

# **Tema 2.4 - Generación de Interfaces I - CSS (Display y Posicionamiento)**



**Módulo: Desarrollo de Interfaces**

Curso 2º DAM

# Comportamiento Predeterminado de las Etiquetas HTML

En HTML, cada etiqueta tiene un comportamiento predeterminado basado en su tipo

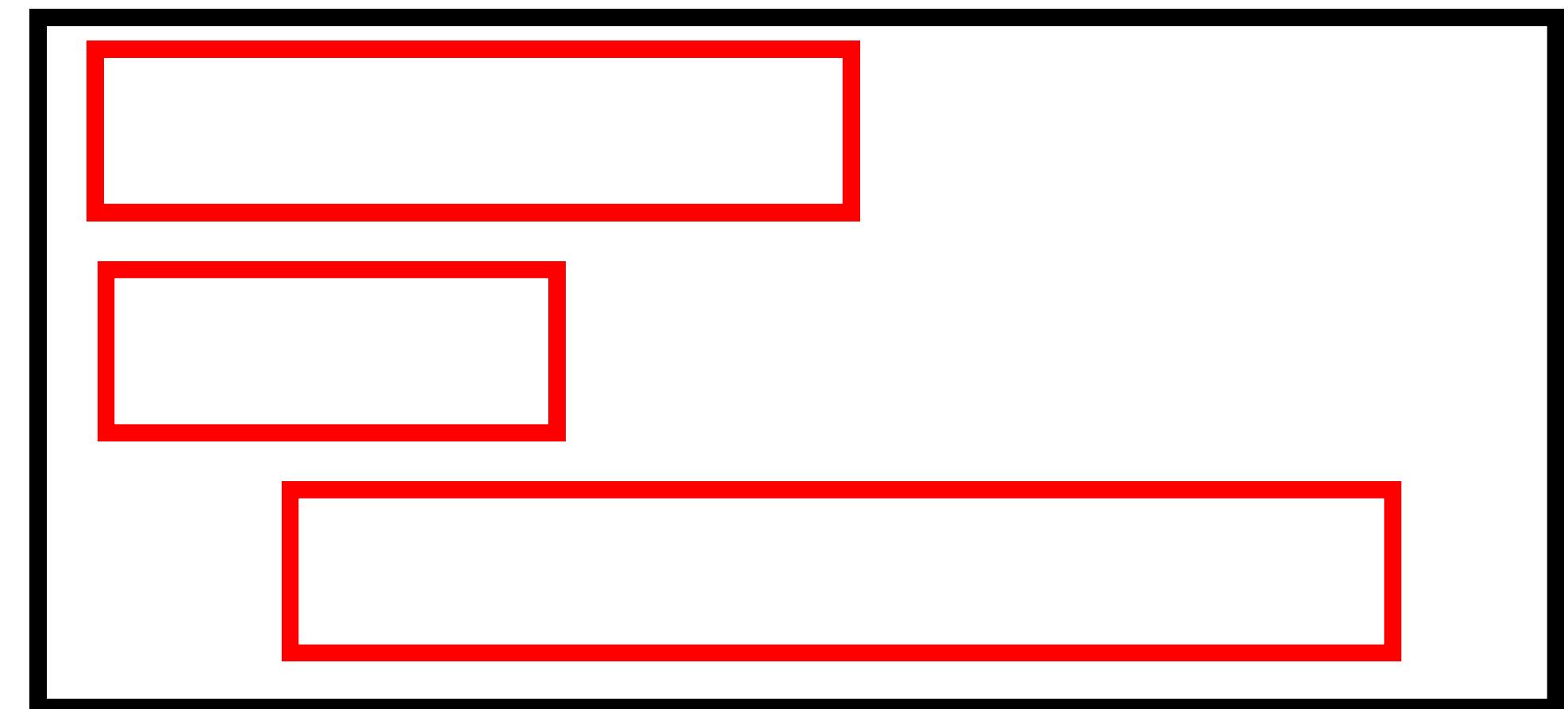
- **Elementos de Bloque (block):**

- Ocupan todo el ancho disponible de su contenedor
- Comienzan en una nueva línea.
- Ejemplos: <div>, <p>, <h1>, <section>, <article>.
- Permiten definir width, height, margin, y padding en todas las direcciones.

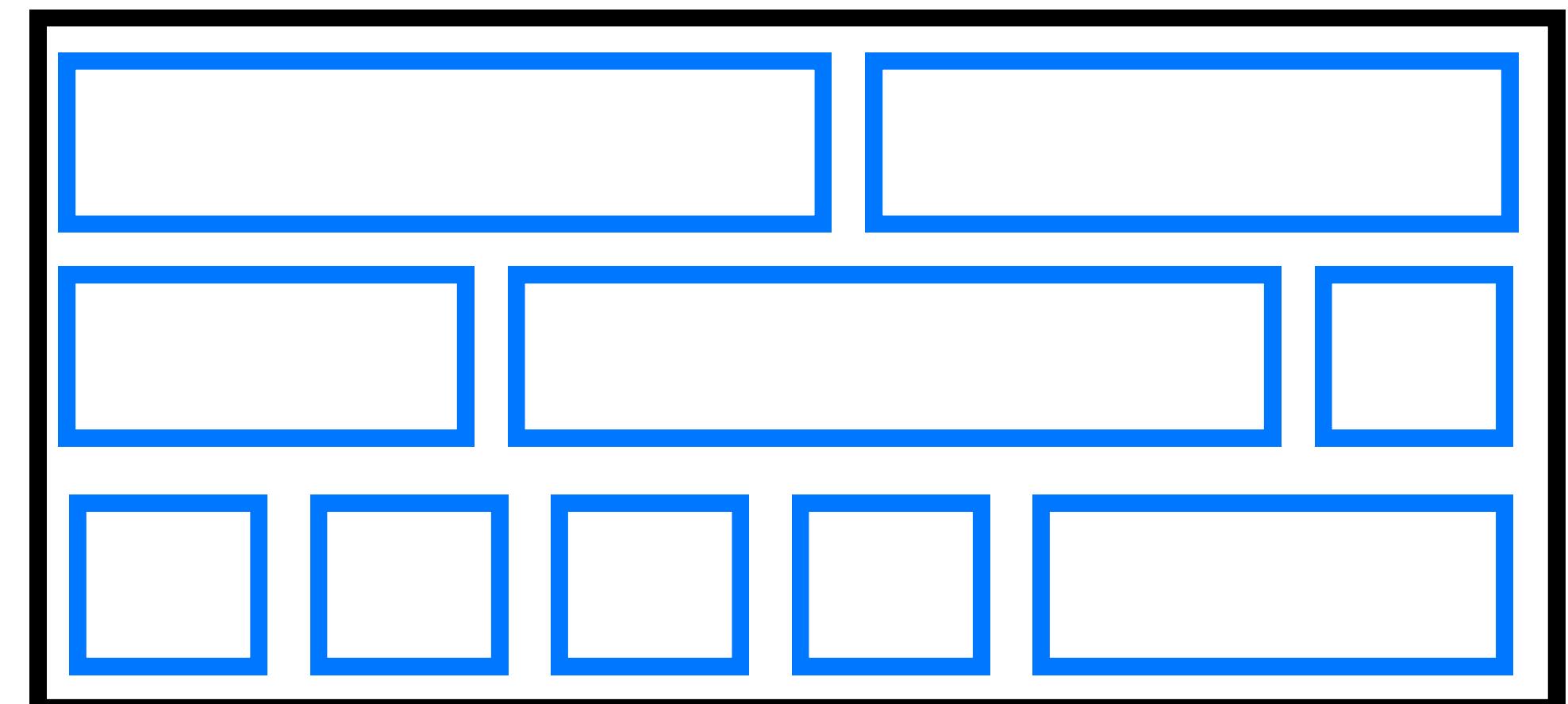
- **Elementos en Línea (inline):**

- Solo ocupan el espacio necesario para su contenido
- No fuerzan un salto de línea.
- Ejemplos: <span>, <a>
- No permiten definir width ni height, aunque sí permiten padding, border y margin en los ejes horizontales.

