CSS avanzado

Guía de estudio

Index

Introducción

CSS es uno de los componentes clave dentro del desarrollo Frontend. Permite agregar y modificar la capa visual de un proyecto web, por medio de diferentes animaciones y estilos, aumentando la estética de la aplicación.

Como todo lenguaje, **tiene un nivel de dificultad progresivo,** permitiéndonos desarrollar estructuras más complejas a mayor tiempo, práctica y estudio le dediquemos.

En la presente guía, desarrollaremos con cierta profundidad aquellos aspectos que nos resultan relevantes para optimizar el aspecto visual de las aplicaciones web.

Propiedades básicas de CSS

Color: Cambia el color del texto que está en el interior de un elemento.

Background-color: Cambia el color de fondo de un elemento.

Ejemplo:

```
.div {
  background-color: black;   /* Color de fondo */
  color: white;              /* Color de texto */
}
```

Propiedades de tamaño

Width: especifica la anchura del área de contenido de un elemento.

Height: especifica la altura del área de contenido de un elemento.

Margin

Los márgenes son los espacios exteriores de un elemento, que establece el margen para los cuatro lados. El tamaño de dichos márgenes se puede alterar en conjunto (de forma general) o de forma específica a cada una de las zonas del elemento (izquierda, derecha, arriba o abajo).

margin-top	Establece un tamaño de margen superior.
margin-left	Establece un tamaño de margen a la izquierda.
margin-right	Establece un tamaño de margen a la derecha.
margin-bottom	Establece un tamaño de margen inferior.

Padding

Los rellenos (padding) son los espacios que hay entre los bordes del elemento y el contenido del elemento (por la parte interior) es decir, desde el borde hacia el interior.

padding-top	Aplica un relleno interior en el espacio superior de un elemento.
padding-left	Aplica un relleno interior en el espacio izquierdo de un elemento.
padding-right	Aplica un relleno interior en el espacio derecho de un elemento.
padding-bottom	Aplica un relleno interior en el espacio inferior de un elemento.

El **padding** es el espacio que hay desde el borde hacia el **interior**.
El **margin** es el espacio que hay desde el borde hacia el **exterior**.

Fonts (fuentes)

Existe un amplio abanico de **propiedades CSS** para modificar las características básicas de las tipografías a utilizar. A continuación, veremos las propiedades CSS más básicas para aplicar a cualquier tipo de tipografía:

font-family	Indica el nombre de la fuente (o una lista de ellas).
font-size	Indica el tamaño de la fuente.
font-style	Indica el estilo de la fuente.
font-weight	Indica el peso (grosor) de la fuente (100-800).

Con ellas podemos seleccionar tipografías concretas, especificar su tamaño, estilo o grosor.

Border (borde):

En CSS es posible especificar el **aspecto que tendrán los bordes** de cualquier elemento HTML, pudiendo incluso, dar diferentes características a zonas particulares del borde como, por ejemplo, el borde superior, izquierdo, derecho o inferior.

Las **propiedades básicas** de los bordes en CSS son las siguientes:

Propiedad	Valor	Significado
border-color	black	Especifica el color que se utilizará en el borde.
border-width	thin | medium | thick	Especifica un tamaño predefinido para el grosor del borde.
border-style	None	Define el estilo para el borde a utilizar (ver más adelante).

En primer lugar, **border-color** establece el color del borde, por ejemplo, “border-color: blue;” el cual aplicaría un borde de color azul.

En segundo lugar, con **border-width** podemos establecer el ancho o grosor del borde utilizando tanto palabras clave predefinidas como un tamaño concreto con cualquier tipo de las unidades ya vistas.

Por último, con la propiedad **border-style** podemos aplicar un estilo determinado al borde de un elemento.

