

REFUERZO

POSICIONAMIENTO
FLOTANTE Y DISEÑO
DE TABLAS

Sumario

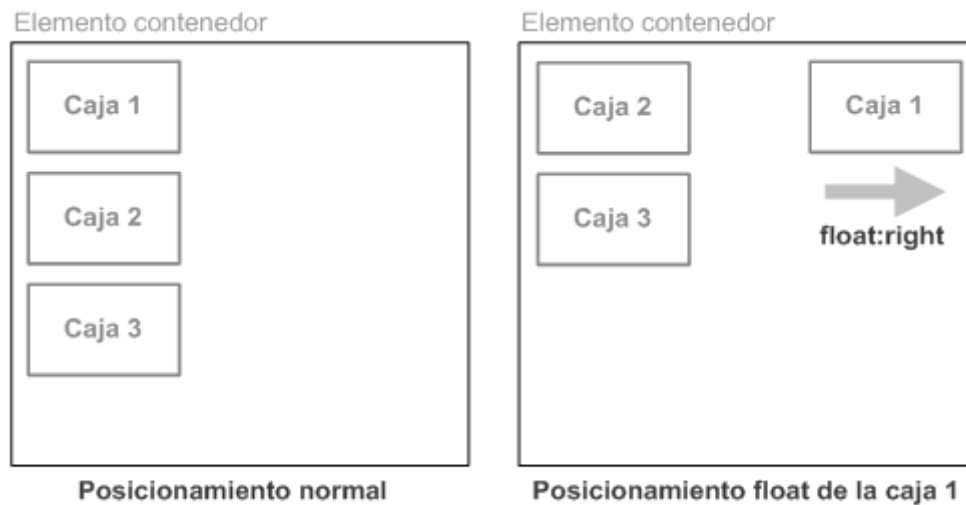
1. Posicionamiento Flotante.....	2
1.1. Ejemplos:.....	5
1.1.1. Letra capital.....	5
1.1.2. Posicionamiento flotante de imágenes.....	6
1.1.3. La propiedad clear.....	9
1.1.4. Tamaño de los elementos que contienen elementos flotantes.....	11
2. Diseño simplista.....	13
2.1. Conclusión sobre diseñar con elementos flotantes.....	20
3. Diseño con tablas.....	21
3.1. La propiedad display.....	21
4. Diseño simple con una tabla.....	22
4.1. Ancho en pantalla de la tabla.....	24
4.2. Elementos anónimos.....	25
4.3. Otros elementos para las tablas.....	25
4.3.1. Párrafos en las celdas.....	25
4.3.2. Listas en tablas.....	26
5. Diseño de las filas.....	27
5.1. Establecer el ancho de las celdas.....	29
5.1.1. Anchos fijos.....	29
5.1.2. Anchos en porcentaje.....	29
5.2. Diseño con una tabla más estructurada.....	30
5.3. Otras propiedades de diseño.....	32
5.3.1. Propiedad table-layout.....	32
5.3.2. Propiedad border-collapse.....	33
5.3.3. Propiedad border-spacing.....	36
5.3.4. Propiedad empty-cells.....	36
5.3.5. Propiedad caption-side.....	37
5.4. Alineación vertical.....	39
6. Conclusión.....	40

1. Posicionamiento Flotante

El posicionamiento flotante es el más difícil de comprender pero al mismo tiempo es el más utilizado. La mayoría de estructuras de las páginas web complejas están diseñadas con el posicionamiento flotante.

Cuando una caja se posiciona con el modelo de posicionamiento flotante, automáticamente se convierte en una *caja flotante*, lo que significa que se desplaza hasta **la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.**

La siguiente imagen muestra el resultado de posicionar de forma flotante hacia la derecha la caja 1:

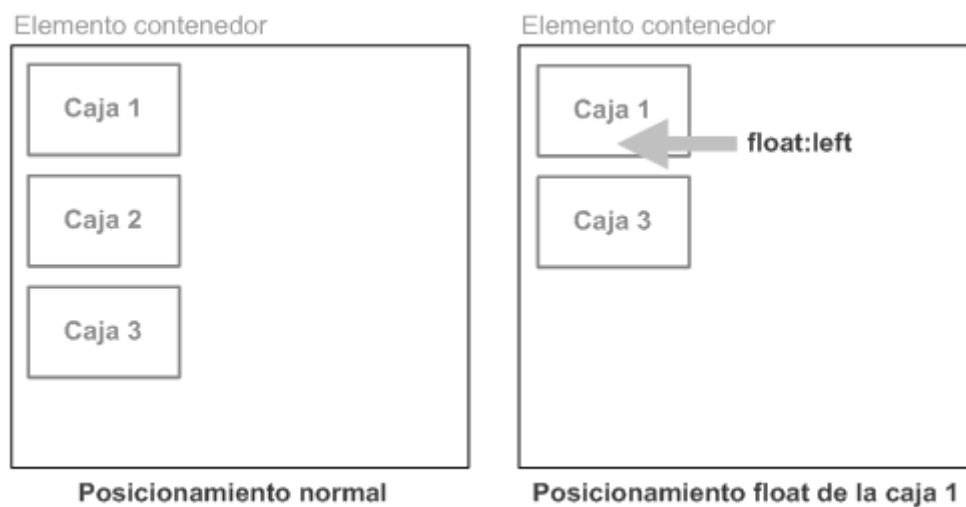


Ejemplo de posicionamiento float de una caja

Cuando se posiciona una caja de forma flotante:

- La caja deja de pertenecer al **flujo normal de la página**, lo que significa que el resto de cajas ocupan el lugar dejado por la caja flotante.
- La caja flotante se posiciona lo **más a la izquierda o lo más a la derecha posible de la posición en la que se encontraba originalmente**.