BLOCK



INLINE



# Modificaciones al comportamiento de las etiquetas (display)



```
.elemento {  
    display: block;  
}
```

## display: block

Fuerza al elemento a comportarse como un bloque, ocupando todo el ancho disponible y comenzando en una nueva línea. **Útil para elementos como el <span>**



```
.elemento {  
    display: none;  
}
```

## display: none

Oculta el elemento por completo. No ocupa espacio ni es visible en la página. **Útil para controlar la visibilidad de elementos en función de ciertos estados o interacciones.**



```
.elemento {  
    display: inline;  
}
```

## display: inline

Hace que el elemento se comporte en línea, ocupando solo el espacio necesario para su contenido sin empezar en una nueva línea. **Útil para elementos como <div>**



```
.elemento {  
    display: inline-block;  
}
```

## display: inline-block

Mezcla el comportamiento de los elementos en línea y de bloque: permanece en línea pero permite definir width, height, padding, y border. **Útil para crear elementos ajustables en línea, como botones y menús.**

## Proximamente

- **display: flex**
- **display: grid**
- **display: table, table-row, table-cell**

# Ocultar elementos: (display: none) vs (visibility:hidden)



```
.elemento {  
    display: none;  
}
```

## display: none

Oculta el elemento por completo. **No ocupa espacio ni es visible en la página.**

VS



```
.elemento {  
    visibility: hidden;  
}
```

## visibility: hidden

Controla si un elemento es:

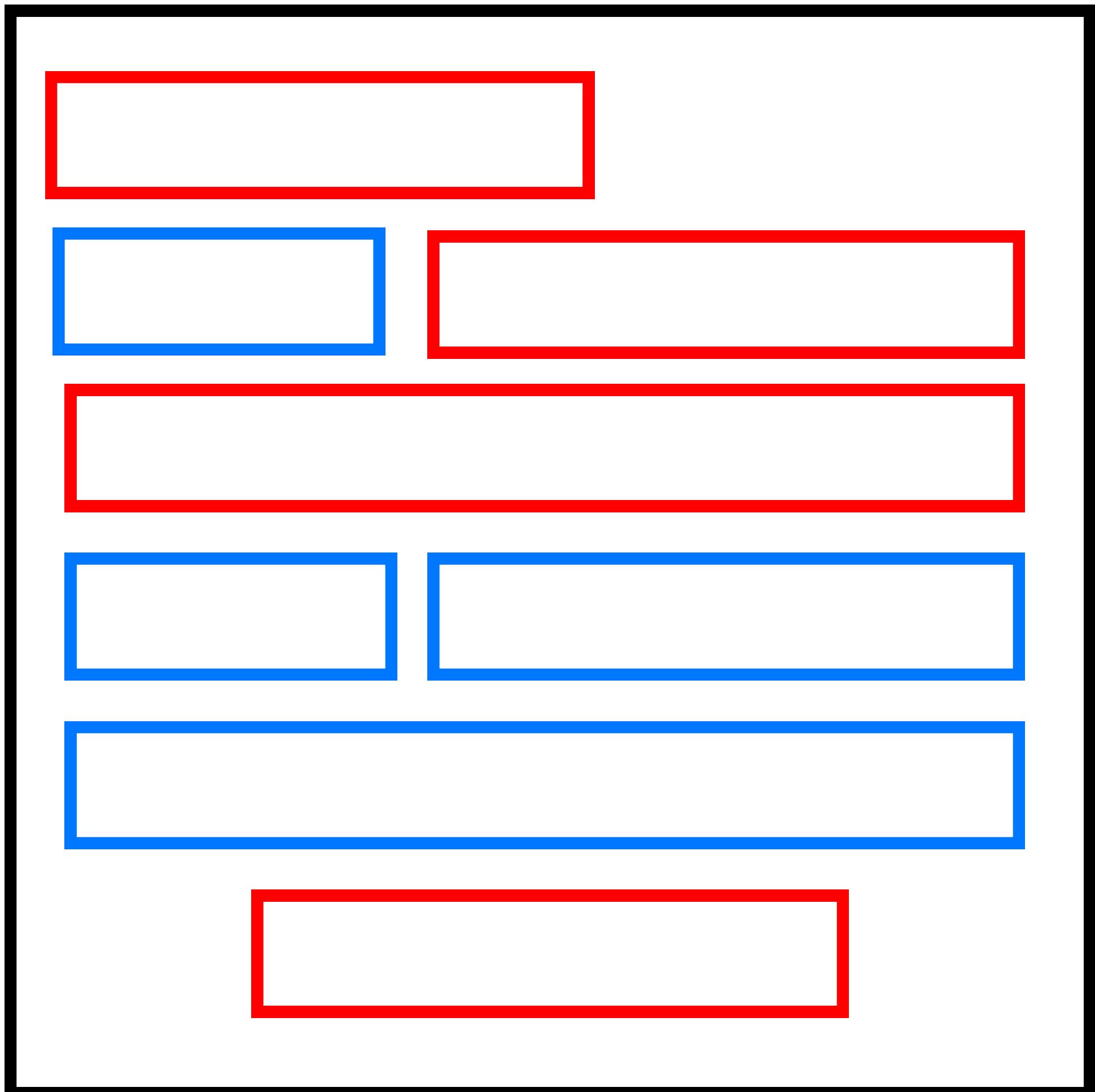
- Visible: **visibility: visible**
- Invisible: **visibility: hidden**

**Reserva el espacio del elemento en el flujo del documento incluso cuando está oculto.**

# Posicionamiento predeterminado en HTML

Por defecto, los elementos en HTML se colocan uno tras otro en el flujo del documento, siguiendo el orden en que aparecen en el código HTML:

- **Los elementos de bloque** se apilan verticalmente, uno encima de otro, respetando la estructura de línea completa.
- **Los elementos en línea** se colocan uno junto a otro dentro de los elementos de bloque, hasta llenar el ancho disponible.