En estilo de borde podemos elegir cualquiera de las siguientes opciones:

Valor	Descripción
Hidden	Idéntico a none, salvo para conflictos con tablas.
Dotted	Establece un borde basado en puntos.
Dashed	Establece un borde basado en rayas (línea discontinua).
Solid	Establece un borde sólido (línea continua).
Double	Establece un borde doble (dos líneas continuas).
Groove	Establece un borde biselado con luz desde arriba.
Ridge	Establece un borde biselado con luz desde abajo. Opuesto a Groove.
Inset	Establece un borde con profundidad «hacia dentro».
outset	Establece un borde con profundidad «hacia fuera». Opuesto a inset.

Veamos un ejemplo sencillo aplicando alguna de estas características:

```css
.div {
  border-color: gray;
  border-width: 1px;
  border-style: dotted;
}
```

Pueden utilizarse cualquiera de los estilos indicados en la tabla anterior e incluso combinar con otras propiedades. **Recuerda que, si no aplica una de ellas, se aplica el valor por defecto.**

Borders

Bordes múltiples (diferentes):

Hasta ahora, sólo hemos utilizado un parámetro en cada propiedad, lo que significa que se aplica el mismo valor para cada borde de un elemento (borde superior, derecho, inferior e izquierdo). Sin embargo, **podemos especificar uno, dos, tres o hasta cuatro parámetros**, dependiendo de lo que queramos hacer:

Propiedad	Valor	Significado
border-color	1 parámetro.	Aplica el mismo color a todos los bordes.
	2 parámetros.	Aplica al borde top/bottom, y al left/right.
	3 parámetros.	Aplica al top, al left/right y al bottom.
	4 parámetros.	Aplica al top, right, bottom y left.

De la misma forma, podemos hacer exactamente lo mismo con las propiedades **border-width** (respecto al ancho del borde) y **border-style** (respecto al estilo del borde). Teniendo en cuenta esto, disponemos de mucha flexibilidad a la hora de especificar esquemas de bordes más complejos:

```
.div {
  border-color: red blue green;
  border-width: 2px 10px 5px;
  border-style: solid dotted solid;
}
```

En el ejemplo anterior **hemos utilizado 3 parámetros**, indicando un elemento con:

- **Borde superior** rojo con un estilo sólido y de 2 píxeles de grosor.
- **Borde izquierdo** como el derecho tienen un estilo punteado de color azul de 10 píxeles de grosor.
- **Borde inferior** verde, con un estilo sólido de 5 píxeles de grosor.

Atajo: Bordes

Ya habremos visto que, con tantas propiedades, para hacer algo relativamente sencillo, nos pueden quedar varias líneas de código complejas y difíciles de leer. Al igual que con otras propiedades CSS, podemos utilizar la propiedad de atajo "**border**", con la que podemos hacer un resumen y no necesitar indicar múltiples propiedades individuales por separado, realizando el proceso de forma más corta:

Border

Propiedad de atajo para simplificar valores.

Por ejemplo:

```
.div {
  border: 1px solid #000000;
}
```

Así estamos aplicando: un borde de 1 píxel de grosor, estilo sólido y color negro a todos los bordes del elemento, ahorrando mucho espacio y escribiéndolo todo en una sola propiedad.

Box-shadow

Es una propiedad de CSS que como su nombre lo indica, es la sombra de la caja. Esta propiedad añade un efecto de sombra alrededor del marco de un elemento.

Funciona de forma muy similar a la que vimos en las sombras de texto, sólo que con algunos añadidos interesantes.

Las sombras **box-shadow** están desactivadas sobre cualquier elemento, lo mismo que ocurriría si aplicamos el valor none a dicha propiedad.

Veamos un **resumen** de valores que podemos indicar a esta propiedad de sombras:

Propiedad	Valor	Significado
Box-shadow	none	Elimina (o simplemente no establece) sombra sobre un elemento.
	PosX PosY	Crea una sombra color negro desplazándola ligeramente en horizontal y/o vertical.
	PosX PosY SIZE	Idem a la anterior, pero desenfocando o difuminando la sombra.
	PosX PosY SIZE SIZE	Idem a la anterior, pero además aplicando un factor de crecimiento a la sombra.
	PosX PosY SIZE COLOR	Idem a la anterior, cambiando el color de la sombra.
	PosX PosY SIZE COLOR INSET	Idem a la anterior, pero estableciendo una sombra interna en lugar de externa.

