

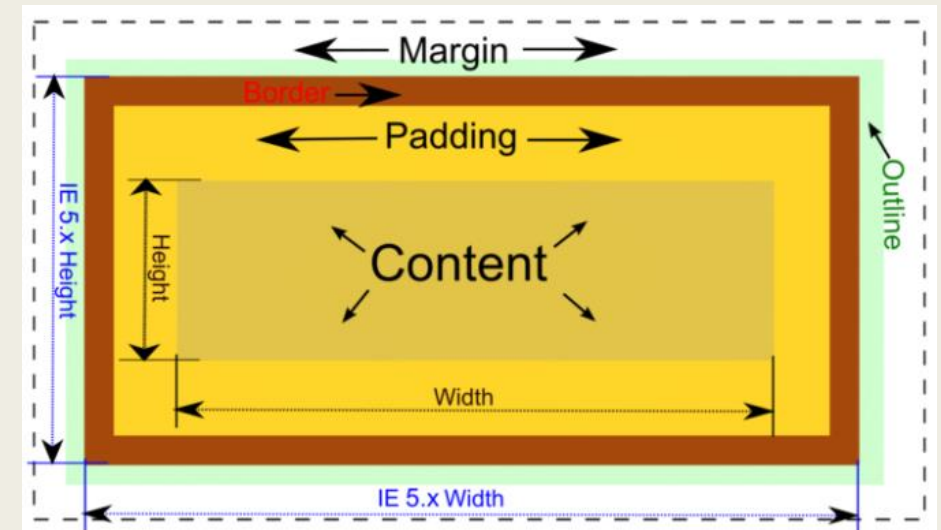


ANEXO I. POSICIONAMIENTO

DIW – FRANCISCO CORBERA NAVAS



POSICIONAMIENTO



POSICIONAMIENTO

■ Unidades de medida

- ***Absolutas:*** *completamente definida, su valor no depende de otros.*
 - **in**, pulgadas ("inches", en inglés). Una pulgada equivale a 2.54 centímetros.
 - **cm**, centímetros.
 - **mm**, milímetros.
 - **pt**, puntos. Un punto equivale a 1 pulgada/72, es decir, unos 0.35 milímetros.
 - **pc**, picas. Una pica equivale a 12 puntos, es decir, unos 4.23 milímetros.

```
body { margin: 0.5in; }  
h1 { line-height: 2cm; }  
p { word-spacing: 4mm; }  
a { font-size: 12pt }  
span { font-size: 1pc }
```

POSICIONAMIENTO

■ Unidades de medida

- ***Relativas:** su valor siempre esta referenciado respecto de otro.*
 - **em**, (no confundir con la etiqueta de HTML) relativa respecto del tamaño de letra del elemento. Si se utiliza una tipografía de 12 puntos, 1em equivale a 12 puntos.
 - **ex**, relativa respecto de la altura de la letra x ("equis minúscula") del tipo y tamaño de letra del elemento.
 - **px**, (píxel) relativa respecto de la resolución de la pantalla del dispositivo en el que se visualiza la página HTML.
 - **%**, relativo al tamaño del elemento que hereda el contenedor o al tamaño del propio contenedor (width por ejemplo)
 - **rem**, toma como referencia el tamaño de la fuente raíz del documento (normalmente el html), evitando así que si se anidan varios elementos el tamaño aumente o disminuya debido a la naturaleza de la herencia.

POSICIONAMIENTO

■ margin

- *Separación entre una caja determinada y las cajas adyacentes.*
- *Es un shorthand → propiedad abreviada que permite asignar el valor de otras muchas al mismo tiempo:*

- *margin-top*
- *margin-right*
- *margin-bottom*
- *margin-left*

```
margin-top: 10px;  
margin-right: 5px;  
margin-bottom: 10px;  
margin-left: 5px;
```

```
margin: 10px 5px 10px 5px;
```

POSICIONAMIENTO

■ margin

- *Los márgenes top y bottom de dos elementos que van seguidos se "colapsan". Es decir, se asume como margen entre ambos elementos el mayor de ellos.*

```
h1 {margin: 10px 20px 10px 20px; }  
h2 {margin: 20px; }
```

