Reglas At

https://developer.mozilla.org/es/docs/Web/CSS/At-rule
Una regla-at es una declaración CSS que comienza con el símbolo arroba, '@' (U+0040 COMMERCIAL AT), seguido por un identificador, e incluye todo el contenido hasta el siguiente punto y coma, ';' (U+003B SEMICOLON), o el siguiente bloque CSS, lo que sea primero.

- Hay varias reglas-at, designadas por sus identificadores, cada una con sintaxis distinta:
- @charset — Define el conjunto de caracteres usado por la hoja de estilos.
- @import — Indica al motor de CSS que incluya una hoja de estilos externa.

Reglas-at anidadas — Un subconjunto de declaraciones anidadas, que pueden ser usadas como declaraciones de estilos, así como grupos de reglas condicionadas internas:

- @media — Un grupo de reglas condicional que aplicará su contenido si el dispositivo cumple los criterios de las condiciones definidas usando un *media query*.
- @supports — Un grupo de reglas condicional que aplicará su contenido si el navegador cumple los criterios de la condición dada.
- @document — Un grupo de reglas condicionadas que aplicará su contenido si el documento donde se aplica la hoja de estilos cumple los criterios de la condición dada. *(diferida al Nivel 4 de la Especificación CSS)*
- @page — Describe los cambios en la disposición de la página que serán aplicados al imprimir el documento.
- @font-face — Describe la configuración de fuentes externas que se descargarán.
- @keyframes — Describe la configuración de pasos intermedios en una secuencia de animación CSS.
- @viewport — Describe los aspectos del viewport para dispositivos de pantalla pequeña. *(actualmente en Borrador)*
- @counter-style — Define estilos de contador específicos que no son parte de los conjuntos de estilos predeterminados. *(en estado de Recomendación Candidata, pero sólo implementada en Gekko al momento de esta publicación)*
- @font-feature-values (junto con @swash, @ornaments, @annotation, @stylistic, @styleset y @character-variant)
  — Define nombres comunes para la propiedad font-variant-alternates. *(en estado de Recomendación Candidata, pero sólo implementada en Gekko al momento de esta publicación)*

Media Queries:

https://developer.mozilla.org/es/docs/Web/CSS/%40media

https://desarrolloweb.com/articulos/css-media-queries.html

Las media queries son uno de los elementos clave para realizar un diseño web adaptable o adaptativo (*responsive web design*). Desde junio de 2012, son una recomendación (estándar) del W3C: Media Queries.

Formas de usar los media queries:
1. En la llamada al archivo CSS

Usando este método le indicamos al navegador que utilice el archivo CSS que estamos cargando solo si las condiciones se cumplen. Veamos un ejemplo:

```
<link rel="stylesheet" media="only screen and (max-width:
768px)" href="ejemplo.css" >
```

estamos cargando la hoja de estilos *ejemplo.css*, pero sus propiedades solo aplicarán si el sitio se está viendo en una pantalla, dentro de una ventana de máximo 768 pixeles de ancho.
El problema con este enfoque es que cada vez que queramos cargar estilos de acuerdo a ciertas condiciones, tendríamos que cargar un archivo nuevo, y eso, por supuesto, hace nuestro sitio más lento al añadir solicitudes HTTP adicionales.

2. Dentro del archivo CSS
Esta es la forma más recomendada, ya que nos permite tener múltiples comprobaciones, sin la necesidad de cargar diferentes archivos por cada una de las *media queries* que hagamos.

```
@media only screen and (max-width: 768px) {
  /* Aquí van todos los estilos CSS */
}
```

Operadores lógicos para las Media Queries
Para combinar diversas condiciones podemos usar los operadores lógicos, de una manera similar a como se usan en lenguajes de programación. Los que tenemos disponibles son:

- **Operador and:** las dos condiciones deben cumplirse para que se evalúe como verdadera.
- **Operador not:** es una negación de una condición. Cuando esa condición no se cumpla se aplicarán las media queries.
- **Operador only:** se aplican las reglas solo en el caso que se cunpla cierta circunstancia.
- **Operador or:** no existe como tal, pero puedes poner varias condiciones separadas por comas y cuando se cumpla cualquiera de ellas, se aplicarán los estilos de las media queries.

```
@media (max-width: 600px) and (orientation: landscape) {
    h1{
        color: red;
    }
}
```

orientation (landscape o portrait)