La palabra clave inset se puede escribir en cualquier otro orden, pero se suele indicar al final. Veamos cada uno de estos parámetros, uno por uno, junto a sus peculiaridades.

Desplazamiento de la sombra

Los dos primeros parámetros POSX y POSY, son los parámetros obligatorios mínimos para hacer funcionar la propiedad box-shadow, donde indicamos el desplazamiento que tendrá la sombra en el eje “x” (horizontal) y en el eje “y” (vertical).

Si queremos desplazar una sombra ligeramente a la derecha (eje x, primer parámetro) y hacia abajo (eje y, segundo parámetro), tendríamos que escribir, como mínimo, algo similar a lo siguiente:

```css
.element {
  box-shadow: 5px 5px;
}
```

En este caso, la sombra se crea de color currentColor (habitualmente black) y sin difuminar, como veremos en los siguientes apartados. Ten en cuenta que valores negativos invierten la dirección de la sombra. Si “5px 5px” mueve la sombra “5 píxeles a la derecha y hacia abajo”, entonces “-5px -5px” movería la sombra “5 píxeles a la izquierda, y 5 píxeles hacia arriba”.

Desenfoque de la sombra

El tercer parámetro de la propiedad **box-shadow** indica la cantidad de desenfoque o difuminado que queremos utilizar en nuestra sombra. Por defecto, tiene un valor de “0”, o lo que es lo mismo, la sombra será igual a la caja original, por lo que será completamente lisa, sin difuminar.

Este valor puede irse ampliando y de esta forma conseguiremos una sombra más desenfocada:

```css
.element {
  box-shadow: 5px 5px 0;      // Sombra sin desenfoque
  box-shadow: 5px 5px 2px;    // Sombra con ligero desenfoque
  box-shadow: 5px 5px 10px;   // Sombra desenfocada
  box-shadow: 5px 5px 40px;   // Sombra con un desenfoque casi disipado
}
```

Color de la sombra

Lo habitual de las sombras creadas con **box-shadow** es que siempre incluyan cuatro valores: desplazamiento de x, de y, nivel de desenfoque y color de la sombra.

Luego, el factor de **crecimiento** y la palabra clave **inset**, **son opcionales y se usan sólo cuando son necesarias.**

```
.element {
  box-shadow: 15px 5px 10px #48529944;
}
```

En el ejemplo anterior utilizamos el color #485299 con una transparencia en hexadecimal de 44, teniendo en cuenta que los valores de canal alfa en hexadecimal van desde 00 (totalmente transparente) hasta ff (totalmente opaco).

Con el **parámetro del color** podemos cambiar el color de la sombra a nuestro antojo, **utilizando palabras clave, valores hexadecimales, funciones rgb() o hsl() y canales alfa** para conseguir cierta transparencia, si así lo deseamos.

Distribución

Espaciados

CSS dispone de ciertas propiedades relacionadas con el texto de una página, pero alejándose de criterios de tipografías, y centrándose más en objetivos de **alineación o tratamiento de espaciados.**

Veamos algunas de estas propiedades:

letter-spacing	Espacio entre letras (tracking)
line-height	Altura de una línea (interlineado)

Alineaciones

Existen varias propiedades CSS que permiten modificar las diferentes alineaciones de los textos en su conjunto.

Propiedad	Valor	Significado
text-align	left | center | right | justify	Justificación del texto
text-justify	auto | inter-word | inter-character | none	Método de justificación de textos
text-overflow	clip | ellipsis | texto	Comportamiento cuando el texto no cabe «[...]»

En el primer caso, se puede establecer los valores “l**eft, right, center o justify**” a la propiedad “**text-align**” para alinear horizontalmente el texto a la izquierda, a la derecha, en el centro o justificar el texto, respectivamente, de la misma forma que lo hacemos en un procesador de texto.

