UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Programación Web Inicial – Frontend Developer

CLASE 6

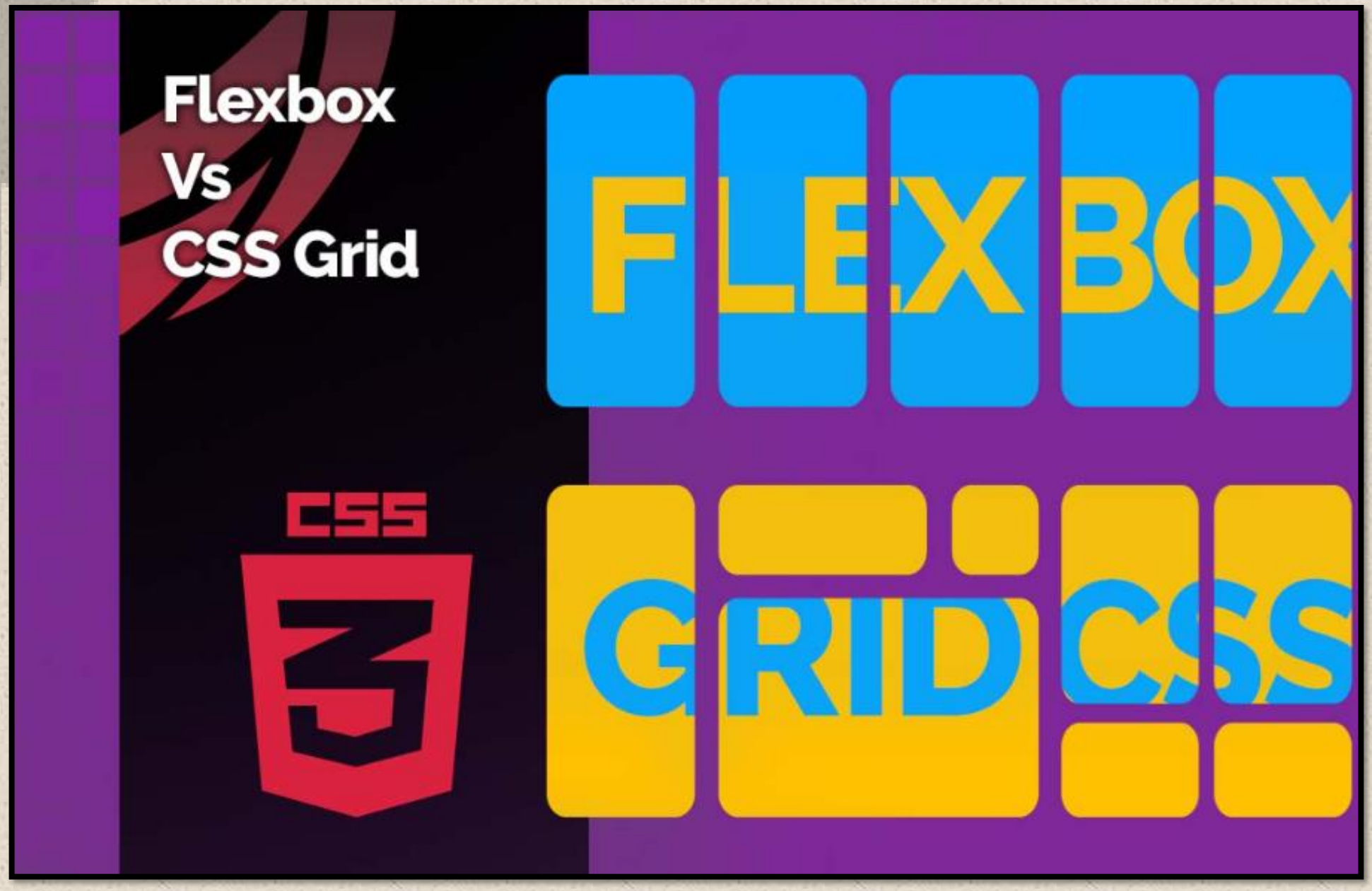
“Sé bueno con los nerds, es muy probable que termines trabajando para uno de ellos”

Bill Gates

MCC

Flexbox y Grid

- Order
- Wrap
- Reverse
- Shrink
- Align-slef-content-items
- Grid System


Flexbox
Vs
CSS Grid
CSS
3
FLEXBOX
GRID CSS

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Flexbox 👉 Caja flexible

Este módulo fue diseñado como un *modelo unidimensional* en el que podemos ordenar elementos como filas o columnas (solo una opción), y como un método que pueda ayudar a distribuir el espacio entre los ítems de una interfaz y mejorar las capacidades de alineación.

Se le asigna este comportamiento mediante la propiedad **CSS**:

display:flex;

```html
<div class="flex-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
</div>
```

EJEMPLO 👉

```css
.flex-container {
    display: flex;
    flex-direction: column-reverse;
    justify-content: flex-start;
}
```

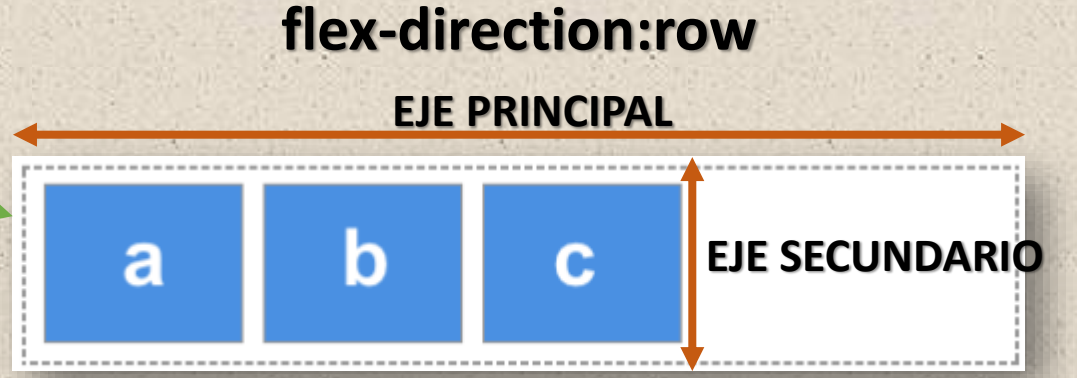
Flexbox 👉 Caja flexible

Cuando trabajamos con flexbox necesitamos pensar en términos de dos ejes: el eje principal y el eje cruzado. El ***eje principal*** está definido por la propiedad **flex-direction**, y el *eje cruzado* es perpendicular a este.

