Guía de ejercicios 1 - Flexbox

 ¡Hola! Te damos la bienvenida a esta nueva guía de ejercicios.

¿En qué consiste esta guía?

Cuando hablamos de construcción de layout lo primero que se nos vendrá a la cabeza será la propiedad floats. Esta propiedad funciona muy bien para la construcción de sitios web, sin embargo, cuando la creación de interfaces se hace más compleja, esta propiedad podría no ser suficiente.

Para solventar este problema, existe una tecnología que nos permitirá posicionar nuestros elementos de manera sencilla usando flexbox.

Durante toda esta unidad conoceremos los elementos que componen a estas cajas flexibles, su comportamiento, además contrastaremos flexbox con la grilla de bootstrap y finalmente aplicaremos esta tecnología en un proyecto.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

Conceptos básicos de Flexbox

Flexbox o contenedores flex es una herramienta que nos permite alinear contenido de forma sencilla. Un caso básico de uso sería:

```
<div style="display:flex">
  <p> Item 1 </p>
  <p> Item 2 </p>
  <p> Item 3 </p>
</div>
```

Imagen 1. Código.
Fuente: Desafío Latam.

Lo que nos daría como resultado:

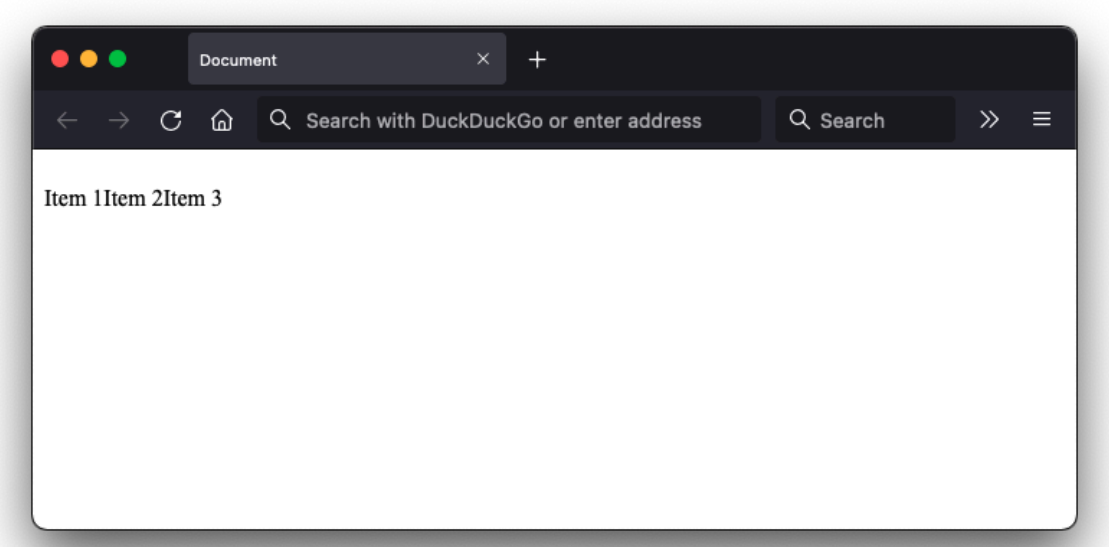

Imagen 2. Screenshot de ejemplo básico de flex.
Fuente: Desafío Latam.

Con este ejemplo, vemos que basta aplicar la propiedad display:flex a un contenedor padre para poder alinear los contenidos uno al lado del otro.

Flexbox funciona con dos elementos principales: el primero es el contenedor de los elementos llamado “Flex container” o contenedor flex, y el segundo, son los ítems al interior del contenedor llamados “Flex ítem” o ítems flex, donde el flex container es el contenedor padre de los Flex ítems.

En nuestro ejercicio, el div sería el contenedor flex y los párrafos los ítems flex.

Alineando el contenido

La propiedad justify-content nos permite alinear el contenido dentro de un contenedor flex. Por ejemplo probemos:

```
<div style="display:flex; justify-content:center">
  <p> Item 1 </p>
  <p> Item 2 </p>
  <p> Item 3 </p>
</div>
```

Imagen 3. Código.
Fuente: Desafío Latam.

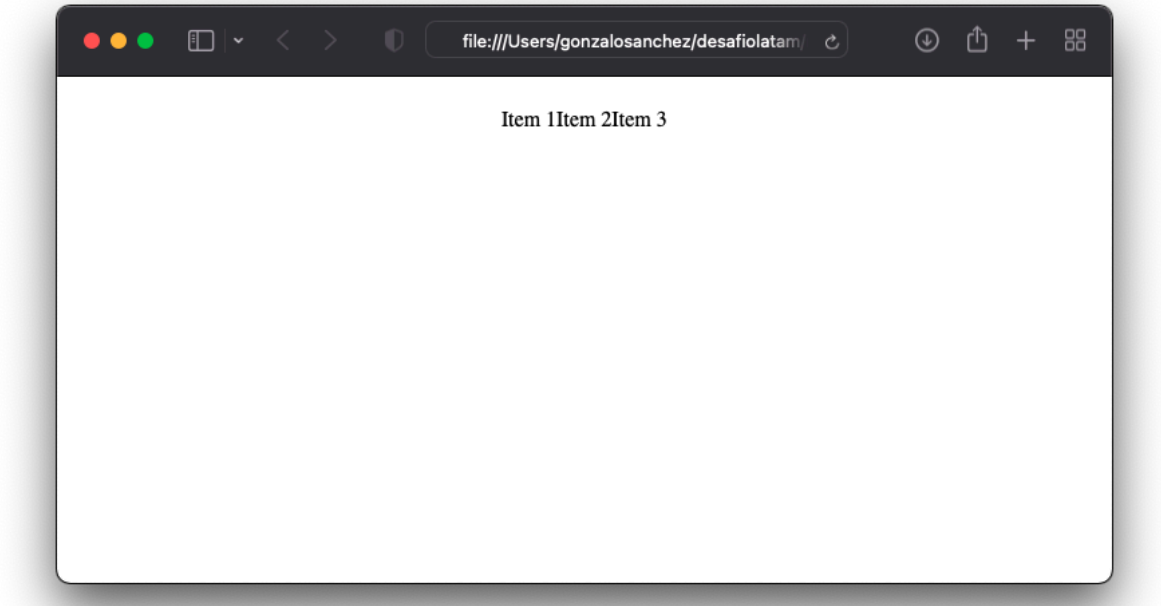

Imagen 4. Flex + justify-content.
Fuente: Desafío Latam.

Justify-content tiene varios valores posibles, la siguiente ilustración muestra las diferencias.

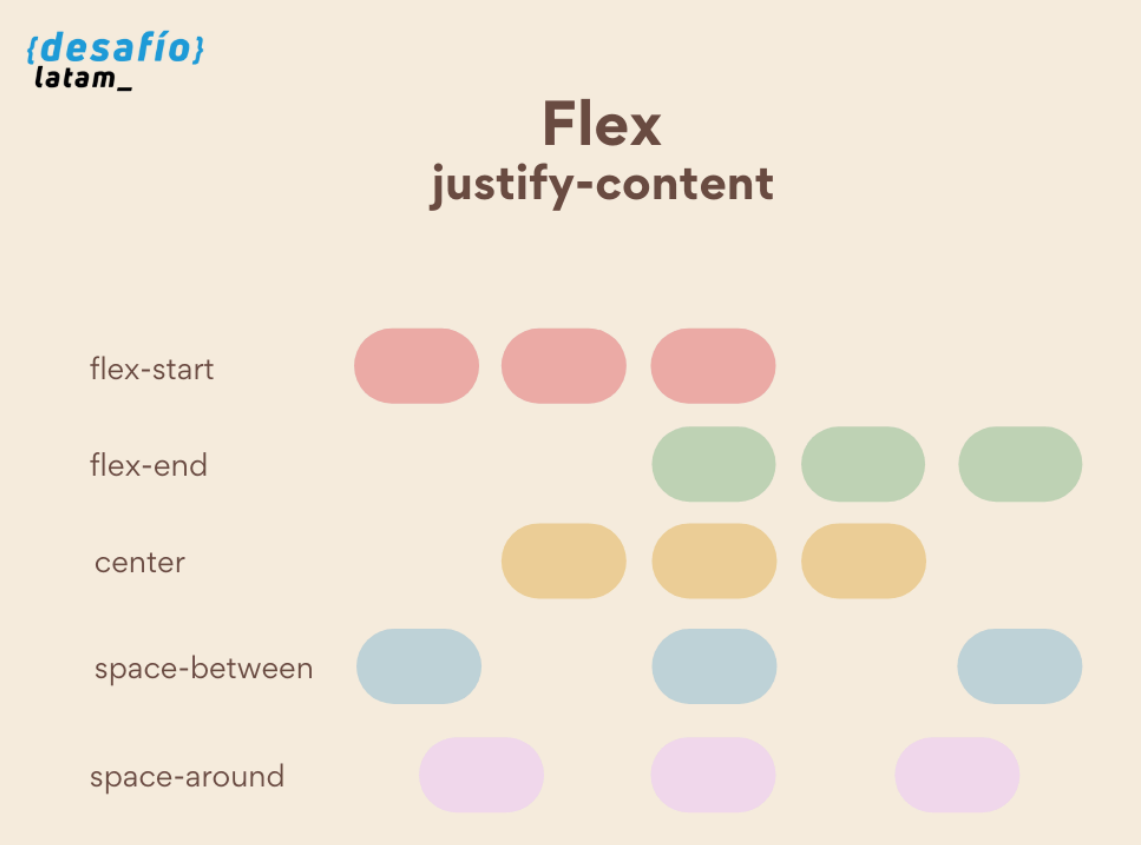

Imagen 5. Propiedad Justify-content.
Fuente: Desafío Latam.