- *En el primer caso el margen superior e inferior es de 10px. En el segundo caso es de 20px. El espacio resultante entre los dos elementos será de 20px.*

POSICIONAMIENTO

■ margin

```
p {  
  margin: 25px 50px 75px;  
}
```

margin: 25px 50px 75px;

- el margen superior es de 25px
- los márgenes derecho e izquierdo son 50px
- el margen inferior es de 75 px

```
p {  
  margin: 25px;  
}
```

margin: 25px;

- los cuatro márgenes son 25px

```
p {  
  margin: 25px 50px;  
}
```

margin: 25px 50px;

- los márgenes superior e inferior son de 25 px
- los márgenes derecho e izquierdo son 50px

POSICIONAMIENTO

■ padding

- *Separación entre el contenido y su borde*
- *Es un shorthand → propiedad abreviada que permite asignar el valor de otras muchas al mismo tiempo:*

- *padding-top*
- *padding-right*
- *padding-bottom*
- *padding-left*

```
padding: 1em;
```

```
padding: 10% 0;
```

```
padding: 10px 50px 20px;
```

```
padding: 10px 50px 30px 0;
```

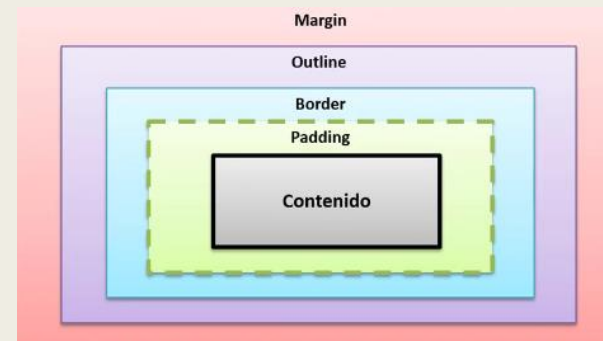

POSICIONAMIENTO

■ border

- *Línea que rodea la caja, lo que va a continuación del padding.*
- *Al igual que margin y padding, es una shorthand:*

- border-width
- border-style
- border-color

```
p {  
  border: 5px solid red;  
}
```



POSICIONAMIENTO

■ border

– *Existen varias combinaciones:*

■ **border-width: 2px;**

(top-right-bottom-left)

■ **border-style: solid dotted;**

(Y X)

■ **border-color:red blue green;**

(top X bottom)

■ **border-width: 2px 3px 4px 5px;**

(top right bottom left)

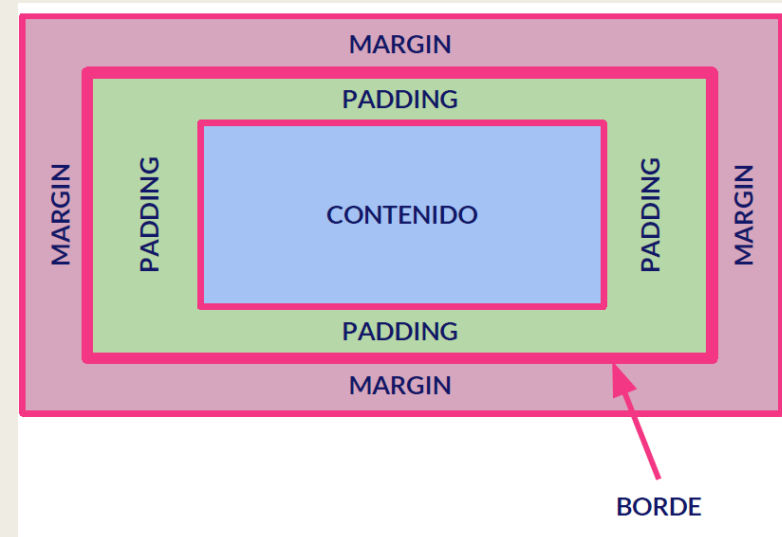
```
border-top(width|style|color)
border-right(width|style|color)
border-bottom(width|style|color)
border-left(width|style|color)
```

