«Talento Tech»

Desarrollo Web 1

Clase 07

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 7 | Conceptos básicos

Temario:

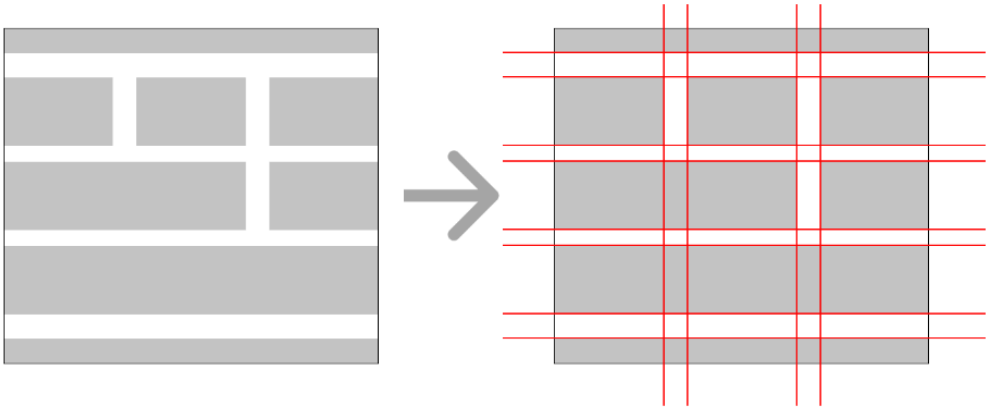
- Construcción de grillas
- Columnas, filas y gap: medidas y espaciados para las grillas
- Justificando y Alineando: elementos en la grilla
- BONUS: Pseudo-Clases

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Sistema de grillas

Las grillas son sistemas o estructuras de diseño que nos permiten organizar y distribuir elementos en una página de manera ordenada y flexible. Se basan en una estructura de filas y columnas que proporciona un marco para alinear y posicionar los diferentes elementos de contenido. Las grillas nos ayudan a crear diseños balanceados y proporcionales, facilitando la creación de diseños responsivos que se adaptan a diferentes tamaños de pantalla.

Supongamos que tenemos una web con este diseño.



Buenos Aires aprende
Agencia de Habilidades para el Futuro

Podemos pensar que podemos armar una grilla con 3x5 (tres columnas y cinco filas), y agregar espacio entre ellas. Luego distribuir en ella los diferentes elementos, como div, textos o imágenes que componen el sitio.

Cómo construimos una grilla.

Al igual que en flexbox, para armar una grilla precisamos un contenedor padre tenga la propiedad display:grid; para indicarle al contenedor que debe comportarse como grilla y que sus elementos hijos se acomoden dentro de ella. Para eso, cada elemento dentro de ese contenedor grilla tendrá sus propias coordenadas para saber dónde ubicarse.

Proceso paso por paso:

1. Creamos un contenedor con la propiedad **display: grid**;

2. Asignamos la cantidad de columnas que va a tener la grilla escribiendo la propiedad grid-template-columns. Por ejemplo:
   grid-template-columns: 100px 100px 100px;
   La cantidad de valores que se escriban en esta propiedad, será la cantidad de columnas que tenga la grilla. En este ejemplo tendrá tres columnas de 100px de ancho cada una.

3. Lo mismo para designar la cantidad de columnas de la grillas. Por ejemplo:
   grid-template-rows: 50px 150px 100px 50px 20px;
   En este caso creamos filas con diferentes anchos.

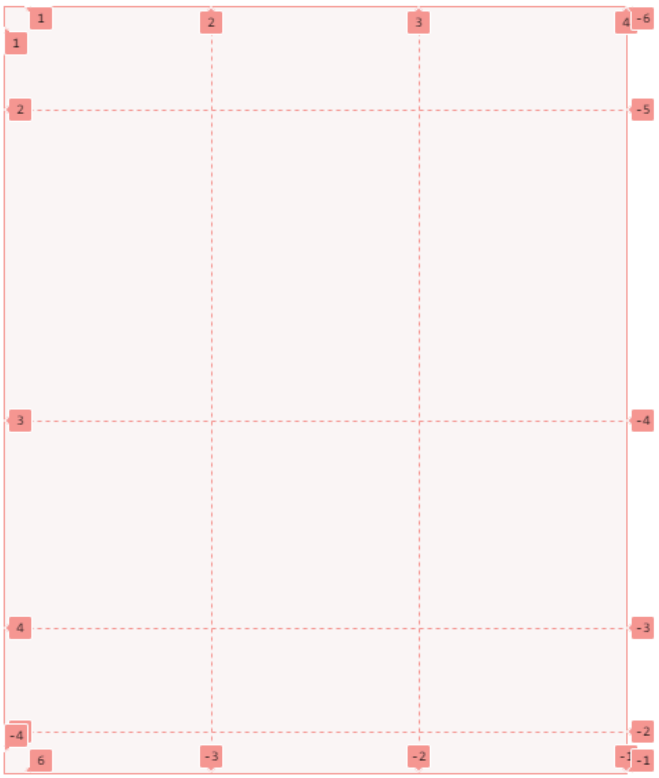
El código css sería así:

```css
.contenedor {
    display:grid;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows: 50px 150px 100px 50px 20px;
}
```