Actividad 1: Alineando contenido con Flex

En el ejemplo anterior, donde se tiene un contenedor con display:flex y 3 párrafos:

- Prueba cambiando la propiedad justify-content del contenedor flex utilizando los valores de la ilustración, observa los resultados.
- También puedes usar el inspector de elementos para probar los cambios.

Actividad 2: Creando una galería de imágenes (parte 1)

Utiliza lo aprendido para crear una galería de imágenes, para lograrlo, tendrás que crear una nueva carpeta y el archivo index.html con una base de HTML. Luego, puedes utilizar la siguiente estructura como base para el ejercicio:

```
<div style="display:flex">
 <!-- Primer elemento de la galería -->
  <div>
```

```
    <img>
    <p>
  </div>
  <!-- Segundo elemento de la galería -->
   <div>
    <img>
    <p>
  </div>
  <!-- Tercer elemento de la galería -->
   <div>
    <img>
    <p>
  </div>
</div>
```

Utiliza justify-content para alinear el contenido de la galería, según estimes que mejor quedan los resultados.

Observaciones:

- Los flex-items también pueden ser un div u otro tipo de elemento.
- El primer div es el flex-container y sus elementos hijos son los flex-items.
- Puedes mover el CSS en un archivo externo o en el header.
- Si las imágenes son muy grandes puedes utilizar CSS para ajustarlas, comienza probando con agregar max-width: 100% sobre la imagen.

Actividad 3: Creando una galería de imágenes (parte 2)

Sobre el ejercicio de la actividad 2:

- Agrega 9 elementos más a la galería de imágenes y observa como se ve la página, deberían quedar a lo largo de una única fila, y los últimos elementos no podrán verse a menos que hagas scroll horizontal.
- Utiliza sobre el flex-container la propiedad flex-wrap: wrap y abre la página de nuevo, ahora deberías poder ver la galería de imágenes bien.

Actividad 4: Creando un menú con flex

- Crea una nueva carpeta llamada primer-menu y el archivo index.html con la base de HTML.
- Copia dentro del body las siguientes etiquetas

```
<menu>
  <ul>
    <li>
      <a href="#"> Inicio </a>
    </li>
    <li>
      <a href="#"> Pag 2 </a>
    </li>
    <li>
      <a href="#"> Pag 3 </a>
    </li>
  </ul>
</menu>
```

- Dentro del Head o un archivo de CSS externo agrega:

```
menu ul {display: flex}
```

- Utiliza CSS para eliminar los márgenes de forma que quede en la parte superior del sitio.
- Agrega un color de fondo al menú.
- Puedes también cambiar el color de los links, el text-decoration y el color.

Eje principal y eje transversal

En flex hay dos ejes, el eje principal y el eje transversal.

Hasta el momento, cuando alineamos el contenido, lo hemos hecho respecto al eje principal usando justify-content, ahora, para alinear el contenido respecto al transversal utilizaremos align-items.

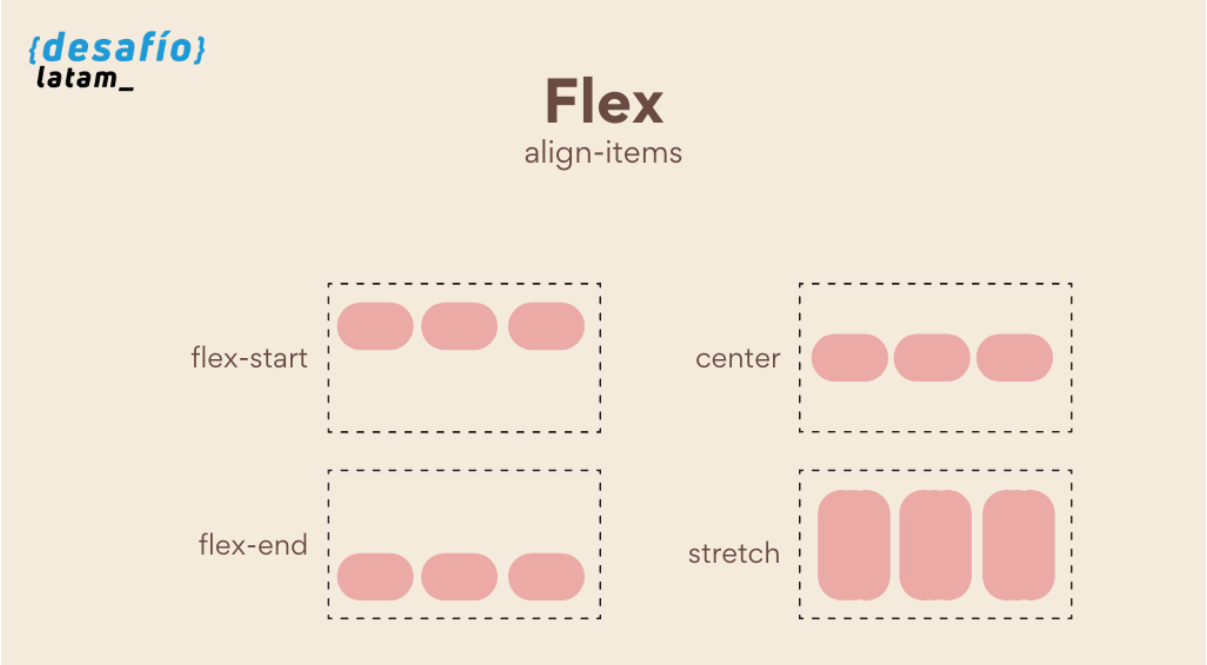

Imagen 6. Propiedad align-items.
Fuente: Desafío Latam.

Esto podemos probarlo en el mismo menú que construimos en la actividad 4, pero necesitaremos darle un alto al menú de forma de poder alinear los elementos respecto a ese alto.

```
menu ul{
 margin: 0;
 padding: 0;
 display: flex;
 align-items: flex-start;
 height: 4em;
}
```

Luego, podemos utilizar el inspector de elementos para observar el efecto con flex-start, flex-end, center y stretch.

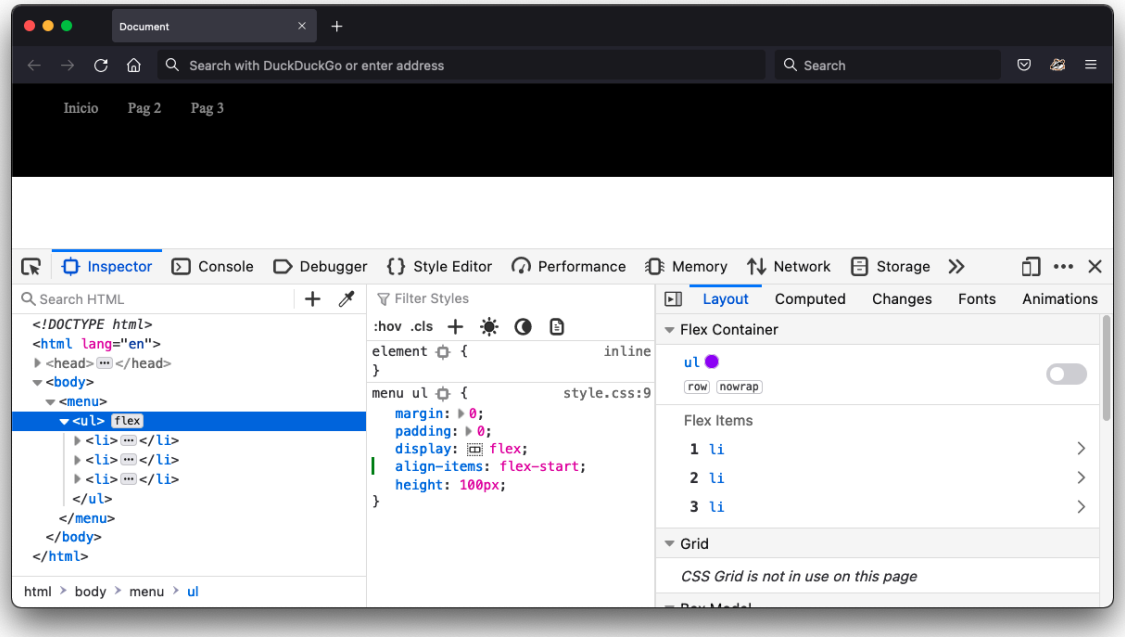

Imagen 7. Explorando la propiedad flex-start con el inspector de elementos.
Fuente: Desafío Latam.