En el eje principal, flex-direction posee 4 posibles valores:

- row
- row-reverse

- colum
- colum-reverse

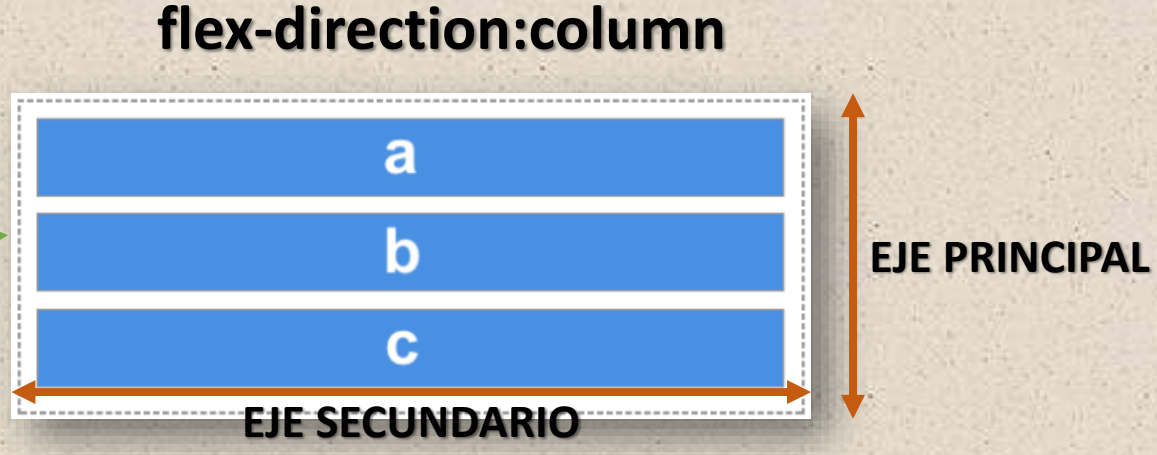


Flexbox 👉 Caja flexible

Ahora veamos como se vería pasando de fila (row) a columna (column):

- row
- row-reverse

- colum
- colum-reverse



Si **flex-direction** marca «**row**», el principal será el eje horizontal y el secundario el vertical; mientras que si marca «**column**», el principal será el eje vertical y el secundario será el horizontal.

Flexbox 👉 Caja flexible

justify-content = eje principal

align-ítems = eje secundario

La propiedad *justify-content* funciona para el eje principal, mientras *align-items* funciona para el eje secundario. No ordenan horizontal ni verticalmente, ya que dependen de la orientación que le dé *flex-direction*.

Flexbox 👉 Caja flexible

Propiedad **order**

La propiedad **order** de *Flexbox* nos permite establecer el orden en el que se van a cargar nuestros elementos **HTML** en la página web. Este orden se establece mediante un valor numérico, sea negativo o positivo. Es decir, el posicionamiento del elemento dependerá de su valor y su relación con los valores de los demás elementos; el elemento se cargará más cerca al inicio del eje principal cuanto menor sea su valor.

El valor por defecto para todos los elementos es de 1.

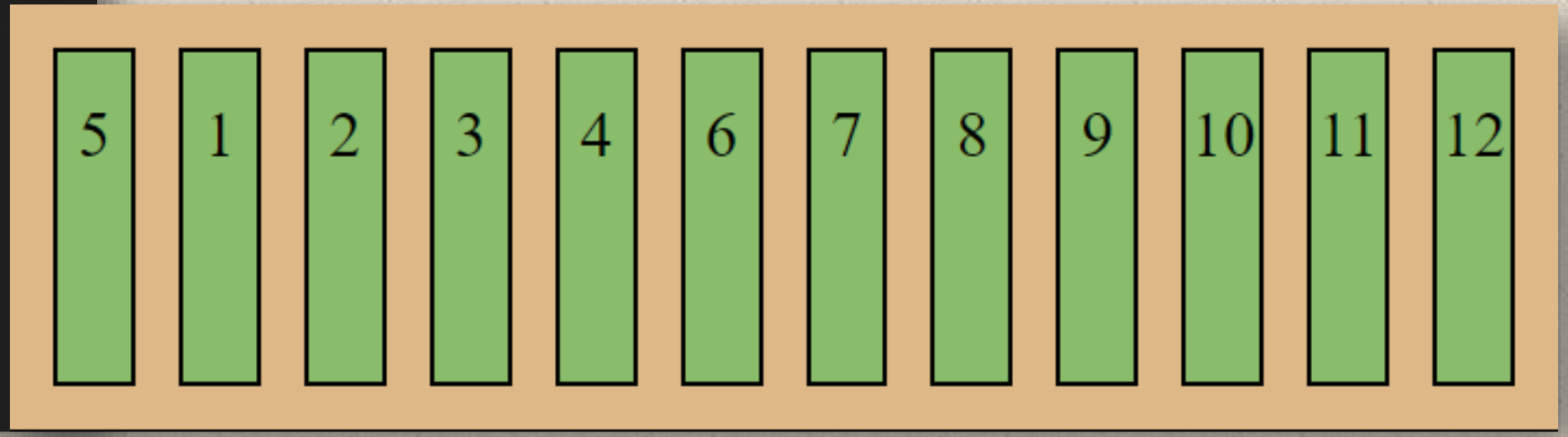
UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Flexbox 👉 Caja flexible

