**FLEXBOX:**

• Metodología que permite maquetar un sitio utilizando una estructura de filas y columnas.

• Ventajas: propone un único flujo, en el que dispondremos de los elementos con mayor libertad para distribuir, redimensionar y reordenar cada uno de ellos en función de ese flujo de trabajo.

**Estructura básica**: está basada en el uso de un contenedor padre (flex-container) y sus elementos hijos (flex-items). Cuando hablamos de un flex-container, hablamos de un elemento HTML que contiene a uno o más elementos. A estos elementos anidados los llamamos flex-items.

• Trabajar con flexbox: en el flex-container es en donde configuramos la mayoría de las propiedades flex. Para eso usamos la propiedad display con el valor flex. De esta forma, estamos habilitando un contexto flex, para trabajar con los hijos directos del elemento. La propiedad display también puede recibir el valor inline-flex. Ej.: .contenedor-padre {display: flex;}.

**Propiedades**:

• Flex-wrap: por defecto, los elementos hijos de un contenedor flex van a tratar de entrar todos en una misma línea. Para aclararle al contenedor que debe respetar el ancho definido de sus hijos usamos la propiedad flex-wrap con el valor *wrap*, esto permitirá que los ítems tomen el ancho definido y que los que no entren en la línea, caigan a la siguiente. Flex-wrap también puede recibir los valores *nowrap* y *wrap-reverse*. Ej.: .contenedor-padre {display: flex; flex-wrap: wrap;}.

• Flex-items: su vez, puede convertirse en un flex-container. Para eso, solo hace falta asignarle la regla display:flex, para que así sus elementos hijos pasen a ser flex-items. Ej.: .elemento-hijo {display: flex;}.

**EJES:**

|  |  |
| --- | --- |
| **Flex-direction:** trabaja con 2 ejes para desarrollar todo su flujo interno: el eje X y el eje Y. Definimos el main axis (eje principal) del contenedor, que puede ser tanto horizontal como vertical. El cross axis (eje transversal) será la dirección perpendicular al main axis. |  |
| **Propiedades**: | |
| • Row: los ítems se disponen en el eje x, de izquierda a derecha. Si no le aclaramos la propiedad flex-direction al contenedor, row es el valor por defecto. |  |
| • Row-reverse: los ítems se disponen en el eje x, de derecha a izquierda. En este caso, estamos invirtiendo el inicio y fin del main-axis. |  |
| • Column: los ítems se disponen en el eje y, de arriba hacia abajo. |  |
| • Column-reverse: los ítems se disponen en el eje y, de abajo hacia arriba. En este caso, estamos invirtiendo el inicio y fin del main-axis. |  |
| **Justify-content**: con esta propiedad alineamos los ítems a lo largo del main axis. Si es horizontal, se alinearán en función de la fila. Si es vertical, se alinearán en función de la columna. |  |
| **Propiedades**: |
| • Flex-start: los ítems se alinean respecto del inicio del main axis que hayamos definido. Si no le aclaramos el justify-content al contenedor, flex-start es el valor por defecto. |  |
| • Flex-end: los ítems se alinean respecto del final del main axis que hayamos definido. |  |
| • Center: los ítems se alinean en el centro del main axis. |  |
| • Space-between: los ítems se distribuyen de manera uniforme. El primer ítem será enviado al inicio del main axis, y el último ítem, al final. El espacio libre se repartirá para separar los ítems. |  |
| • Space-around: los ítems se distribuyen de manera uniforme. El espacio libre disponible se repartirá entre todos los elementos. Del espacio que le toque a cada elemento, la mitad irá a la derecha y la otra a la izquierda (o arriba y abajo en caso de que sean columnas). |  |

