

UD3.3. CSS CON FLEX

Tabla de contenido

UD3.3. CSS CON FLEX	1
1. CONCEPTOS BÁSICOS DE DISPOSICIÓN FLEX	2
1.1. ¿Por qué FLEX?	2
1.2. Presentación de un Ejemplo Sencillo	2
1.3. Valorada qué elementos colocar como cajas flexibles	3
1.4. El Modelo Flexible	5
1.5. Columnas o filas?.....	6
1.6. Delimitar	6
1.7. Propiedades Abreviadas de Flex-flow	8
1.8. Procedimentaldo flexible de elementos flexibles	8
2. ALINEACIÓN HORIZONTAL Y VERTICAL	10
3. ORDENAR LOS ELEMENTOS FLEXIBLES.....	12
3.1. Cajas Flexibles Niadas	13

1. CONCEPTOS BÁSICOS DE DISPOSICIÓN FLEX

1.1. ¿Por qué FLEX?

Durante mucho tiempo, las únicas herramientas fiables con compatibilidad cruzada entre navegadores disponibles para crear diseños CSS fueron cosas como la flotación y el posicionamiento. Están bien y funcionan, pero de alguna manera también limitan bastante y frustran.

Con tales herramientas resulta difícil, si no imposible, conseguir obtener en cualquier forma conveniente y flexible un diseño de página sencillo con unos requisitos como los siguientes:

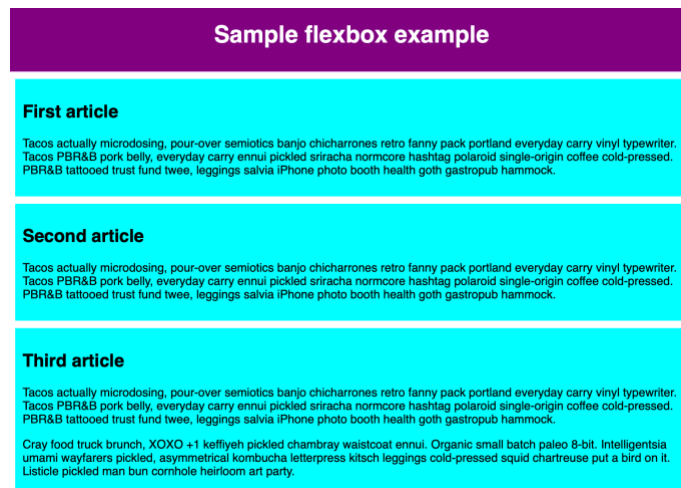
- Centrar verticalmente un bloque de contenido dentro de su elemento padre.
- Hacer que todos los elementos secundarios de un contenedor ocupen una cantidad igual del ancho/alto disponible, independientemente del ancho/alto que haya disponible.
- Hacer que todas las columnas en una compaginación en columnas múltiples adopten la misma altura incluso si contienen cantidades diferentes de contenido.

Como verás en las siguientes secciones, los elementos flexbox facilitan mucho algunas tareas de compaginación. ¡Profundizaremos en esto!

1.2. Presentación de un Ejemplo Sencillo

Te guiaré por una serie de ejercicios para ayudarte a comprender cómo funcionan los elementos flexbox. Para empezar, debes hacer una copia local de la carpeta del AULES de código para esta teoría. El primer archivo que emplearemos será el contenido de la carpeta "PRIMER". Cargalo en un navegador moderno (como Firefox o Chrome) y da un vistazo al código en tu editor de código.

Verás que hay un elemento `<header>` con un encabezado de nivel superior en él, y un elemento `<section>` que contiene tres elementos `<article>`. Los usaremos para crear una compaginación bastante habitual de tres columnas.