Si en el anterior ejemplo la caja 1 se posiciona de forma flotante hacia la izquierda, el resultado es el que muestra la siguiente imagen:

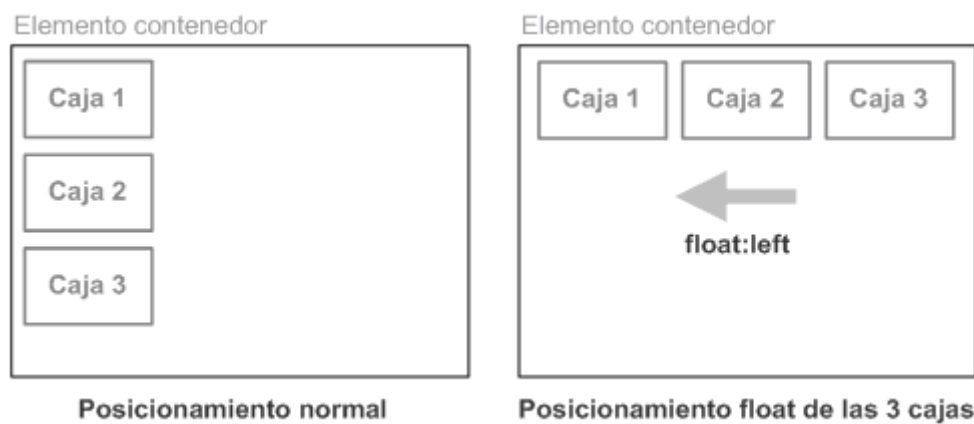


Ejemplo de posicionamiento float de una caja

La caja 1 es de tipo flotante, por lo que *desaparece del flujo normal* de la página y el resto de cajas ocupan su lugar. El resultado es que la caja 2 ahora se muestra donde estaba la caja 1 y la caja 3 se muestra donde estaba la caja 2.

Al mismo tiempo, la caja 1 se desplaza todo lo posible hacia la izquierda de la posición en la que se encontraba. El resultado es que la caja 1 se muestra encima de la nueva posición de la caja 2 y tapa todos sus contenidos.

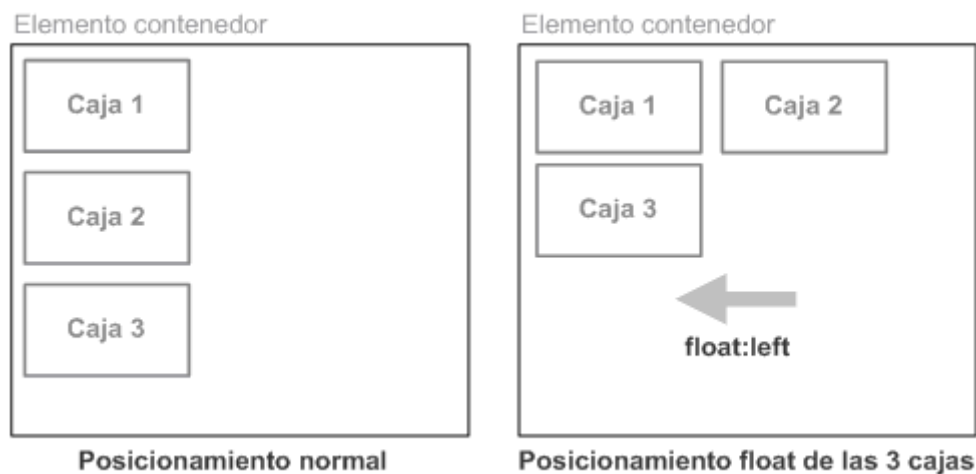
Si existen otras cajas flotantes, al posicionar de forma flotante otra caja, se tiene en cuenta el sitio disponible. En el siguiente ejemplo se posicionan de forma flotante hacia la izquierda las tres cajas:



Ejemplo de posicionamiento float de varias cajas

En el ejemplo anterior, las cajas no se superponen entre sí porque las cajas flotantes tienen en cuenta las otras cajas flotantes existentes. Como la caja 1 ya estaba posicionada lo más a la izquierda posible, la caja 2 sólo puede colocarse al lado del borde derecho de la caja 1, que es el sitio más a la izquierda posible respecto de la zona en la que se encontraba.

Si no existe sitio en la línea actual, la caja flotante baja a la línea inferior hasta que encuentra el sitio necesario para mostrarse lo más a la izquierda o lo más a la derecha posible en esa nueva línea:



Ejemplo de posicionamiento float cuando no existe sitio suficiente

Las cajas flotantes influyen en la disposición de todas las demás cajas. Los elementos en línea *hacen sitio* a las cajas flotantes adaptando su anchura al espacio libre dejado por la caja desplazada. Los elementos de bloque no les hacen sitio, pero sí que adaptan sus contenidos para que no se solapen con las cajas flotantes

1.1. Ejemplos:

1.1.1. Letra capital

Se puede crear una letra capital aplicando la propiedad float a la primera letra de un párrafo. En los ejemplos siguientes se utiliza una etiqueta `` o la pseudo-clase `:first-letter`.

```
span.capital {  
  background-color: pink;  
  color: red;  
  float: left;  
  font-family: monospace;  
  font-size: 400%;  
}
```

La primera letra de este párrafo es una letra capital, es decir, una letra más grande que ocupa varias líneas de texto. Para ello, en la hoja de estilo hay que hacer la letra flotante y aumentar su tamaño. En este caso se ha cambiado también el color y el tipo de letra para resaltar el espacio ocupado por la primera letra.

```
p.capital:first-letter {  
  background-color: pink;  
  color: red;  
  float: left;  
  font-family: monospace;  
  font-size: 400%;  
}
```

La primera letra de este párrafo es una letra capital, es decir, una letra más grande que ocupa varias líneas de texto. Para ello, en la hoja de estilo hay que hacer la letra flotante y aumentar su tamaño. En este caso se ha cambiado también el color y el tipo de letra para resaltar el espacio ocupado por la primera letra.