En la propiedad **text-justify** indicamos el **tipo de justificación de texto que el navegador realizará: automática (el navegador elige)**, ajustar el espacio entre palabras (el resultado de ajustar con la propiedad word-spacing), ajustar el espacio entre par de caracteres (el resultado de ajustar con la propiedad letter-spacing) y justificación desactivada.

Por su parte, la propiedad **text-overflow cambia el comportamiento del navegador cuando detecta que un texto no cabe y se desborda**. En ella podemos utilizar los valores clip, desbordar el contenedor (comportamiento por defecto), elipsis, que muestra el texto «...» cuando no cabe más texto y por último indicar el texto que queremos utilizar en lugar de «...».

Al igual que existe **text-align** para alinear horizontalmente, **también existe la propiedad vertical-align**, que se encarga de la alineación vertical de un elemento, pudiendo establecer como valor las siguientes opciones:

Valor	¿Cómo hace la alineación?
Baseline	La base del elemento con la base del elemento padre.
Sub	El elemento como un subíndice.
Super	El elemento como un superíndice.
Top	La parte superior del elemento con la parte superior del elemento más alto de la línea.

Middle	El elemento en la mitad del elemento padre.
Bottom	La parte inferior del elemento con la parte inferior del elemento más bajo de esa línea.
Text-top	La parte superior del elemento con la parte superior del texto padre.
Text-bottom	La parte inferior del elemento con la parte inferior del texto padre.
Tamaño	Sube o baja un elemento el tamaño o porcentaje especificado.

¡Atención con el vertical-align!
Esta propiedad puede querer utilizarse para centrar verticalmente un elemento, sin embargo, su utilización es un poco menos intuitiva de lo que en un principio se cree, ya que **se debe utilizar para alinear textos respecto a elementos.**

Variaciones

Por último, existen varias propiedades aplicables a los textos para variar su naturaleza. Echemos un vistazo:

Propiedad	Valor	Significado
text-decoration	none | underline | overline | line-through	Indica el tipo de subrayado (decoración)
text-transform	none | capitalize | uppercase | lowercase	Transforma a mayús/minús o texto capitalizado

La propiedad **text-decoration** permite establecer **subrayados (underline)**, subrayados por encima del texto **(overline) y tachados (line-through)**. Indicando el valor none se puede eliminar cualquiera de los formatos anteriores. **Es muy utilizado, por ejemplo, para eliminar el subrayado de los textos que tienen un enlace o hipervínculo.**

Por último, la propiedad **text-transform** es muy útil para convertir textos a **mayúsculas (uppercase) o minúsculas (lowercase),** o incluso capitalizar el texto **(capitalize)**, es decir, poner sólo la primera letra en mayúscula, independientemente de cómo esté escrito en el documento HTML.

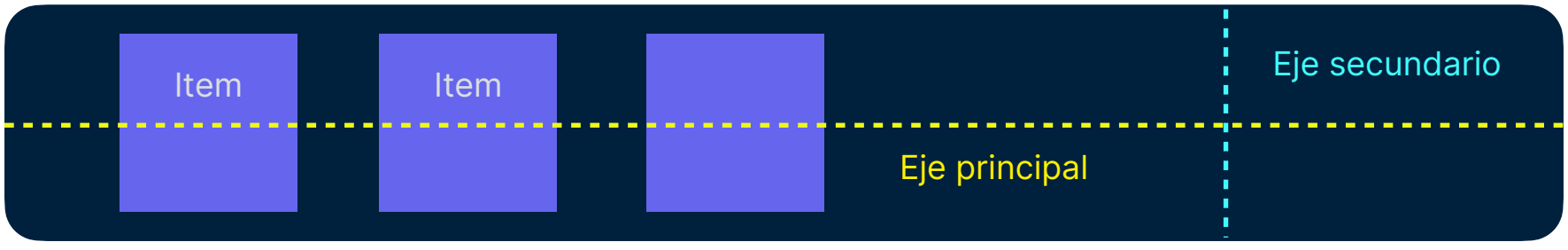
Flexbox css

¿Qué es Flexbox?

