados Caja flexible de CSS

Alinear elementos usando la propiedad de justificar contenido

A veces, los artículos flexibles dentro de un contenedor flexible no llenan todo el espacio del contenedor. Es común querer decirle a CSS cómo alinear y espaciar los elementos flexibles de cierta manera. Afortunadamente, la `justify-content` propiedad tiene varias opciones para hacer esto. Pero primero, hay una terminología importante que debe comprender antes de revisar esas opciones.

Para obtener más información sobre las propiedades de la caja flexible, [haz clic aquí](#)

Recuerda que configurar un contenedor flexible como una fila coloca los elementos flexibles uno al lado del otro de izquierda a derecha. Un contenedor flexible configurado como una columna coloca los elementos flexibles en una fila vertical de arriba abajo. Para cada uno, la dirección en la que se organizan los elementos flexibles denotará eje principal. Para una fila, éste es una línea horizontal que atravesará cada elemento. Y para una columna, éste es una línea vertical o trazo de los elementos.

Hay varias opciones sobre cómo espaciar los elementos flexibles a lo largo de la línea que es el eje principal. Uno de los más utilizados es `flex-start`, que alinea los elementos al inicio del contenedor flexible. Para una fila, esto empuja los elementos a la izquierda del contenedor. Si especificas `flex-end`, alinea los elementos al final del contenedor. Para una fila, esto empuja los artículos a la derecha del contenedor. Para una columna, esto empuja los elementos al final del contenedor. `space-around` alinea los elementos en el centro del eje principal, con espacio adicional colocado entre los elementos. El primer y el último elemento se encogen para dejar el borde del contenedor flexible. Por ejemplo, en una fila, el primer elemento está contra el lado izquierdo del contenedor y el último elemento está contra el lado derecho. Los demás elementos quedan espaciados uniformemente entre los demás elementos. `space-between` similar a `space-around`, excepto que el primer y el último elemento no están bloqueados en los bordes del contenedor; el espacio se distribuye alrededor de todos los elementos con medio espacio en cada extremo del contenedor flexible. `space-equal` Distribuye el espacio uniformemente entre los elementos flexibles con un espacio completo en cada extremo del contenedor flexible.

Un ejemplo ayuda a mostrar esta propiedad en acción. Agrega la propiedad CSS `justify-content: space-around;` al elemento `#box-container` y asigne un valor de `center`.

Bonificación

Prueba las otras opciones para la `justify-content` propiedad en el editor de código para ver sus diferencias. Pero tenga en cuenta que un valor de `center` es el único que superaría este desafío.

Ejecutar las pruebas (Ctrl + Enter) Restablecer todo el código Consigue ayuda + Pruebas

El `#box-container` elemento debe tener una `justify-content` propiedad establecida en un valor de `center`.

```

1 <style>
2   #box-container {
3     background: gray;
4     display: flex;
5     justify-content: space-around;
6     height: 100px;
7   }
8   #box-1 {
9     background-color: dodgerblue;
10    width: 25px;
11    height: 100px;
12  }
13  #box-2 {
14    background-color: orange-red;
15    width: 25px;
16    height: 100px;
17  }
18 </style>
19
20 <div id="box-container">
21   <div id="box-1"></div>
22   <div id="box-2"></div>
23 </div>

```

9:53 25/09/2022

[outlook](#) [B My A!](#) [Forma](#) [CS](#) [freeCodeCamp](#) [freeCodeCamp](#) [freeCodeCamp](#) [freeCodeCamp](#) [traduc](#) [Learn](#) [Comp](#) [Instru](#) [Atribu](#) [Perfil](#) [pepit](#) [Escuci](#) [i Curso](#) [Nuevo](#) + Actualizar

Buscador: Buscar más de 8000 tutoriales freeCodeCamp(A)

Diseño web: Recipientes heredados Caja flexible de CSS

Alinear elementos usando la propiedad de justificar contenido

A veces, los artículos flexibles dentro de un contenedor flexible no llenan todo el espacio del contenedor. Es común querer decirle a CSS cómo alinear y espaciar los elementos flexibles de cierta manera. Afortunadamente, la `justify-content` propiedad tiene varias opciones para hacer esto. Pero primero, hay una terminología importante que debe comprender antes de revisar esas opciones.

Para obtener más información sobre las propiedades de la caja flexible, [haz clic aquí](#)

Recuerda que configurar un contenedor flexible como una fila coloca los elementos flexibles uno al lado del otro de izquierda a derecha. Un contenedor flexible configurado como una columna coloca los elementos flexibles en una fila vertical de arriba abajo. Para cada uno, la dirección en la que se organizan los elementos flexibles denotará eje principal. Para una fila, éste es una línea horizontal que atravesará cada elemento. Y para una columna, éste es una línea vertical o trazo de los elementos.

Hay varias opciones sobre cómo espaciar los elementos flexibles a lo largo de la línea que es el eje principal. Uno de los más utilizados es `justify-content: center`, que alinea todos los elementos flexibles en el centro del contenedor flexible. Para una fila, esto empuja los elementos a la derecha del contenedor. Si especificas `justify-content: space-around`, alinea los elementos a la parte superior del contenedor. Esto es la alineación predefinida si no se especifica. Para una columna, esto empuja los artículos al fondo del contenedor flexible. Para una fila, esto empuja los artículos a la derecha del contenedor. `justify-content: space-between` alinea los elementos en el centro del eje principal, con espacio adicional colocado entre los elementos. El primer y el último elemento se encogen para dejar el borde del contenedor flexible. Por ejemplo, en una fila, el primer elemento está contra el lado izquierdo del contenedor y el último elemento está contra el lado derecho. Los demás elementos quedan espaciados uniformemente entre los demás elementos. `justify-content: space-around` similar a `space-between`, excepto que el primer y el último elemento no están bloqueados en los bordes del contenedor; el espacio se distribuye alrededor de todos los elementos con medio espacio en cada extremo del contenedor flexible. `justify-content: space-equal` Distribuye el espacio uniformemente entre los elementos flexibles con un espacio completo en cada extremo del contenedor flexible.

Un ejemplo ayuda a mostrar esta propiedad en acción. Agrega la propiedad CSS `justify-content: space-around;` al elemento `#box-container` y asigne un valor de `center`.

Bonificación

Prueba las otras opciones para la `justify-content` propiedad en el editor de código para ver sus diferencias. Pero tenga en cuenta que un valor de `center` es el único que superaría este desafío.

Ejecutar las pruebas (Ctrl + Enter) Restablecer todo el código Consigue ayuda + Pruebas

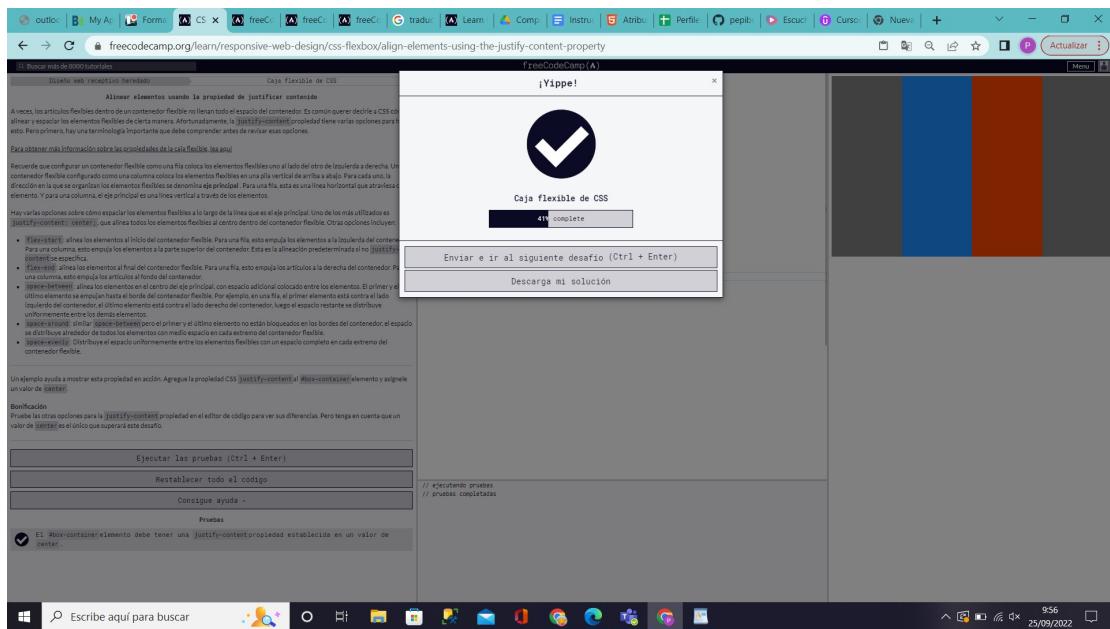
El `#box-container` elemento debe tener una `justify-content` propiedad establecida en un valor de `center`.

```

1 <style>
2   #box-container {
3     background: gray;
4     display: flex;
5     justify-content: space-around;
6     height: 100px;
7   }
8   #box-1 {
9     background-color: dodgerblue;
10    width: 25px;
11    height: 100px;
12  }
13  #box-2 {
14    background-color: orange-red;
15    width: 25px;
16    height: 100px;
17  }
18 </style>
19
20 <div id="box-container">
21   <div id="box-1"></div>
22   <div id="box-2"></div>
23 </div>