1.1.2. Posicionamiento flotante de imágenes

Las imágenes **son elementos en línea**, es decir, que se insertan como si fueran **caracteres**, formando parte del párrafo o del elemento de bloque en el que se insertan. La altura de la línea en la que está insertado el elemento aumenta lo necesario para poder alojar la imagen, como muestra el siguiente ejemplo, en el que la hoja de estilo no contiene ninguna propiedad relacionada con la imagen.

```
img {  
}
```



Este párrafo tiene insertada una imagen al principio del párrafo. Se trata del logotipo de GNU. GNU es un acrónimo recursivo que significa "GNU is Not Unix" (GNU No es Unix). GNU es un proyecto de software libre iniciado por Richard Stallman en 1984. La imagen forma parte del párrafo, es una letra más en la primera línea.

Si se quiere que una imagen aparezca a la izquierda (o a la derecha) de un texto, es decir, que el texto fluya a lo largo de la imagen, hay que utilizar la propiedad `float`. Esta propiedad sólo admite dos valores, `left` y `right`, que sitúan la imagen a la izquierda o a la derecha, como muestran los ejemplos siguientes.

```
img {  
    float: left;  
}
```



Este párrafo tiene insertada una imagen al principio del párrafo. Se trata del logotipo de GNU. GNU es un acrónimo recursivo que significa "GNU is Not Unix" (GNU No es Unix). GNU es un proyecto de software libre iniciado por Richard Stallman en 1984. La imagen debe "flotar" a la izquierda y el texto debe fluir a su derecha.

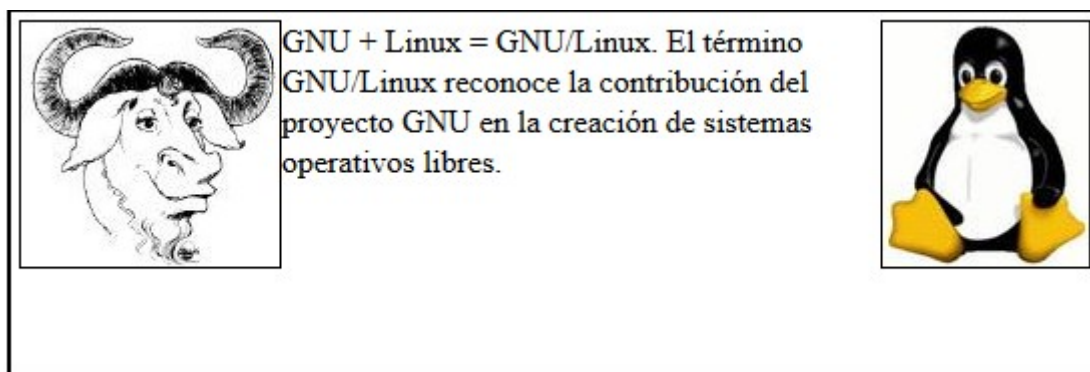
```
img {  
    float: right;  
}
```

Este párrafo tiene insertada una imagen al principio del párrafo. Se trata del logotipo de GNU. GNU es un acrónimo recursivo que significa "GNU is Not Unix" (GNU No es Unix). GNU es un proyecto de software libre iniciado por Richard Stallman en 1984. La imagen debe "flotar" a la derecha y el texto debe fluir a su izquierda.



Si queremos tener una imagen a la izquierda y otra a la derecha, debemos definir clases y asignarlas a la imagen correspondiente

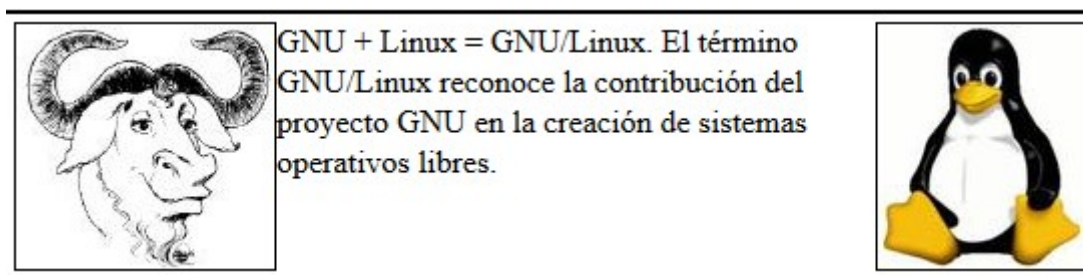
```
img.izquierda {  
    float: left;  
}  
  
img.derecha {  
    float: right;  
}  
  
<p><img class="izquierda" /><img class="derecha" />Texto</p>
```



Para que las imágenes salgan correctamente alineadas con el texto, la imagen debe insertarse al principio del texto, independientemente de la posición final que vaya a tener la imagen. Los siguientes ejemplos muestran las diferencias

