**CSS**

TIPOS DE SELECTORES:

\*Universal: (\*)

\*de tipo: por etiquetas del html

\*clases: se pone un “.” y se llama a la clase de HTML

\*ID: “#” se pone un id en html y se lo llama en el selector

\*Por atributo

\*Descendientes

\*Psedo-Clases

**FLEXBOX!!!! – conceptos básicos**

eje principal y eje secundario:

🡪 el eje principal lo marcamos según como queremos ordenadas las cajas:

*flex-direction*: row 🡪 cuando queremos una caja bajo la otra

flex-direction: column 🡪 cuando queremos una caja al lado de la otra

🡪 el eje secundario va a ser el **contrario** del principal que elegimos, y hará que todos los componentes que estén dentro del contenedor principal del flexbox (ppal flex.direction), se ajusten a su eje secundario. Ejemplo: si elegí ROW, los componentes contenidos en éste adopatarán la forma del ancho y al max de las columnas, del eje secundario dado por defecto, COLUMN.

display-flex 🡪 hace que los componentes contenidos por la caja que le estoy asignando esta propiedad adquieran FLEX, automáticamente.

flex-basis: auto;

flex-wrap: nowrap; 🡪 propiedad por defecto para trabajar con FLEX-BOX.

El resultado es que todos los ítems se alinearán en una solo fila, usando el tamaño del contenedor como su tamaño en el eje principal. Si hay más ítems de los que caben en el contenedor, estos no pasarán más abajo si no que sobrepasarán el margen. Si hay ítems más altos que otros, todos los ítems serán ajustados en el eje cruzado para alcanzar al mayor.

Con **flex-basis** se define **el tamaño de un ítem en términos del espacio que deja como espacio disponible**. El valor inicial de esta propiedad es auto — en este caso el navegador revisa si los ítems definen un tamaño.

Si los ítems no tienen un tamaño, entonces el tamaño de su contenido es usado como flex-basis. Y por eso, apenas declarado display: flex en el padre a fin de crear **ítems flex**, todos estos ítems se ubicaron en una sola fila y tomaron solo el espacio necesario para desplegar su contenido.

Con la propiedad **flex-grow** **definida como un entero positivo**, los ítems **flex** pueden crecer en el eje principal a partir de **flex-basis.** Esto hará que el ítem se ajuste y tome todo el espacio disponible del eje, o una proporción del espacio disponible si otro ítem también puede crecer.

**¡!** Si le damos a todos los ítems un valor **flex-grow de 1**, entonces el espacio disponible en el contenedor flex será compartido igualitariamente entre estos ítems y se ajustarán para llenar el contenedor sobre el eje principal.

**Podemos usar flex-grow apropiadamente para distribuir el espacio en proporciones.** Si otorgamos al primer ítem un valor flex-grow de 2 y a los otros un valor de 1, entonces 2 partes serán dadas al primer ítem (100px de 200px, por ejemplo) y 1 parte para cada uno de los restantes (cada uno con 50px de los 200px en total).

**La propiedad flex-shrink**

Así como la propiedad flex-grow se encarga de añadir espacio sobre el eje principal, la propiedad flex-shrink controla como se contrae. Si no contamos con suficiente espacio en el contenedor para colocar los ítems y flex-shrink posee un valor entero positivo, el ítem puede contraerse a partir de flex-basis. Así como podemos asignar diferentes valores de flex-grow con el fin que un ítem se expanda más rápido que otros — un ítem con un valor más alto de flex-shrink se contraerá más rápido que sus hermanos que poseen valores menores.

El tamaño mínimo del ítem tendrá que ser considerado cuando se determine un valor de contracción que pueda funcionar, esto significa que flex-shrink tiene el potencial de comportarse menos consistentemente que flex-grow .

Nótese que los valores de **flex-grow y flex-shrink** son proporciones. Típicamente, si pusiéramos todos los ítems flex: 1 1 200px y luego quisiéramos que un ítem creciera al doble, deberíamos ponerlo con flex: 2 1 200px. Aunque igualmente podemos colocar flex: 10 1 200px y flex: 20 1 200px si quisiéramos.

**align-items**

La propiedad align-items alineará los ítems sobre el eje cruzado.

El valor inicial para esta propiedad es stretch razón por la cual los ítems se ajustan por defecto a la altura de aquel más alto. En efecto se ajustan para llenar el contenedor flex — el ítem más alto define la altura de este.

En cambio, definimos align-items como flex-start para que los ítems se alineen al comienzo del contenedor flex, flex-end para alinearlos al final, o center para alinearlos al centro. Intente esto en el ejemplo en vivo — He definido en el contenedor flex una altura para que se aprecie que se pueden mover libremente dentro del contenedor. Vea lo que sucede si se coloca el valor align-items como:

stretch

flex-start

flex-end

center

justify-content

La propiedad justify-content es usada para alinear los ítems en el eje principal, cuyo flex-direction define la dirección del flujo. El valor inicial es flex-start que alineará los ítems al inicio del margen del contenedor, pero también se podría definir como flex-end para alinearlos al final, o center para alinearlos al centro.

También podemos usar space-between para tomar todo el espacio sobrante después de que los ítems hayan sido colocados, y distribuir de forma pareja los ítems para que haya un espacio equitativo entre cada ítem. O bien, usamos el valor space-around para crear un espacio equitativo a la derecha e izquierda de cada ítem.

Pruebe con los siguientes valores de justify-content en el ejemplo en vivo:

space-evenly

flex-start

flex-end

center

space-around

space-between