```

9:56 25/09/2022



```

<style>

#box-container {
    background: gray;
    display: flex;
    height: 500px;
}

justify-content: center;

}

#box-1 {
    background-color: dodgerblue;
    width: 25%;
    height: 100%;
}

}

#box-2 {
    background-color: orangered;
    width: 25%;
    height: 100%;
}

}

```

```
</style>

<div id="box-container">

    <div id="box-1"></div>

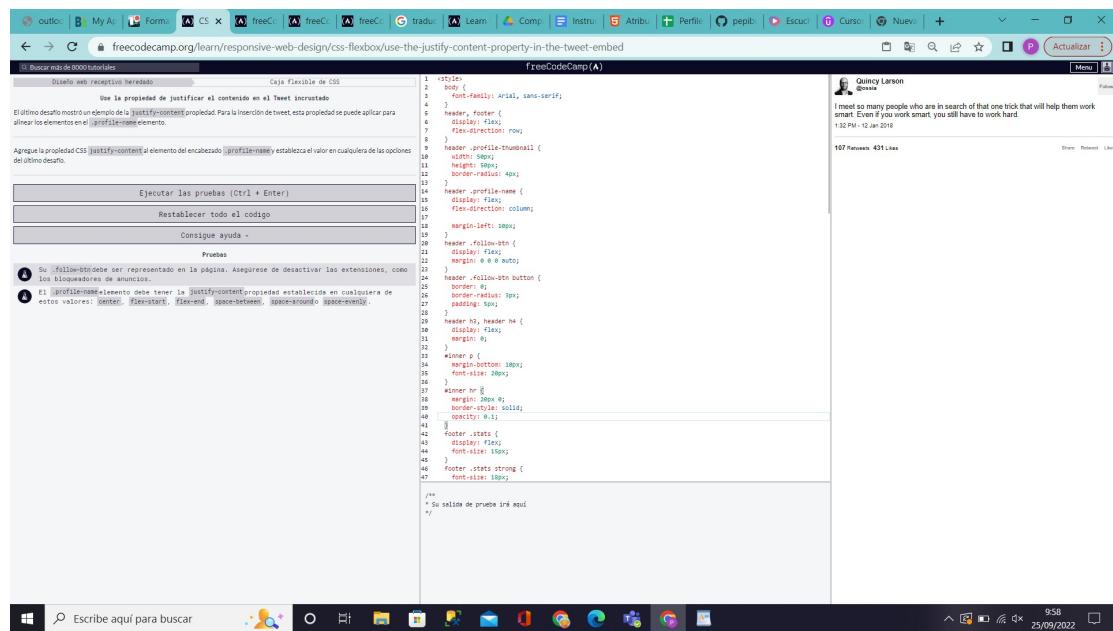
    <div id="box-2"></div>

</div>
```

8. Use la propiedad de justificar el contenido en el Tweet incrustado

El último desafío mostró un ejemplo de la `justify-content` propiedad. Para la inserción de tweet, esta propiedad se puede aplicar para alinear los elementos en el `.profile-name` elemento.

Agregue la propiedad CSS **justify-content** al elemento del encabezado `.profile-name` y establezca el valor en cualquiera de las opciones del último desafío.



```

body {
    font-family: Arial, sans-serif;
}

header, footer {
    display: flex;
    flex-direction: row;
}

header .profile-thumball {
    margin-right: 10px;
    height: 40px;
    border-radius: 4px;
}

header .profile-name {
    display: flex;
    justify-content: space-between;
    margin-left: 10px;
}

header .follow-btn {
    display: flex;
    align-items: center;
}

header hr {
    margin: 20px 0;
    border: 0;
    border-top: 1px solid #ccc;
    opacity: 0.1;
}

footer .stats {
    display: flex;
    justify-content: space-around;
    font-size: 10px;
}

.winner p {
    margin-bottom: 10px;
    font-size: 12px;
}

.winner hr {
    margin: 20px 0;
    border: 0;
    border-top: 1px solid #ccc;
    opacity: 0.1;
}

.winner .stats {
    display: flex;
    justify-content: space-around;
    font-size: 10px;
}

.winner .stats strong {
    font-size: 12px;
}

/*
Salida de prueba irá aquí
*/

```

```

body {
    font-family: Arial, sans-serif;
}

header, footer {
    display: flex;
    flex-direction: row;
}

header .profile-thumball {
    margin-right: 10px;
    height: 40px;
    border-radius: 4px;
}

header .profile-name {
    display: flex;
    justify-content: space-between;
    margin-left: 10px;
}

header .follow-btn {
    display: flex;
    align-items: center;
}

header hr {
    margin: 20px 0;
    border: 0;
    border-top: 1px solid #ccc;
    opacity: 0.1;
}

.winner p {
    margin-bottom: 10px;
    font-size: 12px;
}

.winner hr {
    margin: 20px 0;
    border: 0;
    border-top: 1px solid #ccc;
    opacity: 0.1;
}

.winner .stats {
    display: flex;
    justify-content: space-around;
    font-size: 10px;
}

.winner .stats strong {
    font-size: 12px;
}

/*
Pruebas completadas
*/

```

```

<style>

body {
    font-family: Arial, sans-serif;
}

header, footer {
    display: flex;
}

```

```
flex-direction: row;  
}  
  
header .profile-thumbnail {  
width: 50px;  
height: 50px;  
border-radius: 4px;  
}  
  
header .profile-name {  
display: flex;  
flex-direction: column;  
justify-content: space-between;  
margin-left: 10px;  
}  
  
header .follow-btn {  
display: flex;  
margin: 0 0 0 auto;  
}  
  
header .follow-btn button {  
border: 0;  
border-radius: 3px;  
padding: 5px;  
}  
  
header h3, header h4 {  
display: flex;  
margin: 0;  
}
```

```
#inner p {  
    margin-bottom: 10px;  
    font-size: 20px;  
}  
  
#inner hr {  
    margin: 20px 0;  
    border-style: solid;  
    opacity: 0.1;  
}  
  
footer .stats {  
    display: flex;  
    font-size: 15px;  
}  
  
footer .stats strong {  
    font-size: 18px;  
}  
  
footer .stats .likes {  
    margin-left: 10px;  
}  
  
footer .cta {  
    margin-left: auto;  
}  
  
footer .cta button {  
    border: 0;  
    background: transparent;  
}
```

</style>

9. Alinear elementos usando la propiedad align-items

La **align-items** propiedad es similar a justify-content. Recuerde que la justify-content propiedad alineó elementos flexibles a lo largo del eje principal. Para las filas, el eje principal es una línea horizontal y para las columnas es una línea vertical.

Los contenedores flexibles también tienen un eje transversal que es el opuesto del eje principal. Para las filas, el eje transversal es vertical y para las columnas, el eje transversal es horizontal.

CSS ofrece la align-items propiedad de alinear elementos flexibles a lo largo del eje transversal. Para una fila, le dice a CSS cómo empujar los elementos de toda la fila hacia arriba o hacia abajo dentro del contenedor. Y para una columna, cómo empujar todos los elementos hacia la izquierda o hacia la derecha dentro del contenedor.

Los diferentes valores disponibles para align-items incluyen:

flex-start: alinea los elementos al inicio del contenedor flexible. Para las filas, esto alinea los elementos en la parte superior del contenedor. Para las columnas, esto alinea los elementos a la izquierda del contenedor.

flex-end: alinea los elementos al final del contenedor flexible. Para las filas, esto alinea los elementos con la parte inferior del contenedor. Para las columnas, esto alinea los elementos a la derecha del contenedor.

center: alinea los elementos al centro. Para las filas, esto alinea los elementos verticalmente (el mismo espacio por encima y por debajo de los elementos). Para las columnas, esto las alinea horizontalmente (igual espacio a la izquierda y a la derecha de los elementos).

stretch: estire los artículos para llenar el contenedor flexible. Por ejemplo, los elementos de las filas se estiran para llenar el contenedor flexible de arriba a abajo. Este es el valor predeterminado si no align-items se especifica ningún valor.

