Continuamos viendo las posibilidades, enormes posibilidades que nos brinda el modelo de caja flexible. Esta herramienta de CSS llamada "Flex Box", que ha venido para solucionarlos la vida. Ahora, ¿cómo se distribuye el espacio de un contenedor flexible entre los distintos ítems? o cómo, digamos, cómo se asigna, cómo cada uno se ubica, ¿todos ocupan lo mismo?, ¿puede ser variable?, ¿qué ocurre cuando se agranda y qué ocurre cuando, al contrario, se nos reduce el espacio disponible?, ¿cómo se comportan los elementos dentro de un contenedor flexible? Es un mundo maravilloso de aprender, así que vamos a verlo.

Seguimos viendo en el código y en la pantalla bueno como quedamos en la clase anterior aquí tenemos nuestro diseño, recuerden, esta línea que tenemos comentada es porque, digamos, se contrapone su efectividad “align items”, con “align content”, si no les quedó claro aún cuál es la diferencia entre una y otra, voy a volver a mostrarles. Fíjense que con “align content”, “center”, que funciona cuando tenemos más de una fila, por lo tanto tenemos “flex wrap” habilitado en el modo “wrap”, me está alineando al centro de todo el contenedor, a diferencia de “align items” que, si la descomentamos, van a ver lo que ocurre, me va a alinear al centro, pero de cada fila, noten ustedes, ¿llegan a darse cuenta?, si aquí tuviésemos alguna línea divisoria entre cada una de las filas del contenedor, podrían notar que estos contenedores ahora están al centro de cada fila, tal vez así les quede más claro, vamos a ponerlas al inicio, o al final, si volvemos a ponerlas al centro, quedan al centro, pero es un centro relativo, es el centro de cada una de las filas. ¿Ahora les queda más claro? Creo que sí.

Miren otra vez “align content” al centro, me va a alinear el contenido, es decir, todas, todos los “div”, pero sin tener en cuenta las filas relativas, sino, todo el contenedor, será al centro del contenedor. Si le ponemos “flex end”, no será al final de cada fila, será al final del contenedor y así sucesivamente.

Espero que haya quedado ahora un poco más claro, no se preocupen, es una de las preguntas que más repiten los estudiantes, ¿cuál es la diferencia entre “align content” y “align items”? De hecho, bueno, hay muchas personas que ya no son estudiantes, que están diseñando y todavía no lo recuerdan.

Bien, de hecho, a mí me llevo bastante tiempo recordarlo y comprenderlo cabalmente, así que no se hagan problemas si tienen que repasar esto un par de veces.

Okey. Hasta ahora vemos como las cajas están fluyendo dentro de nuestro contenedor, pero lo están haciendo en la disposición o la dirección por defecto, que es en filas. Si nosotros quisiéramos que estos elementos fluyesen en columnas, tendríamos que voltear la caja contenedora, esto sería como darle vuelta a nuestro contenedor, a nuestro “div” contenedor, ¿bien?, darle un empujoncito de lado, esto podemos hacerlo con la propiedad “flex direction” y vamos a indicar una dirección en columna.

La dirección por defectos en fila y ahora nuestros ítems, se están ordenando en columnas miren uno, dos, tres, bla, bla, bla, hasta el siete. Cuando se termina el espacio del contenedor, crea una nueva columna, en vez de crear una nueva fila como hacía hoy, crea una nueva columna. ¿Okey? Y así hasta terminar, hasta acomodar todos los elementos y este 15 quedó ahí triste, solitario y final porque, bueno pues, no hay otros elementos que ocupen el espacio de esa columna. Aquí noten lo siguiente, si queremos ordenar con “justify content”, “center”, cuando lo indicábamos en una fila

se movía horizontal, ahora lo hará vertical, ¿por qué?, porque la caja se ha volteado, es decir, “justify content” sigue ordenando los los elementos, sigue alineando los elementos en el eje principal, pero al voltear la caja el eje principal que era horizontal, ahora es vertical, es decir, ¿ha cambiado algo?, no, no ha cambiado nada. Simplemente la caja está volteada, los ejes giran, porque están integrados a la caja.

Fíjense que, si aquí ponemos en vez de“justify content center”, “flex start”, solamente se mueve el ítem 15, porque ahora esta es la dirección en la que ordena“justify content”, entonces, alguno estará razonando: si “justify content”, que antes trabajaba en horizontal, es decir, en el eje principal, ahora que volteé la caja, entonces, para alinear en ese eje, mejor dicho, los ejes no cambiaron, siguen siendo los mismos, pero tendemos a pensar en horizontal y vertical, porque nos cuesta un poco internalizar el concepto de que no hay horizontal y vertical en "Flex Box", sino que hay un eje principal y un eje en cruz, o secundario, y que si cambiamos el orden o, mejor dicho, la dirección del contenedor, bueno, intrínsecamente esos ejes también han girado.