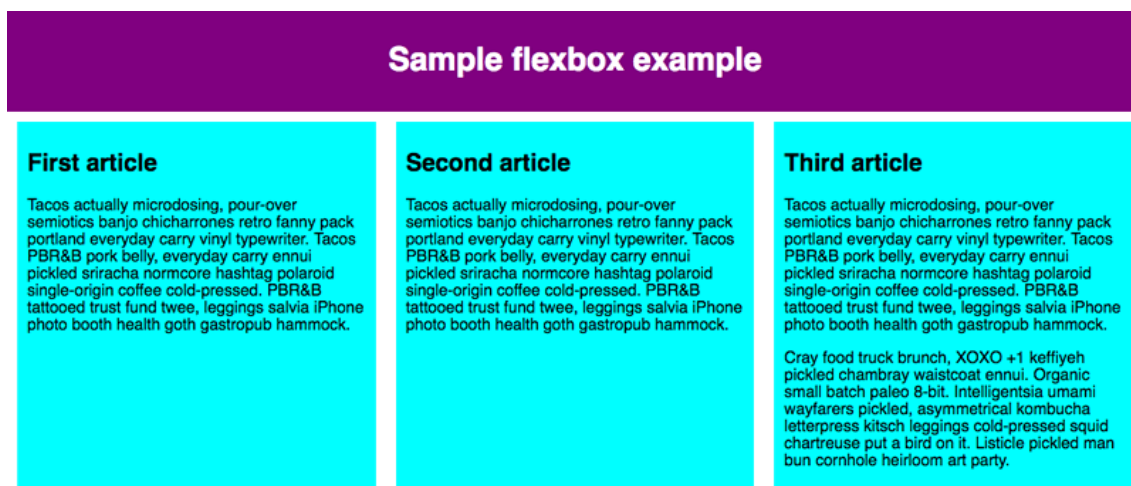


1.3. Valorar qué elementos colocar como cajas flexibles

Para empezar, seleccionaremos qué elementos se presentarán como cajas flexibles. Para ello, establecemos un valor especial de `display` en el elemento padre de los elementos que deseas editar. En este caso, queremos compaginar los elementos `<artículo>`, por lo que lo establecemos en `<section>` (que se convierte en un contenedor flexible):

```
section {  
  display: flex;  
}
```

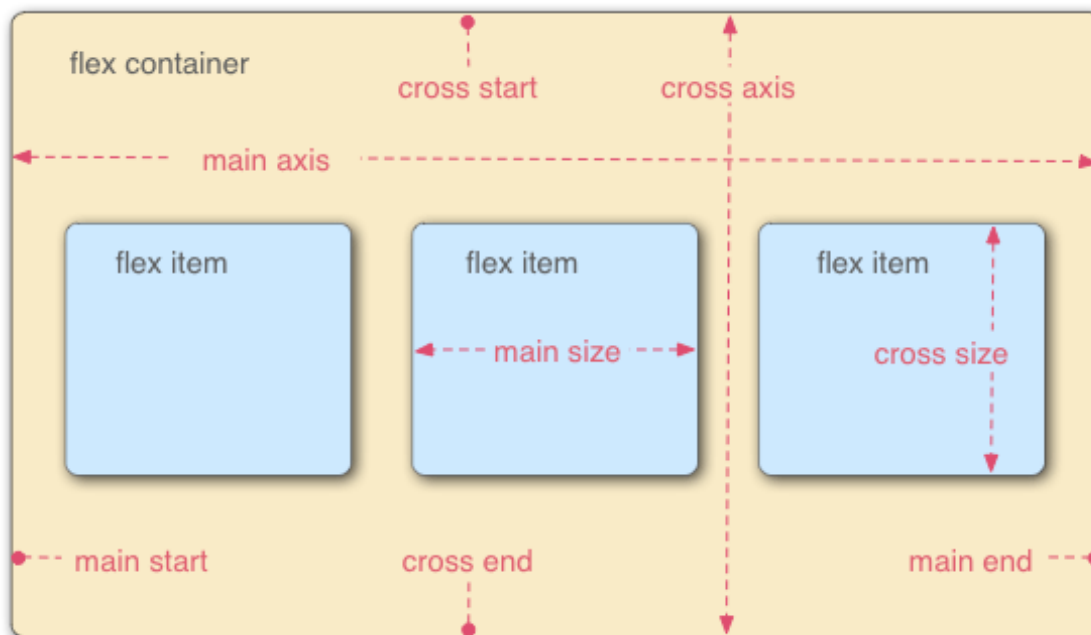
Esto hace que el elemento `<section>` se convierta en **contenedor flex**, y sus hijos en **elementos flexibles**. El resultado de esto debería ser algo así:



Así, esta declaración única nos da todo lo que necesitamos. Increíble, ¿verdad? Tenemos nuestra compaginación en columnas múltiples con columnas de igual tamaño, y todas las columnas tienen la misma altura. Esto se debe a que los valores por defecto que se han asignado a los elementos flexibles (los elementos secundarios del contenedor flexible) están pensados para resolver problemas comunes como este. Veremos más sobre el tema más adelante.