POSICIONAMIENTO

■ Propiedad box-sizing

- *Altura del elemento:*
 - $\text{Altura del contenido} + \text{padding} + \text{borde}$
- *Anchura del elemento:*
 - $\text{Anchura del contenido} + \text{padding} + \text{borde}$

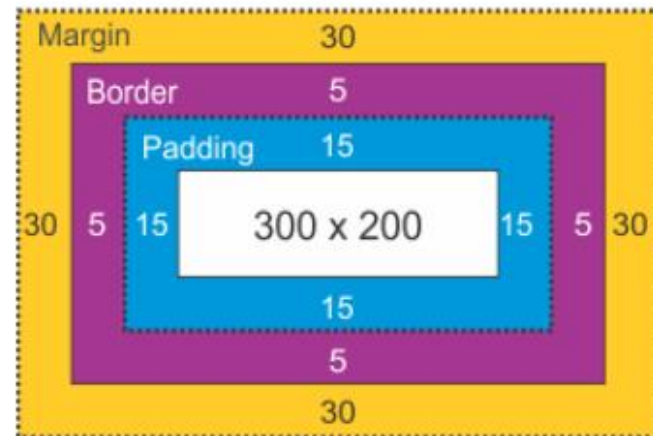
```
* {  
  -webkit-box-sizing: border-box;  
  -moz-box-sizing: border-box;  
  box-sizing: border-box;  
}
```



POSICIONAMIENTO

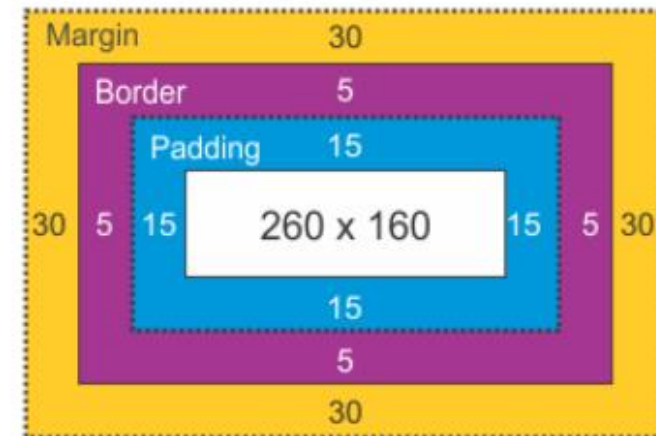
■ Propiedad box-sizing

Box Model is content-box



```
div{  
  width: 300px;  
  height: 200px;  
  padding: 15px;  
  border: 5px solid grey;  
  margin: 30px;  
  -moz-box-sizing: content-box;  
  -webkit-box-sizing: content-box;  
  box-sizing: content-box;  
}
```

Box Model is border-box

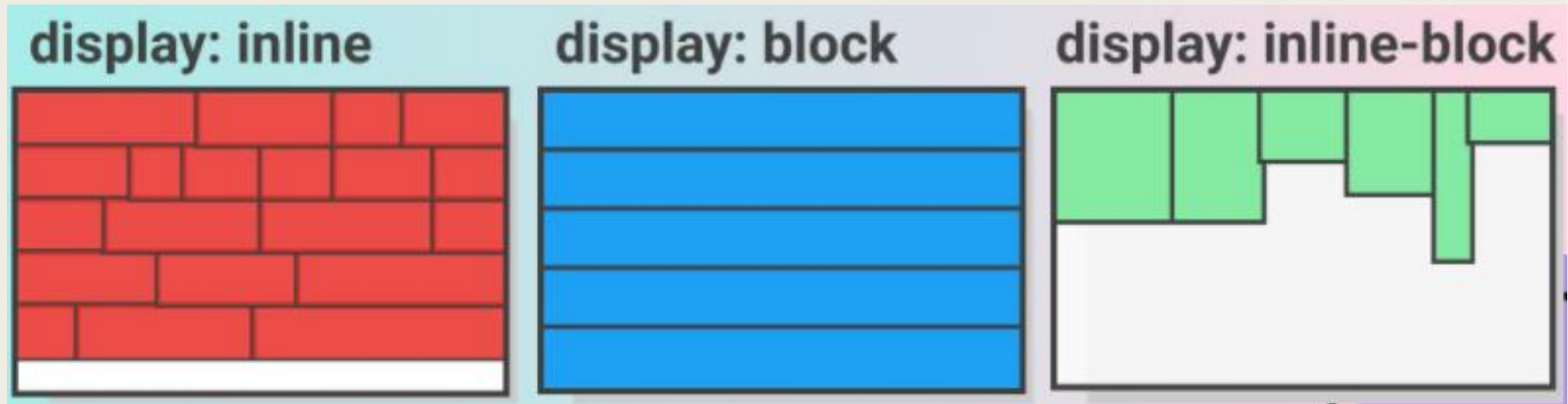


```
div{  
  width: 300px;  
  height: 200px;  
  padding: 15px;  
  border: 5px solid grey;  
  margin: 30px;  
  -moz-box-sizing: border-box;  
  -webkit-box-sizing: border-box;  
  box-sizing: border-box;  
}
```