Pero entiendo que a veces es un poco difícil internalizar el concepto, así que si siguen pensando en horizontal, sepan que aquí, ahora no está el eje principal, está el eje secundario. ¡Ajá! Entonces, “align content” hará lo que antes hacía “justify content”. Si ponemos “align content” “flex end”, ¡boom!, al final. Si lo ponemos al principio bueno, pues, al principio y, ¿funcionarán “space between”, y “space around”, y“space evenly”?, por supuesto, que funciona. “Space evenly”, “space around”. Bueno, no sé si les mostré “space between” pero, ahí está, ¿correcto? Perfecto.

Bueno, muy bien, ahí tenemos entonces otro concepto que quería enseñarles, podemos girar la caja cuando sea necesario para que nuestros, nuestros elementos fluyan de otra manera. ¿Qué pasa si no tenemos esta propiedad habilitada? Tendremos el mismo problema que teníamos en la otra dirección, es decir, en la dirección “row”. Los elementos empiezan a achicarse, a resignar su espacio original para tratar de caber dentro del contenedor. Por eso, también, aunque estemos en “flex direction row” o “column”, o como sea, en cualquiera de las dos vamos a utilizar “flex wrap”, si queremos que los elementos fluyan naturalmente y conserven su tamaño.

Hablando de tamaño, lo que viene a continuación les va a interesar mucho también, supongan que tenemos solamente cuatro contenedores y bueno, no importa, podemos mostrar esto con la dirección por defecto o con cualquier otra no hay problema vamos a empezar a quitar propiedades que ahora no vienen al caso.

Para que sea más sencillo para ustedes entender lo que está pasando, bien, ahí tenemos simplemente el contenedor, con un poquito de espacio entre cada columna, el “road map” aquí no le hace daño a nadie, pero no es necesario, porque no hay más de una fila, por lo tanto no se aplica y vamos a ver cómo se están dividiendo el espacio cada una de estas cajas por defecto, he dicho que cada uno de estos de estos “div” tiene 150 píxeles, lo estoy indicando aquí, con este selector, al decir cada día que está dentro de otro “div”, o sea, que está dentro del container, esto podría haberlo expresado también así, sería igual y tal vez les quede más claro, bien, cada uno tiene 150 píxeles.

Ahora, supongamos que no queremos esto, que queremos indicar las propiedades. Vamos a dejar sí el alto, pero el ancho lo vamos a indicar en cada una de los, de los “div”, en cada uno de los ítems.

Miren, tenemos una propiedad muy interesante que es la propiedad “flex grow”. “Flex grow” se utiliza, es decir, define la capacidad que tiene un ítem flexible de crecer, si es necesario, ¿okey? Esto acepta a un valor que por defecto es cero. Vamos a ponerlo en uno, vamos a hacer esto para cada uno de los ítems y luego veremos qué es lo que ocurre. Bien, ¿qué está ocurriendo aqu?, al tener cada uno de los ítems la habilidad de crecer en una unidad y todos tienen la habilidad de crecer en una unidad, están tomando el tamaño completo de su padre contenedor y se lo dividen en partes iguales. ¿Cuál es el valor por defecto de “flex grow”? Cero. Entonces, supongan que el primero tiene esto se te ha dado en cero, o que directamente no tiene esa propiedades y, digamos, podemos borrarlo también, no es lo mismo, bueno, pues, no crece, no le hemos dado la posibilidad de crecer y, ¿por qué ocupa de este espacio? Bueno, porque el espacio de este “div” está dado por el número uno, por el elemento que tiene aquí dentro. También podría ser, por ejemplo, perfectamente posible que este “div” tuviera un ancho de 50 píxeles, ¿okey?, y ahora tiene 50, pero sólo 50, no crece, okey, bien. ¿Qué ocurre si, por ejemplo el ítem dos tiene un “flex grow” de dos? Van a notar ustedes que crece más que tres y que cuatro, ya dijimos que uno no va a crecer, va a mantener sus 50 pixeles, ¿lo notan?

Hemos dicho que la propiedad “flex grow” del “ítem dos”, es de dos. En tanto que la de tres y cuatro es de uno. Bien, y así sucesivamente. Aquí podemos probar con un tres, bueno, en fin, con el número que ustedes prefieran. Okey.