Propiedad **order**

```html
<div class="flex-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div class="orden">5</div>
    <div>6</div>
    <div>7</div>
    <div>8</div>
    <div>9</div>
    <div>10</div>
    <div>11</div>
    <div>12</div>
</div>
```

```css
}
.flex-container .orden {
    order: -1;
}
```



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

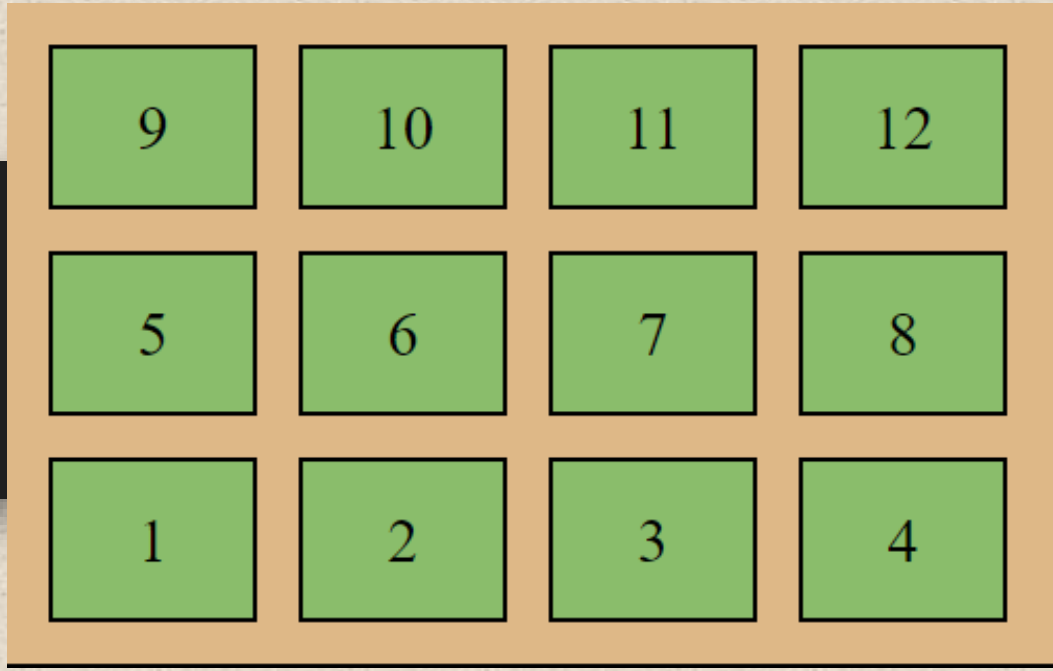
Flexbox 👉 Caja flexible

Propiedad **flex-wrap**

Hace que las cajas mantengan su tamaño y no que se adapten al tamaño de pantalla. Si no caben en la fila se desplazan a la siguiente.

```html
<div class="flex-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
    <div>7</div>
    <div>8</div>
    <div>9</div>
    <div>10</div>
    <div>11</div>
    <div>12</div>
</div>
```

```css
.flex-container {
    display: flex;
    flex-wrap: wrap-reverse;
    background-color: burlywood;
    padding: 10px;
```



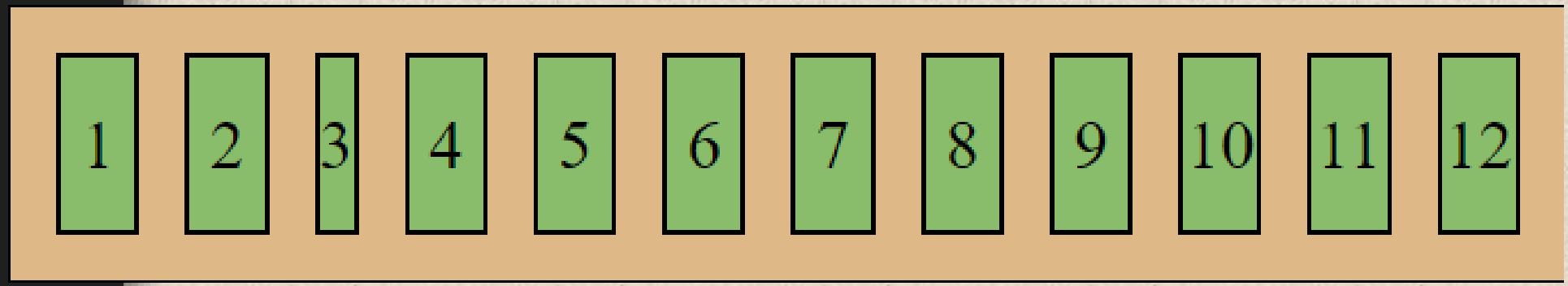
Flexbox 👉 Caja flexible

Propiedad **flex-shrink**

Mientras mas grande sea su valor mas rápido perderá espacio y más rápido se encogerá la caja.

```html
<div class="flex-container">
    <div>1</div>
    <div>2</div>
    <div class="sh">3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
    <div>7</div>
    <div>8</div>
    <div>9</div>
    <div>10</div>
    <div>11</div>
    <div>12</div>
</div>
```

```css
.sh {
    flex-shrink: 5;
}
```



