Sabemos que la propiedad justify-content: space-between hace que el espacio que queda en nuestro elemento padre se distribuya entre los hijos, pero por supuesto ese no es el único valor de la propiedad justify-content. Por ejemplo, ¿de qué otras formas podemos distribuir este espacio?

¿Cuál es el valor de la propiedad justify-content para estas distribuciones de espacio?

Podemos distribuir los elementos dentro del padre de varias maneras, podemos por ejemplo:

Poner todo el espacio a la izquierda, lanzando el contenido a la derecha con justify-content: flex-end.

Poner todo el espacio a la derecha, lanzando el contenido a la izquierda con justify-content: flex-start (que es el modelo).

Poner todo el espacio a la izquierda y a la derecha, lanzando el contenido al centro con justify-content: center.

Poner todo el espacio entre los elementos como vimos antes con justify-content: space-between.

Y una posibilidad muy interesante es también poner espacio alrededor de los elementos. Para ello podemos utilizar justify-content: space-around.

flex: 0=**(flex-grow)** 1=**(flex-shrink)** 30%=**(flex-basis)**

**flex-basis** tambien se puede definir en pixeles

La propiedad flex-basis sirve para definir un ancho para el elemento en caso de que el flex container esté con flex-direction: column.

flex-basis: 500px;

¡Alternativa correcta! La propiedad flex-basis sirve para definir una anchura o altura para el flex item. Se el flex container tiene flex-direction: column, el flex-basis en el flex item servirá para definir una height. En este caso el flex-direction: row funciona como un width.

La propiedad flex-grow vista en el ejercicio anterior ayuda mucho si queremos que un elemento ocupe todo el ancho restante del flex container.

Por ejemplo, si tenemos:

Elemento 1: 200 px.

Elemento 2: 200 px.

Espacio vacío que restó del flex container: 600 px.

Total: 1000 px.

Se ponemos flex-grow: 1 en el primer elemento, este pasa a tener 800 px de ancho, es decir:

Espacio vacío + Elemento 1: 800 px.

Y el segundo elemento sigue teniendo 200 px de ancho.

Ahora, si ponemos flex-grow: 1 en ambos elementos, ¿Qué pasaría? ¿Cuál seria el tamaño del elemento 1? Y el elemento 2?

Imaginemos que en nuestra página web tenemos un video de YouTube, imaginemos que aunque el usuario reduzca la pantalla, nuestro vídeo no debe disminuir de tamaño. ¿Cómo podemos hacerlo?

.videoSobre-video {

flex-shrink: 0

}

¡Alternativa correcta! ¡Es perfecto! Simplemente ponemos flex-shrink: 0 en el elemento que no queremos que se reduzca.

Para ello podemos poner la propiedad flex-direction: column, que hace que los elementos queden uno debajo del otro. Este es el aspecto más importante de la capacidad de respuesta de flexbox.

Considere el código HTML siguiente.

<main class="flexContainer">

<div class="flexItem firstRow"></div>

<div class="flexItem firstRow"></div>

<div class="flexItem firstRow"></div>

<div class="flexItem firstRow"></div>

</main>

Ahora considere que estamos en un contexto flex. Y que cada flexItem tiene 200 px de width.

CSS

.flexContainer {

display: flex;

}

.flexItem {

width: 200px;

}

1) Cuando el ancho de la pantalla llega a 800 px, los elementos mantienen 200 px de ancho cada uno llenando todo el espacio de la pantalla, sin embargo, qué pasa con el ancho de los flexItem cuando disminuimos la pantalla a por ejemplo 600 px.

2) ¿Qué ocurre también con su ancho cuando aplicamos el siguiente CSS a nuestro código?

.flexItem:first-child {

flex-shrink: 2;

}

Sugerencia: Crea un archivo HTML y CSS, escriba este código y haga las pruebas, ¡también use diferentes valores!

VER OPINIÓN DEL INSTRUCTOR

Opinión del instructor

1) Cuando disminuimos la pantalla a 600 px los flex-item disminuyen su tamaño por igual entre ellos, es decir, como la pantalla disminuyó 200 px y tenemos 4 flex items, cada uno disminuyó 50 px. Así que cada flex-item quedó con 150 px al final.

2) Cuando ponemos flex-shrink: 2 en el primero, este se reduce de forma diferente a los demás elementos. Las cuentas que se producen son similares a las que se dan en el flex-grow.

En primer lugar el navegador toma el total de flex-shrink que nuestros flex items tiene, ya que tenemos por defecto flex-shrink: 1 en los flex items, la cuenta se ve así:

2 + 1 + 1 + 1 = 5`. Eso es porque a el primer le ponemos `flex-shrink:2

Como hemos disminuido 200 px del tamaño total, el navegador toma este valor y lo divide por el total de flex-shrink. Queda: 200 px : 5 = 40 px.

De esta manera el navegador toma 2 partes del primer elemento 200px - 80px = 120px y toma 1 parte de los otros, es decir: 200 px - 40 px = 160 px.

En los juegos encontrarás otra propiedad llamada align-self. Esta propiedad es aplicada al flex item y hace que se alinee individualmente.

¿Recuerdas de la propiedad align-items que pusimos en el flex container? El align-self hace lo mismo, sólo que alinea un solo elemento y se coloca en el flex item.