# Modificación del Flujo Predeterminado: (position)

## position: static

- Flujo normal del documento
- No admite propiedades de desplazamiento (top, right, bottom, left).

```
●●●  
.estatico {  
  position: static;  
  background-color: lightblue;  
  padding: 10px;  
}
```

## position: relative

- Permite mover el elemento relativo a su posición original en el flujo.
- Admite top, right, bottom, y left, que desplazan el elemento desde su posición original.

```
●●●  
.relativo {  
  position: relative;  
  top: 20px;  
}
```

## position: absolute

- El elemento se posiciona absolutamente en relación con el contenedor posicionado más cercano **que no sea static**
- Si no hay un contenedor posicionado, se relaciona con el <html>.
- Admite top, right, bottom, y left, que desplazan el elemento desde su posición original.

```
●●●  
/* position: absolute */  
.contenedor {  
  position: relative; /* Contenedor para  
absolute positioning */  
  background-color: lightgray;  
  padding: 20px;  
}  
.absoluto {  
  position: absolute;  
  top: 10px;  
  right: 10px;  
  background-color: lightcoral;  
  padding: 10px;  
}
```



```
.fijo {  
  position: fixed;  
  bottom: 10px;  
  right: 10px;  
  background-color: lightyellow;  
  padding: 10px;  
}
```

## position: fixed

- El elemento se coloca **en relación con la ventana del navegador**, lo que significa que permanece fijo en su lugar aunque se despliegue la página.
- Se elimina del flujo del documento.
- Admite top, right, bottom, y left, que se aplican en relación a la ventana del navegador.



```
.pegajoso {  
  position: sticky;  
  top: 0;  
  background-color: lightpink;  
  padding: 10px;  
}
```

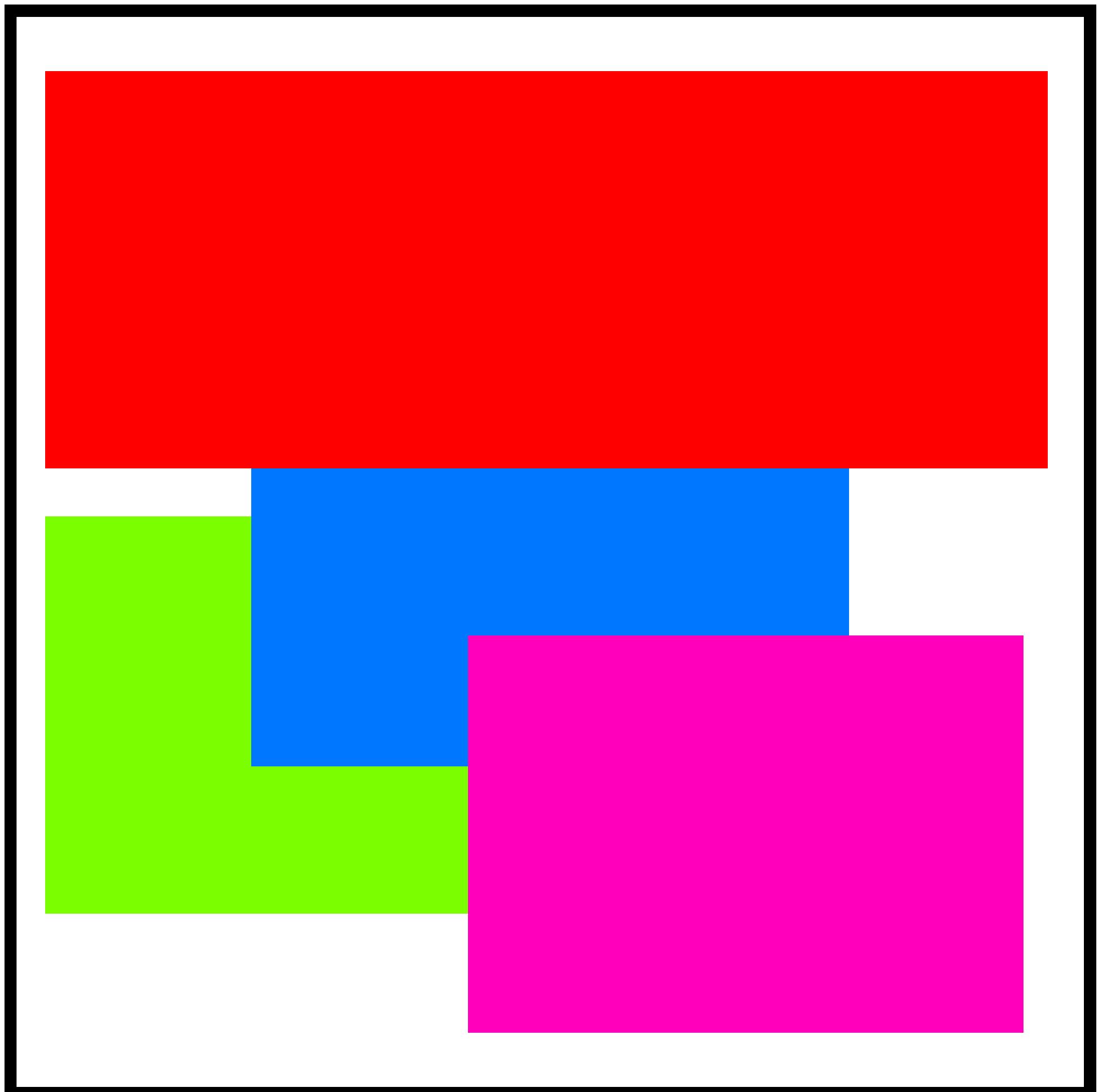
## position: sticky

- Combinación de relative y fixed. El elemento se comporta como relative hasta que alcanza un umbral definido, momento en el cual se vuelve fixed
- Admite top, right, bottom, y left, que determinan el punto donde el elemento se "pega".

# Modificación del solapamiento (z-index)

**Por defecto**, en HTML , al seguir el flujo de ejecución del código y al tener las características de posicionamiento explicadas (block e inline), **los elementos no se superponen**.

No obstante, **al aplicar modificaciones en el CSS como position**, pueden producirse **solapamiento** de cajas. Para controlar de forma específica cómo se superponen los elementos, CSS ofrece la propiedad **z-index**, que permite ajustar el orden de apilamiento.