Importante: Para observar el efecto de asignar el valor a stretch, luego de cambiar la propiedad align-items a stretch se debe hacer clic dentro del inspector de elementos dentro de una de las etiquetas li y con esto dentro del navegador podremos ver resaltada el área vertical.

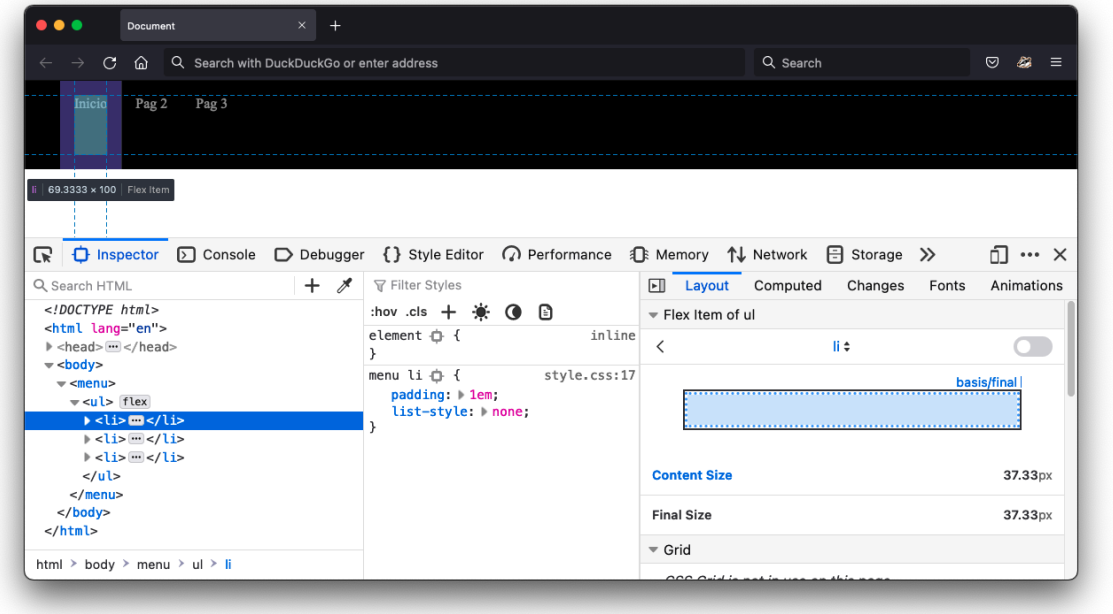

Imagen 8. Explorando la propiedad align-items con el valor stretch.
Fuente: Desafío Latam.

Centrando elementos

Utilizando justify-content y align-items podemos centrar elementos.

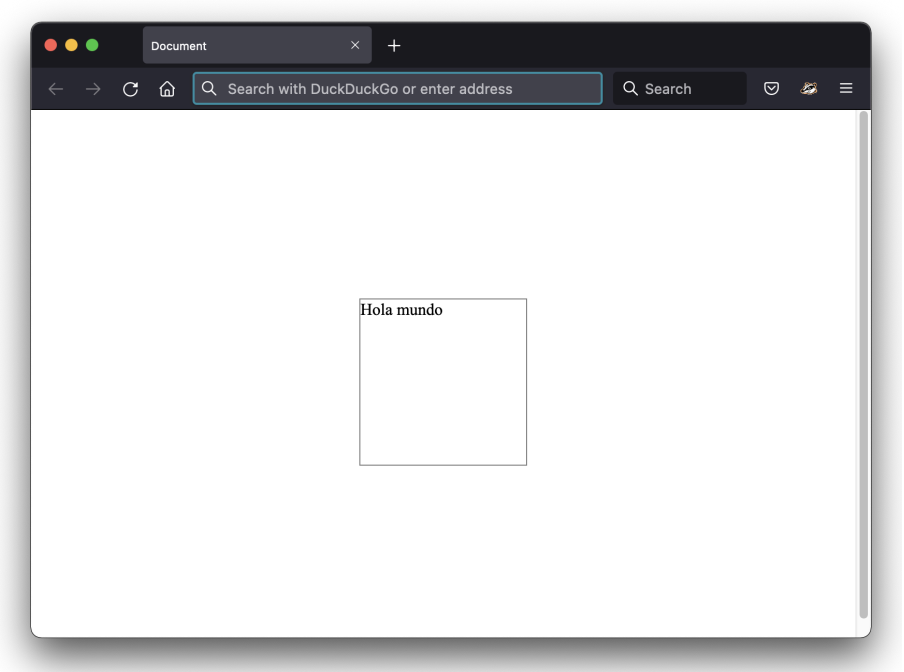

Imagen 9. Screenshot de elemento centrado con justify-content y align-item.
Fuente: Desafío Latam.

Actividad 5

- Crear la carpeta centrando-con-flex y un archivo index.html con la base de HTML.
- En el body agregar:

```
<div class="flex-container">
  <p class="flex-item"> Hola mundo </p>
</div>
```

- En el CSS (puede ser en el head o un archivo externo) agregar:

```
.flex-container{
  display: flex;
  align-items: center;
  justify-content: center;
}
.flex-item{
  width: 10em;
```

```
    height: 10em;
    border: solid 1px gray;
}
```

- Abrir el navegador y observar que hay una diferencia con la imagen mostrada, ya que la página que estamos viendo solo está centrada horizontalmente.
- Utilizar el inspector de elementos para descubrir el alto del contenedor y de su padre.
- En el CSS agregar height: 100vh dentro de flex-container, esto es equivalente a 100% del alto del viewport (la ventana donde se muestra la página).
- Ver que ahora el contenedor está centrado.
- Reparemos la idea del alto del contenedor cuando construyamos un menú vertical.

Flex direction

¿Qué es flex flow direction?

Utilizando flex-direction podemos cambiar la dirección del eje sobre el que funciona flex, por defecto su valor es row (fila):

Imagen 10. Flex-direction:row.
Fuente: Desafío Latam.

Podemos cambiar el flex-direction a column (columna) de forma de disponer el contenido a lo largo de una columna.

Imagen 11. Flex-direction:column.
Fuente: Desafío Latam.

Actividad 6: Construyendo un menú vertical

- Crea una nueva carpeta llamada flex-column y dentro el archivo index.html y agrega la estructura base de un documento html.
- Dentro del body del archivo, copia el siguiente html.
- Abre la página en el navegador:

```
 <div style="display:flex; flex-direction:column; width: 9em;
background-color: aqua">
   <p> Item 1 </p>
   <p> Item 2 </p>
   <p> Item 3 </p>
 </div>
```

Deberías poder ver en el navegador lo siguiente:

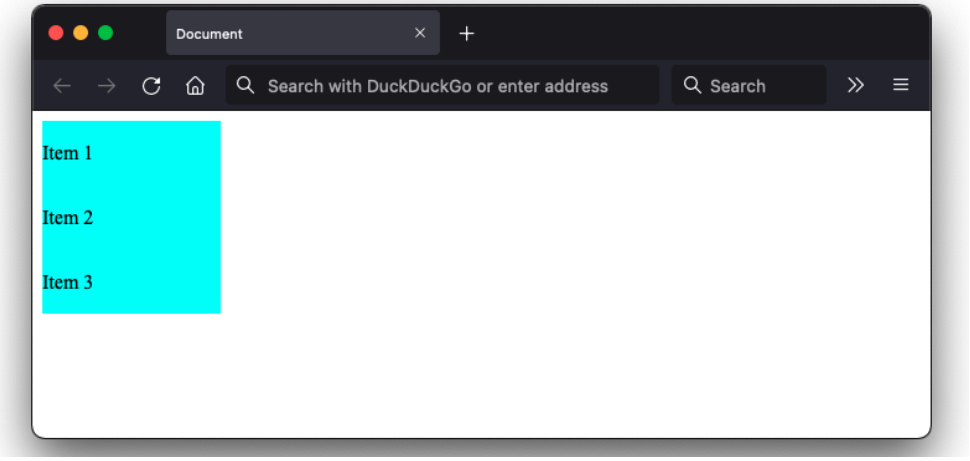

Imagen 12. Screenshot de un menú vertical con Flex.
Fuente: Desafío Latam.