Para poder ver el resultado de la grilla, precisamos hacerla visible en el navegador. Al no tener elementos dentro del contenedor, la grilla está vacía, y por ende, invisible. Pero podemos ver su esqueleto.

Lo que se ve como resultado es la grilla vacía con sus respectiva cantidad de filas y columnas declaradas anteriormente.

Por cada línea divisoria veremos unos números. Esos números nos ayudarán a ubicar elementos dentro de la grilla y servirán como coordenadas.



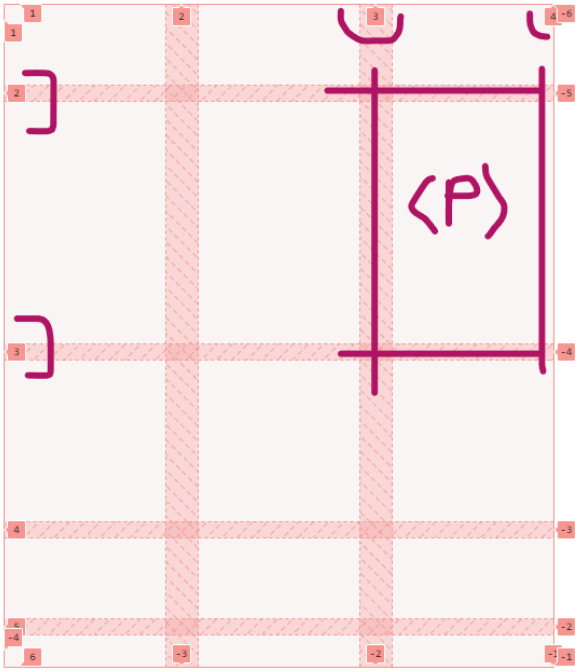
4. Por último, podemos elegir el espacio que puede existir entre las columnas y filas generando un espacio entre sí (muy similar a margin, pero exclusivo para el sistema de grillas). Para lograr esto se utiliza:
   column-gap: 20px; Para dar espacios entre columnas, es decir, de forma vertical.
   row-gap: 10px; Para dar espacios entre filas, es decir, de forma horizontal. El código completo se vería así:

Buenos Aires aprende
Agencia de Habilidades para el Futuro

```
styles.css > ...
1   .contenedor {
2       display:grid;
3       grid-template-columns: 100px 100px 100px;
4       grid-template-rows: 50px 150px 100px 50px 20px;
5       column-gap: 20px;
6       row-gap: 10px;
7   }
8
9
```

Concepto de líneas

Los canales en rosa más oscuro son los espacios entre filas y columnas.



Buenos Aires aprende
Agencia de Habilidades para el Futuro

Ahora pensemos que queremos introducir un párrafo (o cualquier otro elemento) dentro de la grilla.

Una vez elegido el “cuadrante” o área, debemos tener en cuenta entre qué número de líneas está.

Vertical, entre las líneas 3 y 4.
Horizontal, entre las líneas 2 y 3.

¿Cómo le damos esas coordenadas al elemento?

Colocando elementos

Para cada elemento que queramos ubicar en la grilla se le deben dar las coordenadas, es decir, decirle entre qué líneas se ubicará el elemento de forma horizontal en relación a las filas, y vertical, en relación a la columna.
El código en **CSS** sería algo así:

```
p{
    grid-column: 3/4;
    grid-row:2/3;
}
```

Lo que tenemos que pensar es qué línea comienza y termina la **columna (3 y 4)**, y en qué línea comienza y termina la **fila (2 y 3)**.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