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Flexbox 👉 Caja flexible

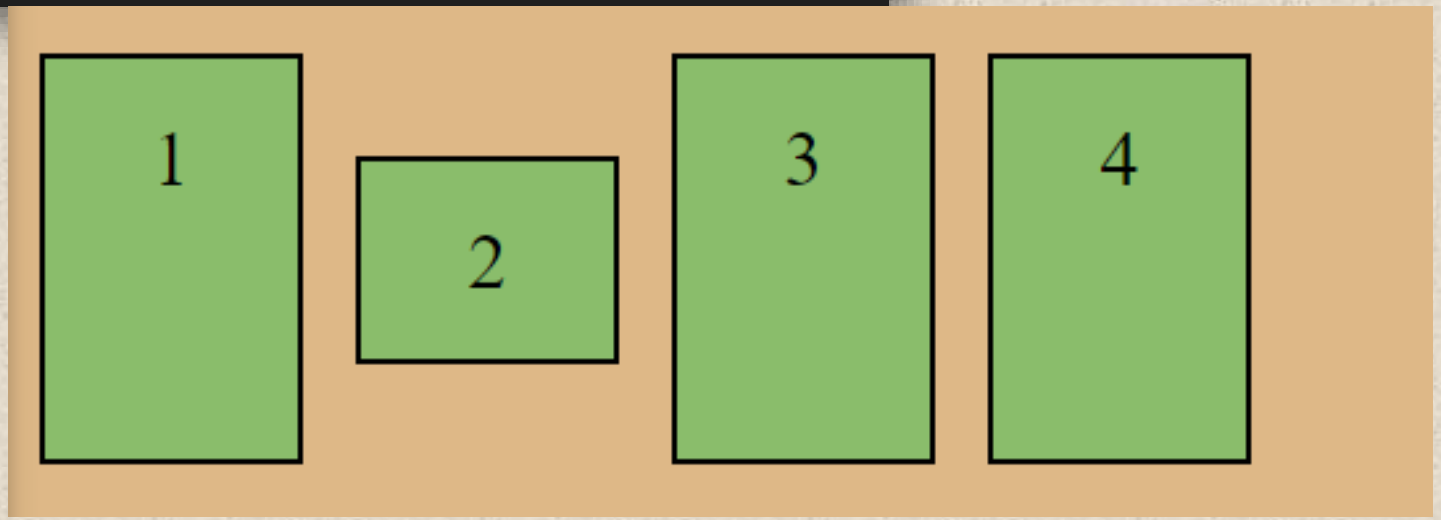
Propiedad **align-slef**

Esta propiedad sobreescribe la propiedad **align-ítems** del *container*.

```html
<div class="flex-container">
    <div>1</div>
    <div class="as">2</div>
    <div>3</div>
    <div>4</div>
</div>
```

```css
.flex-container {
    display: flex;
    background-color: burlywood;
    padding: 10px;
    border: 2px solid black;
    justify-content: start;
    align-content: center;
    height: 200px;
}
```

```css
.as {
    align-self: center;
}
```



MCC

Grid System 👉 Rejilla

CSS grid emplea dos unidades diferentes: contenedores y elementos. El contenedor es el nivel superior y en él se definen las propiedades que luego tendrán todos los elementos. Desde un punto de vista jerárquico, un elemento está dentro de un contenedor. Además de eso, se sigue necesitando HTML para el diseño de la rejilla o grid. En la parte HTML del código fuente, se crean los elementos individuales (texto, gráficos, etc.), que luego son introducidos dentro de CSS grid y se disponen en la posición correcta.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Grid System 👉 Rejilla

Este módulo trabaja de forma *bidimensional*: Al igual que flexbox, tenemos columnas y filas, pero ambos elementos pueden ser definidos a la vez.

Se le asigna este comportamiento mediante la propiedad **CSS**:

display:grid;

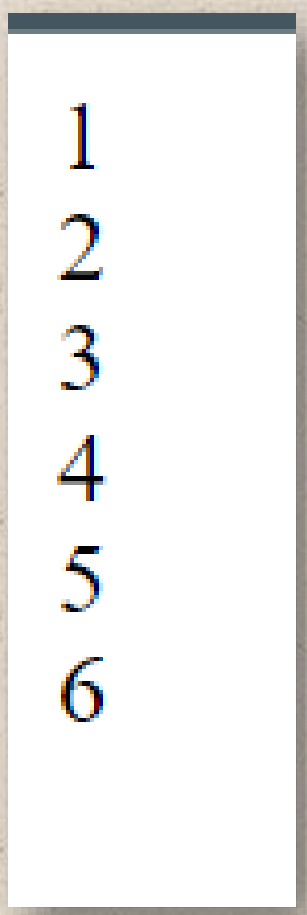
```html
<div class="grid-container">
    <div class="grid-item1">1</div>
    <div class="grid-item2">2</div>
    <div class="grid-item3">3</div>
    <div class="grid-item4">4</div>
    <div class="grid-item5">5</div>
    <div class="grid-item6">6</div>
</div>
```

EJEMPLO 👉

```css
.grid-container {
    display: grid;
}
```

Grid System 👉 Rejilla

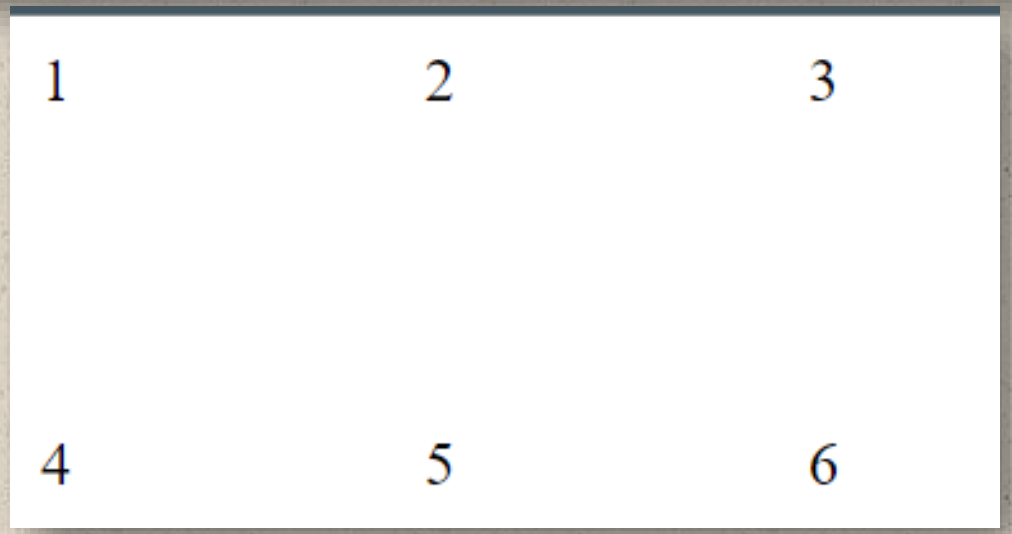
En el ejemplo anterior, hemos creado seis elementos, definimos cada uno de ellos como *grid-item* y los empaquetamos en el *grid-container*. Para activar CSS grid, debemos iniciar la función en el contenedor con **display: grid**. El código generará únicamente 6 números, que aparecerán uno debajo del otro. Después de crearlos, pueden colocarse con relativa libertad en la pantalla.