Es un sistema de elementos flexibles que llega con la idea de olvidar los mecanismos tradicionales, para que nos acostumbremos a utilizar una mecánica más potente, limpia y personalizable, en la que los elementos HTML se adaptan y colocan automáticamente, de forma que es más fácil personalizar los diseños. Está especialmente diseñado para crear, mediante CSS, estructuras de una sóla dimensión.

Conceptos iniciales

Para empezar a utilizar flexbox lo primero que debemos hacer es conocer algunos de



Contenedor los elementos básicos de este nuevo esquema, que son los

Contenedor: Es el elemento padre que tendrá en su interior cada uno de los ítems flexibles. Observa que al contrario que muchas otras estructuras CSS, por norma general, en Flex establecemos las propiedades al elemento padre.

Eje principal: Los contenedores flexibles tendrán una orientación principal específica. Por defecto, el eje principal del contenedor flexbox es en horizontal (en fila).

Eje secundario: De la misma forma, los contenedores flexibles tendrán una orientación secundaria, perpendicular a la principal. Si la principal es en horizontal, la secundaria será en vertical (y viceversa).

Ítem: Cada uno de los hijos que tendrá el contenedor en su interior.

siguientes:

Una vez tenemos claro esto, imaginemos el siguiente escenario:

```
<div class="container"> <!-- Flex container -->
        <div class="item item-1">1</div> <!-- Flex items -->
        <div class="item item-2">2</div>
        <div class="item item-3">3</div>
    </div>
```

Para activar el modo flexbox, hemos utilizado sobre el elemento contenedor la

Tipo de elemento	Descripción
Inline-flex	Establece un contenedor en línea, similar a inline-block (ocupa solo el contendio).
flex	Establece un contenedor en bloque, similar a block (ocupa todo el ancho del padre).

propiedad display que vimos en Tipos de elementos, y especificar el valor flex o inline-flex (dependiendo de cómo queramos que se comporte el contenedor):

Por defecto, y sólo con esto, observaremos que **los elementos se disponen todos sobre una misma línea**. Esto ocurre porque estamos utilizando el modo **flexbox** y estaremos trabajando con ítems flexibles básicos, garantizando que no se desbordarán ni mostrarán los problemas que, por ejemplo, **tienen los porcentajes sobre elementos que no utilizan flexbox.**

display: flex inline-flex

Dirección de los ejes

Propiedad	Valor	Significado
flex-direction	row | row-reverse | column | column-reverse	Cambia la orientación del eje principal.

Existen **dos propiedades principales para manipular la dirección y comportamiento de los ítems** a lo largo del eje principal del contenedor. **Son las siguientes:**

Mediante la propiedad **flex-direction podemos modificar la dirección del eje principal del contenedor** para que se oriente en **horizontal (por defecto) o en vertical**. Además,

Valor	Descripción
row	Establece la dirección del eje principal en horizontal.
row-reverse	Establece la dirección del eje principal en horizontal (invertido).
column	Establece la dirección del eje principal en vertical.
column-reverse	Establece la dirección del eje principal en vertical (invertido).

también podemos incluir el sufijo -reverse para indicar que coloque los ítems en orden inverso.

Esto nos permite tener un control muy alto sobre el orden de los elementos en una página. Veamos la aplicación de estas propiedades sobre el ejemplo anterior, para modificar el flujo del eje principal del contenedor:

```
.container {
  display: flex;
  flex-direction: column;
  background: steelblue;
}

.item {
  background: grey;
}

```

flex-direction: row | column | row-reverse | column-reverse

Contenedor flexbox multilínea

Existe una propiedad llamada **flex-wrap** con la que podemos especificar el comportamiento del contenedor respecto a evitar que se desborde **(nowrap, valor por defecto)** o permitir que lo haga, en cuyo caso, estaríamos hablando de un contenedor

Propiedad	Valor	Significado
flex-wrap	nowrap | wrap | wrap-reverse	Evita o permite el desbordamiento (multilinea).

flexbox multilinea.

