**CSS Flexbox**

Creamos nuestro HTML y CSS base para poder comenzar a estudiar las propiedades de Flexbox. Dichas propiedades serán estilizadas en el HTML para no mezclarnos.

    <div class="container">

      <div class="box box-1">1</div>

      <div class="box box-2">2</div>

      <div class="box box-3">3</div>

      <div class="box box-4">4</div>

      <div class="box box-5">5</div>

    </div>

body {font-family: Verdana, Geneva,

Tahoma, sans-serif; background: "f3f3f3"}

.container {border: 5px solid red}

.box {padding: 30px; text-align: center;

font-size: 35px; color: #f3f3f3}

.box-1 {background: #581f18}

.box-2 {background: #f0a202}

.box-3 {background: #d95d39}

.box-4 {background: #202c59}

.box-5 {background: #51cb20}

* Esta es la BASE.

**Flexbox**

Tendremos un Parten **(Flex-Container)** y los Children **(Flex-Items)**. En nuestro ejemplo tenemos un Flex-Container y 5 Flex-Items.

Para comenzar:

.container {display: flex}

* Flex 🡪 It is a BLOCK element.

.container {display: inline-flex}

* Inline-Flex 🡪 It is an Inline

Properties :

Generalmente las podemos separar en dos categorìas :

* Affect all Children (flex)
* Affect Individual Child (flex item)

Defaults en **Display: Flex :**

* Flex-Direction : Row
* Flex-Wrap : No-Wrap
* Justify-Content : Flex-Start
* Align-Items : Stretch
* Align-Content : Stretch
* Order : 0
* Flex-Grow : 0
* Flex-Shrink : 1

**All**

**Flex-Direction :**

Row 🡪 Default one (1 to 5).

Row-Reverse 🡪 Change direction, 5 to 1.

Column 🡪 Volvemos al Original (default sin habilitar Flexbox).

Column-Reverse  🡪 Original dado vuelta.

**Flex-Wrap :**

Para demostrar Wrap crearemos 5 nuevos divs sin importancia, para luego ser borrados.

**IF** our Flex Container is **ABLE** to fix boxes, Everything will be as before. The moment it is **NOT ABLE**, the user will need to scroll.

No-Wrap  🡪 Default

Wrap 🡪 When theres not enough space : It starts new line. (G, 1 to 5)

Wrap-Reverse 🡪 When there is NO space : Starts new line reversed. (G, 5 to 1 when NO SPACE).

Ej : **Nav Bar** When the screen is smaller we want to see all the items. We would use WRAP. (if not they would need to scroll)

**Justify-Content :**

Responsible for Horizontal Alignment.

Flex-Start 🡪 Default

Flex-End 🡪 Las boxes comienzan del otro lado (derecha) y de 5 to 1.

Center 🡪 Pone todo la centro sin espacios.Solo bordes Izq y Der.

Space-Between 🡪 1 and 5 Tocan las puntas (Izq y Der), luego harbá espacio entre medio de los items.

Space-Around 🡪 Crea margenes al rededor de todos los items. (L-item-R,L-item-R, etc.)

Space-Evenly 🡪 Crea margenes parejos para todos los lados.

* If we start a new line(wrap) : All Lines will be applied with the Justfy-Content chosen.

**Align-Items :**

Risponisble for Vertical Alignment. Necesitamos aclarar previamente un **Height** a la Parent.

Stretch 🡪 Estira los elementos para cubrir todo el Height (Default)

Flex-Start 🡪 Los elementos se alinean al TOP del container

Flex-End 🡪 Tocan el borde de abajo

Center 🡪 Se alinea Verticalmente al centro.

Baseline 🡪 Se alinea dependiendo el CONTENIDO (podrán sobrar bordes arriba o abajo) EJ : Alinear el menu a la misma altura que el logo.

**\*\*Mini Project : Hero Components\*\***

Nuestro objetivo es poner el TITULO exactly in the center of the Hero Section (Vertical and Horizontal). We have a BG image, and the title only.

Que necesitamos ? Flex Parent (hero) y Flex Items (h1)

**.hero** {Min-Height : 100vh} 🡪 Even though the content is less than the 100% , the Size of this Parent element is 100% of the screen. (invisible if we want)

1. Display Flex al Parent
2. Centrar Horizontal
3. Centrar Vertical

**.hero {**Display : Flex ; Justify-Content : Center ; Align-Items : center**}**

**Align-Content :**

Afecta unicamente si tenemos Multiple Lines.

Stretch 🡪 Default

Flex-Start 🡪 Normalmente

Flex-End 🡪 All Lines touches the End of the container

Center 🡪 Alinea los renglones en el Center (vertically)

Space-Around 🡪 Crea espacios entre LINEAS Top y Bot (si es small screen se junta todo)

Space-Between 🡪 Top row Toca, Bot Row Toca, Crea espacio entre las lineas del medio y las Top y Bot.

**Main-Axis Cross-Axis :**

When we have a :

ROW 🡪 Main is Horizontal, Cross is Vertical.

COLUMN 🡪 Main Vertical, Cross Horizontal.