Para que quede claro, reiteramos lo que está sucediendo acá. El elemento al que le hemos dado un valor de display de flex actúa como un elemento a nivel de bloque en términos de cómo interactúa con el resto de la página, pero sus elementos secundarios se presentan como elementos flexibles. La siguiente sección explicará con más detalle qué significa esto. Ten en cuenta también que puede usar un valor de display de inline-flex si desea diseñar los elementos secundarios de un elemento como elementos flexibles, pero hacer que este elemento se comporte como un elemento en línea.

1.4.El Modelo Flexible



Cuando los elementos se presentan como cajas flexibles, se distribuyen con respecto a dos ejes:

- El eje **principal (main axis)** es el eje que corre en la dirección en que se colocan los elementos flexibles (por ejemplo, según se disponen las filas en una página o hacia abajo según se disponen las columnas en una página). El inicio y el final de este eje se denominan **inicio principal (main start)** y **final principal (main end)**.
- El eje **transversal (cross axis)** es el eje que corre perpendicular a la dirección en la que se colocan los elementos flexibles. El inicio y el final de este eje se denominan **inicio transversal (cross start)** y **extremo transversal (cross end)**.
- El elemento padre que tiene establecido `display: flex` (el elemento [<section>](#) en nuestro ejemplo) se llama **contenedor flexible**.
- Los elementos que se presentan como cajas flexibles dentro del contenedor flexible se denominan **elementos flexibles** (son los elementos [<article>](#) de nuestro ejemplo).

Tuve presente esta terminología al avanzar por las secciones posteriores. Si en algún momento te confunde con el uso de estos conceptos, siempre puedes volver atrás a consultarlos.

1.5. ¿Columnas o filas?

Los elementos flexbox proporcionan una propiedad llamada [flex-direction](#) que especifica en qué dirección corre el eje principal (en qué dirección están dispuestos los elementos hijo de un elemento flexbox); por defecto, está establecido en el valor `row`, por lo que se presenta en una fila en la dirección en la que se escribe el idioma predeterminado de tu navegador (de izquierda a derecha, en el caso de un navegador en español).

Afirmó la siguiente declaración a tu regla [<section>](#):

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

Observa que esto vuelve a colocar los elementos en una disposición en columna, al igual que antes de añadir cualquier CSS. Antes de continuar, elimina esta declaración de tu ejemplo.

1.6. Delimitar

Un problema que surge cuando tienes una cantidad fija de ancho o alto en tu diseño es que los hijos de un elemento flexbox eventualmente desbordan el contenedor y rompen el diseño. Puedes crear muchos artículos como en el siguiente ejemplo:

Sample flexbox example							
First article	Second article	Third article	Fourth article	Fifth article	Sixth article	Seventh article	Eighth article
Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth	Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth	Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth	Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth	Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth	Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth	Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth	Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth

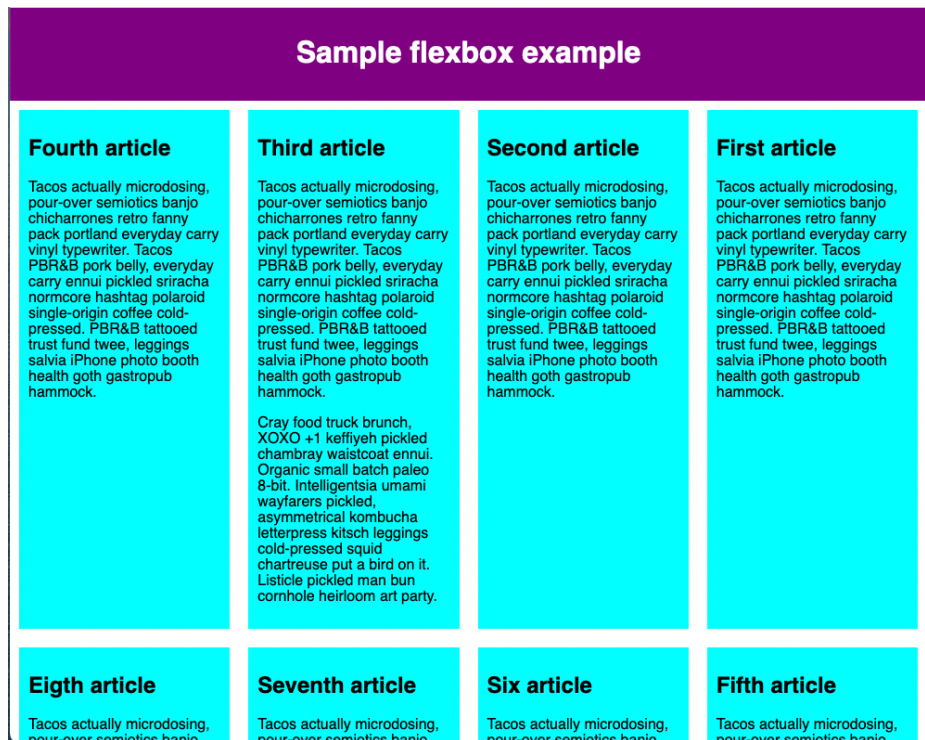
Acá vemos que los elementos hijo se enojan de su contenedor. Una manera de solucionar esto es añadir la siguiente declaración a tu regla `<section>`:

```
flex-wrap: wrap;
```

Afirmó también la siguiente declaración a tu regla `<artículo>`:

```
flex: 200px;
```

Pruebalo; observa que al haberlo incluido el aspecto de la compaginación resulta mucho más agradable:



Ahora hay varias filas y en cada fila caben tantos elementos hijo de un elemento flexbox como sean necesarios, y cualquier desbordamiento hace saltar el elemento hacia la línea siguiente. La declaración `flex: 200px` que hemos establecido en los artículos significa que cada uno tendrá al menos 200 px de ancho; discutiremos esta propiedad con más detalle más adelante. Observa también que los últimos elementos hijo de la última fila se engrandecen hasta rellenar toda la fila.

Pero acá podemos hacer mucho más. En primer lugar, cambia el valor de tu propiedad `flex-direction` a `row-reverse`; ahora verás que todavía tienes tu compaginación en varias filas, pero empieza desde la esquina opuesta de la ventana del navegador y fluye al revés.

1.7. Propiedades Abreviadas de Flex-flow

En este punto vale la pena señalar que hay una propiedad abreviada para [flex-direction](#) y [flex-wrap](#): [flex-flow](#). Así, por ejemplo, puedes reemplazar:

```
flex-direction: row;  
flex-wrap: wrap;
```

con

```
flex-flow: row wrap;
```

1.8. Procedimiento flexible de elementos flexibles

Volvamos ahora a nuestro primer ejemplo y veamos cómo podemos controlar qué proporción de espacio ocupan los elementos flexibles.

Primero, afirma la siguiente regla al final de tu CSS:

```
article {  
  flex: 1;  
}
```

Este es un valor de proporción sin unidades que especifica la cantidad de espacio disponible sobre el eje principal que ocupa cada elemento flexible. En este caso, damos a cada elemento [artículo](#) un valor de 1, lo que significa que todos ocuparán una cantidad igual del espacio libre restante después de que se hayan establecido elementos como el área de relleno y el margen. Es una proporción, lo que significa que dar a cada elemento flexible un valor de 400000 tendría exactamente el mismo efecto.

Ahora afirmó la siguiente regla debajo de la anterior:

```
article:nth-of-type(3) {  
  flex: 2;  
}
```

Al actualizar verás que el tercer [artículo](#) ocupa ahora el doble del ancho disponible que los otros dos; ahora hay cuatro unidades de proporción disponibles en total. Los primeros dos elementos flexibles tienen una cada uno, por lo que ocupan 1/4 del espacio disponible cada uno. El tercero tiene dos unidades, por lo que ocupa 2/4 del espacio disponible (o 1/2).