Grid System 👉 Rejilla

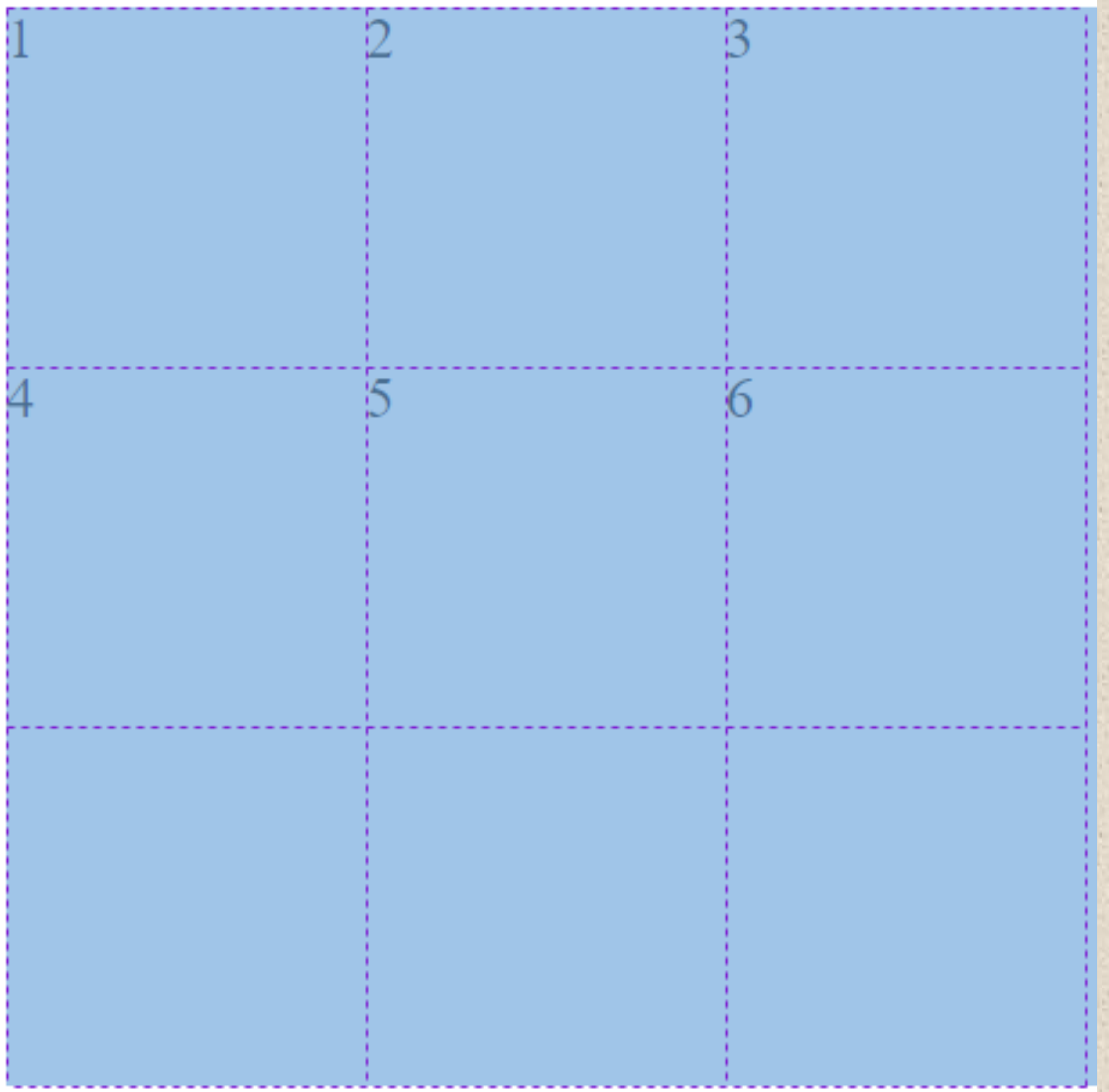
Todavía no tenemos el efecto rejilla. Para ello agregamos los **grid-template-**, tanto para las filas como para las columnas.

```css
.grid-container {
    display: grid;
    grid-template-rows: 100px 100px 100px;
    grid-template-columns: 100px 100px 100px;
}
```



Grid System 👉 Rejilla

Con la última imagen se nos pueden generar algunas dudas debido a que solo vemos los 6 ítems que pusimos en el HTML. Inspeccionando la página veremos **grid** con mayor claridad:

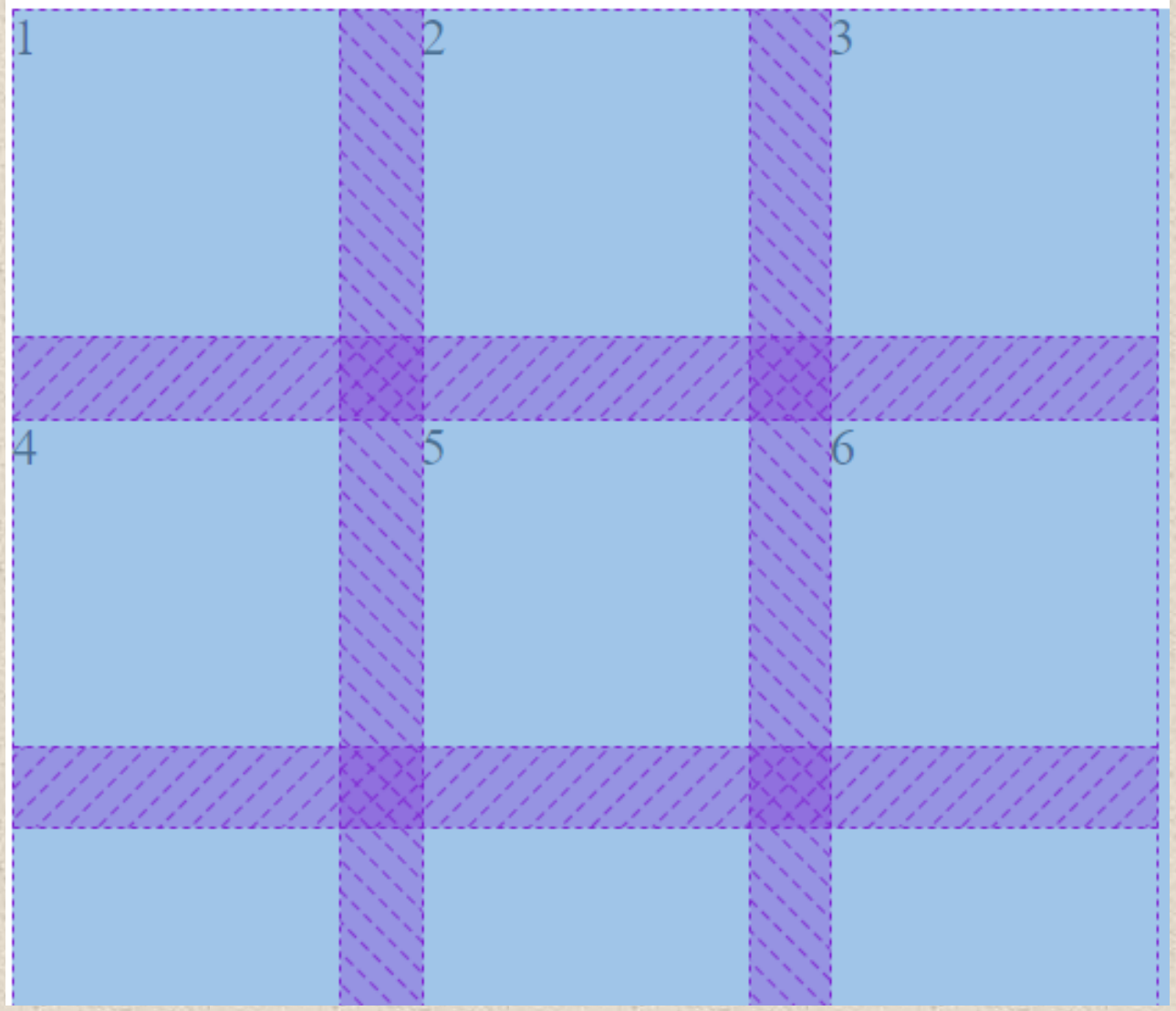


UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Grid System 👉 Rejilla

Con **grid-gap** agregamos una separación entre los cuadros de la rejilla (también llamado espacio vacio):

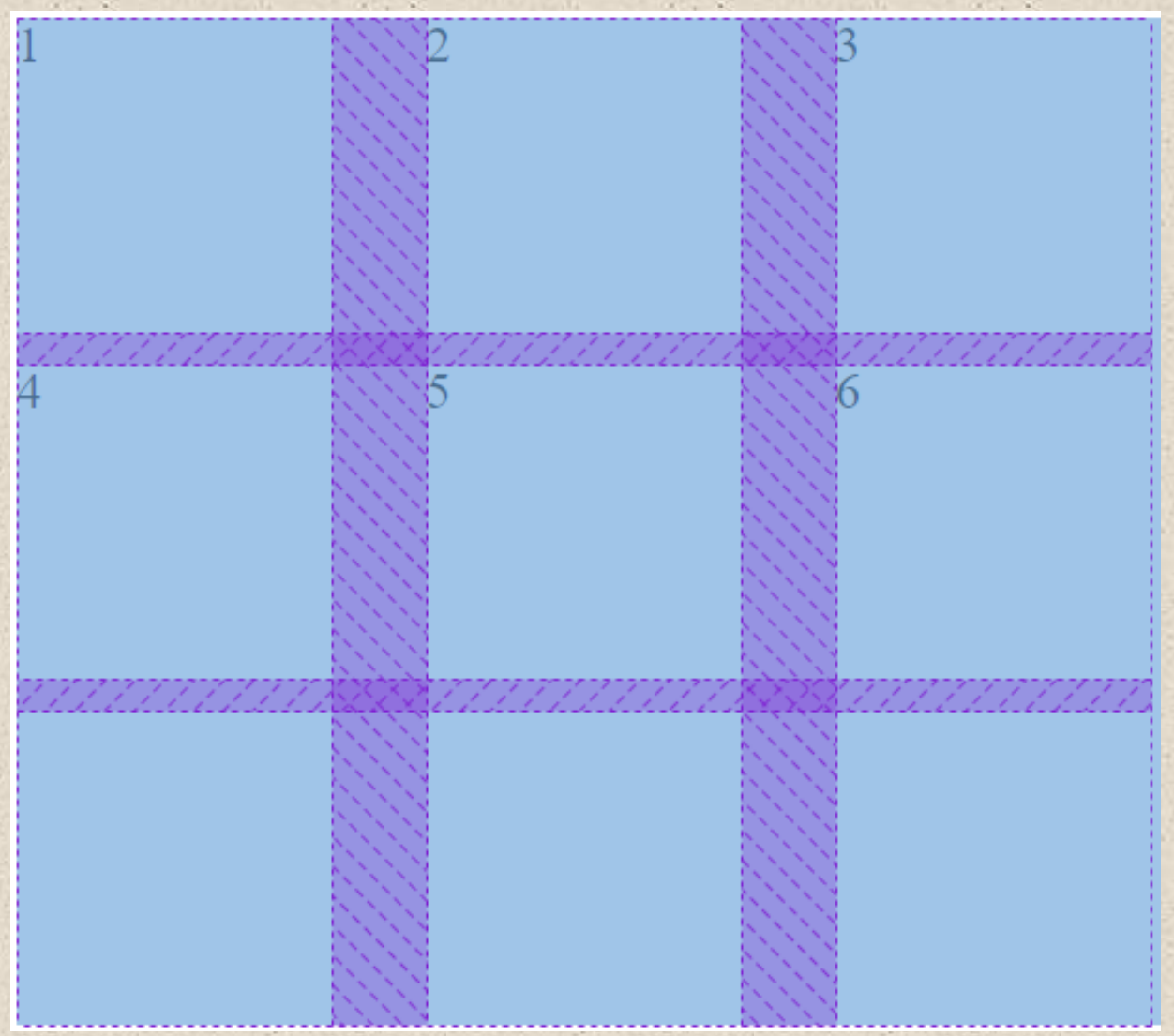
```
.grid-container {
    display: grid;
    grid-template-row
    grid-template-col
    grid-gap: 25px;
}
```