- Al igual que lo hicimos en el ejercicio previo del menú, cuando vamos a centrar el texto en torno al alto del menú, utilizamos align-items.
- Ahora el menú es vertical y tiene un ancho en lugar de un alto, pero la idea es la misma, para centrar el texto debemos ocupar align-items: center.

```
 <div style="display:flex; align-items:center; flex-direction:column;
width: 9em; background-color: aqua">
   <p> Item 1 </p>
   <p> Item 2 </p>
   <p> Item 3 </p>
```

```
</div>
```

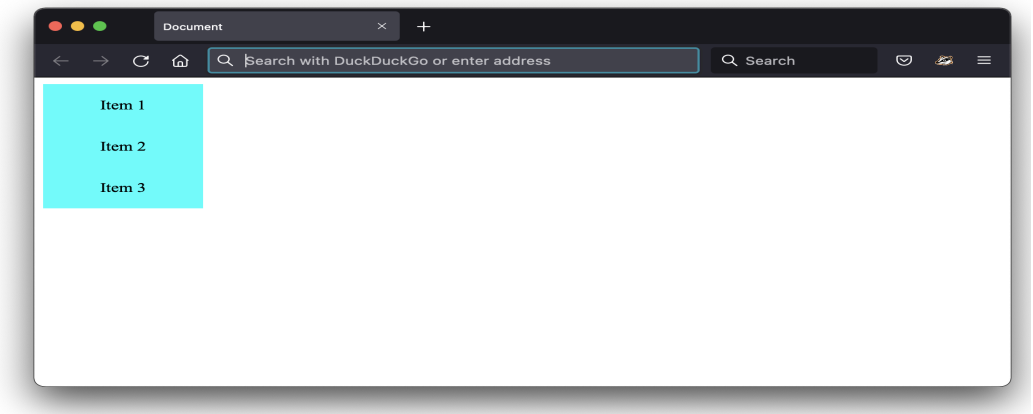

Imagen 13. Centrando los elementos con align-items:center.
Fuente: Desafío Latam.

El problema es que intentamos agregar contenido a la derecha, por lo que terminaremos escribiendo debajo del menú.

```
 <div style="display:flex; align-items: center; flex-direction: column;
width: 9em; background-color: aqua;">
   <p> Item 1 </p>
   <p> Item 2 </p>
   <p> Item 3 </p>
 </div>
 <div>
   Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur
voluptate sapiente corporis quia libero cum tempore, laborum velit
reprehenderit fuga, cumque expedita quibusdam ratione in. Et veritatis rerum
vel fuga.
 </div>
```

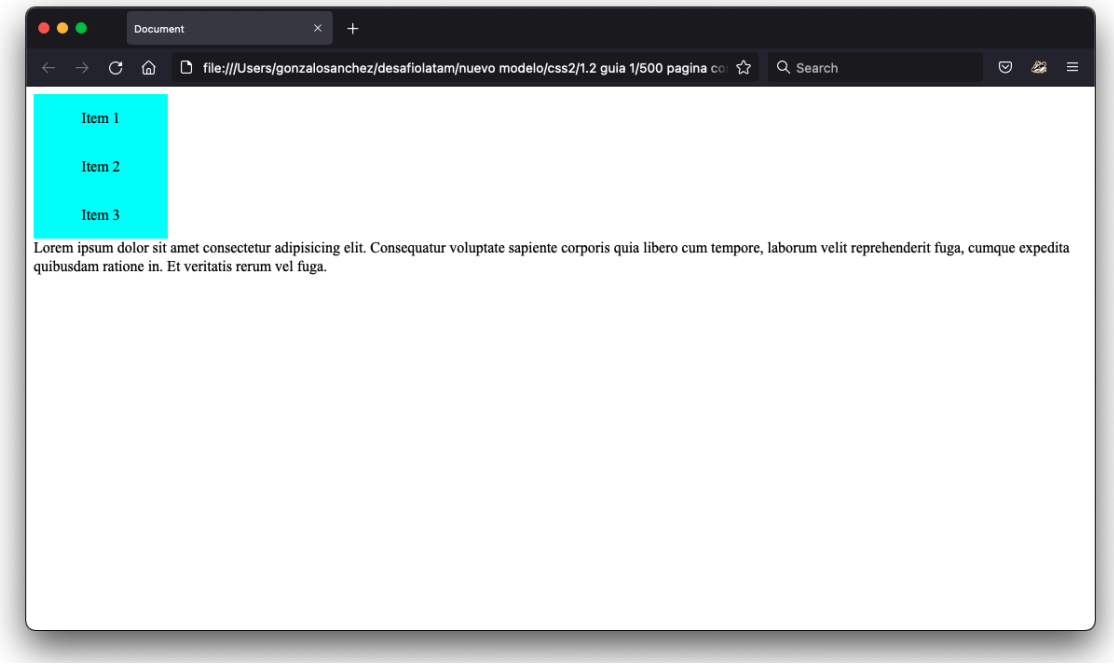

Imagen 14. Screenshot del problema al agregar contenido después del menú.
Fuente: Desafío Latam.

Esto último, aprenderemos a resolverlo cuando aprendamos a anidar los contenedores flex. Por ahora borremos el segundo div.

Controlando el alto del menú

Podemos definir un alto del menú en cualquier unidad de medida, pero si queremos que ocupe el 100% del alto de la página deberíamos utilizar 100vh.

```
 <div style="display:flex; align-items: center; flex-direction: column;
width: 9em; background-color: aqua; height: 100vh;">
   <p> Item 1 </p>
   <p> Item 2 </p>
   <p> Item 3 </p>
 </div>
```

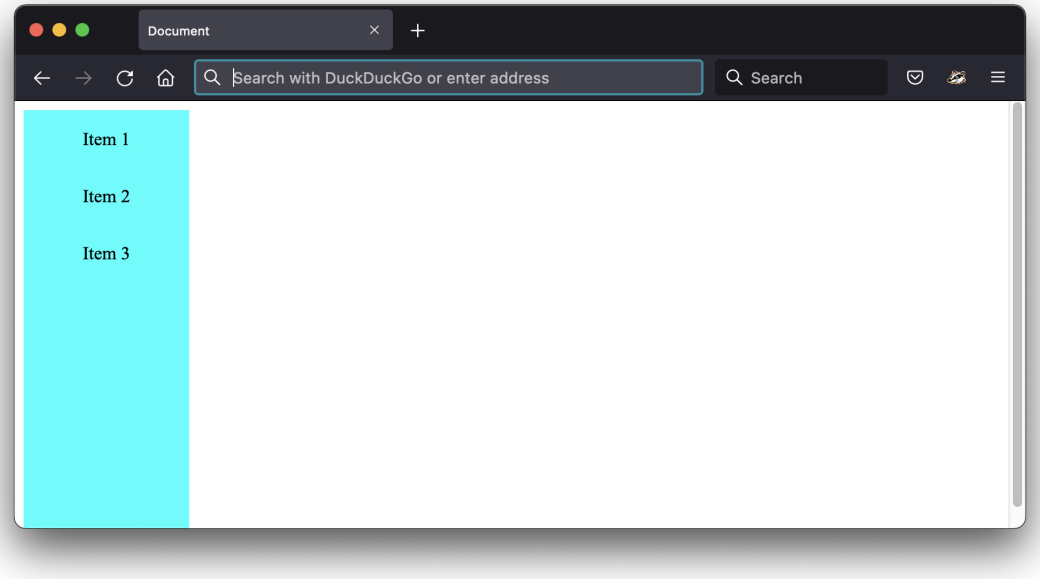

Imagen 15. Screenshot de menú vertical con alto de 100vh.
Fuente: Desafío Latam.

De todas formas tenemos que tener cuidado cuando utilicemos height, debido a que si el contenido, que luego agregaremos a la derecha, es mayor al 100% del alto del viewport, el menú no quedará del alto del documento. Por lo mismo, tenemos que utilizar un alto mínimo de 100vh en lugar de solo alto de vh, de esa forma le damos la opción de crecer en caso de que sea necesario.

```
 <div style="display:flex; align-items: center; flex-direction: column;
width: 9em; background-color: aqua; min-height: 100vh;">
   <p> Item 1 </p>
   <p> Item 2 </p>
   <p> Item 3 </p>
 </div>
```

Anidando flex

En algunas situaciones necesitaremos un contenedor flex que contenga otros contenedores flex. Un ejemplo muy simple de eso es un menú como el siguiente:

Imagen 16. Screenshot de menú con dos secciones (menú anidado).
Fuente: Desafío Latam.

Construyendo un menú anidado

Lograr esto requiere de algunos pasos adicionales. Primero, aquí existen dos contenedores flex, un contenedor padre que separa en dos grupos con espacio entremedio a los contenedores hijo, y los contenedores hijos que tienen los elementos del menú.

