

Usando las cajas flexibles CSS

En este artículo

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.

Muchos diseñadores verán que el modelo "caja flexible" es más sencillo de utilizar. Los elementos "hijos" de una "caja flexible" pueden colocarse en cualquier dirección y pueden tener dimensiones flexibles para adaptarse al espacio visible. Posicionar los elementos "hijos" es por tanto mucho más sencillo, y los diseños complejos pueden hacerse más fácilmente y con código más limpio, ya que el orden de visualización de los elementos es independiente del orden que estos tengan en el código fuente. Esta independencia afecta intencionadamente únicamente a la representación visual, dejando el orden de locución y navegación a lo que diga el código fuente.

Nota: Aunque la especificación del diseño de "cajas flexibles" de CSS está en fase de "candidata a recomendación", no todos los navegadores la han implementado. La implementación de WebKit requiere el prefijo `-webkit`; Internet Explorer implementa una versión antigua de la especificación, con prefijo; Opera 12.10 implementa la última versión, sin prefijo. Revisa la tabla de compatibilidad de cada propiedad para saber cuál es el estado actual de compatibilidad.

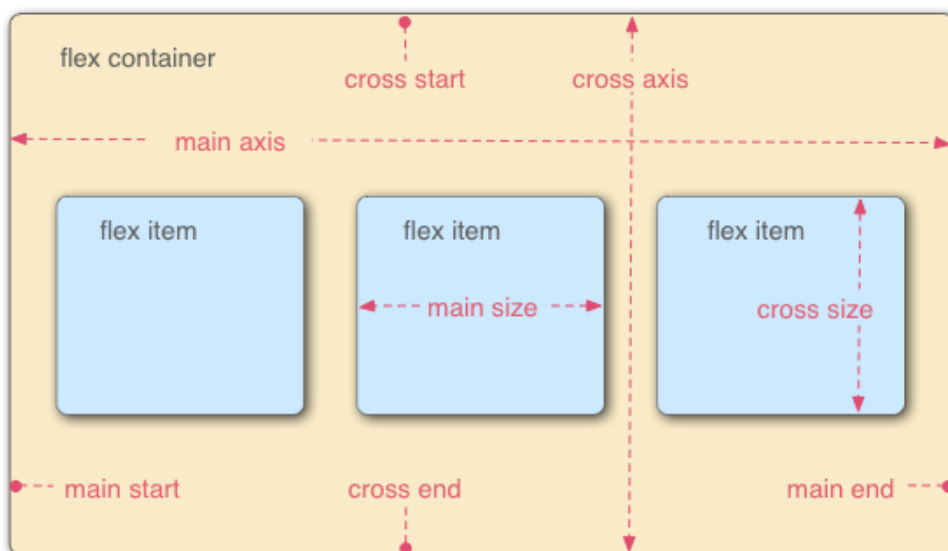
El concepto de "cajas flexibles"

Lo que caracteriza un diseño flexible es su habilidad para alterar el ancho y alto de sus elementos para ajustarse lo mejor posible al espacio disponible en cualquier dispositivo. Un contenedor flexible expande sus elementos para rellenar el espacio libre, o los comprime para evitar que rebasen el área prevista.

El algoritmo del modelo de diseño de "cajas flexibles" no parte de ninguna dirección predeterminada, al contrario de lo que ocurre con el modelo "bloque", que asume una disposición vertical de los elementos, o lo que pasa con el modelo "en línea", que asume una disposición horizontal. Mientras que el modelo "bloque" funciona bien para páginas, se queda muy corto cuando se trata de aplicaciones en las que hay que tener en cuenta el cambio de orientación del dispositivo o los cambios de tamaño realizados por los gestos del usuario. El modelo de "cajas flexibles" es más apropiado para diseños de pequeña escala, mientras que el (emergente) modelo "rejilla" es adecuado para diseños de gran escala. Ambos son parte del gran esfuerzo que el "CSS Working Group" está realizando para proveer de mayor interoperabilidad a las aplicaciones web con todo tipo de usuarios, distintos modos de escritura, y otras necesidades de flexibilidad.

Vocabulario de "cajas flexibles"

Aunque al hablar de las "cajas flexibles" nos olvidamos de términos como "horizontal/en línea" y "vertical/bloque", se hace necesario emplear una nueva terminología. Fíjate en el siguiente diagrama para afianzar el vocabulario empleado en sus elementos. Se muestra un contenedor flexible que tiene una `flex-direction` de tipo `row`, que significa que los elementos flexibles se muestran uno a continuación del otro horizontalmente a lo largo del eje principal (main axis) de acuerdo con el modo de escritura preestablecido, y en este caso, la dirección en que el texto de los elementos fluye es de izquierda-a-derecha.



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`.

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.