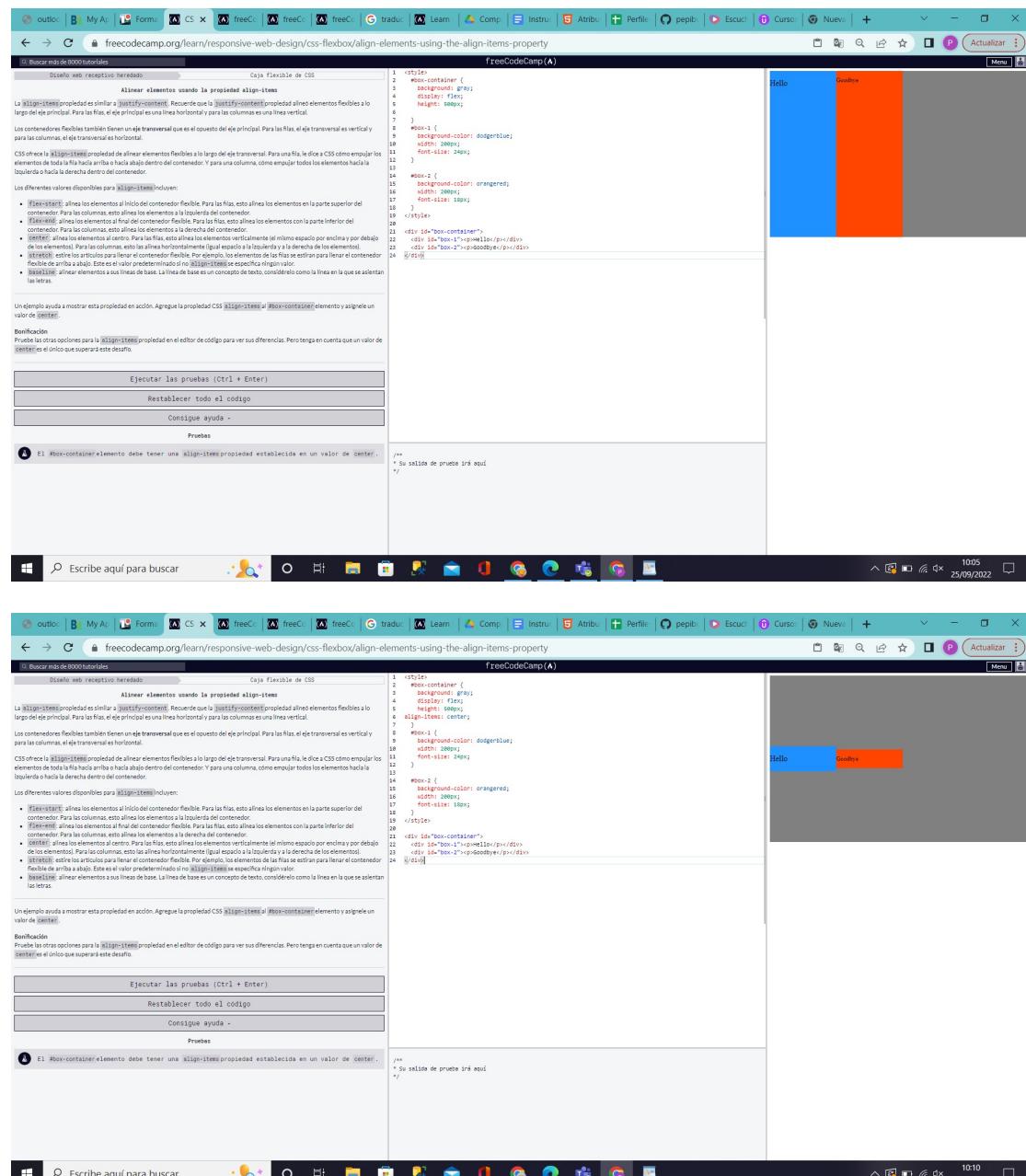
baseline: alinear elementos a sus líneas de base. La línea de base es un concepto de texto, considérelo como la línea en la que se asientan las letras.

Un ejemplo ayuda a mostrar esta propiedad en acción. Agregue la propiedad CSS align-items al #box-containerelemento y asígnele un valor de center.

Bonificación

Pruebe las otras opciones para la align-items propiedad en el editor de código para ver sus diferencias. Pero tenga en cuenta que un valor de center es el único que superará este

desafío.



The screenshot shows a browser window with the URL freecodecamp.org/learn/responsive-web-design/css-flexbox/align-elements-using-the-align-items-property. The page content is about the `align-items` property in CSS flexbox. It includes a code editor with the following CSS:

```
1 #box-container {  
2   background: gray;  
3   display: flex;  
4   height: 300px;  
5 }  
6  
7 #box-1 {  
8   background-color: dodgerblue;  
9   width: 100px;  
10 }  
11  
12 #box-2 {  
13   background-color: orange-red;  
14   width: 200px;  
15 }  
16  
17 /* Su salida de prueba irá aquí  
18 */
```

The browser's status bar at the bottom right shows the date as 25/09/2022 and the time as 10:05.

The screenshot shows a challenge titled "Asalta ese castillo!" on the freeCodeCamp website. The challenge involves creating a flexible CSS box with a central checkmark icon and two buttons at the bottom. The code uses align-items: flex-start;.

```

    .caja {
        width: 100px;
        height: 100px;
        background-color: dodgerblue;
        display: flex;
        align-items: flex-start;
    }
    .caja .checkmark {
        width: 50px;
        height: 50px;
        background-color: orange;
        border-radius: 50%;
        display: flex;
        align-items: center;
        justify-content: center;
        font-size: 1em;
        color: white;
    }
    .caja .button {
        width: 40px;
        height: 20px;
        background-color: red;
        border: none;
        color: white;
        font-size: 0.8em;
        margin-right: 10px;
    }
    .caja .button + .button {
        margin-left: 10px;
    }

```

align-items: flex-end;

The screenshot shows a challenge titled "Caja flexible de CSS" on the freeCodeCamp website. The challenge involves creating a flexible CSS box with a central checkmark icon and two buttons at the bottom. The code uses align-items: flex-end;.

```

    .caja {
        width: 100px;
        height: 100px;
        background-color: dodgerblue;
        display: flex;
        align-items: flex-end;
    }
    .caja .checkmark {
        width: 50px;
        height: 50px;
        background-color: orange;
        border-radius: 50%;
        display: flex;
        align-items: center;
        justify-content: center;
        font-size: 1em;
        color: white;
    }
    .caja .button {
        width: 40px;
        height: 20px;
        background-color: red;
        border: none;
        color: white;
        font-size: 0.8em;
        margin-right: 10px;
    }
    .caja .button + .button {
        margin-left: 10px;
    }

```

align-items: baseline;

```

<style>

#box-container {

background: gray;

display: flex;

height: 500px;

align-items: center;

}

#box-1 {

background-color: dodgerblue;

width: 200px;

font-size: 24px;

}

#box-2 {

background-color: orangered;

width: 200px;

font-size: 18px;

}
    
```

```
</style>

<div id="box-container">

    <div id="box-1"><p>Hello</p></div>

    <div id="box-2"><p>Goodbye</p></div>

</div>
```

10. Use la propiedad align-items en el Tweet incrustado

El último desafío presentó la align-itemspropiedad y dio un ejemplo. Esta propiedad se puede aplicar a algunos elementos de incrustación de tweets para alinear los elementos flexibles dentro de ellos.

Agregue la propiedad CSS align-items al elemento del encabezado .follow-btn. Establezca el valor en center.

Buscar más de 8000 tutoriales

Diseño web heredado Caja flexible de CSS

Use la propiedad align-items en el Tweet insrustado

El último destino presenta la `align-items` propiedad y dice un tiempo. Esta propiedad se puede aplicar a algunos elementos de herencia de tweets para alinear los elementos flexibles dentro de ellos.

Agregue la propiedad CSS `align-items` al elemento del encabezado `<follow-btn>`. Establezca el valor en `center`.

Ejecutar las pruebas (Ctrl + Enter)

Restablecer todo el código

Consigue ayuda -

Pruebas

Atención: Su `follow-btn` debe ser representado en la página. Asegúrese de desactivar las extensiones, como los bloques de anuncios.

Atención: El `follow-btn` elemento debe tener la `align-items` propiedad establecida en un valor de `center`.

```
1 <style>
2   </style>
3   <!--
4     -->
5   header, footer {
6     display: flex;
7     flex-direction: row;
8   }
9   header .profile-thumbnaill {
10    width: 50px;
11    height: 50px;
12    border-radius: 4px;
13  }
14  header .profile-name {
15    display: flex;
16    flex-direction: column;
17    justify-content: center;
18    margin-left: 10px;
19  }
20  header .follow-btn {
21    display: flex;
22    margin: 0 auto 0 auto;
23  }
24  header .follow-btn button {
25    border: 1px solid #ccc;
26    padding: 5px;
27  }
28  header h2, header h4 {
29    display: flex;
30    margin: 0;
31  }
32  header h2 p {
33    margin: 0 10px 0 0;
34  }
35  header h2 button {
36    font-size: 20px;
37  }
38  header hr {
39    margin: 20px 0;
40  }
41  header hr strong {
42    color: #ccc;
43    opacity: 0.4;
44  }
45  footer .stats {
46    display: flex;
47    justify-content: space-between;
48  }
49  footer .stats strong {
50    /* Su salida de prueba irá aquí
51   */
52 }
```

```

<style>
body {
  font-family: Arial, sans-serif;
}
header, footer {
  display: flex;
  flex-direction: row;
}
header .profile-thumball {
  width: 40px;
  height: 40px;
  border-radius: 4px;
}
header .profile-name {
  display: flex;
  flex-direction: column;
  justify-content: center;
  margin-left: 10px;
}
header .follow-btn {
  display: flex;
  align-items: center;
  margin: 0 auto;
}
header .follow-btn button {
  border: 1px solid #ccc;
  border-radius: 1px;
  padding: 5px;
}
header h3, header h4 {
  margin: 0;
}
inner p {
  margin-bottom: 10px;
  font-size: 16px;
}
inner hr {
  border: 0.5px solid #ccc;
  opacity: 0.1;
}
footer .stats {
  display: flex;
  font-size: 14px;
}
footer .stats strong {
  // Executando pruebas
  // Pruebas completadas
}

```

Caja flexible de CSS

¡Un bucle para gobernarlos a todos!