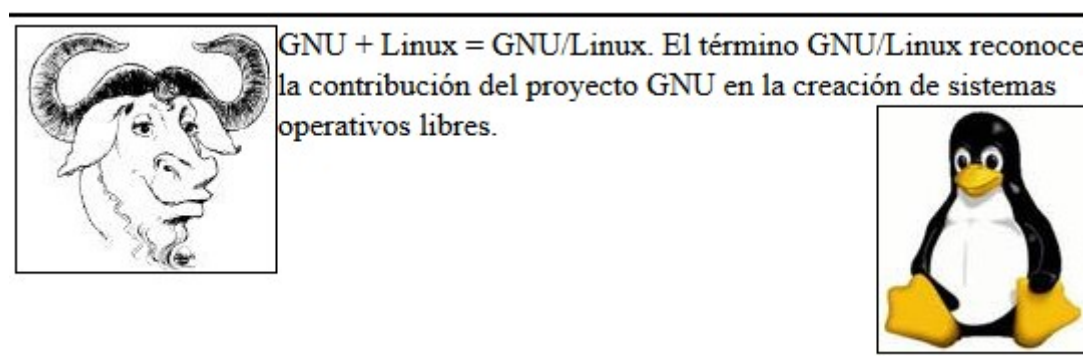
Las dos imágenes están insertadas antes del texto:

```
<p><img class="izquierda" /><img class="derecha" />Texto</p>
```

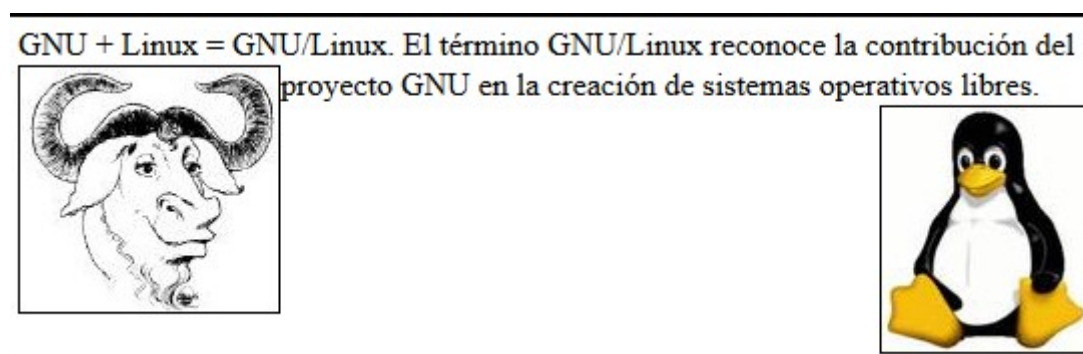
La imagen izquierda está situada antes del texto y la imagen derecha al final del texto:

```
<p><img class="izquierda" />Texto<img class="derecha" /></p>
```



Las dos imágenes están insertadas al final del texto:



```
<p>Texto<img class="izquierda" /><img class="derecha" /></p>
```



1.1.3. La propiedad clear

Al crear una imagen flotante, el navegador sitúa los elementos que se encuentran a continuación de la imagen a su lado mientras haya sitio, aunque no pertenezcan al mismo bloque, como muestra el siguiente ejemplo:

```
img {  
  float: left;}
```

	El logotipo de GNU debe "flotar" a la izquierda y el párrafo debe fluir a su derecha.
	El logotipo de Linux, Tux, debe "flotar" a la izquierda y el párrafo debe fluir a su derecha. Seguramente tanto la imagen como el párrafo están a la derecha del logotipo de GNU.

Para impedir que ocurra esto, es necesario que la flotación de la imagen se interrumpa. La propiedad `clear` hace que un elemento no tenga elementos flotantes a su lado. Los posibles valores de `clear` son:

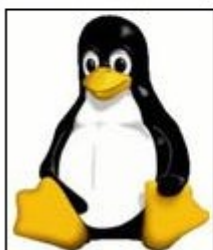
- `left`, que hace que no hayan elementos flotantes a la izquierda,
- `right`, que hace que no hayan elementos flotantes a la derecha,
- `both`, que hace que no hayan elementos flotantes ni a derecha ni a izquierda,
- `none`, que permite que hayan elementos flotantes a derecha y a izquierda (valor por omisión).

Se puede asignar la propiedad `clear` a cualquier elemento. En el ejemplo siguiente se ha asignado a una línea horizontal, de manera que la línea ya no flota a la derecha de la imagen, sino que se muestra a continuación de la imagen:

```
hr {  
  clear: both;  
}  
  
img {  
  float: left;  
}
```



El logotipo de GNU debe "flotar" a la izquierda y el párrafo debe fluir a su derecha.



El logotipo de Linux, Tux, debe "flotar" a la izquierda y el párrafo debe fluir a su derecha. La línea intermedia impide que se monten sobre el párrafo anterior.

1.1.4. Tamaño de los elementos que contienen elementos flotantes

Los elementos flotantes no se tienen en cuenta al calcular el tamaño de los elementos que los contienen. Por ejemplo, si una imagen flotante forma parte de una división con borde, la imagen puede "salirse" del borde, como se ve en siguiente ejemplo:

```
div {  
    border: black 3px solid;  
}  
img {  
    float: left;  
}
```



Este párrafo tiene insertada una imagen flotante.

Este párrafo y el anterior forman parte de una división.

Este párrafo ya está fuera de la división.