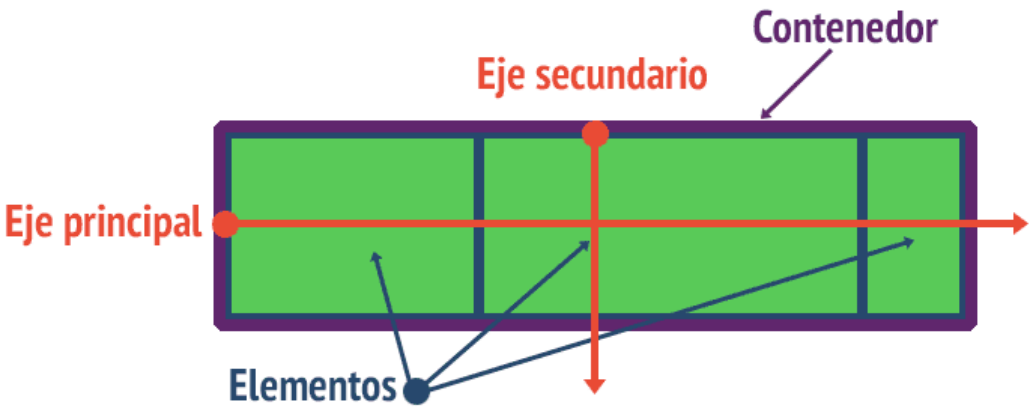
Cajas Flexibles (Flexbox)

https://developer.mozilla.org/es/docs/Web/CSS/CSS_Flexible_Box_Layout/Usando_las_cajas_flexibles_CSS

https://www.w3.org/TR/css-flexbox-1/

La propiedad Flexible Box, o flexbox, de CSS3 es un modo de diseño que permite colocar los elementos de una página para que se comporten de forma predecible cuando el diseño de la página debe acomodarse a diferentes tamaños de pantalla y diferentes dispositivos. Para muchas aplicaciones, el modelo "caja flexible" produce una mejora sobre el modelo "bloque"

porque no utiliza la propiedad `float`, ni hace que los márgenes del contenedor flexible interfieran con los márgenes de sus contenidos.



Contenedor flexible (Flex container)

El elemento "padre" que contiene los elementos flexibles. Un contenedor flexible se define usando los valores flex o inline-flex en la propiedad display.

inline-flex	Establece un contenedor de ítems flexible en línea, de forma equivalente a inline-block.
flex	Establece un contenedor de ítems flexible en bloque, de forma equivalente a block.

Elemento flexible (Flex item)

Cada hijo de un contenedor flex se convierte en un elemento flexible. Si hay texto directamente incluido en el contenedor flexible, se envuelve automáticamente en un elemento flexible anónimo.

flex-grow:	**0** | *[factor de crecimiento]*	Número que indica el crecimiento del ítem respecto al resto.
flex-shrink:	**1** | *[factor de decrecimiento]*	Número que indica el decrecimiento del ítem respecto al resto.
flex-basis:	*[tamaño base]* | **content**	Tamaño base de los ítems antes de aplicar variación.
order:	*[número]*	Número que indica el orden de aparición de los ítems

```
.item {
  /* flex: <flex-grow> <flex-shrink> <flex-basis> */
  flex: 1 3 35%;
}
```

Ejes

Cada diseño de "caja flexible" sigue dos ejes. El **eje principal** es el eje a lo largo del cual los elementos flexibles se suceden unos a otros. El **eje secundario** es el eje perpendicular al **eje principal**.

- **La propiedad `flex-direction` establece el eje principal.**

El atributo flex-direction es el que nos permitirá intercambiar el eje principal y secundario.

```
flex-direction: row;
flex-direction: row-reverse;
flex-direction: column;
flex-direction: column-reverse;
```

- **La propiedad `justify-content` define cómo los elementos flexibles se disponen a lo largo del eje principal en la línea en curso.**

```
justify-content: center;
justify-content: flex-start;
justify-content: flex-end;

/* Baseline alignment */
justify-content: baseline;
justify-content: first baseline;
justify-content: last baseline;

/* Distributed alignment */
justify-content: space-between; /* Distribute items evenly
                                   The first item is flush with the start,
                                   the last is flush with the end */
justify-content: space-around;  /* Distribute items evenly
                                   Items have a half-size space
                                   on either end */
justify-content: space-evenly;  /* Distribute items evenly
                                   Items have equal space around them */
justify-content: stretch;       /* Distribute items evenly
                                   Stretch 'auto'-sized items to fit
                                   the container */

/* Overflow alignment */
justify-content: safe center;
justify-content: unsafe center;
```

flex-start	Agrupa los ítems al principio del eje principal.
flex-end	Agrupa los ítems al final del eje principal.
center	Agrupa los ítems al centro del eje principal.
space-between	Distribuye los ítems dejando (el mismo) espacio entre ellos.
space-around	Distribuye los ítems dejando (el mismo) espacio a ambos lados de cada uno de ellos.