Grid System 👉 Rejilla

En el ejemplo anterior, los espacios entre cuadros son uniformes. Podemos personalizarlos utilizando **grid-row-gap** y **grid-column-gap**:

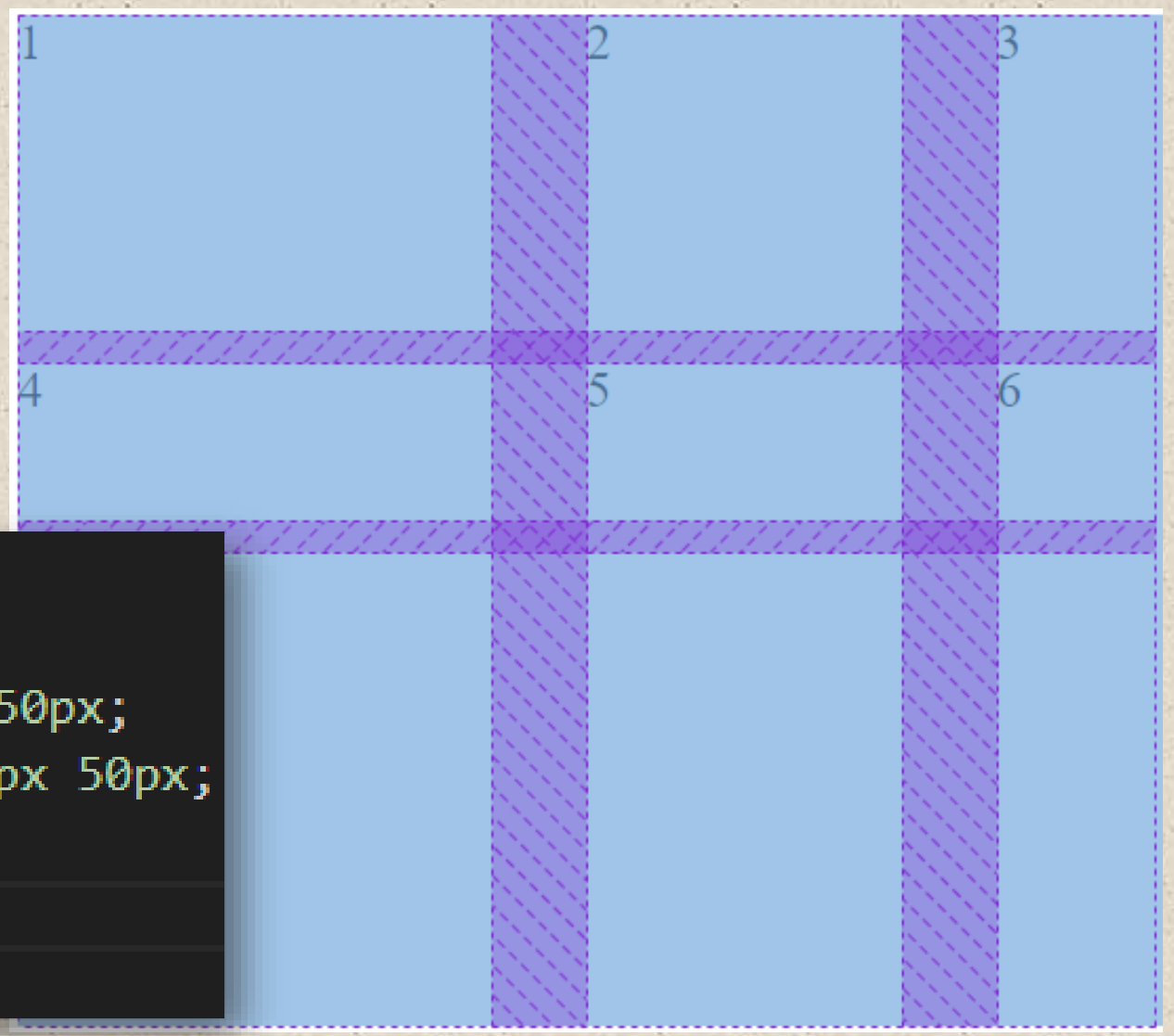
```
.grid-container {
    display: grid;
    grid-template-rows: 100p
    grid-template-columns: 1
    grid-row-gap: 10px;
    grid-column-gap: 30px;
}
```