POSICIONAMIENTO

■ Flujo del HTML

Es el orden en el que aparecen los elementos.



POSICIONAMIENTO

■ Flujo del HTML. Propiedad display

- *Nos permite definir cómo se comporta una caja.*
- *Existen tres modos principales de comportamiento:*
 - **inline:** no rompen el flujo de la línea y se van colocando uno detrás de otro. **Aceptan margin y padding** (solo sirve en horizontal) e **ignoran width y height**. Ejem: ,<a>, ..
 - **inline-block:** igual que inline pero **podemos asignarles width y height**.
 - **block:** los elementos rompen el flujo de la línea y provocan saltos de línea antes y después del elemento. Si no especificamos anchura ocupan toda la del elemento que los contiene. Ejem: <div>,<p>,<h1>,<section> ..

POSICIONAMIENTO

■ Propiedad display

– *Algunos de los valores que puede tener son:*

- **Inline** → la caja se puede comportar como si fuera un elemento en línea
- **Block** → la caja se puede comportar como si fuera un elemento de bloque
- **Inline-block** → la caja se comporta como un elemento en línea pero acepta width y height
- **None** → oculta el elemento pero se sigue renderizando (no se vería pero los recursos se cargan). Con visibility:hidden sí que deja el hueco.
- **Table** → imita el comportamiento de una tabla
- **List-ítem** → imita el comportamiento de una lista
- **(table-cell, table-row, table-colgroup..)**

} Flex y Grid las mejoran

POSICIONAMIENTO

■ Tipos de LAYOUTS

- **Fixed** -> anchura en píxeles (siempre el mismo tamaño)
- **Elastic** -> anchura (width) en em (escala correctamente)
- **Fluid** -> anchura en %, siempre con respecto a la etiqueta padre (proporción de los elementos siempre igual)
- **Max/Min Width** -> la anchura de los bloques puede adaptarse dentro de unos mínimos y máximos (max-width / min-width) expresados en píxeles.
- **Responsive** -> uso de media queries (@media) para que el layout cambie dependiendo de las características de la pantalla.

POSICIONAMIENTO



POSICIONAMIENTO

■ Centrar un elemento

– *Horizontalmente*

- Elementos en Línea: `text-align: center` (al padre)
- Elementos en Bloque: `margin: X auto` (al elemento, que debe tener anchura)
- Elementos Inline-Block: `text-align: center` (al padre) y `display: inline-block` (a los elementos, que deben tener anchura definida)

– *Verticalmente*

- Elementos en Línea: tener el `mismo padding arriba y abajo` y establecer `vertical-align: middle` si estamos dentro de un elemento de tabla o lo estamos simulando con una propiedad display.
- Elementos en Bloque: utilizar `position en el contenedor y en el elemento`.