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

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

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.

Diseñando una "caja flexible"

Para indicar que unos elementos tienen este estilo CSS, asigna la propiedad `display` así:

```
display : flex
```

o

```
display : inline-flex
```

Haciendo esto, se define el elemento como contenedor flexible y todos sus "hijos" como elementos flexibles. El valor `flex` hace que el contenedor flexible sea un bloque dentro del elemento "padre" al que pertenezca. El valor `inline-flex` hace que el contenedor flexible sea un elemento "en línea" dentro del elemento "padre" al que pertenezca.

Nota: Cuando utilices un prefijo para el tipo de navegador, ponlo en la propiedad "display" no en el atributo "display". Por ejemplo, `display : -webkit-flex`.

Consideraciones de los elementos flexibles

El texto que se encuentre directamente dentro de un contenedor flexible, será automáticamente envuelto en un elemento flexible anónimo. Sin embargo, si un elemento flexible contiene solamente espacios en blanco no será mostrado, como si tuviera la propiedad `display:none`.

Los "hijos" de un contenedor flexible que tengan un posicionamiento absoluto, se situarán de manera que su posición estática se determine en referencia a la esquina del **inicio principal (main start)** de su contenedor flexible.

Actualmente, debido a un problema conocido, asignar `visibility:collapse` a un elemento flexible causa que sea tratado como si fuera `display:none` en vez de lo que se supone que debería ocurrir, es decir, como si fuera `visibility:hidden`. La alternativa mientras se resuelve este problema es usar `visibility:hidden` para elementos flexibles que deban comportarse como `visibility:collapse`.

Los márgenes de elementos flexibles adyacentes no se colapsan. Usando márgenes `auto` se absorbe el espacio extra vertical y horizontalmente y puede ser utilizado para alinear o separar

elementos flexibles adyacentes. Ver [Aligning with 'auto' margins](#) en la especificación "W3C Flexible Box Layout Model" para más detalles al respecto.

Para asegurar un tamaño mínimo por defecto de los elementos flexibles, usa `min-width:auto` y/o `min-height:auto`. Para los elementos flexibles, el valor de atributo `auto` calcula la mínima anchura/altura del elemento para que no sea menor que la anchura/altura de su contenido, garantizando que el elemento es mostrado suficientemente grande como para que se vea su contenido. Ver [min-width](#) y [min-height](#) para más detalles al respecto.

Las propiedades de alineación de "cajas flexibles" realizan un "verdadero" centrado en CSS. Esto significa que los elementos flexibles permanecerán centrados, incluso si estos rebasan su contenedor flexible. Esto puede llegar a ser un problema, ya que si sobrepasan el tope superior de la página o el izquierdo (en escritura LTR de izquierda-a-derecha) o el derecho (en escritura RTL de derecha-a-izquierda), no se puede desplazar hacia ese área, incluso habiendo contenido allí. En el futuro, las propiedades de alineación se ampliarán para que tengan una opción "safe" (seguro) para controlar esta situación. De momento, si esto te preocupa, puedes usar los márgenes para conseguir el centrado, ya que estos responderán de modo seguro parando el centrado si se sobrepasan los límites. En vez de usar las propiedades `align-`, simplemente pon márgenes automáticos en los elementos flexibles que quieras centrar. En vez de usar las propiedades `justify-`, pon márgenes automáticos en los límites exteriores del primer y último elemento flexible del contenedor flexible. Los márgenes automáticos se adaptarán asumiendo el espacio sobrante, centrando los elementos flexibles donde sobre espacio, y cambiando a alineación normal donde no sobre espacio. Sin embargo, si tratas de reemplazar `justify-content` con "centrado-basado-en-márgenes" en una "caja flexible" multi-línea, probablemente no funcionará, ya que tendrías que poner márgenes en el primer y último elemento de cada línea. A menos que puedas predecir qué elementos encajarán en cada línea, no tendrás una respuesta fiable usando el "centrado-basado-en-márgenes" en el eje principal al reemplazar la propiedad `justify-content`.

Recuerda que mientras el orden en que se muestran los elementos es independiente de su orden en el código fuente, esta independencia afecta solamente a la representación visual, y no al orden de locución y navegación que seguirán el orden establecido en el código fuente. Incluso la propiedad [order](#) no afectará a la secuencia de locución ni de navegación. Así que los desarrolladores deben preocuparse del orden de los elementos adecuadamente en el código fuente para que no se deteriore la accesibilidad del documento.

Propiedades de las "cajas flexibles"

Propiedades que no afectan a las "cajas flexibles"

Como las "cajas flexibles" emplean un algoritmo diferente, algunas propiedades no tienen sentido para un contenedor flexible.