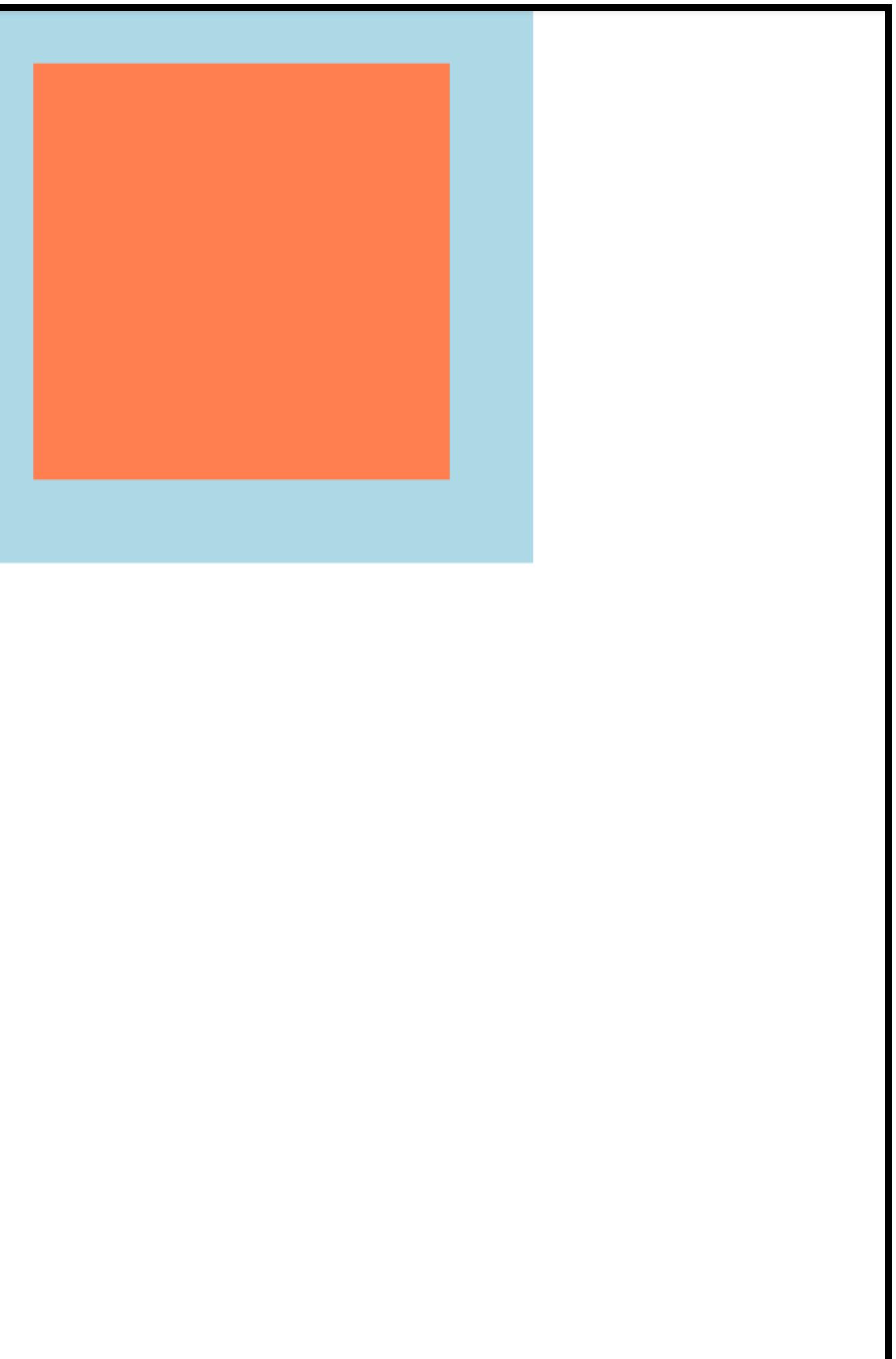
# Funcionamiento: z-index

Para que z-index sea efectivo, el elemento:

- Debe estar posicionado usando **position: relative, absolute, fixed, o sticky.**
- Los elementos con **position: static** (valor por defecto) no responden al z-index.

El z-index establece un valor numérico, donde:

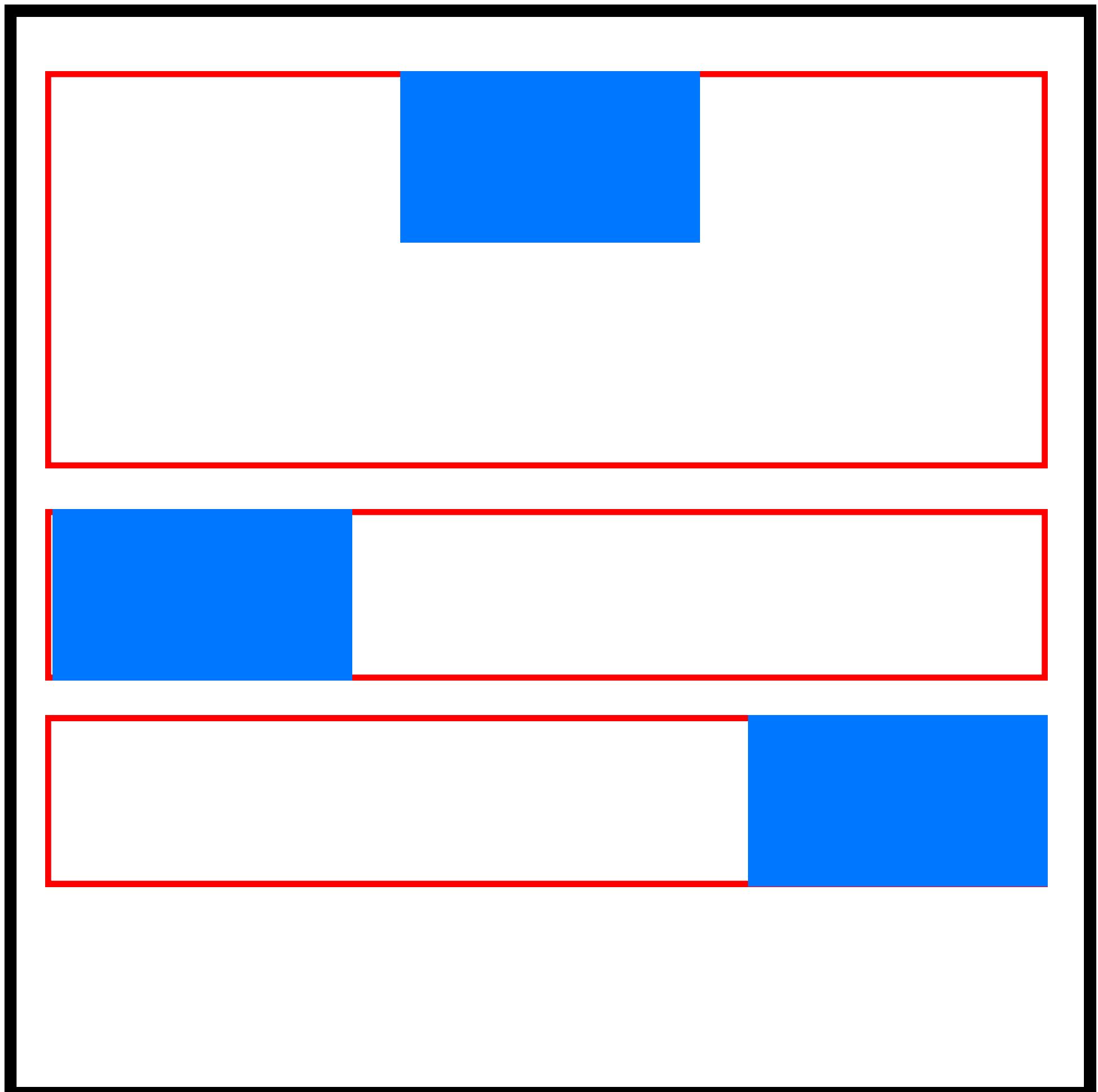
- **Un valor más alto coloca al elemento por encima de otros elementos con un z-index más bajo y viceversa**



```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<title>Ejemplo de z-index</title>
<style>
.fondo {
    position: absolute;
    top: 0;
    left: 0;
    width: 200px;
    height: 200px;
    background-color: lightblue;
    z-index: 1;
}
.frente {
    position: absolute;
    top: 20px;
    left: 20px;
    width: 150px;
    height: 150px;
    background-color: coral;
    z-index: 2;
}
</style>
</head>
<body>
<div class="fondo"></div>
<div class="frente"></div>
</body>
</html>
```

# Centrado de elementos con CSS

En CSS, hay múltiples formas de centrar elementos, dependiendo de si quieres centrar el **contenido horizontalmente, verticalmente, o ambos**. También influye si el elemento es en **línea o de bloque**



# Formas de centrado de elementos



```
.center {  
  margin: auto;  
  width: 300px;  
  border: 3px solid #73AD21;  
}
```



```
.center {  
  text-align: center;  
  border: 3px solid green;  
}
```



```
.center {  
  padding: 70px 0;  
  border: 3px solid green;  
}
```

## Centrado Horizontal → margin: auto

Para centrar un elemento de bloque (como un <div>) horizontalmente, puedes utilizar **margin: auto**.

- Establece un ancho específico con width.
- Aplica margin: auto, el navegador calcula el espacio restante y lo distribuye por igual

## Proximamente

- **display: flex**
- **display: grid**
- **display: table, table-row, table-cell**

## Centrado Horizontal → text-align: (left|center|right)

Utiliza text-align: center **en el contenedor padre** para alinear el texto o los elementos en línea (como <span> o <a>).