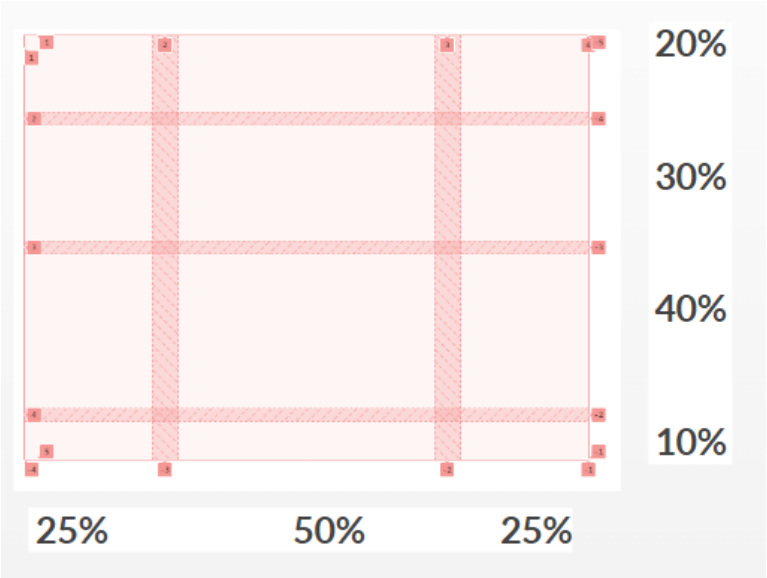
Medidas de las grillas:

Hasta ahora estuvimos trabajando con la unidad de medida px, lo cual no nos permite mucha flexibilidad en el diseño de nuestras grillas. Para obtener diseños más dinámicos y adaptables exploraremos las posibilidades que brindan las unidades de medida **fr**(fracciones) **y %** (porcentaje) al construir grillas.
Por ejemplo, si deseamos crear dos columnas y queremos que cada una ocupe la mitad del contenedor, podemos establecer un ancho del 50% para cada columna utilizando la unidad de medida en porcentaje.

Porcentaje:

Los porcentajes siempre son en relación al tamaño del contenedor. Si un contenedor ocupa **600px** dentro del mismo generamos una grilla, los porcentajes se acomodarán a esas medidas.

```
# styles.css > p
1   .contenedor {
2       height: 300px;
3       width: 400px;
4       display:grid;
5       grid-template-columns: 25% 50% 25%;
6       grid-template-rows: 20% 30% 40% 10%;
7       column-gap: 20px;
8       row-gap: 10px;
9   }
10
```

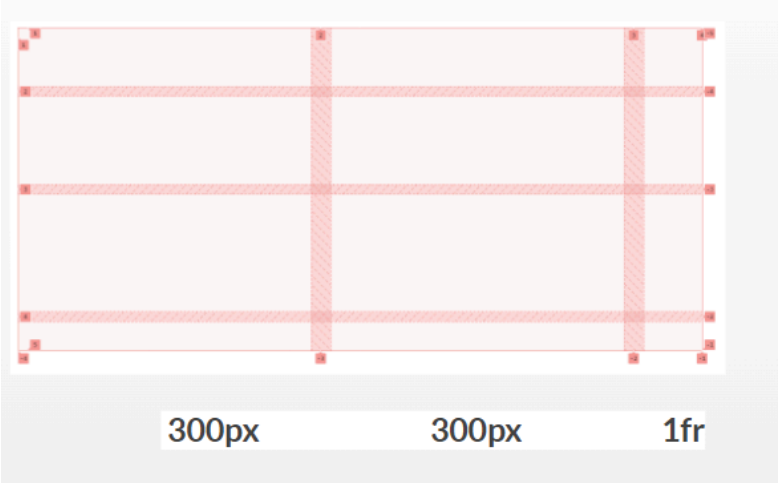


Fracciones

Los **fr** son una unidad trabajada en grids que se maneja en fracciones de pantalla o en espacios libres. Por ejemplo, si tenemos un contenedor que mide **700px** de ancho, y le asignamos dos columnas de **300px**, los **100px** restantes que no asignamos será “el espacio libre”, es decir, **1fr** o una fracción.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

```css
.contenedor {
    height: 300px;
    width: 700px;
    display:grid;
    grid-template-columns: 300px 300px 1fr;
    grid-template-rows: 20% 30% 40% 10%;
    column-gap: 20px;
    row-gap: 10px;
}
```



Las fracciones son unidades flexibles y super útiles al momento de trabajar con grillas. Si usamos **fr** para definir el tamaño de las columnas, el espacio se dividirá proporcionalmente dependiendo del valor asignado.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Alineando el Contenido

Hasta el momento hemos aprendido a crear grillas asignando cantidad de filas y columnas. Aprendimos a asignarles tamaños desde diferentes unidades de medida, aprendimos a darle espacio con el uso de gap, e incluso, conociendo sus líneas, logramos comprender cómo colocar elementos dándole coordenadas.

Ahora, sólo nos queda organizar los elementos allí colocados para que queden visualmente más equilibrados. Para eso existen las propiedades **justify-ítems y align-ítems**

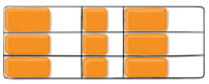
Justify-items alinea los elementos en el eje x o de manera horizontal
Align-items alinea los elementos en el eje y o de manera vertical
Justify-self: permiten alinear a cada elemento hijo de manera individual.