50% completa

Enviar e ir al siguiente desafío (Ctrl + Enter)

Descarga mi solución

```

<style>
body {
  font-family: Arial, sans-serif;
}

header, footer {
  display: flex;
}

```

```
flex-direction: row;  
}  
  
header .profile-thumbnail {  
width: 50px;  
height: 50px;  
border-radius: 4px;  
}  
  
header .profile-name {  
display: flex;  
flex-direction: column;  
justify-content: center;  
margin-left: 10px;  
}  
  
header .follow-btn {  
display: flex;  
align-items: center;  
margin: 0 0 0 auto;  
}  
  
header .follow-btn button {  
border: 0;  
border-radius: 3px;  
padding: 5px;  
}  
  
header h3, header h4 {  
display: flex;  
margin: 0;  
}  
  
#inner p {
```

```
margin-bottom: 10px;  
font-size: 20px;  
}  
  
#inner hr {  
margin: 20px 0;  
border-style: solid;  
opacity: 0.1;  
}  
  
footer .stats {  
display: flex;  
font-size: 15px;  
}  
  
footer .stats strong {  
font-size: 18px;  
}  
  
footer .stats .likes {  
margin-left: 10px;  
}  
  
footer .cta {  
margin-left: auto;  
}  
  
footer .cta button {  
border: 0;  
background: transparent;  
}  
  
</style>
```

11. Use la propiedad flex-wrap para ajustar una fila o columna

CSS flexbox tiene una función para dividir un contenedor flexible en varias filas (o columnas). De forma predeterminada, un contenedor flexible encalará todos los elementos flexibles juntos. Por ejemplo, una fila estará en una sola línea.

Sin embargo, usar la **flex-wrap** propiedad le dice a CSS que ajuste los elementos. Esto significa que los elementos adicionales se mueven a una nueva fila o columna. El punto de ruptura donde ocurre el envoltorio depende del tamaño de los artículos y del tamaño del contenedor.

CSS también tiene opciones para la dirección de la envoltura:

- **nowrap**: esta es la configuración predeterminada y no envuelve elementos.
- **wrap**: envuelve elementos en varias líneas de arriba a abajo si están en filas y de izquierda a derecha si están en columnas.
- **wrap-reverse**: envuelve elementos en varias líneas de abajo hacia arriba si están en filas y de derecha a izquierda si están en columnas.

El diseño actual tiene demasiados cuadros para una fila. Agregue la propiedad CSS **flex-wrap** al elemento `#box-container` y asignele un valor de `wrap`.

```
1 #box-container {  
2   width: 100%;  
3   background: #F0F;  
4   display: flex;  
5   height: 100%;  
6 }  
7  
8 .box {  
9   background-color: dodgerblue;  
10  width: 25%;  
11  height: 50%;  
12 }  
13  
14 .box::before {  
15   background-color: orangered;  
16  width: 25%;  
17  height: 50%;  
18 }  
19  
20 .box::after {  
21   background-color: violet;  
22  width: 25%;  
23  height: 50%;  
24 }  
25  
26 .box::first-child {  
27   background-color: yellow;  
28  width: 25%;  
29  height: 50%;  
30 }  
31  
32 .box::last-child {  
33   background-color: green;  
34  width: 25%;  
35  height: 50%;  
36 }  
37  
38 .box::first-child::before {  
39   background-color: black;  
40  width: 25%;  
41  height: 50%;  
42 }  
43  
44 .box::first-child::after {  
45   background-color: black;  
46  width: 25%;  
47  height: 50%;  
48 }  
49  
50 /*  
51  * Salida de prueba irá aquí  
52 */
```

Caja flexible de CSS

```

1  /*style*/
2  .box-container {
3      background-color: gray;
4      display: flex;
5      flex-wrap: wrap;
6  }
7  .box {
8      width: 25%;
9      height: 50px;
10 }
11 .box-1 {
12     background-color: dodgerblue;
13     width: 25px;
14     height: 50px;
15 }
16 .box-2 {
17     background-color: orange-red;
18     width: 25px;
19     height: 50px;
20 }
21 .box-3 {
22     background-color: violet;
23     width: 25px;
24     height: 50px;
25 }
26 .box-4 {
27     background-color: yellow;
28     width: 25px;
29     height: 50px;
30 }
31 .box-5 {
32     background-color: green;
33     width: 25px;
34     height: 50px;
35 }
36 .box-6 {
37     background-color: black;
38     width: 25px;
39     height: 50px;
40 }
41 <div id="box-container">
42     <div id="box-1"></div>
43     <div id="box-2"></div>
44     <div id="box-3"></div>
45     <div id="box-4"></div>
46     <div id="box-5"></div>
47     <div id="box-6"></div>
48 // Salida de prueba irá aquí
49

```

Ejecutar las pruebas (Ctrl + Enter)

Restablecer todo el código

Consegue ayuda -

Pruebas

El #box-container elemento debe tener la flex-wrap propiedad establecida en un valor de wrap.

Caja flexible de CSS

¡Fuerza del gancho!

Caja flexible de CSS

60% completa

Enviar e ir al siguiente desafío (Ctrl + Enter)

Descarga mi solución

Ejecutar las pruebas (Ctrl + Enter)

Restablecer todo el código

Consegue ayuda -

Pruebas

El #box-container elemento debe tener la flex-wrap propiedad establecida en un valor de wrap.

```

<style>

#box-container {

background: gray;

display: flex;

height: 100%;

flex-wrap: wrap;

```

```
}

#box-1 {
    background-color: dodgerblue;
    width: 25%;
    height: 50%;

}

#box-2 {
    background-color: orangered;
    width: 25%;
    height: 50%;

}

#box-3 {
    background-color: violet;
    width: 25%;
    height: 50%;

}

#box-4 {
    background-color: yellow;
    width: 25%;
    height: 50%;

}

#box-5 {
    background-color: green;
    width: 25%;
    height: 50%;

}

#box-6 {
    background-color: black;
```

```

width: 25%;

height: 50%;

}

</style>

```

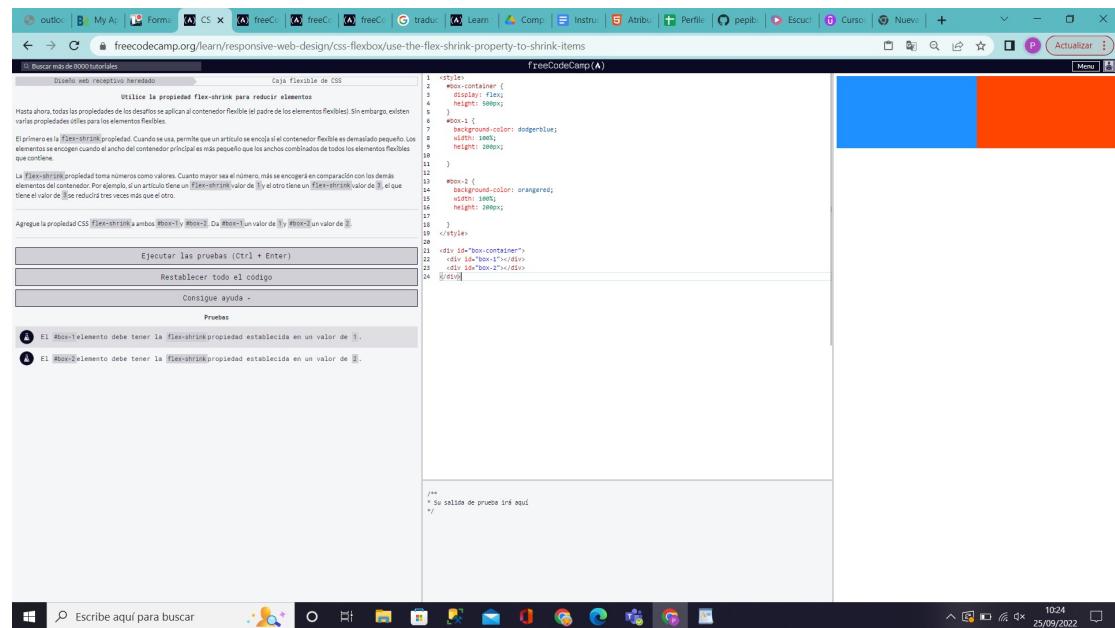
12. Utilice la propiedad flex-shrink para reducir elementos

Hasta ahora, todas las propiedades de los desafíos se aplican al contenedor flexible (el padre de los elementos flexibles). Sin embargo, existen varias propiedades útiles para los elementos flexibles.

El primero es la **flex-shrink** propiedad. Cuando se usa, permite que un artículo se encoja si el contenedor flexible es demasiado pequeño. Los elementos se encogen cuando el ancho del contenedor principal es más pequeño que los anchos combinados de todos los elementos flexibles que contiene.

La **flex-shrink** propiedad toma números como valores. Cuanto mayor sea el número, más se encogerá en comparación con los demás elementos del contenedor. Por ejemplo, si un artículo tiene un **flex-shrink** valor de 1 y el otro tiene un **flex-shrink** valor de 3, el que tiene el valor de 3 se reducirá tres veces más que el otro.