POSICIONAMIENTO

■ Propiedad position

- *Con esta propiedad podemos modificar el flujo normal.*
- *Puede tener los siguientes valores:*
 - Static: por defecto, sigue el flujo normal de la página
 - Relative
 - Absolute
 - Fixed
 - Sticky

ELEMENTO POSICIONADO

POSICIONAMIENTO

■ Propiedad position

- *Los elementos posicionados se pueden mover en los 3 ejes:*
 - **top** → Movemos el elemento por la **parte superior**
 - **right** → Movemos el elemento por la **parte derecha**
 - **bottom** → Movemos el elemento por la **parte inferior**
 - **left** → Movemos el elemento por la **parte izquierda**
 - **Z-index** → Si 2 elementos se solapan podemos elegir cual se ve y cual no, indicando la capa que estará por encima

```
img {  
  position: absolute;  
  left: 0px;  
  top: 0px;  
  z-index: -1;  
}
```

POSICIONAMIENTO

■ position: static

```
.static {  
  position: static;  
}
```

`<div class="static">`

`static` (estático) es el valor por defecto. Un elemento con `position: static;` no está posicionado en ninguna forma en específico. Se dice que un elemento `static`, está *no posicionado* y un elemento con valor establecido de `position` está *posicionado*.

`</div>`

POSICIONAMIENTO

■ position: relative

```
.relative1 {  
  position: relative;  
}  
.relative2 {  
  position: relative;  
  top: -20px;  
  left: 20px;  
  background-color: white;  
  width: 500px;  
}
```

<div class="relative1">

relative (relativo) se comporta de la misma manera que static a menos que le agregues otras propiedades.

<div class="relative2">

Establecer las propiedades top, right, bottom, y left de un elemento con position: relative; causará que su posición normal se reajuste. Otro contenido no se puede ajustar para adaptarse a cualquier hueco dejado por el elemento.

</div>

</div>

POSICIONAMIENTO

■ position: fixed

- *Un elemento fixed (fijo) se posiciona a la ventana del navegador de manera relativa, lo que significa que se mantendrá en el mismo lugar incluso después de hacer scroll en la página*
- *Un elemento con valor fixed no deja espacio en el lugar de la página donde estaba ubicado normalmente.*

Ej. Menú en la parte superior

```
.fixed {  
  position: fixed;  
  bottom: 0;  
  right: 0;  
  width: 200px;  
  background-color: white;  
}
```

POSICIONAMIENTO

■ position: absolute

- *Se comporta como fixed pero es relativo a su ancestro posicionado más cercano en lugar de ser relativo a la ventana del navegador.*
- *Si no tiene ancestros posicionados (no static), usará el elemento body del documento, y se seguirá moviendo al hacer scroll en la página.*

```
<div class="relative">
```

Este elemento está usando `position: relative;`. Si estuviera usando `position: static;` su sucesor con `position: absolute;` escaparía y se posicionaría relativamente al body del documento.

```
<div class="absolute">
```

Este elemento tiene `position: absolute;`. Se posiciona de manera relativa a su ancestro.

```
</div>
```

```
.relative {  
  position: relative;  
  width: 600px;  
  height: 400px;  
}  
.absolute {  
  position: absolute;  
  top: 120px;  
  right: 0;  
  width: 300px;  
  height: 200px;  
}
```


POSICIONAMIENTO

■ position: sticky

- *Es un híbrido entre fixed y relative. Propiedad experimental.*
- *Es tratado como un elemento posicionado relativamente hasta que cruza un umbral especificado, en cuyo punto se trata como fijo hasta que alcanza el límite de su padre.*

```
#one { position: sticky; top: 10px; }
```

- *El elemento se posicionará relativamente hasta que el viewport sea desplazado de manera tal que el elemento esté a menos de 10px del límite superior. Más allá de ese umbral, el elemento es fijado a 10px del límite superior.*

POSICIONAMIENTO

■ position: resumen

static

Es el valor por defecto.
El elemento sigue el flujo que le corresponde. Aunque use top, bottom, left, right o z-index **NO** las aplica.

relative

Como static pero **SÍ** atiende top, bottom, left, right o z-index a partir de la posición que le corresponde por el flujo.