Estas propiedades se le asignan al contenedor que genera la grilla, y afecta a todos los hijos que tiene, muy similar a lo que pasa con su flexbox.

Justify-items y Align items:

Justify-items alinea los elementos en el eje x o de manera horizontal. Acepta los siguientes valores.

```
.container {
  justify-items: start;
}
```



```
.container {
  justify-items: end;
}
```

```
.container {
  justify-items: center;
}
```

```
.container {
  justify-items: stretch;
}
```

Align-items alinea los elementos en el eje y o de manera vertical. Acepta los siguientes valores:

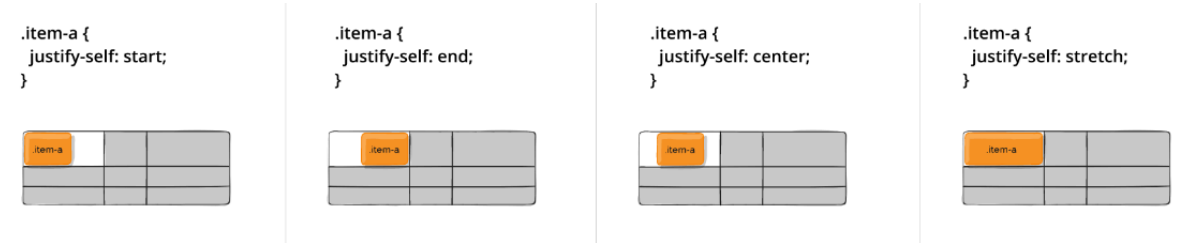
Justify-self y Align-self:

¿Qué pasa si queremos alinear un elemento particular en otra dirección distinta a los otros elementos así como existen?

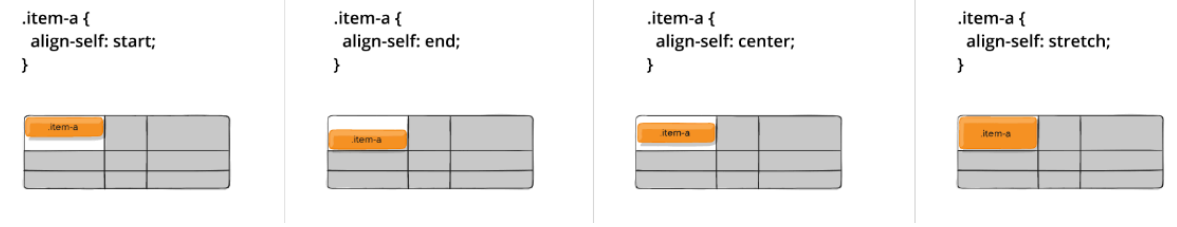
Las propiedades justify-self y align-self permiten alinear a cada elemento hijo de manera individual.

Es importante recordar que esta propiedad se aplica sobre el elemento a ser alineado, no sobre el contenedor.

justify-self



align-self



Grilla Responsive

No todas las pantallas tienen el mismo tamaño, y no todos los usuarios van a ingresar desde un mismo tipo o tamaño de dispositivo, por lo tanto el diseño de nuestro sitio debe “responder” independientemente desde donde se lo visualice asegurando la correcta visibilidad y presentación de la información que tiene todo sitio web. Siempre debemos priorizar la experiencia del usuario y su comodidad visual para que éste permanezca en el

sitio web el tiempo necesario para que adquiera el servicio o producto ofrecido.. En otras palabras, si queremos que las personas permanezcan en nuestro sitio, debemos adaptarlo a todo tipo de tamaño de pantalla. Quedaría muy incómodo ver un sitio de gran tamaño en una pantalla de celular, o una página muy pequeña en un monitor grande.

Las grillas sirven para que nuestro sitio sea responsivo y facilitan su adaptación al modificar la cantidad de filas y columnas teniendo en cuenta el tamaño de pantalla del dispositivo. De esa manera se distribuyen y ordenan los elementos.

Para saber más sobre el sistema responsivo te invitamos a ver las siguientes presentaciones.

En la clase anterior conocimos las media queries. Para las grillas funciona de la misma manera. Para hacer esta práctica, vamos a utilizar **https://cssgrid-generator.netlify.app/**. Este generador de grillas nos va a ayudar a generar grillas completas en cuestión de minutos y nos ofrecerá un código listo. ¡Pero ojo!, es importante leer y entender ese código antes de aplicarlo.