Agregue la propiedad CSS **flex-shrink** a ambos **#box-1** y **#box-2**. Da **#box-1** un valor de **1** y **#box-2** un valor de **2**.



```

1  /*#box-container {
2  width: 100px;
3  display: flex;
4  height: 500px;
5}
6
7  #box-1 {
8    background-color: dodgerblue;
9    width: 100px;
10   height: 200px;
11   flex-shrink: 1;
12}
13
14  #box-2 {
15    background-color: orange;
16    width: 100px;
17   height: 200px;
18   flex-shrink: 2;
19}
20
21 </style>
22
23 <div id="box-container">
24   <div id="box-1"></div>
25   <div id="box-2"></div>
26 </div>

```

Pruebas

- El #box-1 elemento debe tener la `flex-shrink` propiedad establecida en un valor de `1`.
- El #box-2 elemento debe tener la `flex-shrink` propiedad establecida en un valor de `2`.

Kool Aid Man dice ¡oh, sí!

Caja flexible de CSS

7/16 complete

Enviar e ir al siguiente desafío (Ctrl + Enter)

Descarga mi solución

Pruebas

- El #box-1 elemento debe tener la `flex-shrink` propiedad establecida en un valor de `1`.
- El #box-2 elemento debe tener la `flex-shrink` propiedad establecida en un valor de `2`.

*// Ejecutando pruebas
// Pruebas completadas*

```

<style>

#box-container {
  display: flex;
  height: 500px;
}

#box-1 {
  background-color: dodgerblue;
}

```

```

width: 100%;

height: 200px;

flex-shrink: 1;

}

#box-2 {

background-color: orangered;

width: 100%;

height: 200px;

flex-shrink: 2;

}

</style>

<div id="box-container">

<div id="box-1"></div>

<div id="box-2"></div>

</div>

```

13. Use la propiedad flex-grow para expandir elementos

Lo contrario de flex-shrink es la **flex-grow** propiedad. Recuerde que flex-shrink controla el tamaño de los artículos cuando el contenedor se encoge. La flex-grow propiedad **controla el tamaño de los elementos cuando se expande el contenedor principal.**

Usando un ejemplo similar del último desafío, si un artículo tiene un flex-growvalor de 1 y el otro tiene un flex-growvalor de 3, el que tiene el valor de 3crecerá tres veces más que el otro.

Agregue la propiedad CSS flex-grow a ambos #box-1 y #box-2. Da #box-1 un valor de 1 y #box-2 un valor de 2.

outlook | B My A! Formularios CS x freeCodeCamp freeCodeCamp traducir Learn Comp Instrucción Perfil pepita Escuchar Curso Nuevo +

Actualizar

Diseño web heredado Caja flexible de CSS

Uso la propiedad flex-grow para expandir elementos

Lo contrario de flex-shrink es la flex-grow propiedad. Recuerda que flex-grow controla el tamaño de los artículos cuando el contenedor se expande. La flex-grow propiedad controla el tamaño de los elementos cuando se expande el contenedor principal.

Usando un ejemplo similar del último diseño, si un artículo tiene un flex-grow valor de 3, el otro tiene un flex-grow valor de 2, el que tiene el valor de 3 crecerá tres veces más que el otro.

Agregue la propiedad CSS flex-grow a ambos #box1 y #box2. Da #box1 un valor de 3 y #box2 un valor de 2.

Ejecutar las pruebas (Ctrl + Enter)

Restablecer todo el código

Consigue ayuda -

Pruebas

```

1  /*style*/
2  #box-container {
3      display: flex;
4      height: 200px;
5  }
6
7  #box-1 {
8      background-color: dodgerblue;
9      height: 200px;
10 }
11
12 #box-2 {
13     background-color: orange;
14     height: 200px;
15 }
16
17 
```

/* Su salida de prueba irá aquí

Escribe aquí para buscar

1028 25/09/2022

outlook | B My A! Formularios CS x freeCodeCamp freeCodeCamp traducir Learn Comp Instrucción Perfil pepita Escuchar Curso Nuevo +

Actualizar

Diseño web heredado Caja flexible de CSS

Uso la propiedad flex-grow para expandir elementos

Lo contrario de flex-shrink es la flex-grow propiedad. Recuerda que flex-grow controla el tamaño de los artículos cuando el contenedor se expande. La flex-grow propiedad controla el tamaño de los elementos cuando se expande el contenedor principal.

Usando un ejemplo similar del último diseño, si un artículo tiene un flex-grow valor de 3, el otro tiene un flex-grow valor de 2, el que tiene el valor de 3 crecerá tres veces más que el otro.

Agregue la propiedad CSS flex-grow a ambos #box1 y #box2. Da #box1 un valor de 3 y #box2 un valor de 2.

Ejecutar las pruebas (Ctrl + Enter)

Restablecer todo el código

Consigue ayuda -

Pruebas

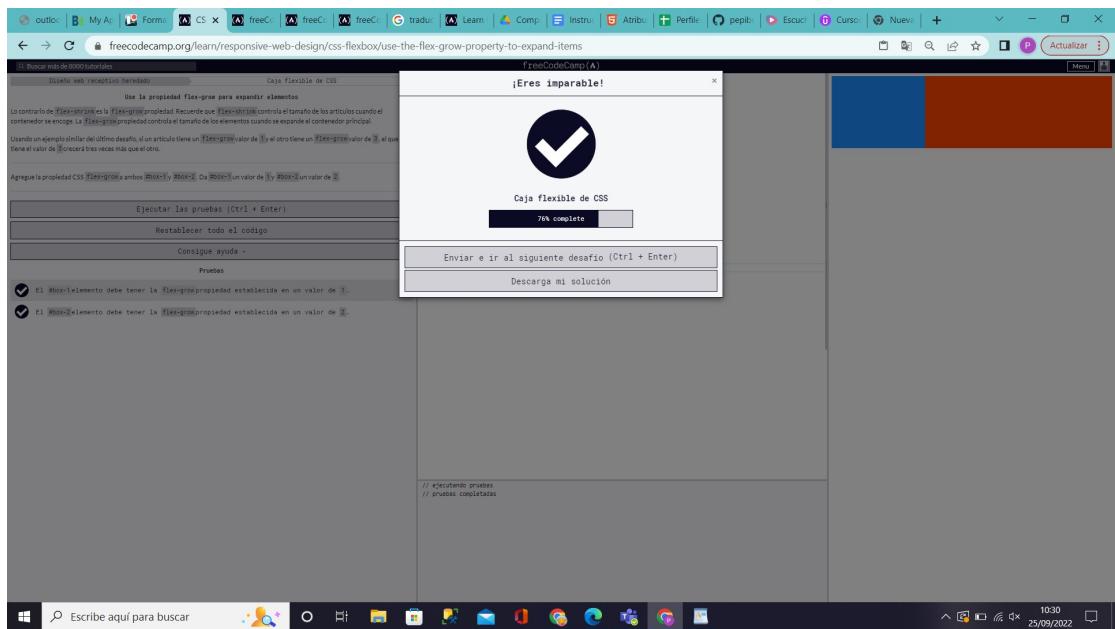
```

1  /*style*/
2  #box-container {
3      display: flex;
4      height: 200px;
5  }
6
7  #box-1 {
8      background-color: dodgerblue;
9      height: 200px;
10 }
11
12 #box-2 {
13     background-color: orange;
14     height: 200px;
15 }
16
17 
```

/* Su salida de prueba irá aquí

Escribe aquí para buscar

1030 25/09/2022



```
<style>
#box-container {
    display: flex;
    height: 500px;
}

#box-1 {
    background-color: dodgerblue;
    height: 200px;
flex-grow: 1;
}

#box-2 {
    background-color: orangered;
    height: 200px;
flex-grow: 2;
}

</style>

<div id="box-container">
```

```

<div id="box-1"></div>
<div id="box-2"></div>
</div>

```

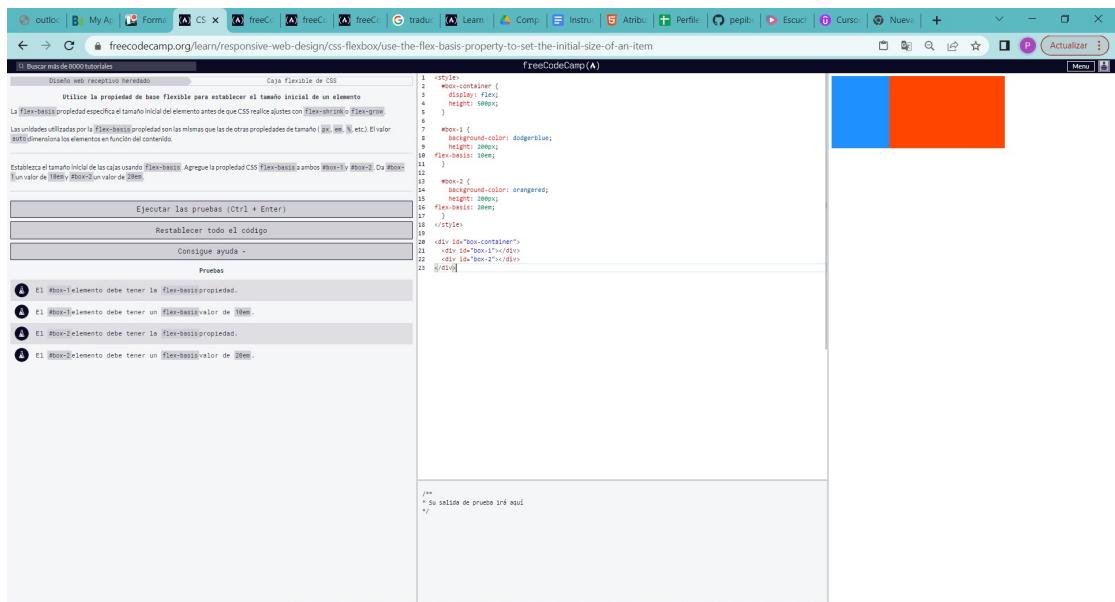
14. Utilice la propiedad de base flexible para establecer el tamaño inicial de un elemento