fixed

Se le aplica top, bottom, left, right o z-index **en relación al documento**. **No** atiende al scroll. Permanece siempre en el mismo sitio.

absolute

Se comporta como **fixed** pero en relación a la primera etiqueta antecesora que tenga **position: relative**.

sticky

relative hasta llegar a una posición de scroll y a partir de entonces **fixed**.

inherit

La propiedad **position** no se propaga en cascada, Si queremos que sea así añadiremos el valor **inherit** a los hijos que queremos que hereden

POSICIONAMIENTO

■ position: centrado vertical

- *Si conocemos la altura*
- *No conocemos la altura*

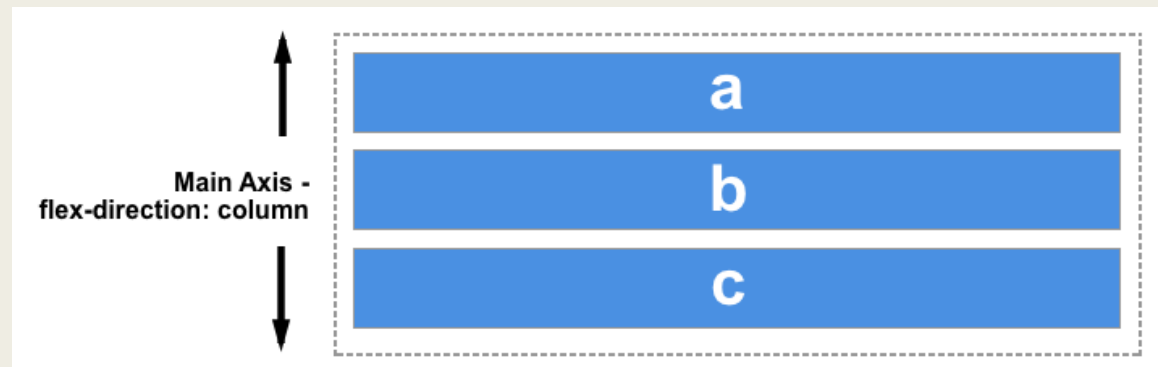
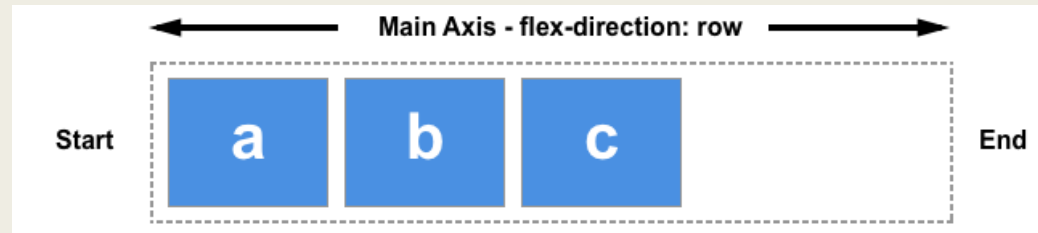
```
.contenedor {  
    position: relative;  
}  
  
.elemento_a_centrar {  
    height: 150px;  
    margin-top: -75px; /** La mitad de la altura **/  
    position: absolute;  
    top: 50%;  
}
```

```
.contenedor {  
    position: relative;  
}  
  
.elemento_a_centrar {  
    position: absolute;  
    top: 50%;  
    transform: translateY(-50%);  
}
```