Si se quiere que la división incluya la imagen, se puede conseguir de varias maneras:

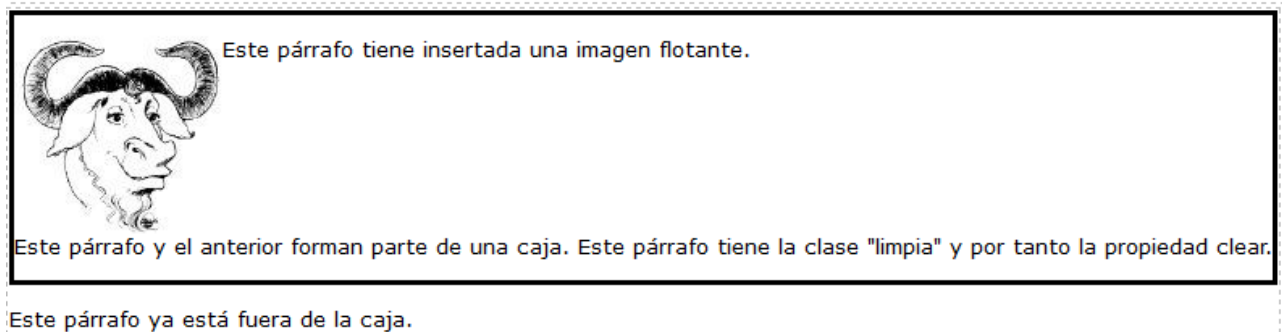
Una primera solución podría ser dar la propiedad `clear: both` al segundo párrafo

```
div {  
    border: black 3px solid;
```

```

}
p.limpia {
    clear: both;
}
img {
    float: left;
}

```



Una segunda solución podría ser insertar un tercer párrafo vacío con la propiedad clear: both (y no ponersela al segundo párrafo). En la primera solución, el segundo párrafo ya no fluye a la derecha de la imagen, mientras que en la segunda sí lo hace.

```

div {
    border: black 3px solid;
}
p.limpia {
    clear: both;
}
img {
    float: left;
}

```

Este párrafo tiene insertada una imagen flotante.



Este párrafo y el anterior forman parte de una división. Además a continuación de este párrafo hay un tercer párrafo vacío con la clase "limpia" y por tanto con la propiedad clear.

Este párrafo ya está fuera de la caja.

Una tercera solución sería dar una altura muy pequeña a la división y establecer la propiedad: overflow: hidden.

```
div {  
    border: black 3px solid;  
    height: 1%;  
    overflow: hidden;  
}  
img {  
    float: left;  
}
```



Este párrafo tiene insertada una imagen flotante.

Este párrafo y el anterior forman parte de una división. La caja tiene una altura y la propiedad overflow: hidden.

Este párrafo ya está fuera de la división.

2. Diseño simplista

Veamos un diseño simplista para comprender el problema de la maquetación con cajas flotantes.

Este es el esquema de lo que queremos hacer:

```
<div id="contenedor">
```

```
<div id="cabecera">
```

```
<div id="contenido">
```

```
<div id="principal">
```

```
<div id="aside">
```

```
<div id="pie">
```

Queremos que la distribución tenga un ancho fijo y que esté centrada.

También queremos que las cajas tengan un color de fondo y que las cajas `<main>` y `<aside>` estén separadas por una línea vertical.

Esta es la estructura HTML:

```
<div id="contenedor">
  <div id="cabecera">
    <h1>Non equidem invideo...</h1>
  </div>
  <div id="contenido">
    <div id="principal">
      <div id="articulo">
        <h2>Prima luce, cum quibus mons aliud
consensu ab eo.</h2>
        <p>Pellentesque habitant morbi tristique
senectus et netus. Excepteur sint obcaecat cupiditat non
proident culpa. Immensae subtilitatis, obscuris et malesuada
fames. Cum sociis natoque penatibus et magnis dis parturient.
Non equidem invideo, miror magis posuere velit aliquet. Cum
ceteris in veneratione tui montes, nascetur mus. Excepteur
sint obcaecat cupiditat non proident culpa. Ambitioni dedisse
scripsisse iudicaretur.</p>
      </div>
      <div id="articulo">
        <h2>Morbi fringilla convallis sapien, id
pulvinar odio volutpat.</h2>
        <p>Nec dubitamus multa iter quae et nos
invenerat. Nec dubitamus multa iter quae et nos invenerat.
Mercedem aut nummos unde unde extricat, amaras. At nos hinc
posthac, sitientis piros Afros. Tu quoque, Brute, fili mi,
nihil timor populi, nihil! Ab illo tempore, ab est sed
immemorabili. Paullum deliquit, ponderibus modulisque suis
ratio utitur. Ab illo tempore, ab est sed immemorabili. Cum
ceteris in veneratione tui montes, nascetur mus. Etiam
```

habebis sem dicantur magna mollis euismod. Prima luce, cum quibus mons aliud consensu ab eo.</p>

</div>

<div id="articulo">

<h2>Ambitioni dedisse scripsisse iudicaretur.</h2>

<p>Contra legem facit qui id facit quod lex prohibet. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Pellentesque habitant morbi tristique senectus et netus. Quam temere in vitiis, legem sancimus haerentia. Donec sed odio operae, eu vulputate felis rhoncus. Paullum deliquit, ponderibus modulisque suis ratio utitur. Excepteur sint obcaecat cupiditat non proident culpa. Immensae subtilitatis, obscuris et malesuada fames. Non equidem invideo, miror magis posuere velit aliquet. Quae vero auctorem tractata ab fiducia dicuntur.</p>

</div>

</div>

<div id="aside">

<h3>Unam incolunt Belgae</h3>

<p>Prima luce, cum quibus mons aliud consensu ab eo. A communi observantia non est recedendum.</p>

<h3>Aliam Aquitani, tertiam</h3>

<p>Curabitur blandit tempus ardua ridiculus sed magna. Immensae subtilitatis, obscuris et malesuada fames. Paullum deliquit, ponderibus modulisque suis ratio utitur.</p>

</div>

</div>

<div id="pie">

<p>Etiam habebis sem dicantur magna mollis euismod. Prima luce, cum quibus mons aliud consensu ab eo.</p>

</div>


```
</Div>
```

Y estos son los selectores aplicados:

```
#contenedor {  
    width: 960px;  
    margin: 0 auto;  
}  
#principal {  
    width: 760px;  
    float: left;  
}  
#aside {  
    width: 200px;  
    float: left;  
}
```

¿Cuál es resultado?:

No está para nada perfecta.