Para agregar el espacio simplemente sobre el contenedor principal tendremos que agregar: **display: flex** y **justify-content: space-between**

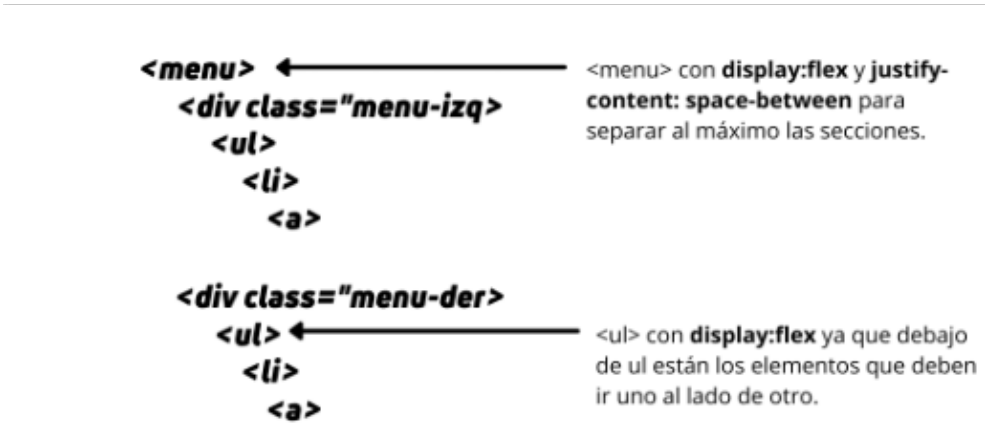

Imagen 17. Explicación de secciones del menú anidado.
Fuente: Desafío Latam.

Para lograr el menú compuesto con secciones, podemos utilizar el siguiente HTML como base:

```
<menu>
  <div class="menu-left">
    <ul>
      <li>
        <a href="#"> Inicio </a>
      </li>
      <li>
        <a href="#"> Pag 2 </a>
      </li>
    </ul>
  </div>
  <div class="menu-right">
    <ul>
      <li>
        <a href="#"> Pag 3 </a>
```

```
        </li>
      </ul>
    </div>
  </menu>
```

Y el siguiente CSS:

```
body, menu{
 margin: 0;
}

menu {
 background-color:black;
 margin: 0;
 padding: 0;
 display: flex;
 justify-content: space-between;
}

menu li{
 padding: 1em;
 list-style: none;
}

menu a{
 text-decoration: none;
 color: white;
 color: rgba(255, 255, 255, 0.6);
}

menu a:hover{
 color: white;
}

menu ul{ /* Estos son todos los ul dentro de menús, sean hijos directos o no
*/
 display: flex;
}
```

También podemos haber logrado lo mismo cambiando el bloque dentro de menu ul {} por:

```
.menu-left ul{
 display: flex;
```

```
}

.menu-right ul{
 display: flex;
}
```

Construyendo una página con dos secciones

Anteriormente, construimos un menú vertical, el problema es que si queríamos agregar contenido a la derecha no lo podremos hacer directamente.

```
  <div style="display:flex; align-items: center; flex-direction: column;
width: 9em; background-color: aqua;">
    <p> Item 1 </p>
    <p> Item 2 </p>
    <p> Item 3 </p>
  </div>
  <div>
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur
voluptate sapiente corporis quia libero cum tempore, laborum velit
reprehenderit fuga, cumque expedita quibusdam ratione in. Et veritatis
rerum vel fuga.
  </div>
```

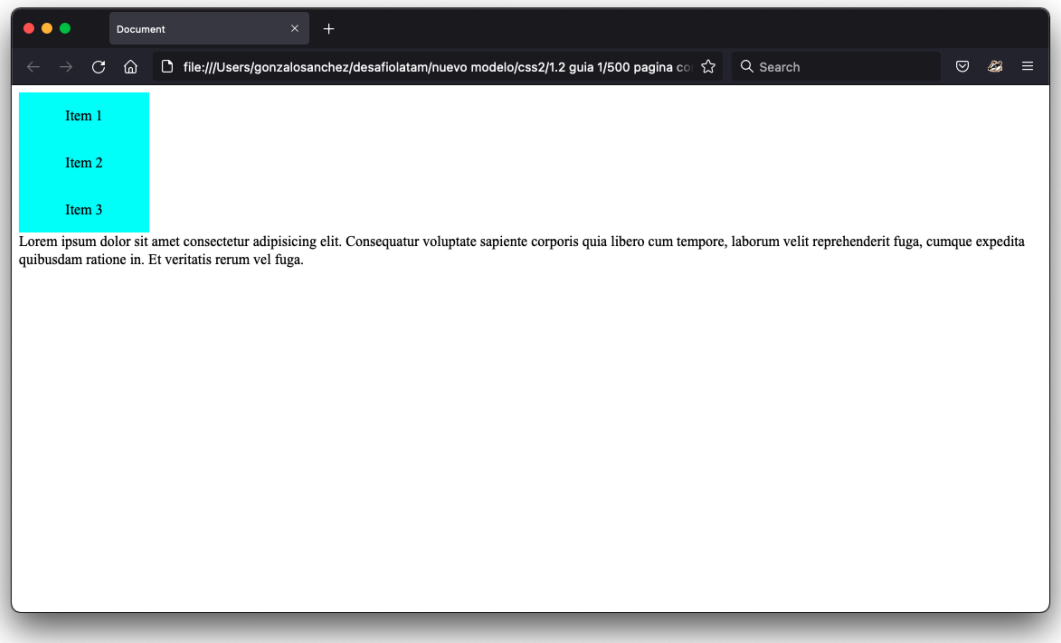

Imagen 18. Screenshot del problema al agregar contenido después del menú vertical.
Fuente: Desafío Latam.

Para resolverlo, envolveremos ambos divs en nuevo contenedor flex:

```
<div style="display: flex"> <!-- Nuevo contenedor flex -->
      <div
        style="
          display: flex;
          align-items: center;
          flex-direction: column;
          width: 9em;
          background-color: aqua;
          min-height: 100vh;
        "
      >
        <p>Item 1</p>
        <p>Item 2</p>
        <p>Item 3</p>
      </div>
      <div>
        Lorem ipsum dolor sit amet consectetur adipisicing elit.
Consequatur
        voluptate sapiente corporis quia libero cum tempore, laborum
velit
        reprehenderit fuga, cumque expedita quibusdam ratione in. Et
veritatis
        rerum vel fuga.
      </div>
    </div>
```

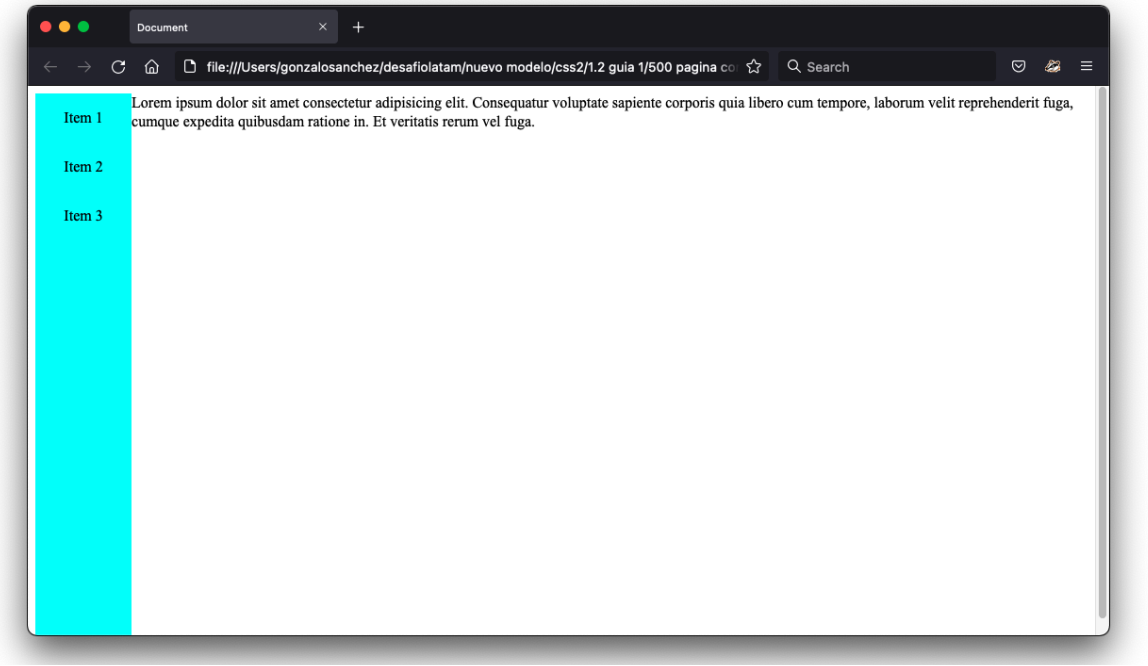

Imagen 19. Screenshot después de crear dos secciones con flex resolviendo el problema de agregar contenido.
Fuente: Desafío Latam.

Flex grow, basis and shrink

En el ejercicio anterior, creamos un layout sencillo con dos columnas; en esta sección, aprenderemos algunas propiedades básicas de flex que nos permitan controlar el tamaño de los elementos. Estas son flex-grow, flex-basis, flex-shrink.