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Grid System 👉 Rejilla

Así como variamos el espacio entre cuadros, también podemos variar el tamaño de los cuadros:



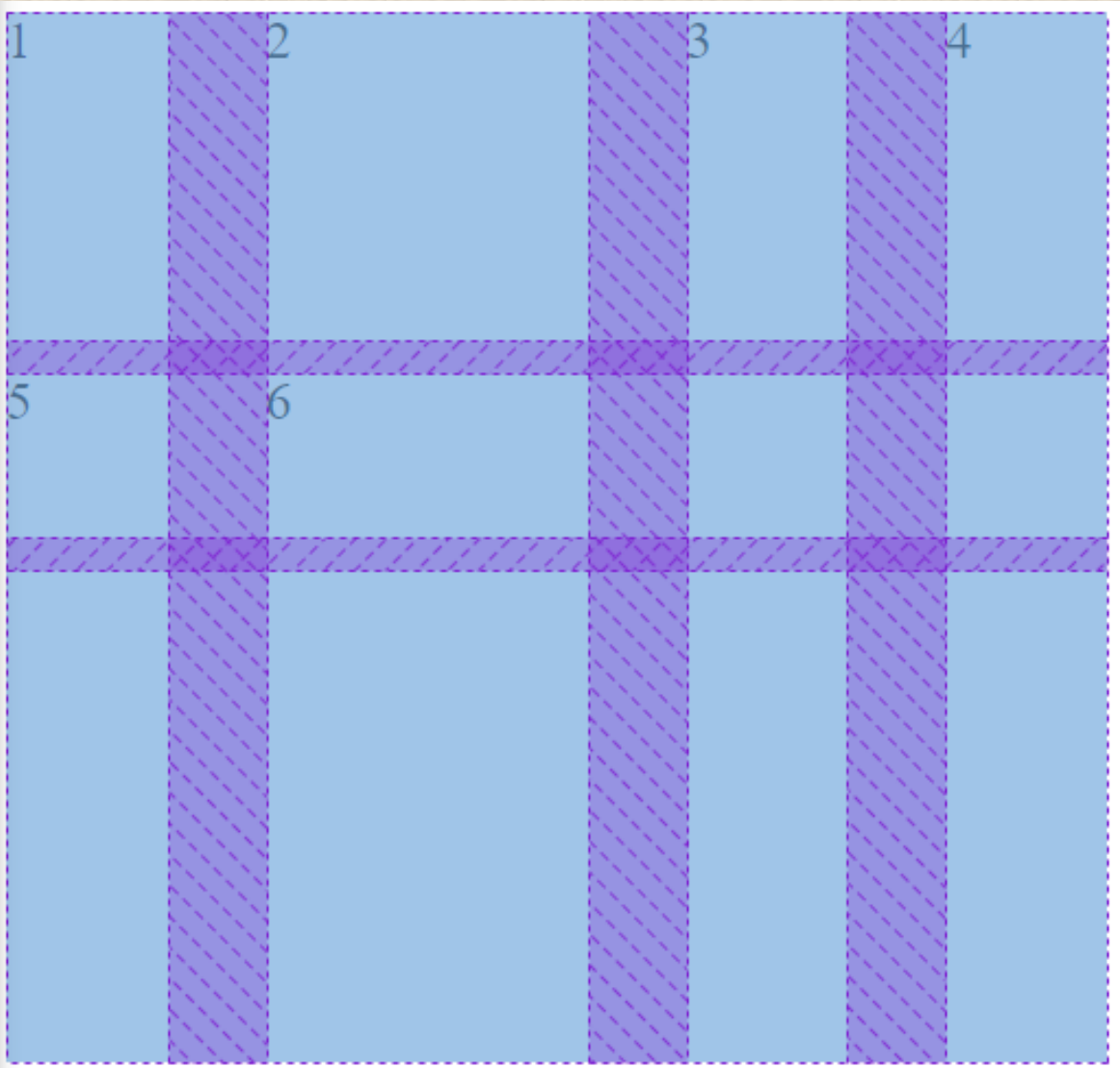

```
.grid-container {
    display: grid;
    grid-template-rows: 100px 50px 150px;
    grid-template-columns: 150px 100px 50px;
    grid-row-gap: 10px;
    grid-column-gap: 30px;
}
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Grid System 👉 Rejilla

Esto último también se puede hacer con fracciones o porcentajes:

```
.grid-container {
    display: grid;
    grid-template-rows: 100px 50px 150px;
    grid-template-columns: 1fr 2fr 1fr 1fr;
    grid-row-gap: 10px;
    grid-column-gap: 30px;
}
```



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Grid System 👉 Rejilla

Disposición de los elementos:

Después de definir la rejilla, se colocan en ella los elementos. Para esto, primero debemos crear los elementos y también especificar los valores de inicio y fin. No todos los elementos tienen que ocupar necesariamente una sola celda dentro de la cuadrícula.

VAMOS AL CÓDIGO Y REALICEMOS UN EJEMPLO

👩‍💻 👩‍💻

Flexbox y Grid

	Flexbox	CSS Grid
Aprendizaje	Esfuerzo inicial moderado	Esfuerzo inicial alto
Ordenación	Control de ordenación en una dimensión	Control de ordenación en dos dimensiones
Uso ideal	Menús de navegación	Cuadrículas de fotos de diferentes tamaños y proporciones
Relación con el contenido	Funciona bien con contenido dinámico que se va ordenando según las reglas creadas	Funciona bien con contenido fijo, permitiendo más control sobre el diseño visual
Superposición	No está preparado para superponer elementos	Preparado para superponer elementos según la ordenación que les demos