El pie de página ocupa el espacio disponible y está debajo de la caja <aside>. Hay que prohibirle cualquier elemento flotante adyacente.

```
{  
    clear: both;  
}
```

Primer problema resuelto:

¿Cuál es resultado?:

Ahora especificaremos los colores de fondo para el encabezado y el pie:

```
#cabecera, #pie {  
    background-color: #ccc;
```

```
}
```

¿Cuál es resultado?:

A continuación, aplicaremos un color al fondo en la parte central (principal) y en la columna lateral derecha (<aside>):

```
#principal {  
    background-color: #ededed;  
}  
#aside {  
    background-color: #a4a4a4;  
}
```

¿Cuál es resultado?

El problema es que mientras lo hacemos no sabemos (¡y nunca lo sabremos!) cuánto contenido habrá en estos dos elementos, no conoceremos la altura de los dos elementos y no podremos tener colores en los fondos a la misma altura.

En nuestro esquema, también tenemos un borde que separa las dos cajas.

Aplicaremos un borde a la derecha de la primera caja <main>. Pero hay que modificar el ancho (width) de la primera caja para que aparezca correctamente en pantalla. Hay que quitar 1 píxel de ancho para tener en cuenta el borde de 1 píxel.

```
#principal{  
    width: 759px;  
    float: left;  
    border-right: 1px solid #000;  
}
```

¿Cuál es resultado?

Tenemos el mismo problema con el borde, que solo se aplica en una caja.

Para arreglarlo, la solución es "sencilla", pero costará su tiempo y recursos en caso de producirse algún cambio: utilizar la técnica de las "columnas falsas". Crearemos una imagen que será el color de fondo de las dos cajas y se la aplicaremos a la parte inferior de la caja <div id="contenido">.

Creamos una imagen de 1 píxel en altura, con un ancho de 960 píxeles y con los colores utilizados. La llamaremos **linea-fondo.png** y la puede descargar.

A continuación, borramos el borde de la derecha de la caja <principal> y restauramos el ancho a 760 píxeles:

```
#principal{
    width: 760px;
    float: left;
}
```

Ahora aplicamos la imagen al fondo de la caja <div id="contenido">:

```
#contenido {
    background-image: url(linea-fondo.png);
    background-repeat: repeat-y;
    overflow: auto;
}
```

¿Cuál es resultado?

A partir de ahora, independientemente del contenido de las cajas <main> y <aside>, el color de fondo y el borde estarán equilibrados entre las dos "columnas" de este diseño:

Ahora nos gustaría tener un poco más espacio en las cajas <main> y <aside>. Hay que aplicar la propiedad padding y modificar consecuentemente el ancho para tener un total de 760 píxeles y 200 píxeles en el área de visualización.

```
#principal {
    width: 740px;
    padding: 10px;
    float: left;
}
#aside {
    width: 180px;
    padding: 10px;
```

```
float: left;
}
```

¿Cuál es resultado?

Terminaremos la maquetación aplicando márgenes a 0 para los elementos en el encabezado y en el pie de página:

```
header h1 {
    margin: 0;
}
#pie p {
    margin: 0;
}
```

¿Cuál es resultado?

2.1. Conclusión sobre diseñar con elementos flotantes

Como acabamos de ver, diseñar con elementos flotantes puede causar ciertos "problemas".

- Problema con los colores en los fondos de las "columnas".
- Problema con los bordes de separación de las columnas.
- Utilizar muchas prohibiciones de flotación con un diseño más elaborado.
- Problema a la hora de calcular el ancho si queremos cambiar una propiedad de las cajas (margen, borde, relleno y contenido).
- Con la imagen de las "columnas falsas", habrá problemas si cambian las dimensiones, ya que será necesario rehacer la imagen.
- Problema con el tiempo de carga de las imágenes de fondo si son muchas y no están optimizadas.

Esta técnica de diseño con elementos flotantes, que durante mucho tiempo fue la única que tuvimos a mano, tiene muchas limitaciones de construcción y mantenimiento. Aunque se siga utilizando mucho, en el capítulo siguiente veremos otra técnica, mucho más flexible y con mejores resultados.

3. Diseño con tablas

En el anterior punto hemos diseñado con las técnicas de las **cajas flotantes** y de las cajas en posición relativa y absoluta y evaluamos los inconvenientes resultantes.

Ahora veremos una de las técnicas más populares a día de hoy: **¡las tablas!** No, no hemos vuelto 20 años atrás cuando las webs se diseñaban con tablas HTML. En este caso vamos a diseñar tablas con la propiedad `display: table`.