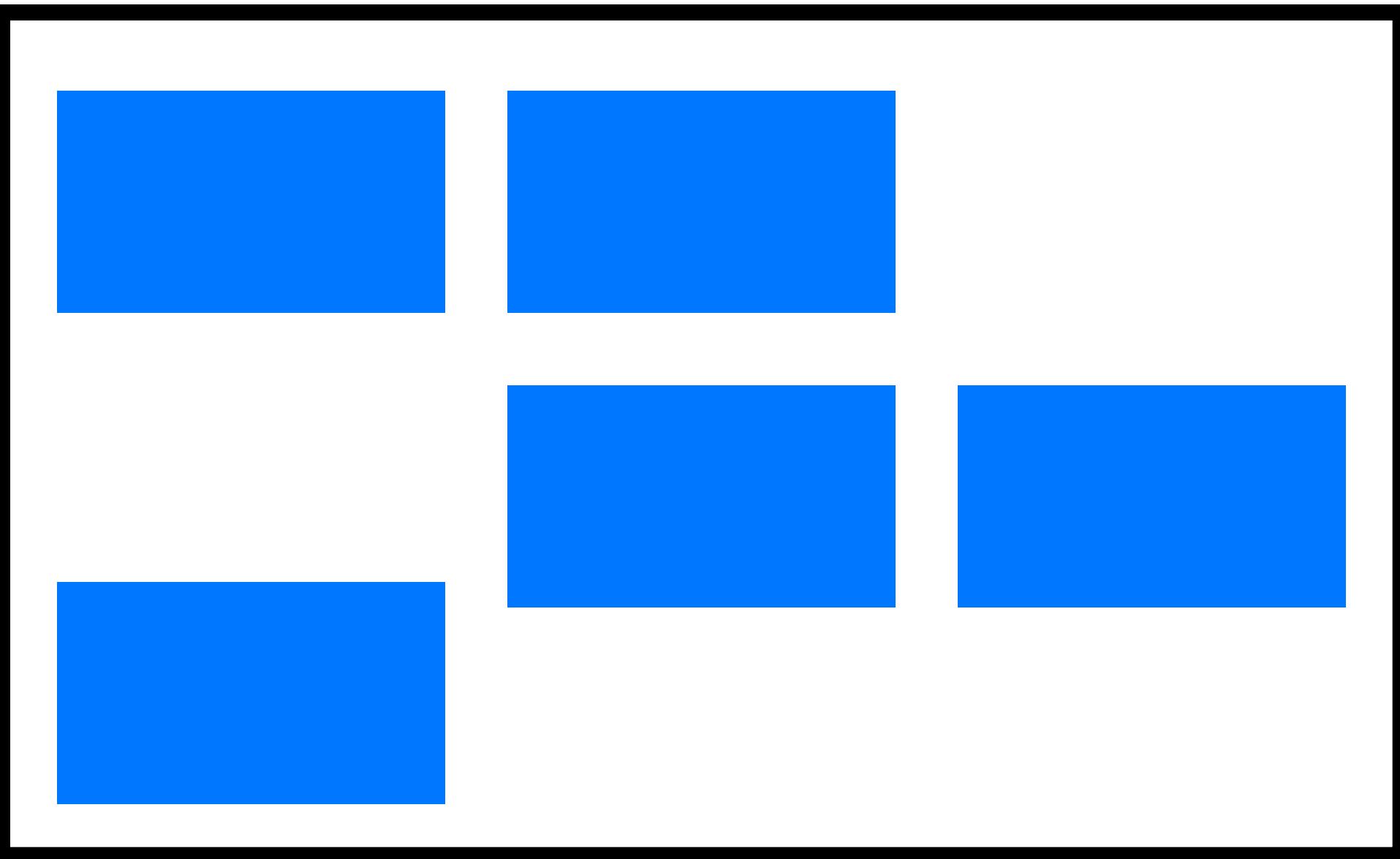
## Centrado Vertical → padding sin height

Una manera sencilla de alinear el contenido de un bloque es agregar padding verticalmente sin añadir height.

## Disposición dinámica: flex

La idea principal detrás de Flexbox es darle al contenedor la capacidad de **ajustar el ancho, la altura e incluso el orden de sus elementos** para ocupar el espacio disponible de la mejor manera posible.

Un contenedor flexible **permite expandir o reducir los elementos** para que ocupen el espacio libre o para evitar el desbordamiento.



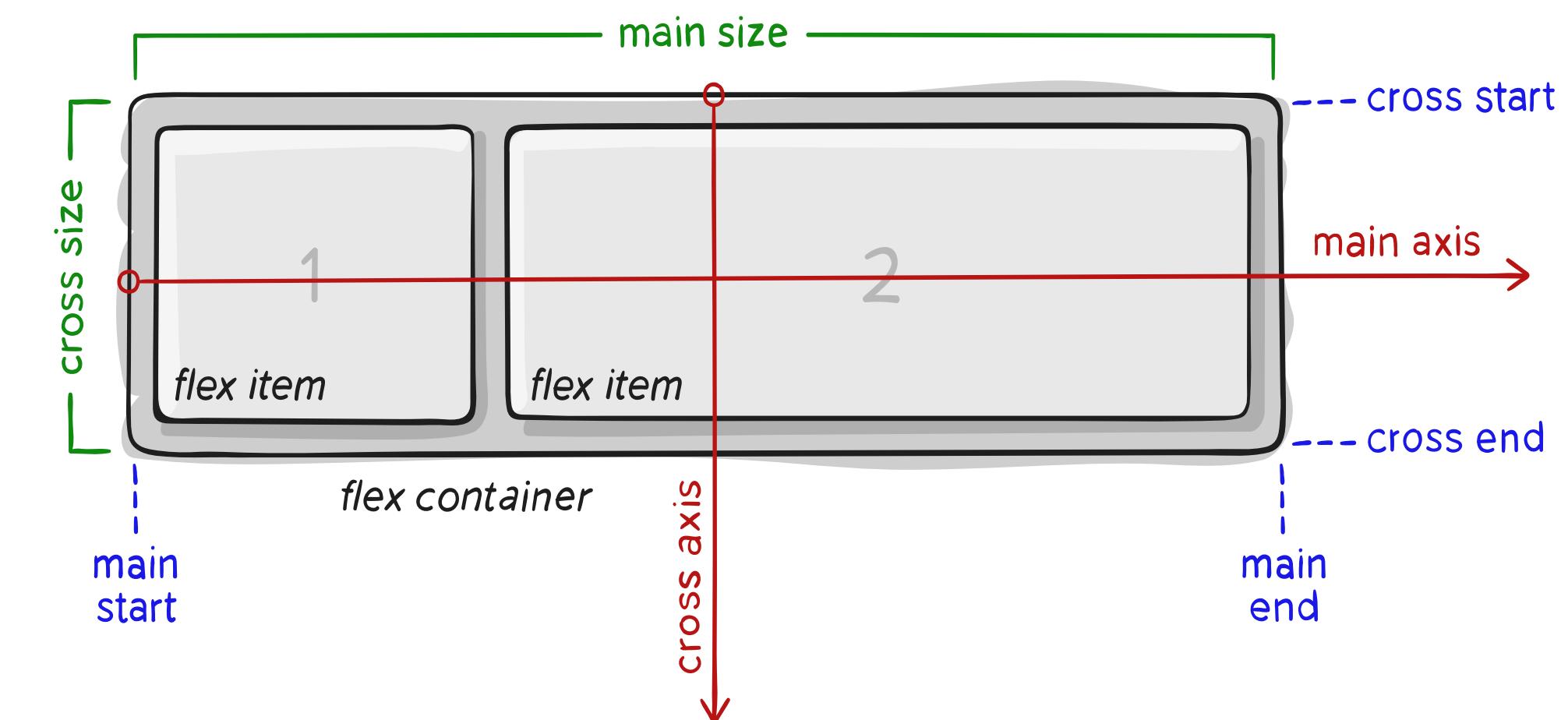
# Conceptos Básicos y Terminología de Flexbox