- Propiedades `column-*` del [Módulo Multicol](#) no tienen ningún efecto en un elemento flexible.
- `float` y `clear` no tienen ningún efecto en un elemento flexible. Usar `float` causa que la propiedad `display` del elemento se comporte como `block`.
- `vertical-align` no tiene efecto en la alineación de los elementos flexibles.

Ejemplos

Ejemplo básico "flex"

Este ejemplo básico muestra como aplicar "flexibilidad" a un elemento y como sus "hijos" se comportan flexiblemente.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <style>

      .flex
      {
        /* basic styling */
        width: 350px;
        height: 200px;
        border: 1px solid #555;
        font: 14px Arial;

        /* flexbox setup */
        display: -webkit-flex;
        -webkit-flex-direction: row;

        display: flex;
        flex-direction: row;
      }

      .flex > div
      {
        -webkit-flex: 1 1 auto;
        flex: 1 1 auto;

        width: 30px; /* To make the transition work
```

"width:auto" are buggy in C
See <http://bugzil.la/731886>

```
-webkit-transition: width 0.7s ease-out;
transition: width 0.7s ease-out;
}

/* colors */
.flex > div:nth-child(1){ background : #009246;
.flex > div:nth-child(2){ background : #F1F2F1;
.flex > div:nth-child(3){ background : #CE2B35;

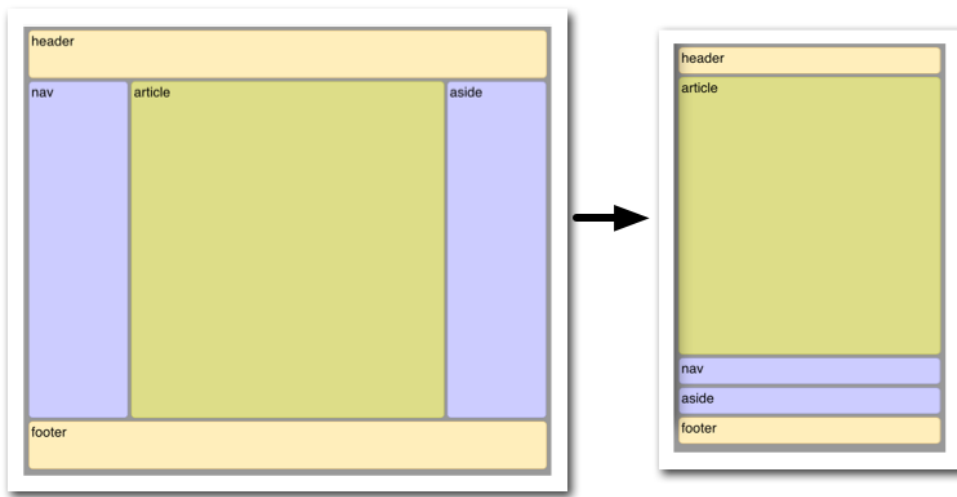
.flex > div:hover
{
    width: 200px;
}

</style>

</head>
<body>
<p>Flexbox nuovo</p>
<div class="flex">
    <div>uno</div>
    <div>due</div>
    <div>tre</div>
</div>
</body>
</html>
```

Ejemplo de "Diseño del Santo Grál"

Este ejemplo muestra como la "caja flexible" proporciona la habilidad de cambiar dinámicamente el diseño para distintas resoluciones de pantalla. El diagrama siguiente ilustra la transformación.



Aquí se muestra el caso en que el diseño de la página adaptado a un navegador tiene que mostrarse óptimamente en un smartphone. No solamente los elementos se reducen de tamaño, sino que el orden en que se muestran también cambia. La "caja flexible" lo hace muy sencillo.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <style>

    body {
      font: 24px Helvetica;
      background: #999999;
    }

    #main {
      min-height: 800px;
      margin: 0px;
      padding: 0px;
      display: -webkit-flex;
      display: flex;
      -webkit-flex-flow: row;
      flex-flow: row;
    }

    #main > article {
      margin: 4px;
      padding: 5px;
      border: 1px solid #cccc33;
      border-radius: 7pt;
```



```
background: #ddd88;
-webkit-flex: 3 1 60%;
    flex: 3 1 60%;
-webkit-order: 2;
    order: 2;
}

#main > nav {
    margin: 4px;
    padding: 5px;
    border: 1px solid #888bb;
    border-radius: 7pt;
    background: #ccccff;
    -webkit-flex: 1 6 20%;
        flex: 1 6 20%;
    -webkit-order: 1;
        order: 1;
}

#main > aside {
    margin: 4px;
    padding: 5px;
    border: 1px solid #888bb;
    border-radius: 7pt;
    background: #ccccff;
    -webkit-flex: 1 6 20%;
        flex: 1 6 20%;
    -webkit-order: 3;
        order: 3;
}

header, footer {
    display: block;
    margin: 4px;
    padding: 5px;
    min-height: 100px;
    border: 1px solid #eebb55;
    border-radius: 7pt;
    background: #ffeebb;
}

/* Too narrow to support three columns */
@media all and (max-width: 640px) {
```

```
#main, #page {
  -webkit-flex-flow: column;
  flex-flow: column;
}

#main > article, #main > nav, #main > aside {
  /* Return them to document order */
  -webkit-order: 0;
  order: 0;
}

#main > nav, #main > aside, header, footer {
  min-height: 50px;
  max-height: 50px;
}
}

</style>
</head>
<body>
<header>header</header>
<div id='main'>
  <article>article</article>
  <nav>nav</nav>
  <aside>aside</aside>
</div>
<footer>footer</footer>
</body>
</html>
```