3.1. La propiedad display

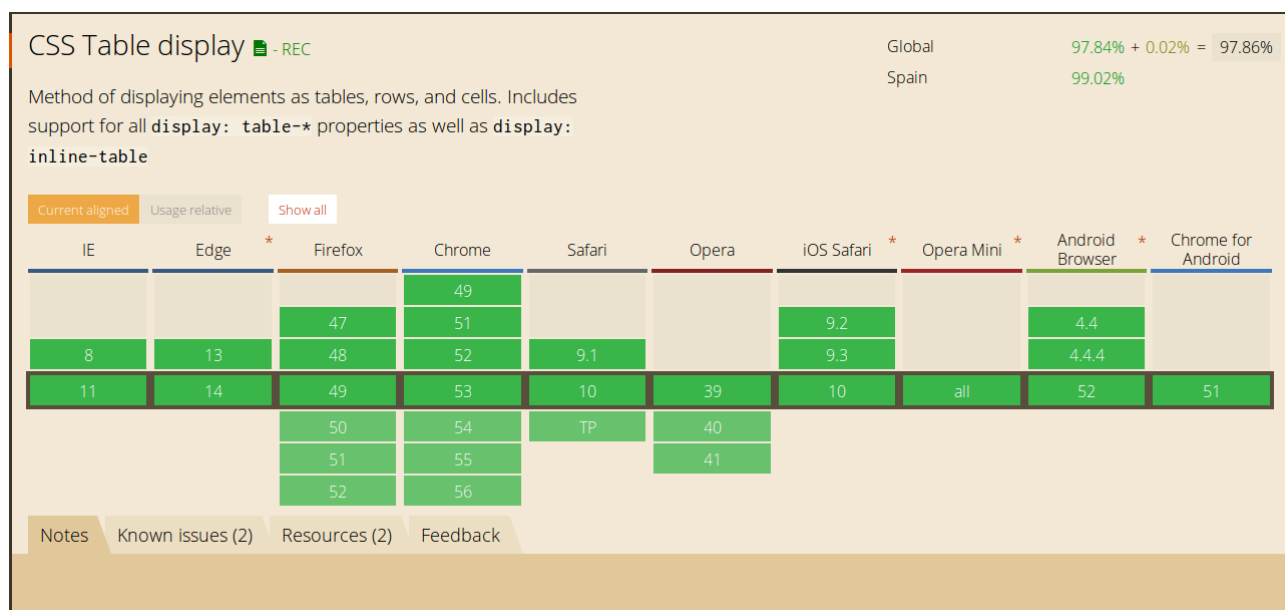
La propiedad `display` forma parte de la norma del W3C de las CSS 2.1:

<http://www.w3.org/TR/CSS21/visuren.html#propdef-display>

Con respecto a la compatibilidad de los navegadores, solo las versiones 6 y 7 de Internet Explorer no reconocen los valores tabulares de la propiedad `display`.

Podemos comprobar la compatibilidad con los navegadores en el sitio **Can I use**:

<http://caniuse.com/#feat=css-table>



Para el diseño con tablas, utilizaremos varios valores de la propiedad `display`.

Estos son los valores:

- **table**, muestra el elemento que utiliza esta propiedad en forma de tabla. El bloque ocupa solo el ancho del contenido.

- **Inline-table**, sirve para diseñar una tabla de tipo en línea. El elemento se incluirá en la línea donde esté ubicado.
- **table-row**, se consigue un diseño de tipo fila. Hace que cualquier elemento se muestre como si fuera una fila de una tabla.
- **table-row-group**, reagrupa una o varias filas
- **table-header-group**, muestra los elementos correspondientes antes de las filas de la tabla y después de la leyenda, si la hay. De este modo se pueden crear encabezados para la tabla.
- **table-footer-group**, muestra los elementos correspondientes después de las filas de las tablas y después de la leyenda, si la hay. De este modo se pueden crear pies de página para la tabla.
- **table-column**, muestra una columna en la tabla.
- **table-column-group**, reagrupa una o varias columnas de la tabla
- **table-cell**, crea y muestra una celda de una tabla.
- **table-caption**. crea y muestra la leyenda de la tabla. Aparece, por defecto, en la parte superior de la tabla, pero la posición se puede modificar con la propiedad `caption-side`

4. Diseño simple con una tabla

Veamos cómo diseñar una tabla simple, con dos celdas. Esta es la estructura HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Mi diseño</title>
  <meta charset="UTF-8" />
```

```

<style>
  #tabla {
    display: table;
  }
  .celda1, .celda2 {
    display: table-cell;
  }
  .celda1 {
    background-color: lightgreen;
  }
  .celda2 {
    background-color: lightblue;
  }
</style>
</head>
<body>
<div id="tabla">
  <div class="celda1">
    <p>Prima luce, cum quibus mons aliud consensu ab
eo.</p>
  </div>
  <div class="celda2">
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit.</p>
  </div>
</div>
</body>

```