Flexbox es un conjunto de propiedades que se configuran:

- **En el contenedor flexible (elemento padre)**
- **En los elementos flexibles (elementos hijos).**

En Flexbox, los elementos se disponen siguiendo uno de los dos ejes:

1. **Eje Principal (main axis):** Es el eje principal a lo largo del cual se colocan los elementos. Este puede ser horizontal o vertical.
2. **Inicio y Fin del Eje Principal:** Los elementos flexibles se distribuyen desde el inicio del eje principal (main-start) hasta el final del eje principal (main-end).
3. **Tamaño Principal:** Es el ancho o la altura del elemento flexible a lo largo del eje principal.
4. **Eje Cruzado (cross axis):** Es el eje perpendicular al eje principal, cuya dirección depende de la orientación del eje principal.
5. **Inicio y Fin del Eje Cruzado:** Las líneas flexibles se llenan y se colocan dentro del contenedor desde el inicio hasta el fin del eje cruzado.
6. **Tamaño Cruzado:** Es el ancho o altura del elemento flexible a lo largo del eje cruzado.



# Propiedades del padre (flex container)



```
.container {  
  display: flex;  
  /* o inline-flex */  
}
```

## display: flex

Esto define un contenedor flexible (en línea o bloque). Habilita un contenedor flex para todos sus hijos directos.



```
.container {  
  flex-direction: row;  
}
```

## flex-direction

Establece el eje principal, definiendo la dirección en la que se colocan los elementos flexibles en el contenedor.

- row (por defecto): de izquierda a derecha.
- row-reverse: de derecha a izquierda.
- column: de arriba hacia abajo.
- column-reverse: de abajo hacia arriba.



```
.container {  
  flex-wrap: wrap ;  
}
```

## flex-wrap

Por defecto, los elementos flexibles intentan encajar en una sola línea. Puedes cambiar esto y permitir que los elementos se envuelvan según sea necesario.

- nowrap (por defecto): todos los elementos flexibles estarán en una línea.
- wrap: los elementos flexibles se envolverán en varias líneas, de arriba hacia abajo.
- wrap-reverse: los elementos se envolverán en varias líneas, de abajo hacia arriba.

# Propiedades del padre (flex container)



```
.container {  
  flex-flow: column wrap;  
}
```

## flex-flow

Esta es una abreviatura de las propiedades flex-direction y flex-wrap, que juntas definen los ejes y cruzado del contenedor flexible.

**El valor por defecto es row nowrap.**



```
.container {  
  justify-content: space-between  
}
```

## justify-content

Define la alineación a lo largo del eje principal.

- **flex-start (por defecto)**: los elementos se alinean hacia el inicio del eje principal.
- **flex-end**: los elementos se alinean hacia el final.
- **center**: los elementos se centran en la línea.
- **space-between**: los elementos se distribuyen de manera uniforme, con el primer elemento en el inicio y el último en el final.
- **space-around**: los elementos tienen espacios iguales a su alrededor.
- **space-evenly**: Todos los espacios son iguales.



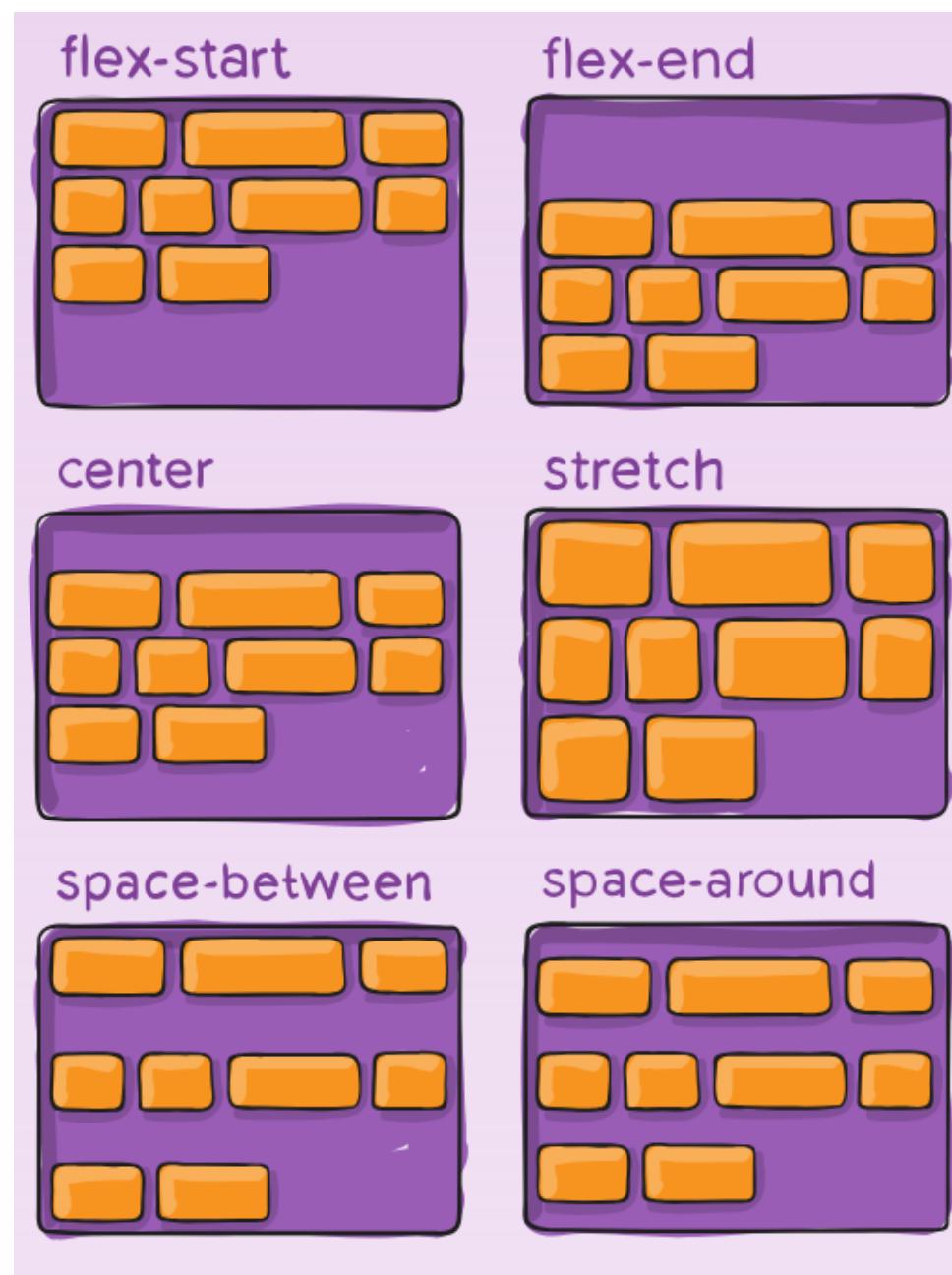
```
.container {  
  align-items: center  
}
```

## align-items

Define la alineación a lo largo del eje cruzado.

- **stretch (por defecto)**: los elementos se estiran para llenar el contenedor.
- **flex-start**: los elementos se colocan al inicio del eje cruzado.
- **flex-end**: los elementos se colocan al final del eje cruzado.
- **center**: los elementos se centran en el eje cruzado.
- **baseline**: los elementos se alinean según sus líneas base.