Grid-Generator

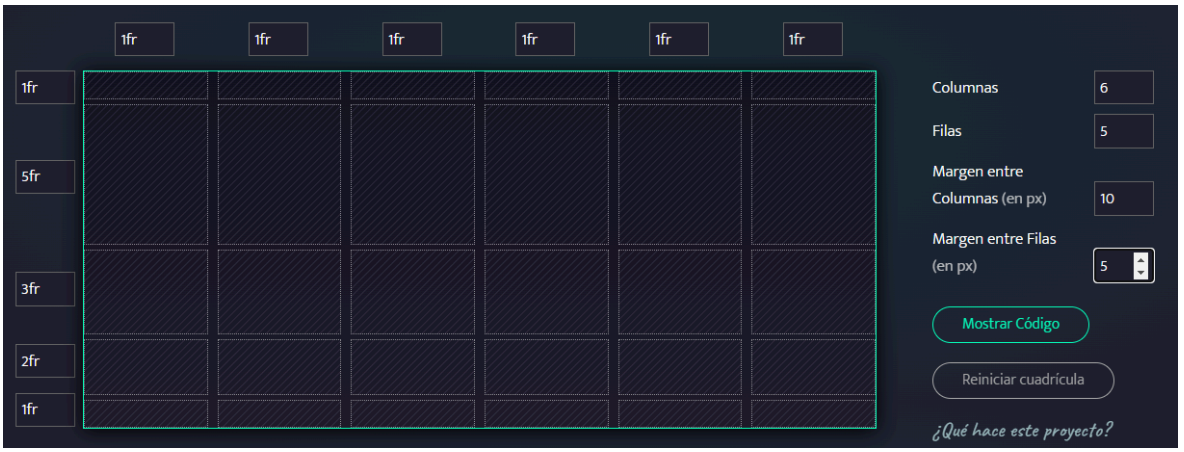
Este generador de grillas nos permitirá generar un layout en cuestión de minutos. Ingresá a https://cssgrid-generator.netlify.app/ y conozcamos su pantalla.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Paso a paso

1. El primer paso es crear la grilla con la cantidad de filas y columnas que se requiera.
2. Luego, haciendo clic en cada input fr, cambiar la medida que se desee.
3. Determina el gap (recuerda que es en pixeles).
4. Esto se configura con los inputs del lado derecho de la pantalla.

¡Inventemos una grilla!



Una vez creada la grilla, ya se pueden introducir los div o contenedores. Para eso se hace un clic en cada cuadrante. Esto depositara un div en cada espacio.

Para que un div ocupe más de un cuadrante se debe hacer clic sostenido, y sin soltar, recorrer todos los cuadrantes que se quieren ocupar.

Haciendo clic en **“Mostrar código”**, podremos ver el **HTML y CSS** de la grilla creada.

```html
<div class="parent">
  <div class="div1"> </div>
  <div class="div2"> </div>
  <div class="div3"> </div>
  <div class="div4"> </div>
  <div class="div5"> </div>
  <div class="div6"> </div>
  <div class="div7"> </div>
  <div class="div8"> </div>
</div>
```

```css
.parent {
  display: grid;
  grid-template-columns: repeat(6, 1fr);
  grid-template-rows: 1fr 5fr 3fr repeat(2, 1fr);
  grid-column-gap: 10px;
  grid-row-gap: 5px;
}
.div1 { grid-area: 1 / 1 / 2 / 7; }
.div2 { grid-area: 2 / 1 / 3 / 3; }
.div3 { grid-area: 2 / 3 / 3 / 5; }
.div4 { grid-area: 2 / 5 / 3 / 7; }
.div5 { grid-area: 3 / 1 / 4 / 4; }
.div6 { grid-area: 3 / 4 / 4 / 7; }
.div7 { grid-area: 4 / 1 / 5 / 7; }
.div8 { grid-area: 5 / 1 / 6 / 7; }
```

Vamos a ver cosas nuevas en este código. Analicemos en detalle:

¿Qué significa repeat(6, 1fr)?
Significa que se crearán 6 columnas iguales de una fracción cada una.
Repetir 6 columnas de una fracción”

¿Qué significa .div1 { grid-area: 1 / 1 / 2 / 7; } ?
La propiedad grid-area se utiliza para especificar la ubicación y el tamaño del elemento dentro de una grilla. En este caso, la declaración 1 / 1 / 2 / 7 define los siguientes valores:
El primer valor 1 indica que el elemento comienza en la primera fila de la grilla.
El segundo valor 1 indica que el elemento comienza en la primera columna de la grilla.
El tercer valor 2 indica que el elemento termina en la segunda fila de la grilla.
El cuarto valor 7 indica que el elemento termina en la séptima columna de la grilla.
Es otra manera de ubicar elementos dentro de una grilla.

¡Vamos al proyecto!

Llevemos estos códigos al proyecto.