|  |  |
| --- | --- |
| **Align-items**: con esta propiedad alineamos los ítems a lo largo del cross axis. Si no aclaramos esta propiedad, el valor por defecto es stretch, en otras palabras, los ítems ocuparán todo el espacio disponible en el cross axis. |  |
| **Propiedades**: | |
| • Stretch: los ítems se ajustan para abarcar todo el contenedor. Si el cross axis es vertical, se ajustan en función de la columna. Si el cross axis es horizontal, se ajustan en función de la fila. |  |
| • Flex-start: los ítems se alinean al inicio del cross-axis. |  |
| • Flex-end: los ítems se alinean al final del eje transversal. |  |
| • Center: los ítems se alinean al centro del eje transversal. |  |
| **Align-content:** si tenemos un contenedor de una sola línea (donde flex-flow se establece como no-wrap) utilizaremos align-items, pero en el caso de que estemos trabajando con contenedores multilínea debemos utilizar align-content.  Con esta propiedad alineamos los ítems a lo largo del cross axis cuando los contenedores flexibles incluyen de varias líneas (donde flex-flow se establece en wrap o wrap-reverse). Los valores que admite la propiedad align-content son similares a los que podemos utilizar para align-items. | |
| **Propiedades**: | |
| • Flex-start | flex-end: | |
| • Center | stretch: | |
| • Space-between | space-around: | |

**ITEMS:** posibilidad de aplicarle propiedades directamente a cada ítem para poder manipularlos por separado y tener mayor control.

|  |  |
| --- | --- |
| **Order:** controlamos el orden de cada ítem, sin importar el orden original que tengan en la estructura HTML. Esta propiedad recibe un *nº entero*, *positivo* o *negativo*, como valor. Por defecto, todos los ítems flex tienen un order: 0 *implícito*. Las cajas se irán ordenando respetando la secuencia desde los nº negativos hacia los positivos. | |
| **Propiedades:** | |
| • Nº Positivo: si le asignamos a la caja Q (que posee la clase caja-q) la propiedad order con valor 1, esta pasará al final de la fila por ser el nº más alto ya que por defecto, el valor del orden de cada ítem es 0. | Ej: .caja-p {orden: 1;} |
| • Nº Negativo: si le asignamos a la caja D la propiedad order con un -1 como valor, esta pasará al principio de la fila. Colocando al ítem con el orden más pequeño  primero. | Ej.: .caja-q {orden: -1;} |

|  |  |  |
| --- | --- | --- |
| **Flex-grow:** configura un crecimiento flexible para el elemento. | | |
| • Si *ambos ítems* tienen la propiedad con valor 1, a medida que el contenedor se agrande, irán abarcando el espacio disponible en partes iguales. Ej.: .caja-a, .caja-b {flex-grow: 1;} |  | |
| • Si un *solo ítem* tienen la propiedad, este intentará ocupar el espacio libre disponible, a medida que el contenedor se agrande, según la proporción que definamos con el valor. Ej.: .caja-b {flex-grow: 1;} |  | |
| • El nº que le asignamos determina qué cantidad de espacio disponible dentro del contenedor tiene que ocupar ese ítem. *1 = 100% del espacio, y 0 = 0%*. Podemos usar cualquier valor en el medio como *0.25 = 25%.* |  | |
|  | | |
| **Align-self**: alinea, sobre el cross axis, a cada ítem al que le apliquemos la propiedad, independientemente de la alineación que se haya definido en el contenedor flex con align-items. Estas propiedades aplicarán para los flex-items siempre y cuando el contenedor padre sea un flex-container. | | |
| **Propiedades**: | | |
| • Flex-end: el ítem se alínea al final del eje transversal. Ej.: .contenedor-padre { align-items: flex-start; } / .caja-dos { align-self: flex-end;} | |  |
| • Center: el ítem se alínea al centro del eje transversal. Ej.: .contenedor-padre {align-items: flex-start;} / .caja-dos {align-self: center;} | |  |
| • Flex-start: el ítem se alínea al inicio del eje transversal. Ej.: .contenedor-padre {align-items: flex-end;} / .caja-dos {align-self: flex-start;} | |  |
| • Stretch: el ítem se ajusta hasta abarcar todo el cross axis, es el comportamiento por defecto. Funciona siempre que el elemento no tenga definida una altura. Ej.: .caja-uno, .caja-tres {align-self: flex-start;} / .caja-dos {align-self: stretch;} | |  |