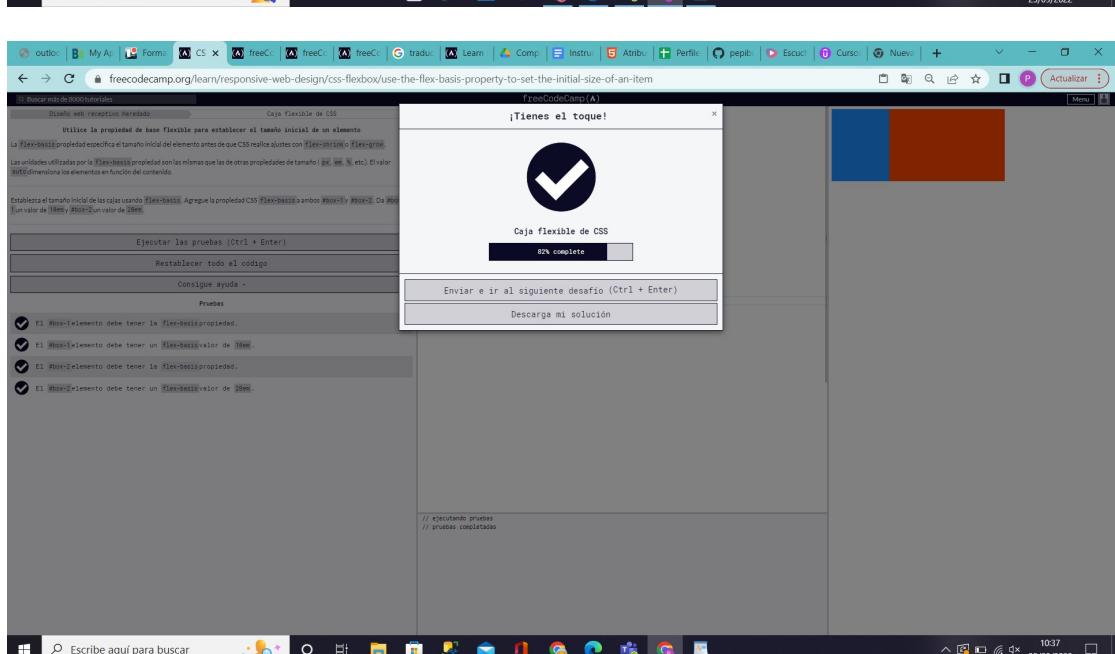
La **flex-basis** propiedad especifica el tamaño inicial del elemento antes de que CSS realice ajustes con **flex-shrink** o **flex-grow**.

Las unidades utilizadas por la **flex-basis** propiedad son las mismas que las de otras propiedades de tamaño (px, em, %, etc.). El valor autodimensiona los elementos en función del contenido.

Establezca el tamaño inicial de las cajas usando **flex-basis**. Agregue la propiedad CSS **flex-basis** a ambos **#box-1** y **#box-2**. Da **#box-1** un valor de **10em** y **#box-2** un valor de **20em**.

48





<style>

```
#box-container {
    display: flex;
    height: 500px;
}

#box-1 {
```

```

background-color: dodgerblue;
height: 200px;
flex-basis: 10em;
}

#box-2 {
background-color: orangered;
height: 200px;
flex-basis: 20em;
}

</style>

<div id="box-container">
<div id="box-1"></div>
<div id="box-2"></div>
</div>

```

15. Usar la propiedad abreviada flex

Hay un **atajo** disponible para **configurar varias propiedades flexibles a la vez**. Las propiedades flex-grow, flex-shrink y flex-basis se pueden configurar juntas mediante la **flexpropiedad**.

Por ejemplo, **flex: 1 0 10px;** establecerá el elemento en **flex-grow: 1;**, **flex-shrink: 0;** y **flex-basis: 10px;**

La configuración de propiedad **predeterminada** es **flex: 0 1 auto;**

Agregue la propiedad CSS flex a ambos #box-1 y #box-2.

Da #box-1 los valores así:

```

flex-grow: 2;
flex-shrink: 2;
flex-basis: 150px;

```

Da #box-2 los valores así:

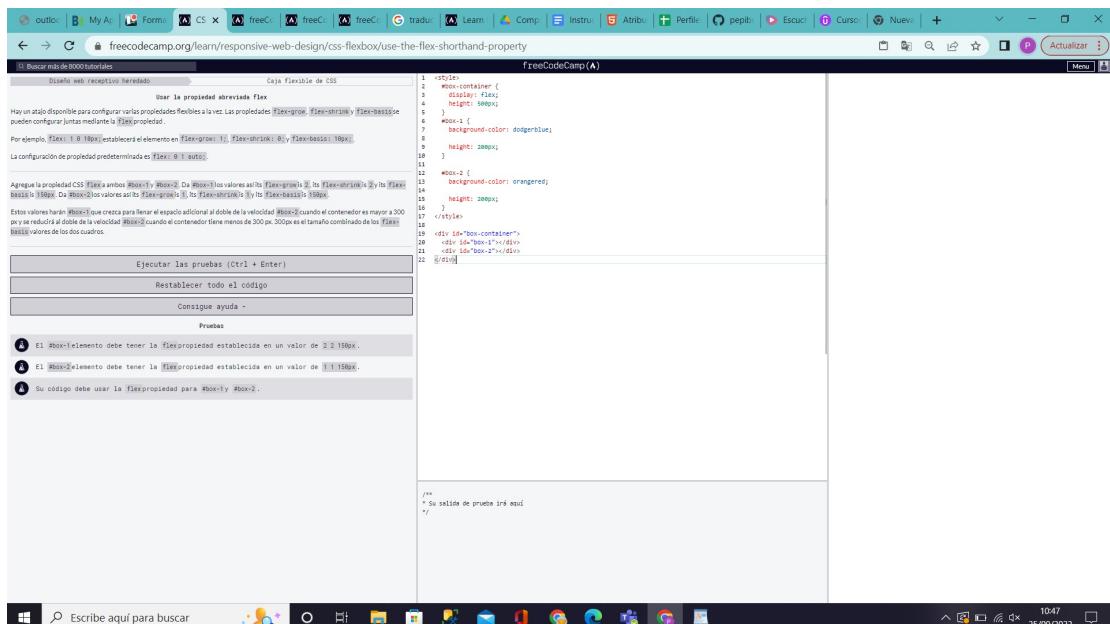
flex-grow: 1;

flex-shrink: 1;

flex-basis: 150px;

Estos valores harán #box-1 que **crezca** para **llenar el espacio adicional al doble de la velocidad** #box-2 cuando el contenedor es mayor a 300 px y se **reducirá al doble de la velocidad** #box-2 cuando el contenedor tiene menos de 300 px.

300px es el tamaño combinado de los flex-basis valores de los dos cuadros.



```
1 /* Caja flexible de CSS */
2 #box-container {
3   display: flex;
4   height: 300px;
5 }
6 #box-1 {
7   background-color: dodgerblue;
8   width: 200px;
9   height: 200px;
10 }
11 #box-2 {
12   background-color: orange-red;
13   width: 150px;
14   height: 200px;
15 }
16 
```

Ejecutar las pruebas (Ctrl + Enter)

Restablecer todo el código

Consigue ayuda -

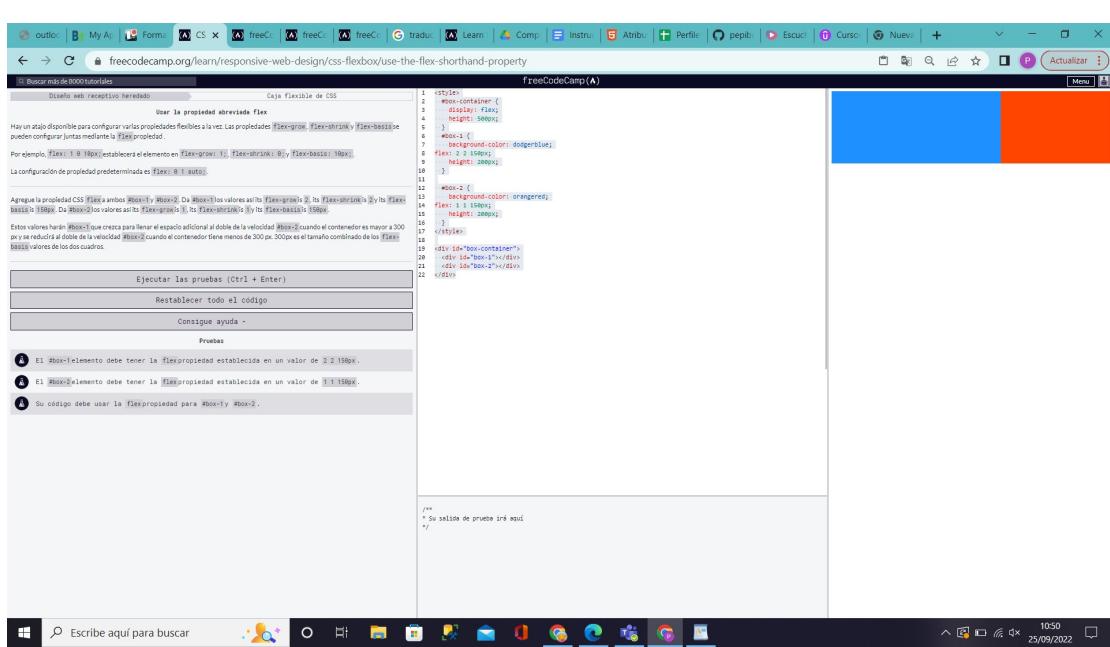
Pruebas

El #box-1 elemento debe tener la flex propiedad establecida en un valor de 2 2 150px.