Por defecto, dentro de un contenedor flex el tamaño de los flex-item depende de su contenido interno. Creemos un ejemplo sencillo donde podamos ver esto.

```
<!-- index.html -->
  <div class="flex">
   <div class="flex-item item-1">
     1
   </div>
   <div class="flex-item item-2">
     2
   </div>
   <div class="flex-item item-3">
     3
   </div>
 </div>
```

```
/* style.css */
   .flex{
     display:flex;
   }
   .flex-item{
     background-color: aqua;
     border: solid 1px black;
     margin: 1px;
   }
```

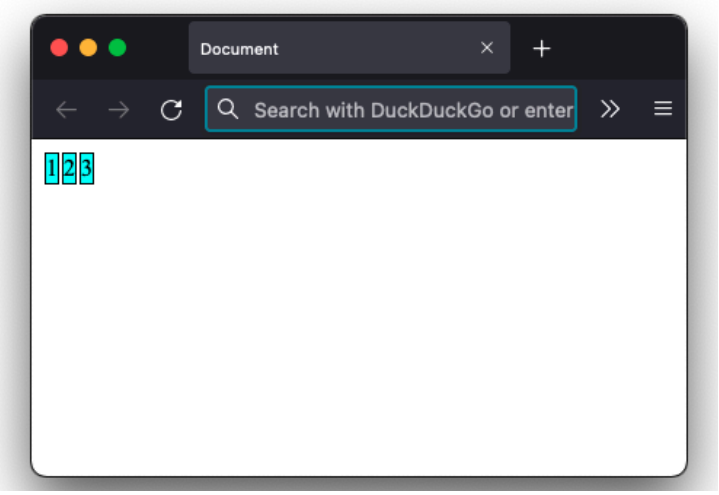

Imagen 20. Screenshot de uso de Flex resaltando los distintos ítems con fondo de color.
Fuente: Desafío Latam.

Si cambiamos el texto de 2 por hola a todos, obtendremos:

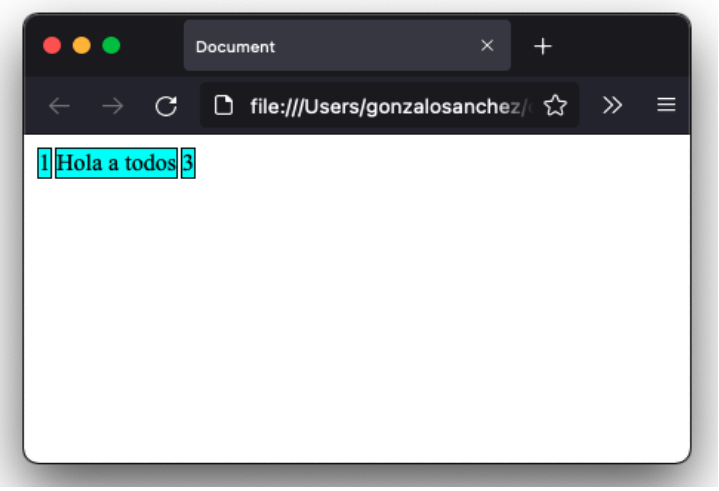

Imagen 21. Screenshot demostrando que el tamaño de los ítems flex se adapta al contenido dentro del ítem.
Fuente: Desafío Latam.

Flex-grow

La propiedad flex-grow especifica un factor de crecimiento para los ítems. Partamos probando con flex-grow: 1, esto lo agregaremos a nuestro CSS anterior:

```
.flex-item{
  background-color: aqua;
  border: solid 1px black;
  margin: 1px;
  flex-grow: 1;
}
```

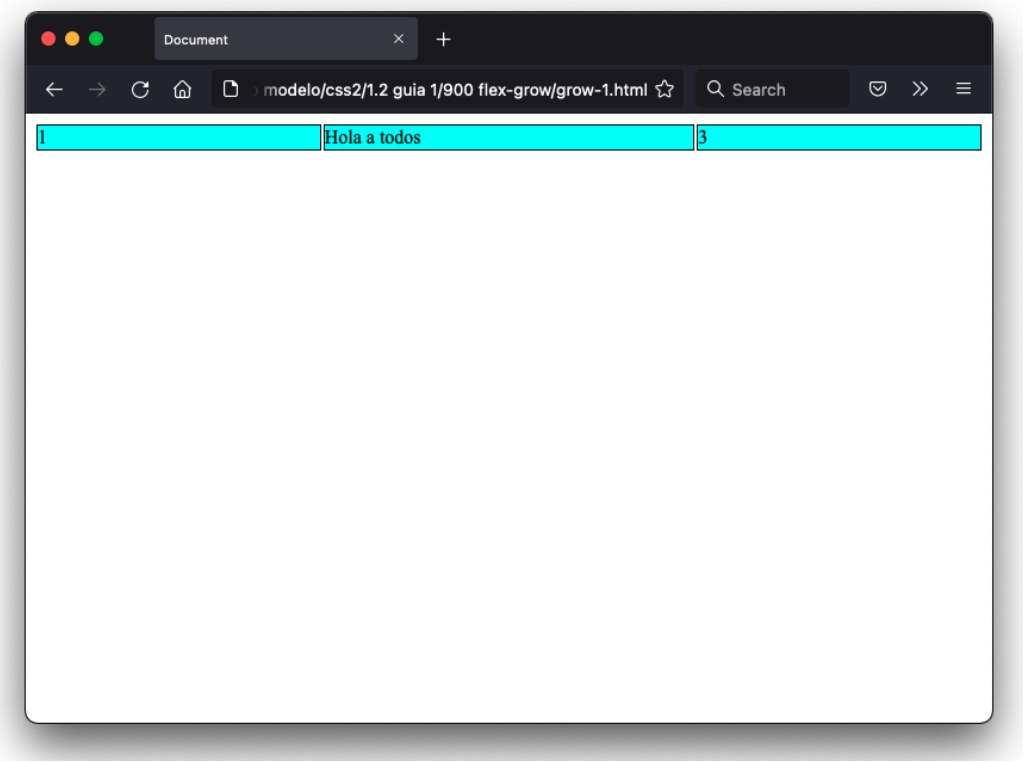

Imagen 22. Screenshot tras aplicar la propiedad flex-grow con el valor 1.
Fuente: Desafío Latam.

flex-grow especifica un factor de crecimiento. Para entender a qué nos referimos con esto, vamos a probar cambiar la propiedad de flex-grow de solo 1 de los elementos.

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

```
    background-color: aqua;
    border: solid 1px black;
    margin: 1px;
    flex-grow: 1;
  }
  .item-2{
    flex-grow: 3;
  }
```

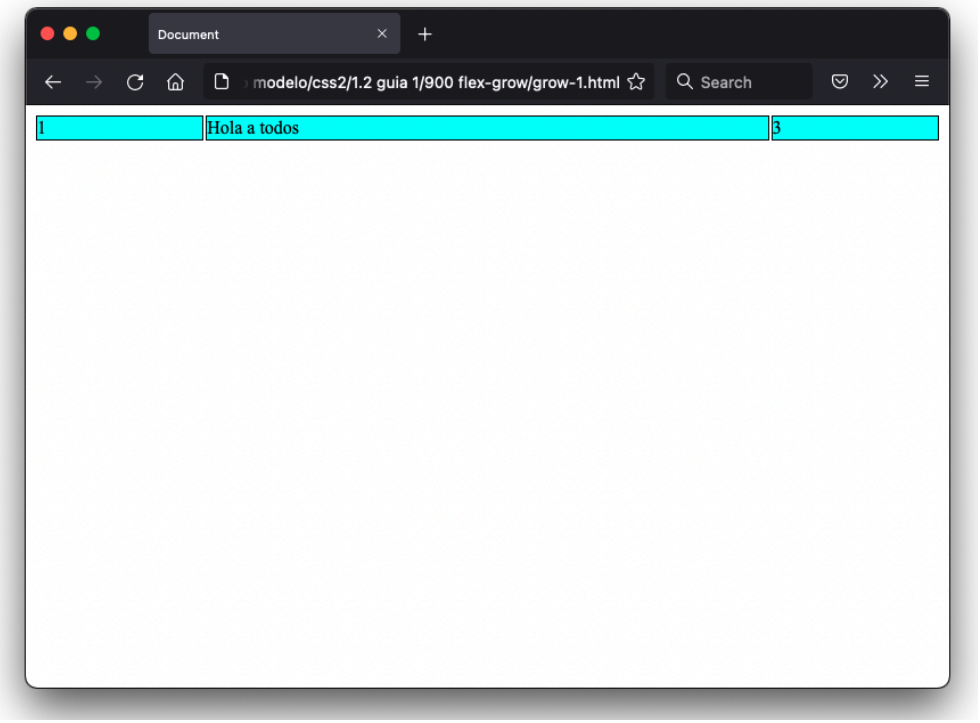

Imagen 23. Screenshot tras aplicar la propiedad flex-grow con el valor 3, el elemento con flex-grow 3 ocupa mucho más espacio que los otros elementos.
Fuente: Desafío Latam.