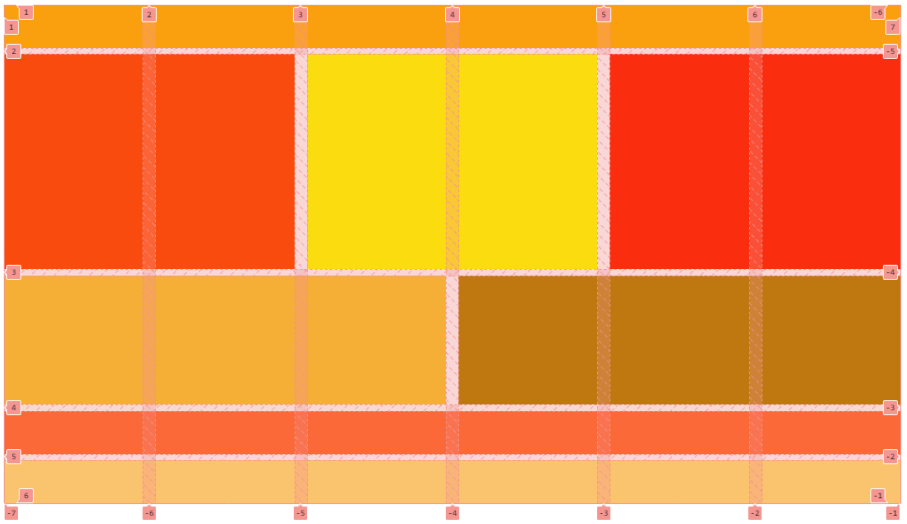
Te recomendamos darle una **height** (altura) al div contenedor para poder ver los resultados. Recordá que la grilla y los div que contiene están vacíos.

Tip: comienza a experimentar en un proyecto nuevo para no afectar el desarrollo que llevas hasta ahora.

Para poder visualizar cada div, te proponemos colorearlos. ¿Cómo? Colocando un **background-color** a cada uno de ellos.

Ejemplo:

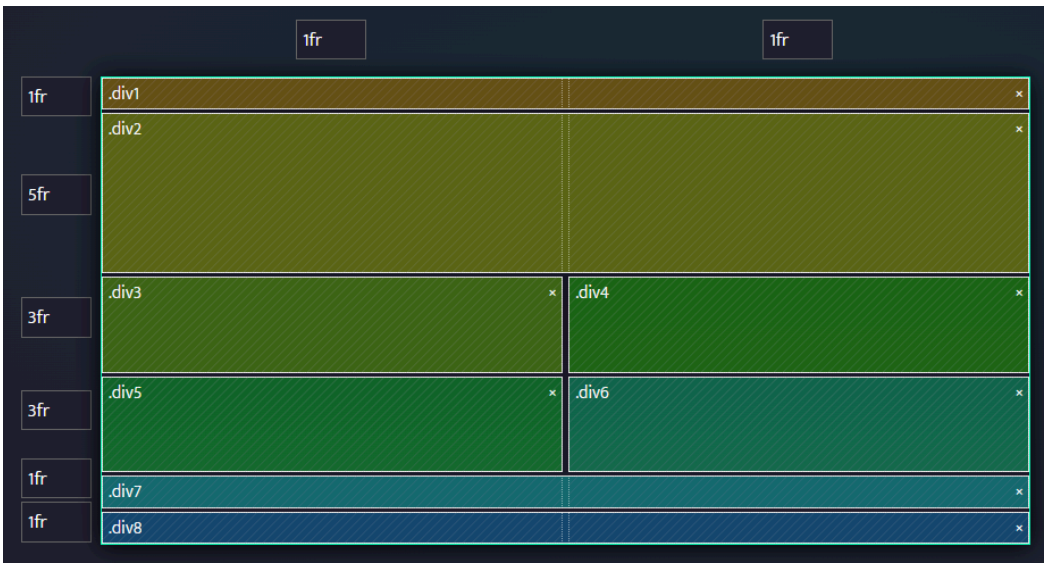
```
.div1 {
  grid-area: 1 / 1 / 2 / 7;
  background-color: orange;
}
```



Ya creamos una grilla completa y le colocamos div vacíos para ver la disposición de los mismos. Para lograr que esta grilla cambie la disposición de sus filas y columnas, y además, cambié la distribución de los div, debemos seguir estos pasos:

- Crear una nueva grilla para un tamaño de pantalla más pequeño
- Crear una nueva distribución de TODOS los div en la nueva grilla. Si en el ejemplo anterior hay ocho divs, la misma cantidad debe haber en la grilla más pequeña.
- Integrar estas nuevas propiedades del contenedor dentro de una media query.

Dibujamos una nueva grilla con una cantidad de filas y columnas diferente, ajustamos sus medidas (los fr), y colocamos en ella los ocho div anteriores.