El #box-2 elemento debe tener la flex propiedad establecida en un valor de 1 1 150px.

El código debe usar la flex propiedad para #box-1 #box-2.

/* Se saldrá de prueba iré aquí */



```
1 /* Caja flexible de CSS */
2 #box-container {
3   display: flex;
4   height: 300px;
5 }
6 #box-1 {
7   background-color: dodgerblue;
8   flex: 2 2 150px;
9   height: 200px;
10 }
11 #box-2 {
12   background-color: orange-red;
13   flex: 1 1 150px;
14   height: 200px;
15 }
16 
```

Ejecutar las pruebas (Ctrl + Enter)

Restablecer todo el código

Consigue ayuda -

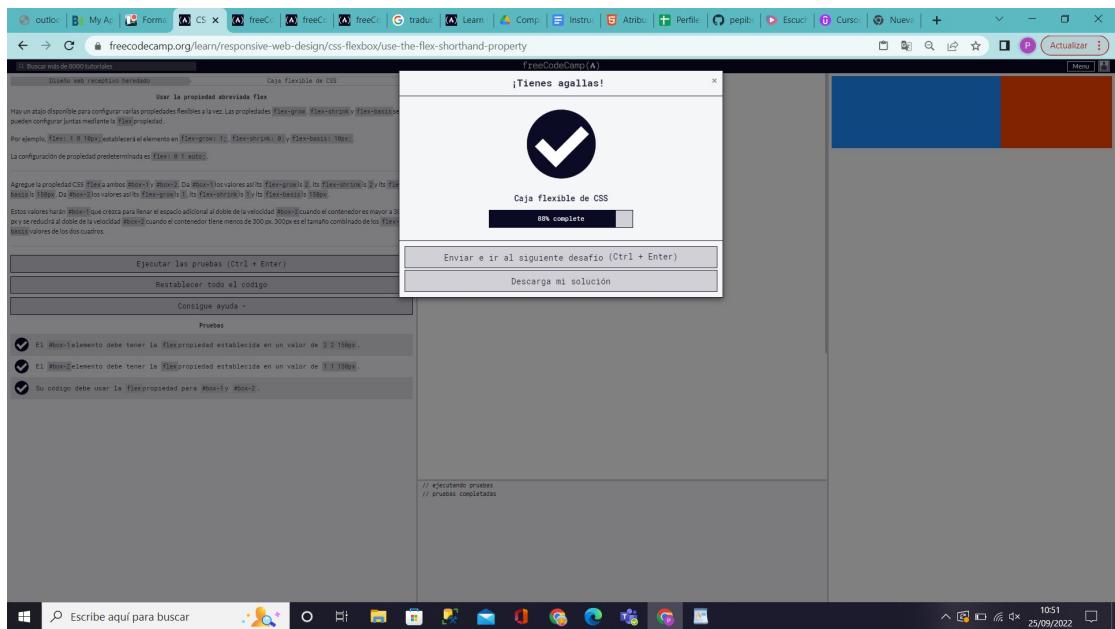
Pruebas

El #box-1 elemento debe tener la flex propiedad establecida en un valor de 2 2 150px.

El #box-2 elemento debe tener la flex propiedad establecida en un valor de 1 1 150px.

El código debe usar la flex propiedad para #box-1 #box-2.

/* Se saldrá de prueba iré aquí */



```

<style>

#box-container {
    display: flex;
    height: 500px;
}

#box-1 {
    background-color: dodgerblue;
    flex: 2 2 150px;
    height: 200px;
}

#box-2 {
    background-color: orangered;
    flex: 1 1 150px;
    height: 200px;
}

</style>

<div id="box-container">
    <div id="box-1"></div>

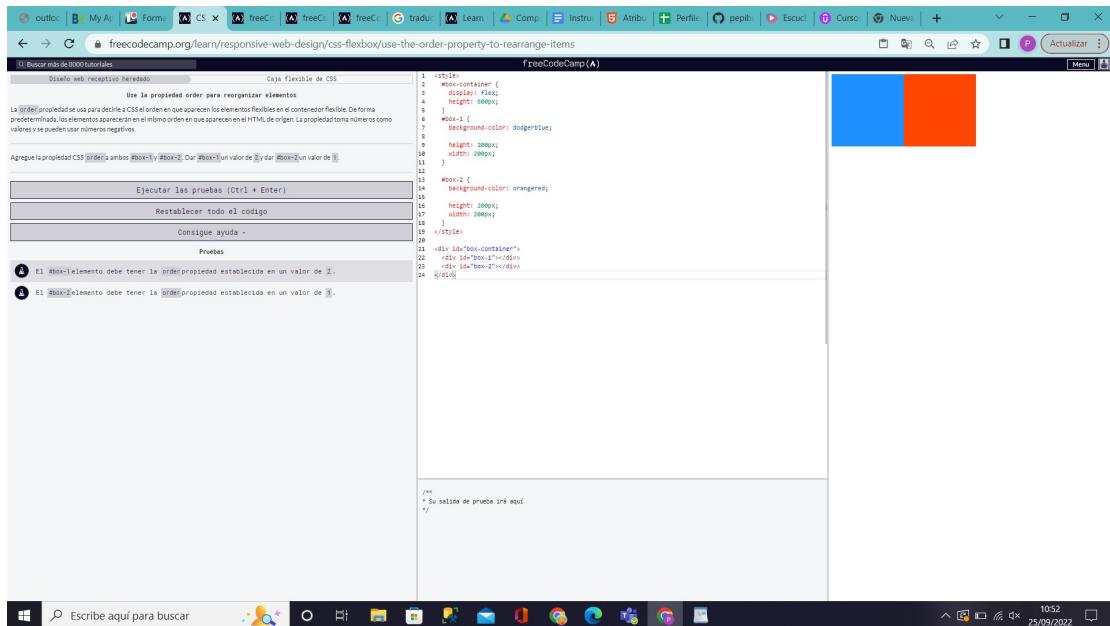
```

```
<div id="box-2"></div>  
</div>
```

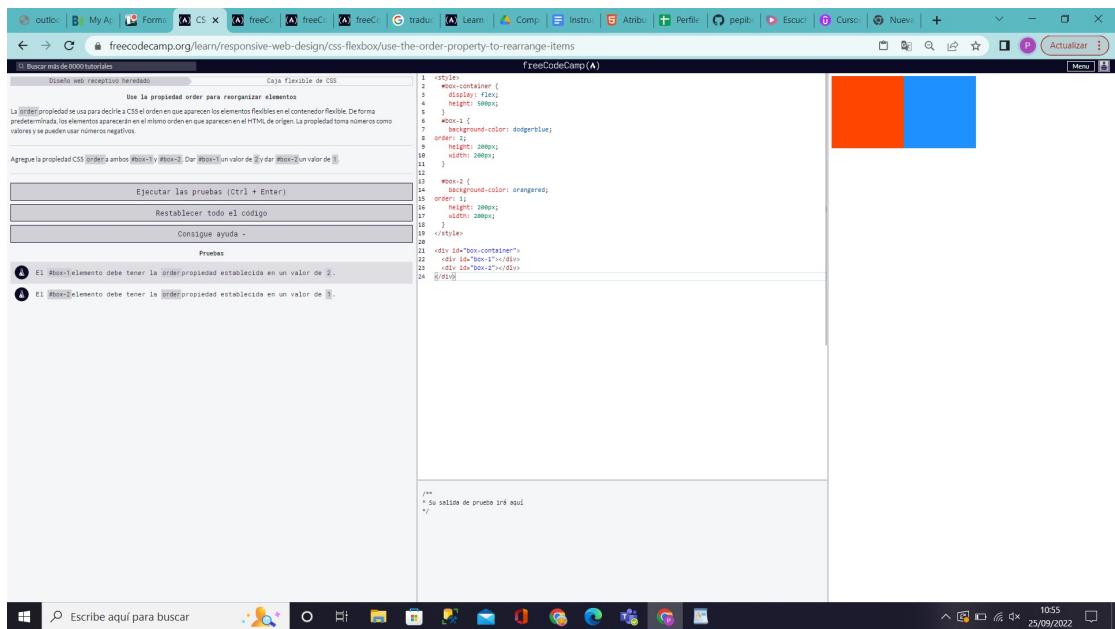
16. Use la propiedad order para reorganizar elementos

La **order** propiedad se usa para decirle a CSS el orden en que aparecen los elementos flexibles en el contenedor flexible. De forma predeterminada, los elementos aparecerán en el mismo orden en que aparecen en el HTML de origen. **La propiedad toma números como valores y se pueden usar números negativos.**

Agregue la propiedad CSS order a ambos #box-1y #box-2. Dar #box-1 un valor de 2 y dar #box-2 un valor de 1.



```
title:  
#box-1{  
    background-color: dodgerblue;  
    height: 200px;  
    width: 200px;  
}  
#box-2{  
    background-color: orangered;  
    height: 200px;  
    width: 200px;  
}  
/* Su salida de prueba irá aquí */
```



Caja flexible de CSS

La `order` propiedad se usa para decirle CSS en qué orden aparecen los elementos flexibles en el contenido flexible. De forma predeterminada, los elementos aparecen en el mismo orden en que aparecen en el HTML de origen. La propiedad `order` toma números como valores y se pueden usar números negativos.