Alineando el contenido de cards con flexbox-grow

Un caso muy útil de flex-grow son los cards. A continuación veremos de qué tratan.

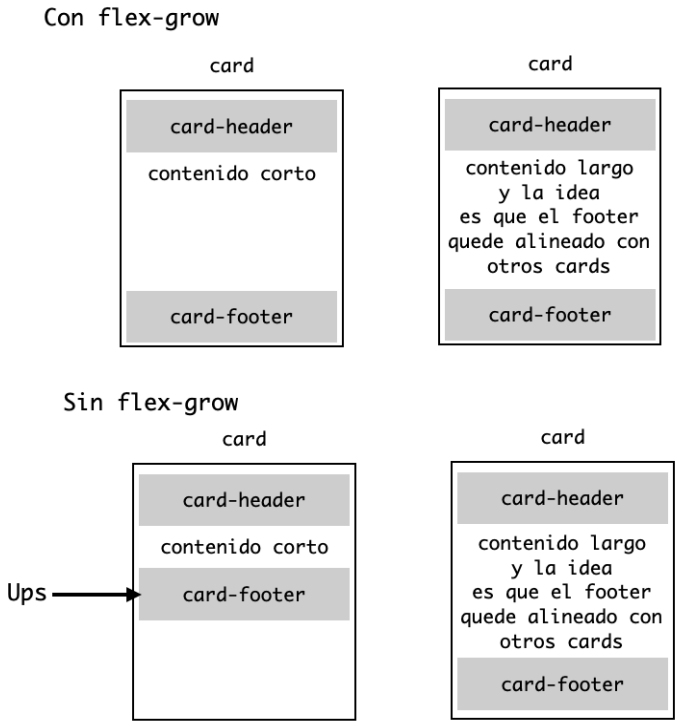

Imagen 24. Usando flex-grow con cards.
Fuente: Desafío Latam.

Para que las cards estén alineadas una al lado de la otra, las envolveremos en otro contenedor flex.

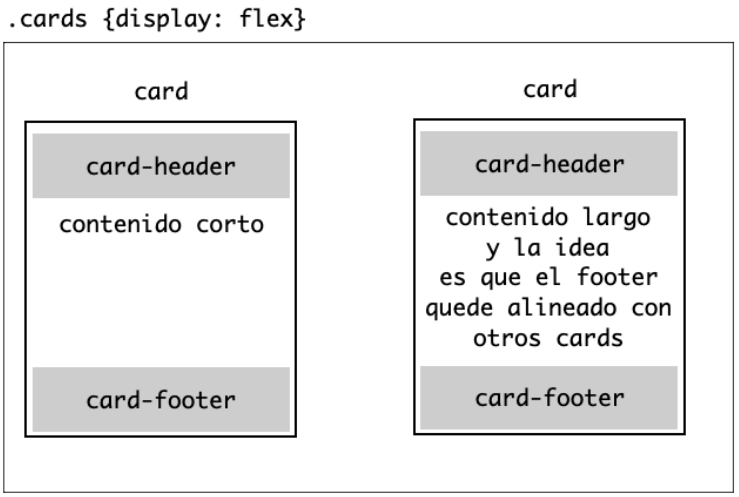

Imagen 25. Los cards deben estar dentro de un contenedor Flex.
Fuente: Desafío Latam.

De esta forma, la tarjeta con más contenido define el alto del contenedor, y el resto se ajusta automáticamente.

Actividad 7

- Crea la carpeta flex-cards y dentro un archivo index.html con la base de html.
- Agrega el siguiente contenido dentro del body del index.html:

```
    <div class="cards">
      <div class="card">
        <div class="card-header">Card 1</div>
        <div class="card-body">
          Lorem ipsum dolor sit amet consectetur
        </div>
        <div class="card-footer">Footer</div>
      </div>
      <div class="card">
        <div class="card-header">Card 2</div>
        <div class="card-body">
          Lorem ipsum dolor sit amet consectetur adipisicing elit. Ea
voluptate
          unde, qui natus amet deleniti esse iste excepturi sit sequi a
nesciunt
          error nostrum numquam, id quidem nisi. Eius, maxime?
        </div>
        <div class="card-footer">Footer</div>
      </div>
    </div>
```

- Agrega el siguiente contenido dentro del CSS:

```
      .cards{
        display: flex;
      }
      .card {
        display: flex;
        flex-direction: column;
        width: 10em;
        border: solid 1px;
        margin: 5px;
        padding: 5px;
      }
      .card-body {
```

```
        flex-grow: 1;
      }
```

- Revisa la página en el navegador.
- En el código, remueve la propiedad flex-grow:1 (o cambia el valor por cero, el cual es el valor por defecto).
- Revisa la página nuevamente.

Ancho de un flex-item

Por defecto en flex, el tamaño inicial de un item-flex depende de su contenido, lo cual puede ser lo que necesitemos en algunos casos, pero en otros no.

```
<!-- index.html -->
  <div class="flex">
   <div class="flex-item item-1">
     1
   </div>
   <div class="flex-item item-2">
      Lorem ipsum dolor sit, amet consectetur adipisicing elit. Vitae ipsa
sed voluptate dolorum, facere magni sapiente quam odio eaque dolore velit.
Dolor saepe atque molestiae eligendi vel maxime libero possimus?
   </div>
   <div class="flex-item item-3">
     3
   </div>
 </div>
```

```
/* style.css */
   .flex{
     display:flex;
   }
   .flex-item{
     background-color: aqua;
     border: solid 1px black;
     margin: 1px;
   }
```

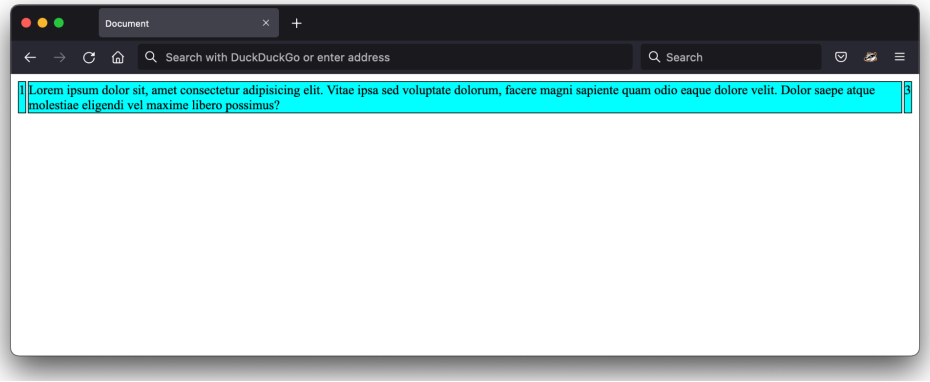

Imagen 26. Screenshot mostrando que el ancho mínimo depende de la cantidad de contenido.
Fuente: Desafío Latam.

Si ahora intentamos dividir la página en 3 columnas iguales utilizando flex-grow:1, no obtendremos un resultado distinto al que vemos sobre los flex-items porque el tamaño de la columna se calcula en función de su contenido.

```
/* style.css */
   .flex{
     display:flex;
   }
   .flex-item{
     background-color: aqua;
     border: solid 1px black;
     margin: 1px;
     flex-grow: 1;
   }
```

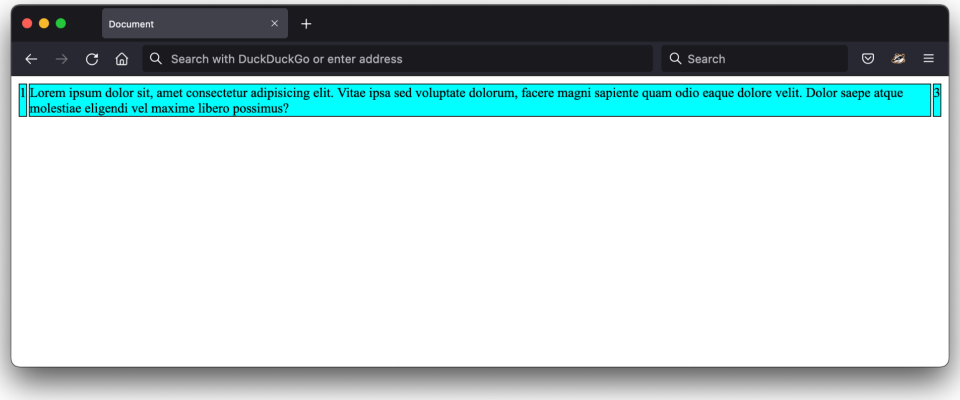

Imagen 27. Screenshot mostrando que flex-grow es ignorado producto de que el tamaño de la columna se calcula en función de su contenido.
Fuente: Desafío Latam.