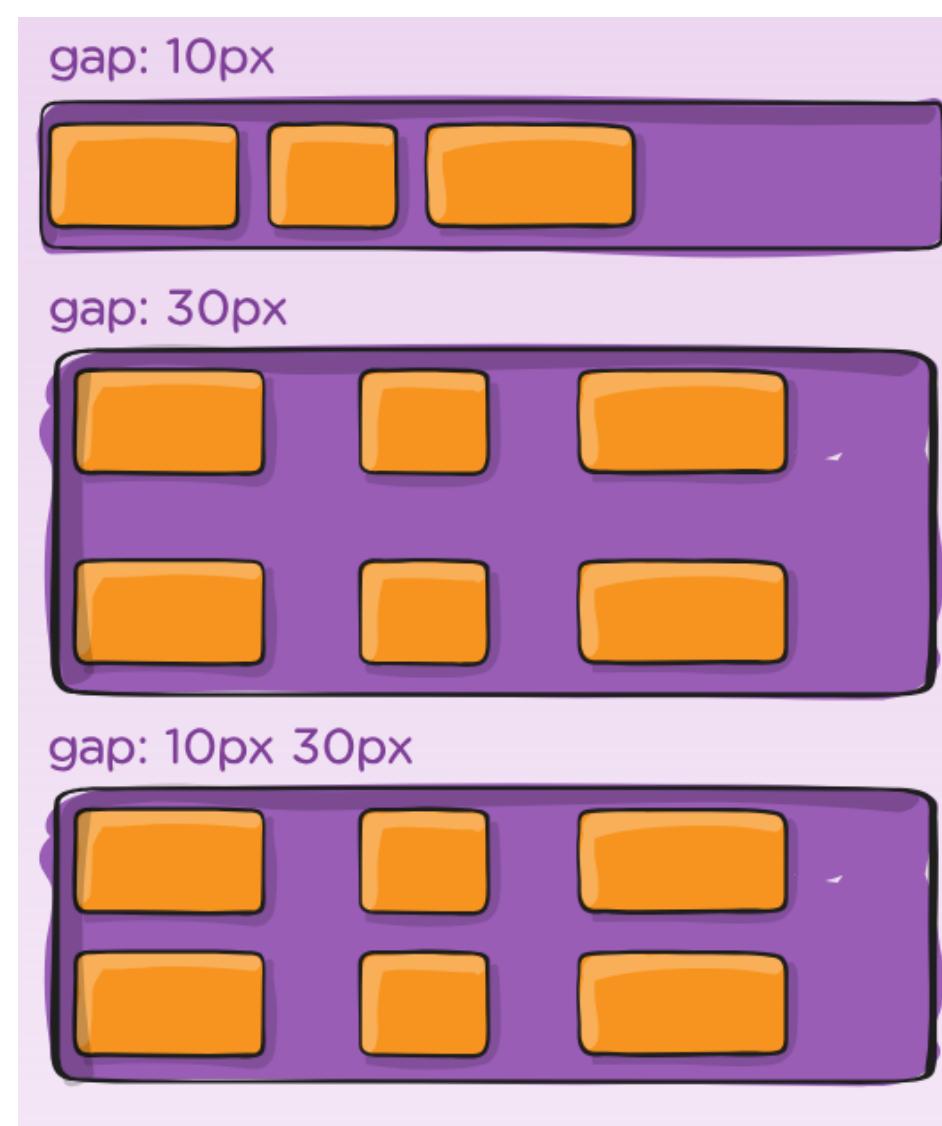
# Propiedades del padre (flex container)



## align-content

Alinea las líneas del contenedor flexible cuando se distribuye en forma de grilla (flex-wrap: wrap)

- **normal (por defecto)**: Los elementos se colocan en su posición predeterminada.
- **flex-start / start**: Los elementos se alinean al inicio del contenedor
- **flex-end / end**: Los elementos se alinean al final del contenedor.
- **center**: Los elementos se centran en el contenedor.
- **space-between**: Los elementos se distribuyen uniformemente, con el primero en el inicio y el último en el final.
- **space-around**: Los elementos se distribuyen con espacio igual alrededor de cada uno.
- **space-evenly**: Los elementos se distribuyen con espacios iguales entre ellos y los bordes del contenedor.
- **stretch**: Las líneas de elementos se estiran para ocupar el espacio disponible.

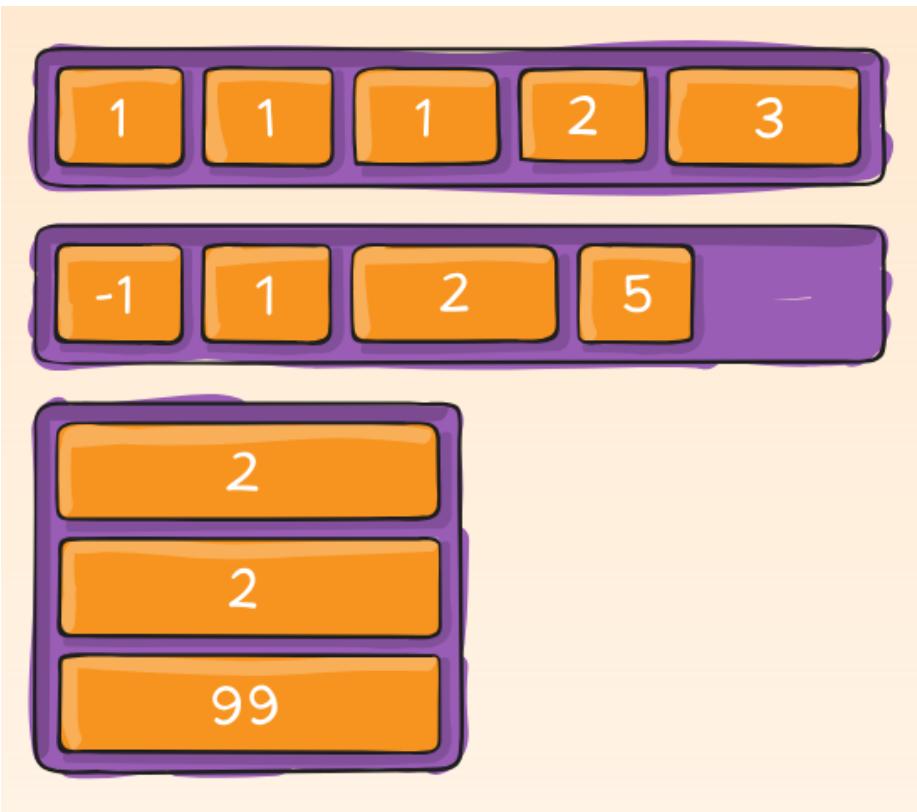


```
● ● ●  
.container {  
  display: flex;  
  ...  
  gap: 10px;  
  gap: 10px 20px; /* row-gap column gap */  
  row-gap: 10px;  
  column-gap: 20px;  
}
```