Valor	Descripción
nowrap	Establece los ítems en una sola línea (no permite que se desborde el contenedor).
wrap	Establece los ítems en modo multilínea (permite que se desborde el contenedor).
wrap-reverse	Establece los ítems en modo multilínea, pero en dirección inversa.

Los valores que se pueden aplicar con flex-wrap son:

Teniendo en cuenta estos valores de la propiedad **flex-wrap, podemos conseguir resultados como el siguiente:**

```css
.container {
  display: flex;
  flex-wrap: wrap;  /* Comportamiento por defecto: nowrap */
  background: steelblue;
  width: 200px;
}

.item {
  background: grey;
  width: 50%;
}

```


En el caso de especificar nowrap (u omitir la propiedad **flex-wrap**) en el contenedor, **los 3 ítems se mostrarían en una misma línea del contenedor.** En ese caso, cada ítem debería tener un 50% de ancho (o sea, 100px de los 200px del contenedor).

Un tamaño de 100px por ítem, sumaría un total de 300px, que no cabrían en el

contenedor de 200px, por lo que **flexbox reajusta los ítems flexibles para que quepan todos en la misma línea**, manteniendo las mismas proporciones.

Sin embargo, si especificamos **wrap** en la propiedad **flex-wrap**, lo que permitimos es que **el contenedor se pueda desbordar, pasando a ser un contenedor multilínea,** que mostraría el ítem 1 y 2 en la primera linea (con un tamaño de 100px cada uno) y el ítem 3 en la línea siguiente, dejando un espacio libre para un posible ítem 4.

Atajo: Dirección de los ejes

Recuerda que existe una propiedad de atajo (short-hand) llamada **flex-flow**, con la que podemos resumir los valores de las propiedades **flex-direction y flex-wrap**, especificándolas en una sola propiedad y **ahorrándonos utilizar las propiedades concretas.**

```
.container {
  /* flex-flow: <flex-direction> <flex-wrap>; */
  flex-flow: row wrap;
}

```

Propiedades de alineación

Ahora que tenemos un control básico del contenedor de estos ítems flexibles,

Propiedad	Valor	Eje
justify-content	flex-start | flex-end | center | space-between | space-around | space-evenly	Principal
align-content	flex-start | flex-end | center | space-between | space-around | space-evenly | stretch	Secundario
align-items	flex-start | flex-end | center | stretch | baseline	Secundario
align-self	auto | flex-start | flex-end | center | stretch | baseline	Secundario

necesitamos conocer las propiedades existentes dentro de flexbox para disponer los

En esta lista, tenemos que centrarnos en la primera y la tercera propiedad (justify-content y align-items), que son las más importantes (las otras dos son casos particulares que explicaremos más adelante):

- **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).

ítems dependiendo de nuestro objetivo. Echemos un vistazo a 4 propiedades interesantes, la primera de ellas actúa en el eje principal, mientras que el resto en el eje secundario:

Sobre el eje principal:

La primera propiedad, justify-content, sirve para colocar los ítems de un contenedor mediante una disposición concreta a lo largo del eje principal:

Valor	Descripción
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 máximo espacio para separarlos.
space-around	Distribuye los ítems dejando el mismo espacio alrededor de ellos (izq/dcha).

space-evenly	Distribuye los ítems dejando el mismo espacio (solapado) a izquierda y derecha.

> Una vez entendido este caso, debemos atender a la propiedad **align-content**, que es un caso particular del anterior. **Nos servirá cuando estemos tratando con un contenedor flex multilinea**, que es un contenedor en el que los ítems no caben en el ancho disponible y, por lo tanto, el eje principal se divide en múltiples líneas (por ejemplo, usando flex-wrap: wrap).

De esta forma, **align-content** servirá para alinear cada una de las líneas del **contenedor multilínea**. Los valores que puede tomar son los siguientes:

Valor	Descripción
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 máximo espacio para separarlos.
space-around	Distribuye los ítems dejando el mismo espacio alrededor de ellos (izq/dcha).
stretch	Estira los ítems para ocupar de forma equitativa todo el espacio.