También puedes especificar un valor de tamaño mínimo dentro del valor flexible. Actualiza las reglas para tu artículo de la siguiente manera:

```
article {  
  flex: 1 200px;  
}  
  
article:nth-of-type(3) {  
  flex: 2 200px;  
}
```

Esto establece básicamente que «a cada elemento flexible se le da primero 200px del espacio disponible. Tras esto, el resto del espacio disponible se reparte de acuerdo con las unidades de proporción». Actualiza y observa de qué modo se reparte ahora el espacio.

Sample flexbox example

First article

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

Second article

Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

Third article

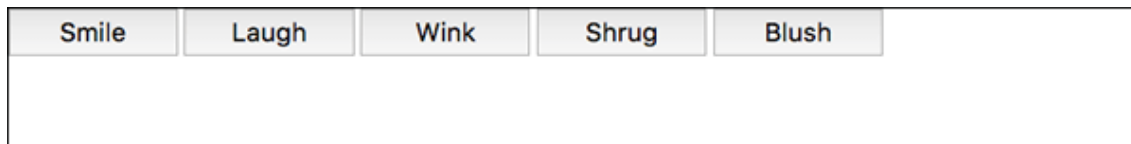
Tacos actually microdosing, pour-over semiotics banjo chicharrones retro fanny pack portland everyday carry vinyl typewriter. Tacos PBR&B pork belly, everyday carry ennui pickled sriracha normcore hashtag polaroid single-origin coffee cold-pressed. PBR&B tattooed trust fund twee, leggings salvia iPhone photo booth health goth gastropub hammock.

Cray food truck brunch, XOXO +1 keffiyeh pickled chambray waistcoat ennui. Organic small batch paleo 8-bit. Intelligentsia umami wayfarers pickled, asymmetrical kombucha letterpress kitsch leggings cold-pressed squid chartreuse put a bird on it. Listicle pickled man bun cornhole heirloom art party.

El valor real del elemento flexbox se puede ver en su flexibilidad/adaptabilidad: si cambias el tamaño de la ventana del navegador o afianzas otro elemento <artículo>, el diseño sigue funcionando bien.

2. ALINEACIÓN HORIZONTAL Y VERTICAL

También puedes usar las funciones de los elementos flexbox para alinear elementos flexibles sobre el eje principal o transversal. Exploramos este aspecto a partir de un ejemplo nuevo dentro de la carpeta "SEGUNDO" que convertiremos en una barra de herramientas/botones ordenada y flexible. En este momento puedes ver una barra de menú horizontal, con algunos botones pegados en línea en la esquina superior izquierda.



Primero, haz una copia local de este ejemplo.

Ahora, dijo en la parte inferior del CSS del ejemplo lo siguiente:

```
div {  
  display: flex;  
  align-items: center;  
  justify-content: space-around;  
}
```

Actualiza la página y observa que los botones ahora están convenientemente centrados, horizontal y verticalmente. Lo hemos hecho a partir de dos propiedades nuevas.

align-items controla donde se sitúan los elementos flexibles en el eje transversal.

- Por defecto, el valor es `stretch`, que ensancha todos los elementos flexibles para rellenar el elemento primario en la dirección del eje transversal. Si el elemento padre no tiene un ancho fijo en la dirección del eje transversal, todos los elementos flexibles son tan largos como los elementos flexibles más largos. Así es como nuestro primer ejemplo obtuvo columnas de igual altura por defecto.
- El valor `center` que utilizamos en nuestro código anterior mantiene las dimensiones intrínsecas de los elementos pero los centra sobre el eje transversal. Es por ello que los botones de nuestro ejemplo ahora están centrados verticalmente.
- También puedes tener valores como `flex-start` y `flex-end`, que alinean todos los elementos al inicio y al final del eje transversal, respectivamente.

Puedes anular el comportamiento align-items para elementos flexibles individuales aprobándoles la propiedad align-self. Por ejemplo, dijo a tu CSS lo siguiente:

```
button:first-child {  
  align-self: flex-end;  
}
```

Observa qué efecto tiene esto y retirándolo de nuevo al acabar.

justify-content controla donde se sitúan los elementos flexibles sobre el eje principal.

- El valor por defecto es `flex-start`, que asiente todos los elementos al comienzo del eje principal.
- Puedes usar `flex-end` porque se asienten al final.
- `center` también es un valor de `justify-content` (para alinear contenido), que asiente los elementos flexibles sobre el centro del eje principal.
- El valor `space-around` que hemos usado antes es útil porque distribuye todos los elementos de manera uniforme sobre el eje principal y deja un poco de espacio en cada extremo.
- Hay otro valor, `space-between`, que es muy similar a `space-around`, pero no deja espacio en los extremos.