Área de juego

Hay varias áreas de juego de "cajas flexibles" disponibles on-line para experimentar:

- [Flexbox Playground](#)
- [Flexy Boxes](#)

Cosas a tener en mente

El algoritmo de las "cajas flexibles" puede ser un poco complejo de entender a veces. Hay una serie de cosas a considerar para evitar






"sorpresas" cuando se utilizan las "cajas flexibles".

Las "cajas flexibles" se comportan en función del [modo de escritura](#) establecido. Esto significa que **inicio principal (main start)** y **fin principal (main end)** se disponen de según la posición de **inicio (start)** y **fin (end)**.

inicio secundario (cross start) y **fin secundario (cross end)** confían en la definición de la posición **inicio (start)** o **antes(before)** que depende del valor de [direction](#).

Los saltos de página son posibles en el diseño de "cajas flexibles" siempre que la propiedad `break-` lo permita. Las propiedades CSS3 `break-after`, `break-before` y `break-inside` así como las propiedades CSS 2.1 `page-break-before`, `page-break-after` y `page-break-inside` se aceptan en los contenedores flexibles, también en los elementos flexibles que ellos contienen, y también en los elementos que esos elementos flexibles a su vez contienen.

Compatibilidad de Navegadores

	Desktop	Mobile				
Caraterística	Firefox (Gecko)	Chrome	Internet Explorer	Opera	Safari	
Soporte Básico	18.0 (18.0)  -moz (Behind a pref) [2] 22.0 (22.0)	21.0  -webkit	10  -ms (partial)	12.1	3.1  -webkit (partial) 6.1  -webkit 9	


Notas

[1] Internet Explorer 10 y Safari soportan un antiguo borrador de la especificación que es incompatible. No han sido actualizados para soportar la versión final.

[2] Firefox soporta solamente la "caja flexible" con una sola línea. Para activar el soporte de "caja flexible" el usuario tiene que cambiar la preferencia `about:config "layout.css.flexbox.enabled"` a `true`.

[3] El navegador de Android hasta la versión 4.3 soporta un borrador antiguo e incompatible de la especificación. Android 4.4 ha sido actualizado para dar soporte a la versión final.

[4] Mientras que en la implementación inicial en Opera 12.10 `flexbox` no estaba en el prefijo, obtuvo prefijos en las

versiones de la 15 a la 16 de Opera y 15 a 19 de Opera Mobile con -webkit . Este prefijo, fue eliminado de nuevo en Opera 17 y Opera Mobile 24.

[5] Hasta Firefox 29, especificar `visibility: collapse` en un elemento flex causaba que fuera tratado como si fuera `display: none` en vez del comportamiento pretendido, tratándolo como si fuera `visibility: hidden` . El método alternativo sugerido es usar `visibility: hidden` para los elementos flex que debieran comportarse como si hubieran sido designados `visibility: collapse` . Para más información, ver [bug 783470](#).

Ver también

- [El Proyecto Flexbugs](#) para información de errores en implementación de flexbox en navegadores.

¿Te resultó útil este artículo?



Colaboradores en esta

página: [Tonylu11](#), [javier_junin](#), [AlePerez92](#), [MMariscal](#), [fscholz](#), [ArcangelZith](#), [FNK](#), [rippe2hl](#), [StripTM](#), [joan.le](#)

Última actualización por: [Tonylu11](#), 29 dic. 2016 4:50:58

Ver también

CSS

Referencia CSS

CSS Flexible Box Layout

Guías

[Advanced layouts with flexbox](#) [\[Traducir\]](#)

[Usando las cajas flexibles CSS](#)

[Usando flexbox para componer aplicaciones web](#)

Propiedades

[align-content](#)

[align-items](#)

[align-self](#)

[flex](#)

[flex-basis](#)

[flex-direction](#)

[flex-flow](#)

[flex-grow](#)

`flex-shrink` [\[Traducir\]](#)

`flex-wrap`

`justify-content`

`order`