Copiamos los nuevos códigos **CSS** que nos ofrece dentro de una media query. Ajustamos lo que creamos necesario, ya sea color, una nueva distribución, etc.

```css
@media (max-width: 922px) {
  .parent {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    grid-template-rows: 1fr 5fr repeat(2, 3fr) repeat(2, 1fr);
    grid-column-gap: 5px;
    grid-row-gap: 3px;
  }

  .div1 {
    grid-area: 1 / 1 / 2 / 3;
```

Probamos la grilla con el inspector, achicando la pantalla para ver si cambia la grilla.

¡Y listo!
¡Aprendiste a hacer una grilla responsiva!

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Conclusiones

Flexbox y grids son herramientas de suma importancia en el diseño y desarrollo de sitios web responsivos. De acá en adelante repensá el diseño de tu proyecto en base a una pantalla más pequeña como por ejemplo, celular.

1. ¿Qué otros dispositivos podrían mostrar páginas web?
2. ¿Qué otros tamaños de pantalla existen?
3. Con los nuevos celulares que se pliegan, ¿Qué tamaño corresponde?

Esta y muchas otras preguntas te las dejamos para investigar y descubras un mundo de posibilidades con la cantidad de tamaños de pantallas que existen hoy.

Información adicional:

Medidas de pantalla para tus **Media Queries:**

No existe un estándar 100% definido, pero lo mas común es utilizar **320px, 480px, 768px y 1024px**

Ejercicio Práctico:

- Ingresa en el siguiente enlace y analizá el código propuesto:

  https://stackblitz.com/edit/web-platform-vn3ybp?file=index.html

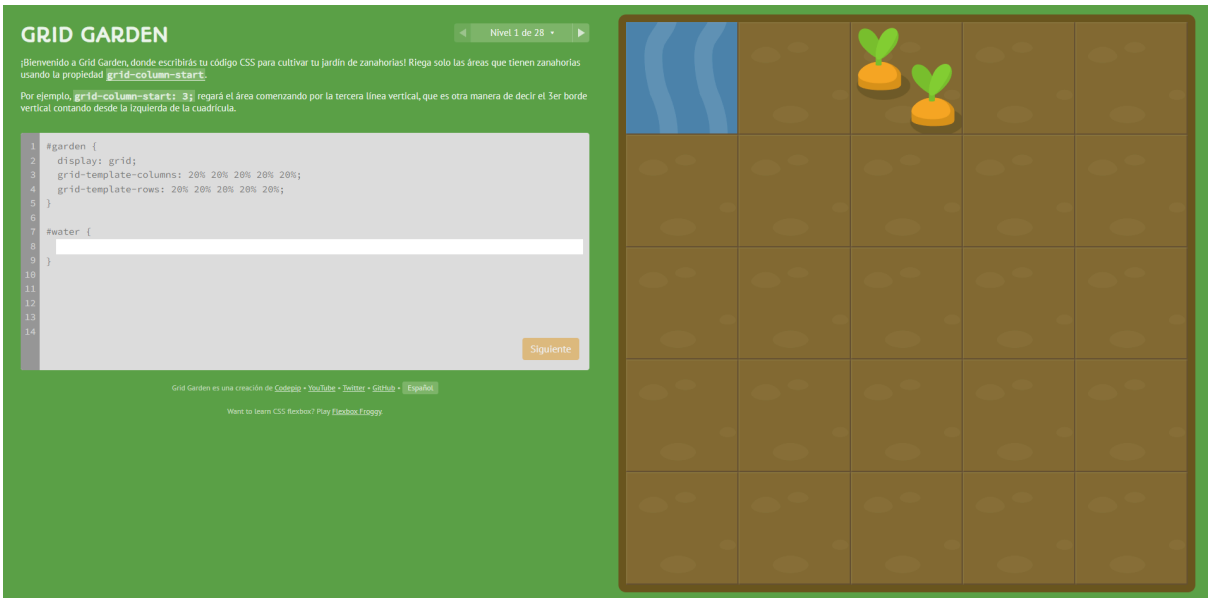
Encontrarás 2 contenedores grid.

- Analizá su funcionamiento y cómo se comportan en diferentes tamaños de pantalla.
- Agregá y quita elementos para modificar el layout. Experimentá sin miedo para conocer a fondo las herramientas antes de aplicarlas en tu proyecto.