Con estos valores, vemos cómo cambiamos la disposición en vertical (porque partimos de un ejemplo en el que estamos utilizando flex-direction: row, y el eje principal es horizontal) **de los ítems que están dentro de un contenedor multilínea.**

A continuación, veremos que al indicar un contenedor de 200 píxels de alto con ítems de 50px de alto y un flex-wrap establecido para tener contenedores multilinea, **podemos utilizar la propiedad align-content para alinear los ítems de forma vertical de modo que se queden en la zona inferior del contenedor.**

```
.container {
  background: #CCC;
  display: flex;
  width: 200px;
  height: 200px;

  flex-wrap: wrap;
  align-content: flex-end;
}

.item {
  background: #777;
  width: 50%;
  height: 50px;
}
```

Sobre el eje secundario

La otra propiedad importante de este apartado es **align-items**, que se encarga de alinear los ítems en el eje secundario del contenedor. **Atención con no confundir align-content con align-items**, puesto que el primero actúa sobre cada una de las líneas de un contenedor multilinea (no tiene efecto sobre contenedores de una sola línea), mientras que **align-items** lo hace sobre la línea actual.

Los valores que puede tomar son los siguientes:

Valor	Descripción
flex-start	Alinea los ítems al principio del eje secundario.
flex-end	Alinea los ítems al final del eje secundario.

center	Alinea los ítems al centro del eje secundario.
stretch	Alinea los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor.
baseline	Alinea los ítems en el contenedor según la base del contenido de los ítems del contenedor.

> Por otro lado, la propiedad **align-self** actúa exactamente **igual que align-items, sin embargo, es la primera propiedad de flexbox que vemos que se utiliza sobre un ítem hijo específico** y no sobre el elemento contenedor. Salvo por este detalle, funciona exactamente igual que align-items.

Gracias a ese detalle, **align-self** nos permite **cambiar el comportamiento de align-items** y sobre-escribirlo con comportamientos específicos para ítems concretos que no queremos que se comporten igual que el resto.

La propiedad puede tomar los siguientes valores:

Valor	Descripción
flex-start	Alinea los ítems al principio del contenedor.
flex-end	Alinea los ítems al final del contenedor.
center	Alinea los ítems al centro del contenedor.
stretch	Alinea los ítems estirándolos al tamaño del contenedor.

baseline	Alinea los ítems en el contenedor según la base de los ítems.
auto	Hereda el valor de align-items del padre (si no se ha definido, es stretch).

Atajo: Alineaciones

Existe una propiedad de atajo con la que se pueden establecer los valores de **align-content** y de **justify-content** de una sola vez, **denominada place-content:**

```
.container {
  display: flex;
  place-content: flex-start flex-end;

  /* Equivalente a... */
  align-content: flex-start;
  justify-content: flex-end;
}

```

Si se especifica el valor auto a la propiedad **align-self, el navegador le asigna el valor de la propiedad align-items del contenedor padre**, y en caso de no existir, **el valor por defecto: stretch**.

Propiedades de hijos

A excepción de la propiedad align-self, todas las propiedades que hemos visto hasta ahora se aplican sobre el elemento contenedor. Las siguientes propiedades, sin embargo, se aplican sobre los ítems hijos.

Propiedad	Valor	Descripción
flex-grow	0	Número que indica el factor de crecimiento del ítem respecto al resto.
flex-shrink	1	Número que indica el factor de decrecimiento del ítem respecto al resto.
flex-basis	content	Tamaño base de los ítems antes de aplicar variación.
order	0	Número (peso) que indica el orden de aparición de los ítems.