POSICIONAMIENTO

■ display: flex

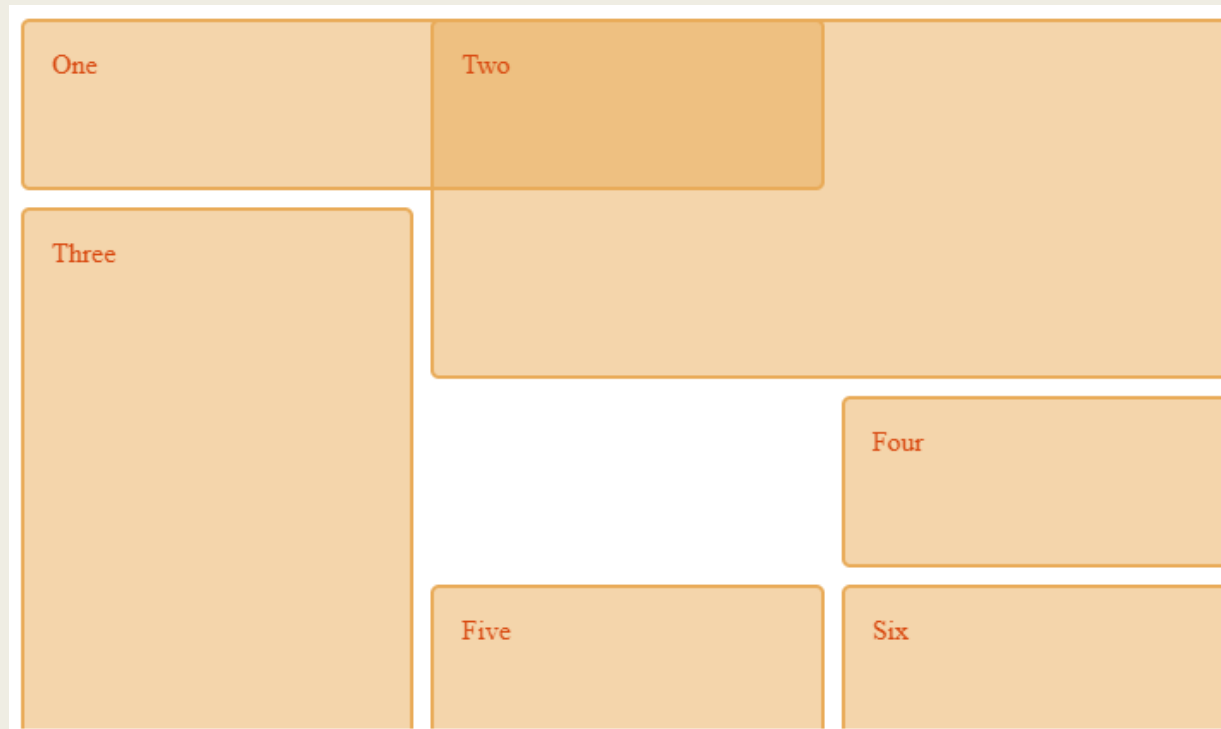
- Usado para maquetar componentes.
 - Ej. Menú de navegación
- Diseñado como un modelo unidimensional de Layout



POSICIONAMIENTO

■ display: grid

- *Usado para maquetar el Layout.*
- *Diseño multidimensional donde trabajo con filas y columnas.*



POSICIONAMIENTO

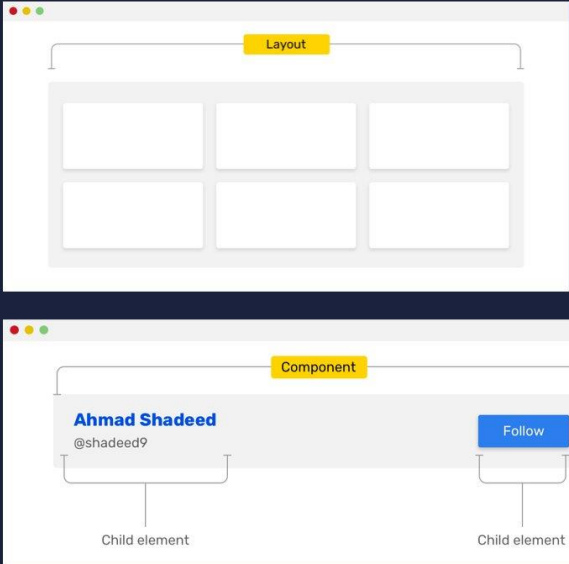
■ Flexbox vs Grid for Layout

- *No hay una forma correcta o incorrecta de usarlos.*

Use...

Grid for layout
e.g if you see columns and rows.

Flexbox for components
e.g if the component you are viewing has all of its child items displayed inline.



- *Demo para aclarar conceptos. ([Ahmad Shadeed](#))*