Vamos a avanzar un poquito más, ¿qué ocurre si tenemos, además estela que resolver cómo se comportan cuando disminuye el tamaño? Bueno, podríamos indicar por ejemplo una propiedad que es antagónica de “flex grow”, es decir, que hace lo mismo, pero en sentido contrario, es el “flex shrink”. “Flex shrink” indica cuánto va a reducir su tamaño, si es necesario un contenedor que está dentro de un contenedor flexible, o sea, un elemento dentro de un contenedor flexible, por caso, podríamos decir que la propiedad “flex shrink” de nuestro ítem número dos, sea de dos, la propiedad por defecto es uno. Okey, la propiedad por defecto es uno, entonces, noten lo que va a ocurrir ahora, hay una suerte de justicia distributiva, porque lo podríamos mostrar un poco más claro, así, porque si bien nuestro hábitat en dos reclama más espacio para crecer, cuando empezamos a decrecer, okey, también es el que más se reduce proporcionalmente, ¿si?, proporcionalmente es el que más se reduce, tal vez aquí cueste verlo un poco, pero podríamos forzarlo, podríamos forzarlo poniéndole, por ejemplo, un “flex shrink” de cero al ítem tres y un “flex shrink” de cero, al ítem dos. Y lo que va a ocurrir ahora es, bueno, que cuando se reducen de tamaño, se reducen menos proporcionalmente que ítem dos, okey. Éstos son números, tal vez a veces un poco difíciles de ver aquí, porque son números muy pequeños, podríamos forzarlo poniendo números más grandes, ¿okey?, pero por ahora vamos a, también debo comentarles, que los números negativos no son válidos aquí, solamente podemos poner números positivos.

Podríamos indicar, por ejemplo, también una propiedad “flex bases”, es decir, ponerle una base a cada uno de nuestros ítems. La base definiría el tamaño por defecto de un elemento antes de que se redistribuya el espacio que quede remanente.

Vamos a ver qué ocurre aquí si, por ejemplo, vamos a quitar los “flex shrink” y vamos a poner un “flex bases” aquí. “Flex bases” de 100 píxeles. Okey, bien y aquí lo que pueden ustedes observar, es que me faltaría para que efectivamente haya un cambio, darle un “flex grow” también a nuestro ítem uno, porque nos había quedado huérfano de esa propiedad hoy, así que vamos a darle la

propiedad “flex grow” uno. Recuerden, tenemos el primer ítem, este tiene una base de 100 píxeles, este no tiene ninguna base, entonces se reparte el espacio remanente, pero tiene un valor dos en la propiedad “flex grow”, así que reclama un poco más de espacio, bueno, el doble de espacio reclama, que tres y cuatro, que tienen la propiedad “flex grow” en uno.

Ahora, ¿qué ocurre cuándo comenzamos a crecer? Noten que uno parte con una cierta ventaja sobre tres y cuatro, ¿por qué?, porque tiene una base, tiene, parte de una base de 100 píxeles, de hecho, si dijésemos que parte de una base de 300, se impone inclusive a “flex”, al item dos, que tiene una propiedad mayor de “flex grow”, porque porque tiene una base, ¿okey?, tiene una base que nunca va a ser inferior que los 300 píxeles.

Bien, perfecto, ahí podemos ver entonces algunas formas en las que se distribuye el espacio utilizando "Flex Box", con las propiedades “flex grow” y “flex shrink”. También hay un “short canon”, un atajo para esto, que es escribir directamente la propiedad “flex”. Si escribimos la propiedad “flex” a secas aquí, por ejemplo, podríamos indicar los tres valores en uno, es decir, primero vamos a tener combinados “flex grow”, “flex shrink” y “flex bases”. Aquí, por ejemplo, podríamos decir dos, dos, y 200 píxeles, o puede ser cualquier otra medida, ¿okey?, bien. Por supuesto elimino esa línea aquí.

Podríamos indicar para todas las demás, podríamos ponerlas en uno de “grow”, cuatro de “shrink” y 100 píxeles. Esta propiedad la voy a copiar a los ítems dos, tres y cuatro. Recuerden, tenemos una propiedad distinta en el número uno; el número uno tiene un “flex grow” de dos, un “flex shrink” de dos, también, y 200 píxeles de base, entonces, ¿qué ocurre cuando crecen?, ¿y qué ocurre cuando disminuyen su tamaño? Este es el concierto de los valores que estamos aplicando aquí, okey, es el concierto de estos valores. Si pusiéramos todos los ítems con una base de 100, la diferencia para el ítem uno, se vería principalmente cuando tiene espacio para crecer, pero mientras más se reduce el espacio, más se acerca al valor de sus hermanos, ¿por qué?, bueno, porque a pesar de que tiene la propiedad de 100 píxeles en la base, tiene un “flex shrink” de dos y, ¿qué ocurre si lo ponemos exactamente igual que sus hermanos, en cuatro? Bueno, va a quedar exactamente igual cuando el tamaño sea más pequeño, mientras más pequeño sea el tamaño. Pero se va a diferenciar cuando haya más tamaño para repartir, ¿por qué?, porque tiene un “flex grow” diferente al de sus hermanos.

Podríamos hacerlo más exagerado, podríamos poner el grupo en cuatro y el “shrink” también en cuatro y noten que, cuando hay poco espacio, está apenas un poquito más grande, pero cuando empieza a crecer el espacio para repartirse, bueno, es como que tiene derecho a más herencia, ¿verdad?, por ponerlo en términos sucesorios, si se quiere.

Bien, con esto completamos nuestro tutorial de "Flex Box". A partir de la próxima clase comenzaremos a ver “Grid” y luego, ya trabajando en conjunto con “Grid” y con "Flex Box", tendremos nuestra web responsive construida.