En primer lugar, **tenemos la propiedad flex-grow para indicar el factor de crecimiento de los ítems en el caso de que no tengan un ancho específico.** Por ejemplo, si con **flex-grow** indicamos un valor de 1 a todos sus ítems, tendrían el mismo tamaño cada uno de ellos. Pero si colocamos un valor de 1 a todos los elementos, salvo a uno de ellos, que le indicamos 2, ese ítem será más grande que los anteriores. **Los ítems a los que no se le especifique ningún valor, tendrán por defecto valor de 0.**

En segundo lugar, tenemos la propiedad **flex-shrink** que es la opuesta a flex-grow. Mientras que la anterior indica un factor de crecimiento, **flex-shrink hace justo lo contrario, aplica un factor de decrecimiento.** De esta forma, los ítems que tengan un valor numérico más grande, serán más pequeños, mientras que **los que tengan un valor numérico más pequeño serán más grandes, justo al contrario de cómo funciona la propiedad flex-grow.**

Por último, tenemos **la propiedad flex-basis, que define el tamaño por defecto (de base) que tendrán los ítems antes de aplicarle la distribución de espacio.** Generalmente, se aplica un tamaño (unidades, porcentajes, etc....), pero también se puede aplicar la palabra clave content que ajusta automáticamente el tamaño al contenido del ítem, que es su valor por defecto.

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

```

Atajo: Propiedades de hijos

Existe una propiedad llamada **flex** **que sirve de atajo para estas tres propiedades de los ítems hijos. Funciona de la siguiente forma:**

Propiedad	Valor	Descripción
row-gap	normal	Espacio entre filas (sólo si flex-direction: column)
column-gap	normal	Espacio entre columnas (sólo si flex-direction: row)

Ten en cuenta que **sólo una de las dos propiedades tendrá efecto, dependiendo de si la propiedad flex-direction está establecida en column o en row.** Eso sí, es posible usar ambas si tenemos la propiedad flex-wrap definida a wrap y, por lo tanto, disponemos de multicolumnas flexbox.

Huecos (gaps)

Existen **dos propiedades de flexbox que han surgido recientemente: row-gap y column-gap.** Dichas propiedades, **permiten establecer el tamaño de un «hueco» entre ítems desde el elemento padre contenedor**, y sin necesidad de estar utilizando padding o margin en los elementos hijos.

- **justify-content**: flex-start | flex-end | center | space-between | space-around space-evenly
- **align-items:** flex-start | flex-end | center | stretch | baseline
- **align-content:** flex-start | flex-end | center | stretch | baseline
- **flex-wrap:** nowrap | wrap | wrap-reverse
- **row-gap:** normal 5px 25px
- **column-gap:** normal 5px 25px

Propiedad	Valor	Descripción
gap	0	Aplica el tamaño indicado para el hueco en ambos ejes.
gap	0	Aplica los tamaños indicados para el hueco del eje X y el eje Y.

Atajo: Huecos

En el caso de que queramos utilizar una propiedad de atajo para los huecos, podemos utilizar la propiedad gap:

Veámosla en práctica:

```
.container {
  /* gap: <row> <column> */
  gap: 4px 8px;

  /* 1 parámetro: usa el mismo para ambos */
  gap: 4px;
}

```

De esta forma **podemos recolocar fácilmente los ítems incluso utilizando media queries o responsive design.**

Orden de los ítems

Por último, y quizás una de las propiedades más interesantes, es **order**, que **modifica y establece el orden de los ítems según una secuencia numérica.**

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.

order (item-1):

order (item-2):

order (item-3):

Se recomienda **profundizar con diferentes fuentes y practicar de manera continua** para mejorar su implementación en proyectos personales.

Conclusiones

El lenguaje css es intuitivo y útil para generar diferentes efectos visuales dentro de un proyecto. Su manejo es fundamental para todo desarrollo web, especialmente en el área frontend, y **requiere de una organización y método claros, que permitan la determinación de clases y correcta asignación de atributos.**

Es importante destacar que la presente guía es una selección acotada de los aspectos que consideramos fundamentales para el desarrollo frontend en CSS. No busca abarcar el enorme universo que representa dicho lenguaje, sino aproximar al lector al mundo de los estilos y animaciones.