Agregue la propiedad CSS `order` a ambos `#box-1` y `#box-2`. Dar `#box-1` un valor de `2` y dar `#box-2` un valor de `1`.

Ejecutar las pruebas (Ctrl + Enter)

Restablecer todo el código

Consigue ayuda -

Pruebas

El `#box-1` elemento debe tener la `order` propiedad establecida en un valor de `2`.

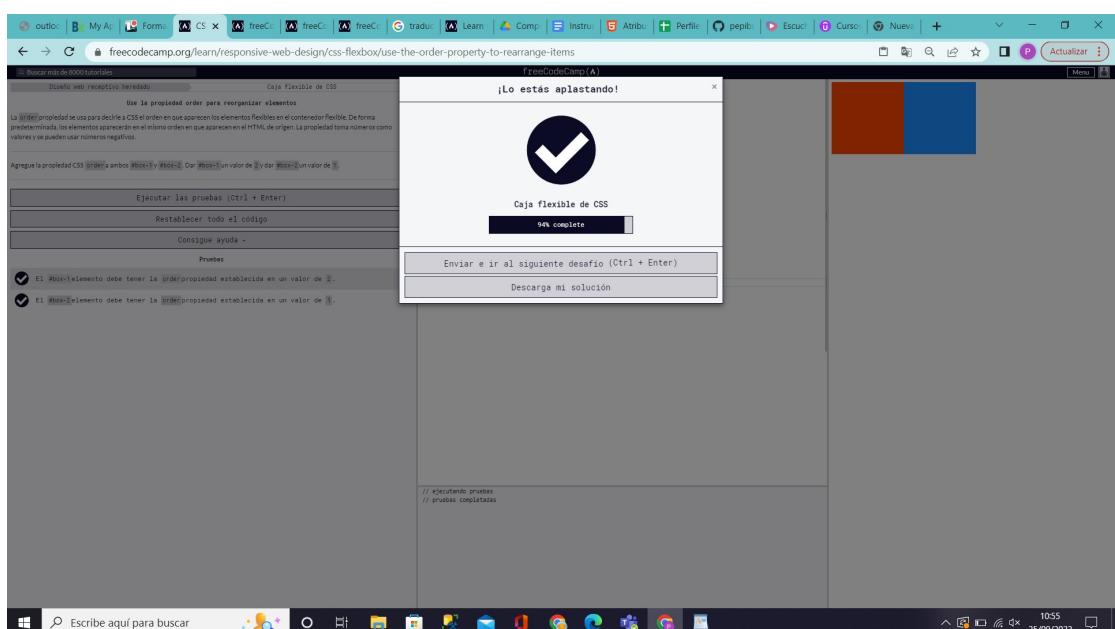
El `#box-2` elemento debe tener la `order` propiedad establecida en un valor de `1`.

freecodecamp(A)

```
1  /*style*/
2  #box-container {
3    display: flex;
4    height: 500px;
5  }
6  #box-1 {
7    background-color: dodgerblue;
8    order: 2;
9    height: 200px;
10   width: 200px;
11 }
12 #box-2 {
13   background-color: orange;
14   order: 1;
15   height: 200px;
16   width: 200px;
17 }
18 
```

/* Salida de prueba irá aquí */

1055 25/09/2022



¡Lo estás aplastando!

Caja flexible de CSS

¡Gen complete!

Enviar e ir al siguiente desafío (Ctrl + Enter)

Descargo mi solución

freecodecamp(A)

```
1  /*style*/
2  #box-container {
3    display: flex;
4    height: 500px;
5  }
6  #box-1 {
7    background-color: dodgerblue;
8    order: 2;
9    height: 200px;
10   width: 200px;
11 }
12 #box-2 {
13   background-color: orange;
14   order: 1;
15   height: 200px;
16   width: 200px;
17 }
18 
```

// Ejecutando pruebas
// Pruebas completadas

1055 25/09/2022

<style>

```
#box-container {  
  display: flex;  
  height: 500px;  
}  
  
#box-1 {
```

```
background-color: dodgerblue;  
order: 2;  
height: 200px;  
width: 200px;  
}  
  
#box-2 {  
background-color: orangered;  
order: 1;  
height: 200px;  
width: 200px;  
}  
  
</style>
```

```
<div id="box-container">  
  <div id="box-1"></div>  
  <div id="box-2"></div>  
</div>
```

17. Utilice la propiedad align-self

La propiedad final para elementos flexibles es **align-self**. Esta propiedad le **permite ajustar la alineación de cada elemento individualmente**, en lugar de configurarlos todos a la vez. Esto es útil ya que otras técnicas de ajuste comunes que usan las propiedades CSS float, clear y vertical-align no funcionan en elementos flexibles.

align-self acepta los mismos valores que align-items y anulará cualquier valor establecido por la align-items propiedad.

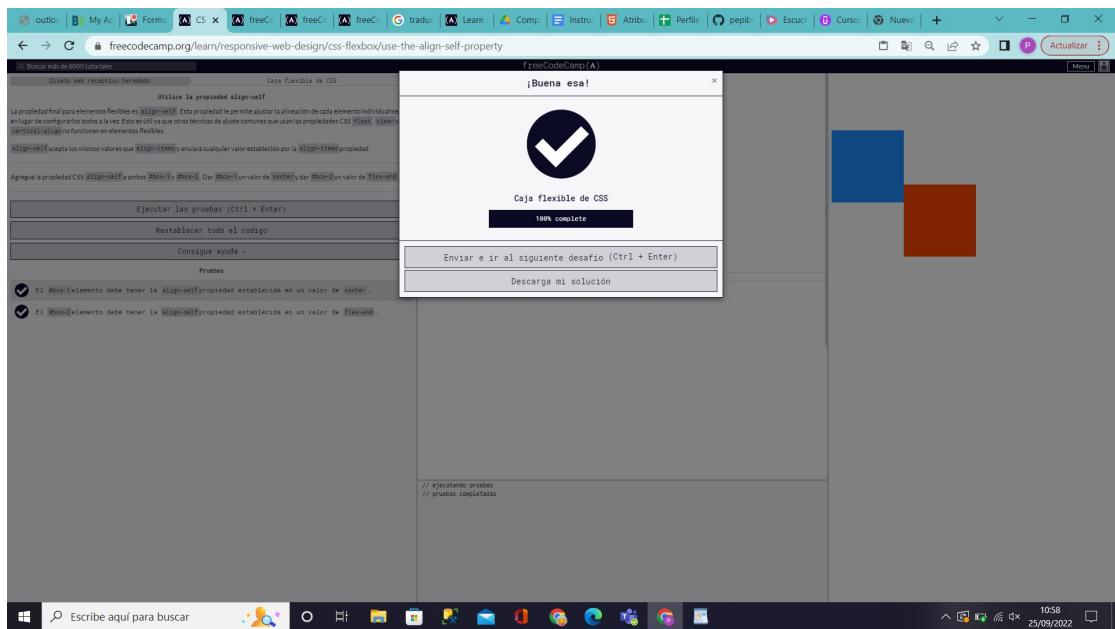
Agregue la propiedad CSS align-self a ambos #box-1 y #box-2. Dar #box-1 un valor de center y dar #box-2 un valor de flex-end.

The screenshot shows a browser window with the URL [freeCodeCamp.org/learn/responsive-web-design/css-flexbox/use-the-align-self-property](https://www.freecodecamp.org/learn/responsive-web-design/css-flexbox/use-the-align-self-property). The page title is "Caja flexible de CSS". The main content area contains a code editor with the following CSS and HTML:

```
1  /*style*/
2  #box-container {
3    display: flex;
4    height: 200px;
5  }
6  #box-1 {
7    background-color: dodgerblue;
8    height: 200px;
9    width: 200px;
10 }
11 #box-2 {
12   background-color: orangered;
13   height: 200px;
14   width: 200px;
15 }
16 
```

```
<html>
<head>
<title>Caja flexible de CSS</title>
</head>
<body>
<div id="box-container">
<div id="box-1"></div>
<div id="box-2"></div>
</div>
</body>
</html>
```

The preview area shows two squares, one blue and one orange, side-by-side. Below the browser window is a taskbar with various icons.



```
<style>

#box-container {
    display: flex;
    height: 500px;
}

#box-1 {
    background-color: dodgerblue;
    align-self: center;
    height: 200px;
    width: 200px;
}

#box-2 {
    background-color: orangered;
    align-self: flex-end;
    height: 200px;
    width: 200px;
}
```

```
</style>

<div id="box-container">
  <div id="box-1"></div>
  <div id="box-2"></div>
</div>
```