Ej : Tenemos una BOX Rows y una BOX2 Columns. Objetivo Centrar todo en la screen (vertical y horizontal)

**Box** **{**display: Flex ; Justify-Content: Center ; Align-Items: Center**}**

**Box2 {**display: Flex ; Flex-Direction: Column ; Align-Items: Center; Justify-Content: Center**}**

* En BOX 🡪 Justify : Alinea Verticalmente,

Align : Horizontalmente

* En BOX2 🡪 Align : Vertical

Justify : Horizontal.

**Order Property :**

By Default all Children have an ORDER of ZERO. Podríamos cambiar su orden SIN tocar el HTML.

Ej : Queremos mover 1 al puesto 3.

Primero damos Order a TODOS LOS CHILD (order :1, 2, 3, 4, 5)

Luego 1 {order :3} 2{order :1} 3{order :4}, etc.

**Align-Self :**

We can specify the direction of each and every CHILD element.

Ej : I want 5 at the end. **5 {Align-Self :Flex-End}**

Stretch 🡪 Estira los elementos para cubrir todo el Height (Default)

Flex-Start 🡪 Los elementos se alinean al TOP del container

Flex-End 🡪 Tocan el borde de abajo

Center 🡪 Se alinea Verticalmente al centro.

Baseline 🡪 Se alinea dependiendo el CONTENIDO (podrán sobrar bordes arriba o abajo) EJ : Alinear el menu a la misma altura que el logo.

**Flex-Grow :**

0 🡪 Default. Thats why we see the empty space on our container.

1 🡪 **All Children will fill all the Container Equally.**

Box1 {Flex-Grow : 2}

Box2 {Flex-Grow : 1}

* Es proporcional. Cada item irá creciendo respetando la proporcion y sin dejar espacios vacios.

**Flex-Shrink :**

1 🡪 Default. They start already Shriking for making space for mroe items.

Ej : **Box2** {width :300px ;}

Ocupará 300px si hay espacio, en caso de achicar la pantalla se ajustará para hcer lugar para el resto de las boxes.

**Box2** {width :300px ; Flex-Shrink :0}

Ocupará 300px y no se modificará su tamaño. Haciendo que las demas se encojan a medida que se achique la pantalla. Hasta sus Base Values aclarados en el CSS.

**Flex-Basis and Flex (shortcuts) :**

* Cuando trabajemos con FLEXBOX no es recomendable usar Paddings Right or Left.

Flex-Basis : 150px 🡪 Establece un valor que al ser tocado, aplicará los demas valores de flexbox.

Ej : Container {display :flex ; flex-wrap :wrap ;}

Box {flex-basis :150px }

* Una vez Tocados los 150px se aplica WRAP, haciendo que la box4 descienda a una nueva Linea.

Shortcut 🡪 **Flex :** 1 0 200px

**Grow Shrink Base**

Ej : Container {display :flex ; flex-wrap :wrap}

Box {**Flex :** 1 0 200px}

* **IF** there is space : Everything will be in **One Line** ;

**IF** any box gets **Less than 200px.** It will wrap **New Line.**

Because **Shrink 0** it will ocuppy the whole space (line).

How to set **Column Layouts** for the Project : Usaremos porcentajes para separar la layout en fracciones.

**IMPORTANTE : No tener paddings a los lados.**

**.box** {flex : 0 0 25%}

Ej : Queremos 4 boxes repartidas en parejo con una separacion entre ellas. Cuando no haya espacio, crear una nueva linea

.container {display : flex ; flex-wrap :wrap ; **Justify-Content** : space-around}

.box {flex : 0 0 **clalc(**25% - 1rem**)**}

* Primero ponemos el **CALC()** 🡪 25% : Lo que ocupan las columnas. //  1REM : margen de separacion
* Segundo el **Justify**: Una vez esablecido el margen ponemos que las boxes tengan un espacio entre ellas.

**Datos Piolas**

* Parent con Display :Flex.
* Defaults en **Display: Flex :**
* Flex-Direction : Row
* Flex-Wrap : No-Wrap
* Justify-Content : Flex-Start
* Align-Items : Stretch
* Align-Content : Stretch
* Order : 0
* Flex-Grow : 0
* Flex-Srhink : 1
* **Main Axis // Cross :**
* ROW 🡪 Main is Horizontal, Cross is Vertical.
* COLUMN 🡪 Main Vertical, Cross Horizontal.
* **Order :** Podemos dar Order a cada Item sin tocar HTML.
* **Grow :** Hacerlos crecer para ocupar espacio vacios (def :0)
* **Shrink :** Achicarlos para dejar espacio libre (def :1)
* **Flex-Basis :** Valor que al ser tocado Se aplicarán los Flex dados a la Parent.
* **Flex : grow srhink basis 🡪** Shortcut

**Recomendations**

* **NO** usar paddings a Left o Right.
* **CENTER TITLE IN HERO SECTION : .hero {**Display : Flex ; Justify-Content : Center ; Align-Items : center**}**
* **COLUMN LAYOUTS :**

**.child** **{**flex : 0 0 **clalc(**25% - 1rem**)}**

**🡪 25%** area de columns // **1rem** margin de separacion

**.parent {**display : flex ; flex-wrap :wrap ; **Justify-Content** : space-around}

🡪**Justify-content :**Dará el margen que deseemos (alrededor, que la primera y ultima columna toquen bordes, etc.)