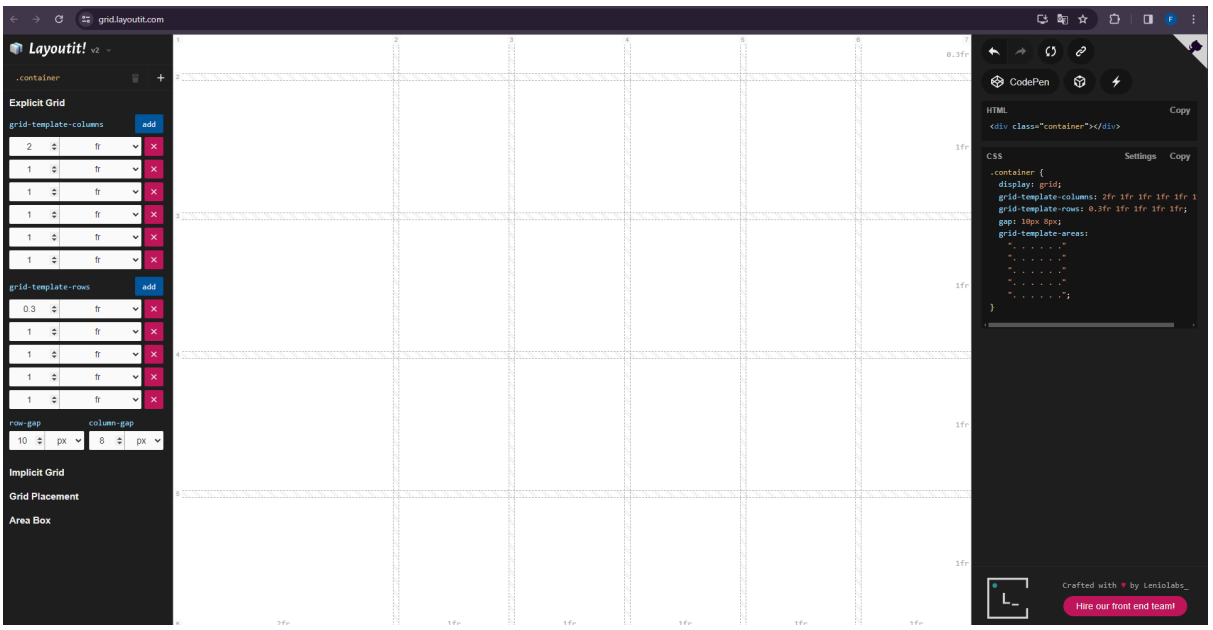
Buenos Aires aprende
Agencia de Habilidades para el Futuro

Practicá jugando:

Practicá jugando **Grid Garden** 🥕
https://cssgridgarden.com/#es



Revisá y probá Grid Layoutit para entender mejor los conceptos de grids con este generador https://grid.layoutit.com/



Links de interés:

http://cssgridgarden.com/#es
https://gridbyexample.com/examples/
https://github.com/rachelandrew/gridbugs
https://mozilladevelopers.github.io/playground/css-grid
https://css-tricks.com/getting-started-css-grid/
https://escss.blogspot.com.ar/2015/12/guia-css-grid-layout.html
https://neliosoftware.com/es/blog/css-grid-futuro-diseno-web-ya-esta-aqui/
https://css-tricks.com/video-screencasts/153-getting-started-with-css-grid/

Preguntas de reflexión

1. ¿Cuál es la diferencia entre Grid y Flexbox?
2. ¿Si uso flexbox puedo usar grids o viceversa?.

3. ¿Cómo sé cuándo usar grid-areas, fraction, y demás métodos de CSS Grid?

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Desafío N° 7:

Consigna:

- **Implementa un diseño responsive:** Garantizar que el sitio web se vea y funcione correctamente en una variedad de dispositivos y tamaños de pantalla, mejorando la experiencia del usuario.
- **Usa técnicas modernas de diseño:** Aplicar las propiedades de CSS Grid o Flexbox para crear una estructura de diseño flexible y adaptable, permitiendo una distribución eficiente de los elementos en la página.
- **Mejorar la usabilidad en dispositivos móviles:** Asegurar que la navegación y la presentación de contenido sean intuitivas y eficientes en dispositivos móviles, Evitar barra de desplazamiento lateral

Instrucciones:

- **Identificación de contenedores y elementos clave:** Analizar la estructura actual del sitio y determinar los contenedores y elementos clave que deben ajustarse para una mejor adaptabilidad.
- **Aplicar CSS Grid o Flexbox:** Elegir entre CSS Grid o Flexbox o ambos según las necesidades específicas del diseño y aplicar las propiedades correspondientes para crear una estructura flexible.
- **Establecer puntos de quiebre (media queries) para dispositivos diferentes:** Definir puntos de quiebre en los que el diseño se ajustará para adaptarse a distintos tamaños de pantalla, proporcionando una experiencia fluida en dispositivos desde escritorios hasta teléfonos móviles.

- **Ajustar estilos y diseño existentes:** Modificar los estilos CSS existentes para mejorar la presentación en diferentes dispositivos,

- **Probar en varios navegadores y dispositivos**: Realizar pruebas exhaustivas en el inspeccionar del navegador.


Buenos Aires
aprende
Buenos
Aires
Ciudad