Aquí tenemos dos opciones para evitar esto. Podemos definir un ancho sobre la columna, por ejemplo:

```
.item2{width: 50%}
```

O podemos utilizar la propiedad flex-basis.

Flex-basis

Flex-basis nos permite definir el tamaño inicial de un flex-item:

```
/* style.css */
   .flex{
     display:flex;
   }
   .flex-item{
     background-color: aqua;
     border: solid 1px black;
     margin: 1px;
     flex-grow: 1;
     /* width: 50%; */
     flex-basis: 50%;
   }
```

También podemos dejar el valor de flex-basis como cero, de esta forma le damos espacio a crecer en la proporción indicada por flex-grow.

```
/* style.css */
   .flex{
     display:flex;
   }
   .flex-item{
     background-color: aqua;
     border: solid 1px black;
     margin: 1px;
     flex-grow: 1;
     /* width: 50%; */
     flex-basis: 0;
   }
```

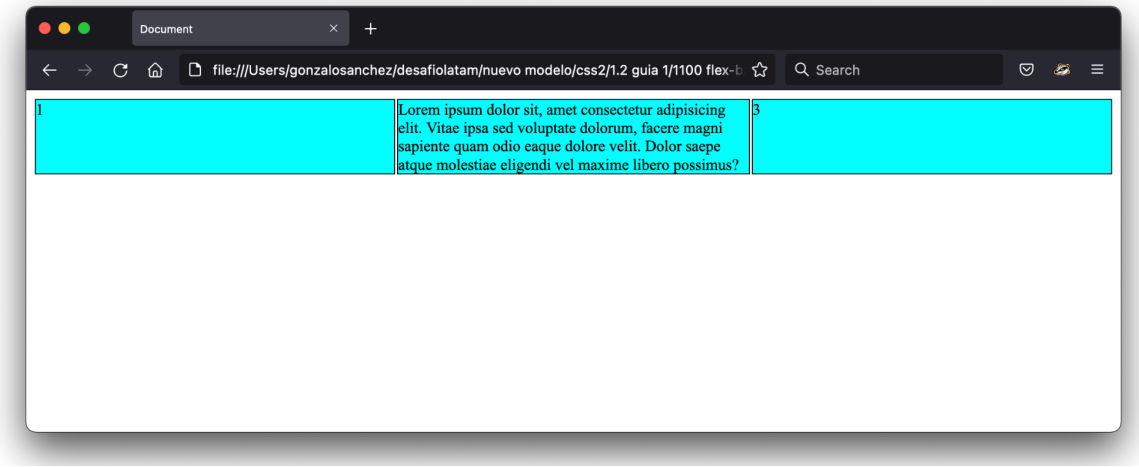

Imagen 28. Screenshot mostrando que podemos resolver el problema del ancho de las columnas con flex-basis.
Fuente: Desafío Latam.

Podemos lograr lo mismo con width: 0

Flex-basis vs width

Entonces, ¿Cuál de los dos utilizamos, flex-basis o width?

En general, cuando trabajamos con flexbox deberíamos ocupar flex-basis por los siguientes motivos:

1. Flex-basis tiene prioridad sobre width, por lo que si definimos ambas flex-basis dominará.
2. Flex-basis no es exactamente lo mismo que width, ya que el tamaño se define en función del eje principal, o sea cuando se trabajó con flex-direction:column, entonces el tamaño se calculará en función del alto.
    a. Cuando realicemos diseños responsivos con media queries será más cómodo definir el tamaño en función de flex-basis.

Flex-shrink

Así como flex-grow permite definir un factor de crecimiento que le permite ocupar proporcionalmente todo el espacio disponible a los flex-items; flex-shrink define un factor de decrecimiento en caso de que no haya espacio suficiente.

Por defecto, flex-shrink tiene un valor 1, esto quiere decir que en caso de no haber espacio suficiente, los flex-items se reducen para ajustarse al contenedor.

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

```
  .flex{
    display:flex;
    width: 500px;
    border: solid 1px;
  }
  .flex-item{
    background-color: aqua;
    border: solid 1px black;
    margin: 1px;
    flex-basis: 200px;
    flex-shrink: 1;
  }
  .item-2{
  }
```

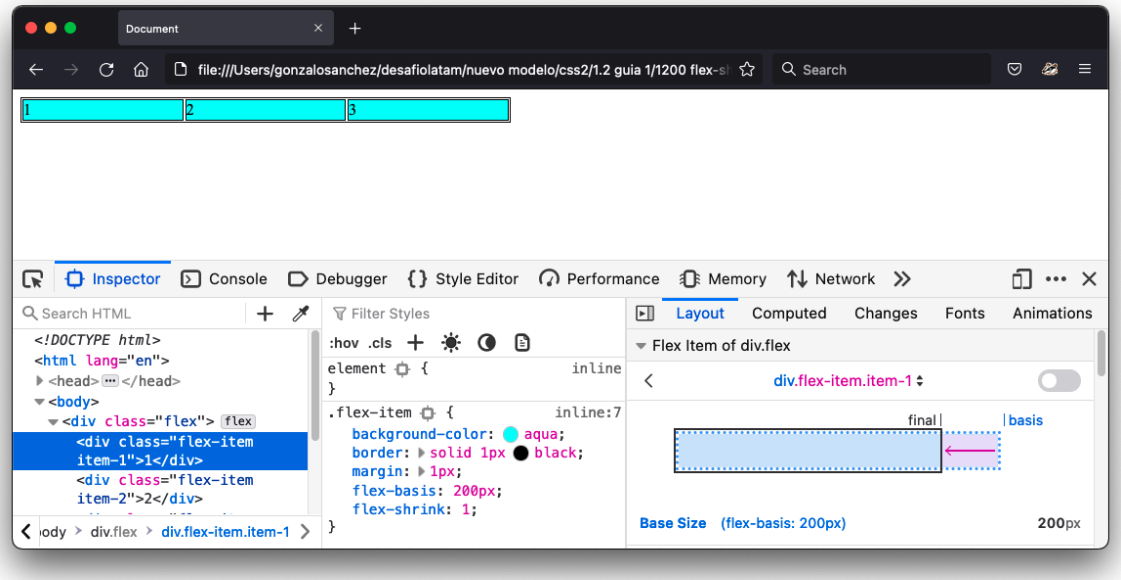

Imagen 29. Screenshot observando la reducción de tamaño producto de flex-shrink.

Fuente: Desafío Latam.

En la página podemos ver que si bien tenemos un contener de 500 píxeles y 3 flex-items de 200 píxeles, los 3 ítems quedaron dentro del contenedor, pero esto se debe a que el tamaño se redujo automáticamente para ajustarse al espacio disponible, esto es gracias a que flex-shrink tiene valor 1.

También podemos ver en el inspector de elementos, que el tamaño final es menor a basis. Esta herramienta es muy útil para poder descubrir donde hay un problema cuando el contenido no se muestra como queremos.

Actividad 8

- Cambia el valor de flex-shrink a 0 y observa qué sucede en la página web.
- Manteniendo flex-shrink 0 para todos los ítems, agrega solo para item-2 flex-shrink: con el valor 1. Explica qué sucede.
- Define como base para todos los flex-items la propiedad flex-shrink con el valor 1, para el ítem 2 ajústalo en 4.
- Observa el resultado en la página web.

Flex 3x1. flex-grow, flex-shrink, flex-basis

Podemos definir grow, shrink y basis con una única propiedad flex.

Sobre un flex-item agregaremos: flex: 1 1 auto;

Esto significa

- flex-grow: 1;
- flex-shink: 1;
- flex-basis: auto;

También podríamos agregar: flex 1 0 0;

Esto significa

- flex-grow: 1;
- flex-shink: 0;
- flex-basis: 0;

Resumen

- Al utilizar flex, tenemos que distinguir nuestro contenedor flex de los flex-items, ya que al agregar la propiedad display:flex el contenedor automáticamente pasa a ser flexible y sus etiquetas hijas flex-items.

- Utilizando flex-wrap:wrap, podemos cambiar el comportamiento típico de flex de forma que los elementos se distribuyan en más de una línea. Esto es útil si queremos hacer una galería de imágenes, considerando que flex-wrap por defecto utiliza el valor nowrap.

- flex-direction nos permite cambiar la orientación del eje principal, de esta forma podemos tener contenedores flexibles que son filas y otros que son columnas.

- Por defecto, la propiedad flex-direction es row, o sea, el eje principal es a lo largo y el transversal a lo alto.

- Podemos alinear el contenido en el eje principal con justify-content.

- Podemos alinear el contenido en el eje transversal con align-items.

- Utilizando alineado de contenido, tanto en el eje transversal como en el vertical, podemos centrar contenido o distribuirlo como necesitemos.

- Podemos tener items-flex que sean también contenedores flex de otros ítems, esto nos permite crear menús con distintas secciones o crear una sección de cards con los footers anidados.

- El tamaño inicial de un item-flex depende de su contenido.