- **La propiedad `align-items` define cómo los elementos flexibles se disponen a lo largo del eje secundario de la línea en curso.**

```
/* Basic keywords */
align-items: normal;
align-items: stretch;

/* Positional alignment */
align-items: center; /* Pack items around the center */
```

```
align-items: flex-start; /* Pack flex items from the start */
align-items: flex-end; /* Pack flex items from the end */

/* Baseline alignment */
align-items: baseline;
align-items: first baseline;
align-items: last baseline; /* Overflow alignment (for positional
alignment only) */
align-items: safe center;
align-items: unsafe center;
```

justify-content: Se utiliza para alinear los ítems del **eje principal** (*por defecto, el horizontal*).

align-items: Usada para alinear los ítems del **eje secundario** (*por defecto, el vertical*).

- **La propiedad `align-self` define cómo cada elemento flexible se alinea respecto al eje secundario, y sustituye al valor por defecto establecido por `align-items`.**

justify-content:	flex-start | flex-end | center | space-between | space-around	Eje principal
align-content:	flex-start | flex-end | center | space-between | space-around | stretch	Eje principal
align-items:	flex-start | flex-end | center | stretch | baseline	Eje secundario
align-self:	auto | flex-start | flex-end | center | stretch | baseline	Eje secundario

Direcciones

Los lados **inicio principal/fin principal (main start/main end)** e **inicio secundario/fin secundario (cross start/cross end)** del contenedor flexible describen el origen y final del flujo de los elementos flexibles. Estos siguen los eje principal y secundario según el vector establecido por writing-mode (izquierda-a-derecha, derecha-a-izquierda, etc.).

- **La propiedad `order` asigna elementos a grupos ordinales y determina qué elementos aparecen primero.**

Por defecto, todos los ítems flex tienen un order: 0 implícito, aunque no se especifique. Si indicamos un order con un valor numérico, irá recolocando los ítems según su número, colocando antes los ítems con número más pequeño (*incluso valores negativos*) y después los ítems con números más altos.

- **La propiedad `flex-flow` property combina las propiedades `flex-direction` y `flex-wrap` para colocar los elementos flexibles.**

```
#contenedor {
  /* flex-flow: <flex-direction> <flex-wrap>; */
  flex-flow: row wrap;
}
```

flex-direction:	**row** | row-reverse | column | column-reverse	Cambia la orientación del eje principal.
flex-wrap:	**nowrap** | wrap | wrap-reverse	Evita o permite el desbordamiento (multilinea).

Líneas

Los elementos flexibles pueden disponerse en una sola o varias líneas de acuerdo con la propiedad flex-wrap, que controla la dirección del eje secundario y la dirección en la que las nuevas líneas se apilan.

Dimensiones

Los términos equivalentes a "altura" y "anchura" usados en los elementos flexibles son **tamaño principal (main size)** and **tamaño secundario (cross size),** que respectivamente siguen al eje principal y al eje secundario del contenedor flexible.

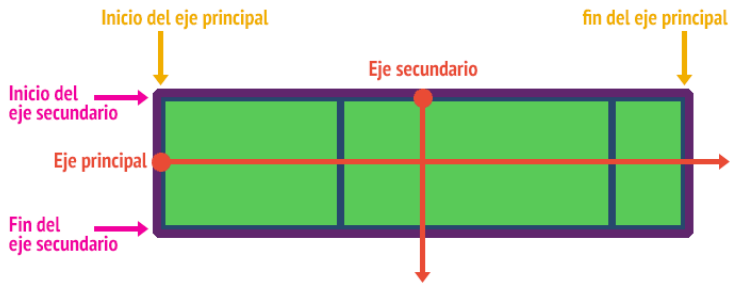
- **La propiedades `min-height` y `min-width` tienen un nuevo valor, `auto` que establece el tamaño mínimo de un elemento flexible.**
- **La propiedad `flex` combina las propiedades `flex-basis`, `flex-grow`, y `flex-shrink` para establecer el grado de flexibilidad de los elementos flexibles.**

Ver ejemplos:

https://www.desarrolloweb.com/manuales/manual-flexbox-css.html

A la hora de alinear un texto, por ejemplo, con CSS tenemos los conceptos left y right, indicando que queremos una alineación a izquierda o derecha. Esto no funciona justamente igual en Flexbox. En este caso tenemos los conceptos de inicio y fin (start / end).

Como se ha dicho, en flex tenemos dos ejes: principal y secundario, pues existirá un inicio y un fin para cada uno de los ejes.



Mirar ejemplos en este enlace:

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

https://lenguajecss.com/p/css/propiedades/flexbox

Ejercicios:

1. Una página, si está en posición apaisada te debe mostrar una imagen de fondo y si está en posición vertical otra diferente.

2. Dividir la página en 6 bloques con div. Los 6 bloques ocuparán toda la pantalla horizontalmente y tendrán un color de fondo diferente uno de otro. Estarán dentro de una etiqueta main. Si la anchura de la página es más pequeña a 720px, los bloques aparecerán verticalmente.

3. En el ejercicio anterior, poner encabezado y pie de página que siempre estarán en su sitio, independiente de la anchura de la página.