## gap, row-gap, column-gap

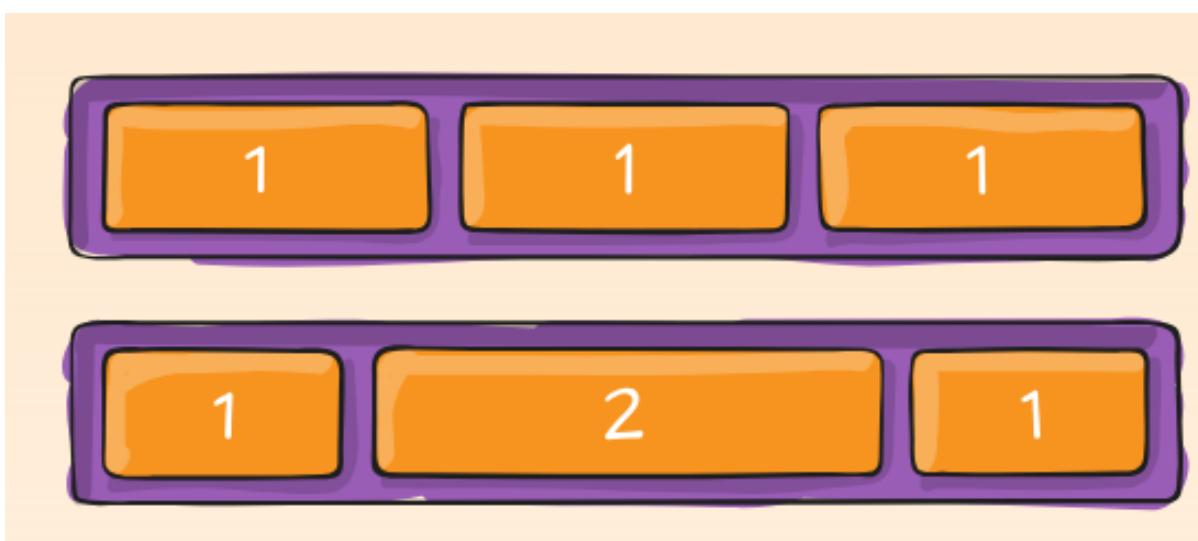
Controla explícitamente el espacio entre los elementos cuando se distribuye en forma de grilla (flex-wrap: wrap)

# Propiedades del hijo (flex items)



## order

Por defecto, los elementos flexibles se disponen según el orden en el que aparecen en el código fuente. Sin embargo, la propiedad `order` permite controlar el orden en el que aparecen dentro del contenedor flexible.



## flex-grow

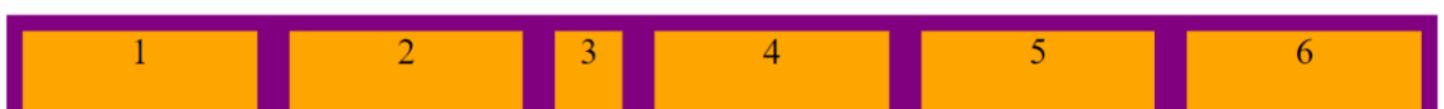
Define la capacidad de un elemento flexible para crecer si es necesario. Acepta un valor que actúa como una proporción para determinar cuánto espacio adicional ocupará el elemento en comparación con los demás.

- Si todos los elementos tienen `flex-grow: 1`, el espacio sobrante se distribuye por igual.
- Si uno de los elementos tiene `flex-grow: 2`, ocupará el doble de espacio que los demás.

`flex-shrink: 1` (default value) for all flex items



`flex-shrink: 2` on the third item



## flex-shrink

Establece la capacidad de un elemento flexible para encogerse si es necesario.

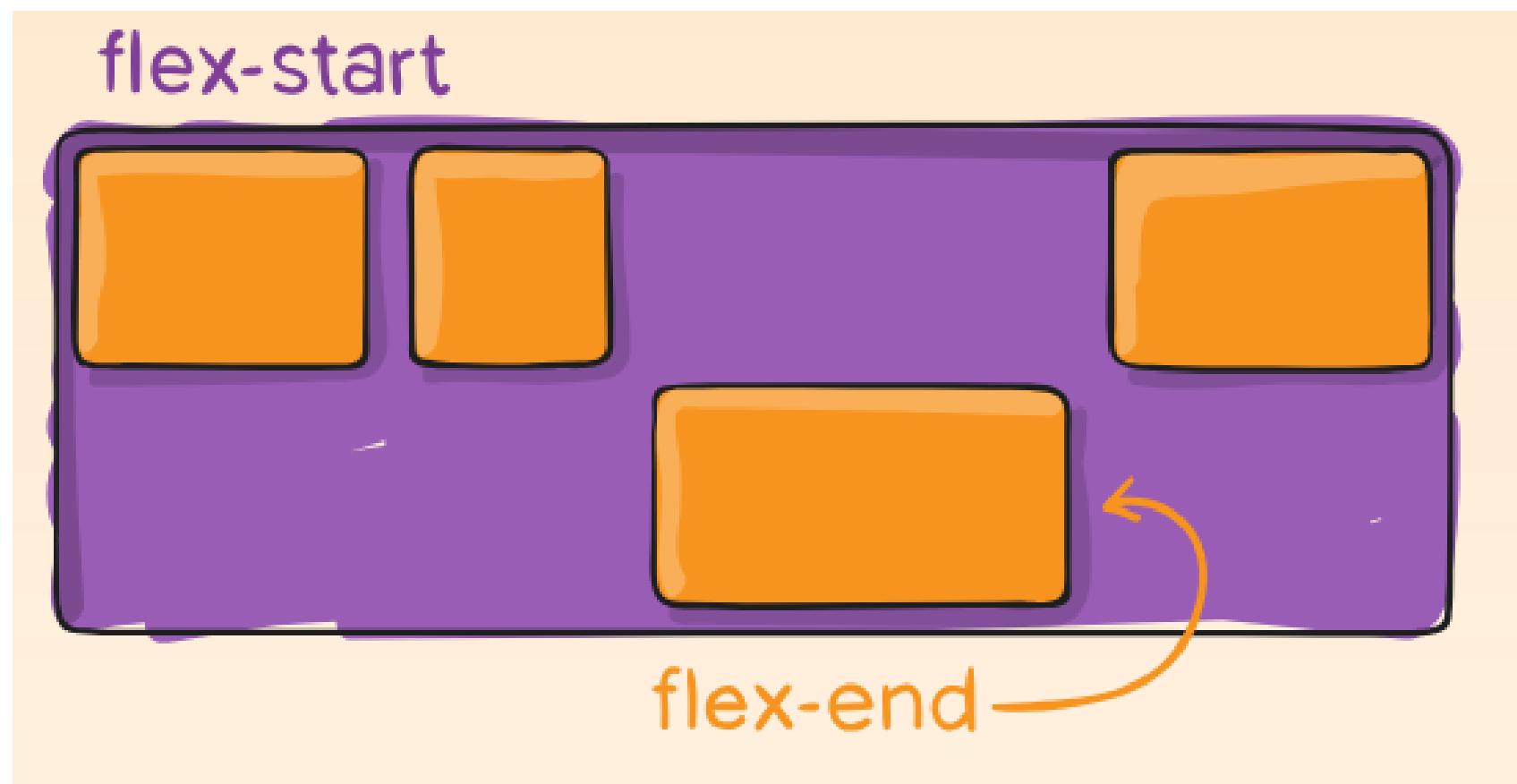
Establece la capacidad de un elemento flexible para encogerse si es necesario.

# Propiedades del hijo (flex items)



## flex-basis

Define el tamaño predeterminado de un elemento antes de que se distribuya el espacio sobrante. Puede ser una longitud (e.g. 20%, 5rem) o una palabra clave como auto, que indica que el tamaño se basará en las propiedades width o height.



## align-self

Mientras que la propiedad align-items controla cómo se alinean todos los elementos dentro de un contenedor flex en el eje cruzado, **align-self** te permite definir una alineación única para un elemento individual.

**Los valores son los mismo que align-items**

# Ejercicios

**<https://flexboxfroggy.com/>**

**<http://www.flexboxdefense.com/>**

**<https://mastery.games/flexboxzombies/>**