Te anime a jugar con estos valores para ver cómo funcionan antes de continuar.

3. ORDENAR LOS ELEMENTOS FLEXIBLES

Los elementos flexbox también tienen una función para cambiar el orden de disposición de los elementos flexibles, sin que ello afecte su orden en el código fuente. Esto es algo que resulta imposible de hacer con los métodos de compaginación tradicionales.

El código para ello es simple: afirme el siguiente CSS en el código de ejemplo de la barra de controles:

```
button:first-child {  
  order: 1;  
}
```

Actualiza, y observa que el botón «Smile» se ha movido al final del eje principal. Observamos cómo funciona esto con un poco más de detalle:

- Por defecto, todos los elementos flexibles tienen un valor [order](#) de 0.
- Los elementos flexibles que tengan establecidos los valores de orden más altos aparecerán más tarde en el orden de visualización que los elementos que tengan valores de orden más bajos.
- Los artículos flexibles con el mismo valor de orden aparecerán en su orden de origen. Luego, si hay cuatro elementos cuyos valores de orden se han establecido en 2, 1, 1 y 0, respectivamente, su orden de visualización será 4º, 2º, 3º y, finalmente, 1º.
- El tercer elemento aparece después del segundo porque tiene el mismo valor de orden y está después de él en el orden del código de origen.

Puedes establecer valores de orden negativos para que los elementos aparecieran antes que los elementos establecidos con el valor 0. Por ejemplo, puedes hacer que el botón «Blush» aparezca al comienzo del eje principal utilizando la siguiente regla:

```
button:last-child {  
  order: -1;  
}
```

3.1. Cajas Flexibles Niadas

Los elementos flexbox permiten crear algunos diseños de página bastante complejos. Es perfectamente aceptable configurar un elemento flexible para que también sea un contenedor flexible, de manera que los elementos secundarios también se disponen como cajas flexibles. Da un vistazo al contiguo de la carpeta "TERCER".



El HTML para ello es bastante simple. Hay un elemento `<section>` que contiene tres elementos `<artículo>`. El tercer elemento `<artículo>` contiene tres elementos `<div>`:

```
section - article
  article
    article - div - button
      div button
      div button
      button
      button
```

Veamos el código que hemos usado para esta compaginación.

En primer lugar, configuramos los elementos secundarios de `<section>` porque se presentan como cajas flexibles.

```
section {
  display: flex;
}
```

A continuación, establecemos algunos valores flexibles en los elementos `<artículo>`. Presta una atención especial a la segunda regla: configuramos el tercer elemento `<artículo>` porque sus hijos también se presentan como elementos flexibles, pero esta vez los disponemos en una columna.

```
article {  
  flex: 1 200px;  
}  
  
article:nth-of-type(3) {  
  flex: 3 200px;  
  display: flex;  
  flex-flow: column;  
}
```

A continuación, seleccionamos el primer elemento `<div>`. Primero usamos `flex:1 100px;` para darle una altura mínima efectiva de 100 px, luego configuramos sus elementos secundarios (los elementos `<button>`) porque también se presentan como elementos flexibles. Acá los colocamos en una fila que los delimita y los alineamos en el centro del espacio disponible, como hicimos en el ejemplo del botón individual que vimos antes.

```
article:nth-of-type(3) div:first-child {  
  flex: 1 100px;  
  display: flex;  
  flex-flow: row wrap;  
  align-items: center;  
  justify-content: space-around;  
}
```

Finalmente, establecemos un tamaño para el botón, pero lo más interesante es que le demos un valor flexible de 1 auto. Esto tiene un efecto muy interesante, que puedes observar si cambias el tamaño del ancho de la ventana de tu navegador. Los botones ocuparán tanto espacio como puedan y se asentarán puntos en la misma línea como quepan, pero cuando ya no quepan con comodidad en la misma línea, saltarán de línea y crearán líneas nuevas.

```
button {  
  flex: 1 auto;  
  margin: 5px;  
  font-size: 18px;  
  line-height: 1.5;  
}
```