```
</html>
```

La primera caja `<div>` tiene el identificador `#tabla`. El selector CSS `#tabla` hará que aparezca en forma de tabla: `display: table;`

En esta caja tenemos que crear otras dos cajas `<div>` mediante dos clases: `.celda1` y `.celda2`. Los dos selectores utilizan una diseño en forma de celda: `display: table-cell` y con colores en el fondo.

¿cómo se verá el ejemplo anterior?

4.1. Ancho en pantalla de la tabla

Las celdas muestran todo su contenido. En este ejemplo no hay mucho texto, aparece al completo y las celdas no son muy anchas. Ahora veamos qué sucede si aumentamos la cantidad de texto.

Para el primer párrafo añade: *“Prima luce, cum quibus mons aliud consensu ab eo. Tu quoque, Brute, fili mi, nihil timor populi, nihil! Quae vero auctorem tractata ab fiducia dicuntur. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Plura mihi bona sunt, inclinet, amari petere vellent”*

Y en le segundo añade: *“Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ut enim ad minim veniam, quis nostrud exercitation. Quam diu etiam furor iste tuus nos eludet? Me non paenitet nullum festiviorem excogitasse ad hoc. Morbi odio eros, volutpat ut pharetra vitae, lobortis sed nibh. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Ut enim ad minim veniam, quis nostrud exercitation. Quisque ut dolor gravida, placerat libero vel, euismod. Plura mihi bona sunt, inclinet, amari petere vellent. Ut enim ad minim veniam, quis nostrud exercitation. Sed haec quis possit intrepidus aestimare tellus”*

¿Cuál es el resultado?

La tabla ocupa todo el ancho disponible en la ventana del navegador y las celdas también ocupan todo el espacio de su elemento padre: la tabla, mostrando así todo su contenido.

Pero lo más importante, como puede comprobar, la altura por defecto de las celdas es equilibrada, independientemente del ancho de la ventana del navegador. Es una particularidad de la vista **`table-cell`**.

4.2. Elementos anónimos

Veamos otra característica de la vista tabular. En el ejemplo anterior no hemos creado la fila, `table-row`. En ese caso, los navegadores crean automáticamente los elementos que faltan para obtener una estructura completa. Ocurriría lo mismo si no hubiéramos definido la tabla con `table`, los navegadores la habrían creado. El hecho de eliminar la caja `<div id="tabla">` no repercute ni en la validez de la sintaxis ni en la visualización. **Comprueballo**

Sin embargo, te aconsejo que introduzca un elemento con un `display: table`, siempre es mejor tener una estructura mínima.

4.3. Otros elementos para las tablas

4.3.1. Párrafos en las celdas

En los ejemplos anteriores utilizamos cajas `<div>` para crear la tabla y las celdas. Para simplificar la estructura, puede utilizar cualquier otro elemento para crear las celdas, un párrafo `<p>`, por ejemplo.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Mi diseño</title>
  <meta charset="UTF-8" />
  <style>
    #tabla {
      display: table;
    }
    .celda1, .celda2 {
      display: table-cell;
    }
    .celda1 {
      background-color: lightgreen;
    }
    .celda2 {
```

```

        background-color: lightblue;
    }
</style>
</head>
<body>
<div id="tabla">
    <p class="celda1">Prima luce...</p>
    <p class="celda2">Lorem ipsum dolor...</p>
</div>
</body>
</html>

```

¿Y el resultado es?

El resultado puede verse de inmediato: los párrafos `<p>` **no tienen el relleno superior e inferior por defecto**. El texto está "pegado" a los bordes del bloque. Para un diseño simplificado puede ser bastante útil.

4.3.2. Listas en tablas

Si quiere, no tiene que utilizar la caja `<div>`, en su lugar, puede utilizar una lista ``. El elemento `` aparece como una tabla y los elementos de la lista `` representan las celdas de la tabla.

Este es el código utilizado:

```

<!DOCTYPE html>
<html lang="es">
<head>
    <title>Mi diseño</title>
    <meta charset="UTF-8" />
    <style>
        #tabla {
            display: table;
        }
    </style>
</head>
<body>
    <div id="tabla">
        <ul>
            <li>Prima luce...</li>
            <li>Lorem ipsum dolor...</li>
        </ul>
    </div>
</body>
</html>

```

```

        .celda1, .celda2 {
            display: table-cell;
        }
        .celda1 {
            background-color: lightgreen;
        }
        .celda2 {
            background-color: lightblue;
        }
    </style>
</head>
<body>
<ul id="tabla">
    <li class="celda1">Prima luce...</li>
    <li class="celda2">Lorem ipsum dolor...</li>
</ul>
</body>
</html>

```

¿Y obtenemos ?

5. Diseño de las filas

Ahora vamos a crear una tabla con filas con la propiedad **display: table-row**. Este es el código utilizado:

```

<!DOCTYPE html>
<html lang="es">
<head>
    <title>Mi diseño</title>
    <meta charset="UTF-8" />
    <style>

```

```

#tabla {
    display: table;
}
.fila1, .fila2 {
    display: table-row;
}
.fila1 {
    background-color: lightgreen;
}
.fila2 {
    background-color: lightblue;
}
</style>
</head>
<body>
<div id="tabla">
    <div class="fila1">
        <p>Prima luce...</p>
    </div>
    <div class="fila2">
        <p>Lorem ipsum dolor sit amet...</p>
    </div>
</div>
</body>
</html>

```

¿Y obtenemos ?

Las dos filas ocupan todo el espacio disponible en el ancho de la tabla padre, que en este ejemplo ocupa todo el ancho disponible en la ventana del navegador.

5.1. Establecer el ancho de las celdas

5.1.1. Anchos fijos

En los ejemplos anteriores no habíamos establecido ni el ancho de la tabla ni el de las celdas. Los elementos ocupan el espacio disponible en sus elementos padre.

Ahora ya podemos establecer el ancho de la tabla y de las celdas.

Si solo establece el ancho de la tabla (`#tabla {display: table; width: 800px;}`), las celdas utilizarán un ancho calculado para mostrar el contenido de **forma distribuida**.

¿Y se verá?

Si establece un ancho para la celda mayor o menor que las de la tabla, se utilizará el ancho de la tabla.

Por el contrario, si no establece el ancho de la tabla y sí el de las celdas (`width: 500px;` para la primera y `width: 300px;` para la segunda), evidentemente, será el ancho que se va a utilizar.

¿Y se verá?

Si lo desea, puede establecer un único ancho fijo. En este ejemplo, se ha establecido un ancho para la primera celda de 500 píxeles (`width: 500px;`), para la segunda no se ha hecho. En ese caso, la primera celda tendrá siempre un ancho de 500 píxeles y la segunda ocupa el resto del espacio disponible en su elemento padre.

5.1.2. Anchos en porcentaje

Otra forma de establecer el ancho de las tablas y de las celdas es utilizar los porcentajes. Al cambiar el tamaño de la ventana del navegador, cambiarán todas las dimensiones.

Estos son los estilos aplicados a la tabla:

```
#tabla {  
    display: table;  
    width: 60%;  
}
```

Y estos a las celdas:

```
.celda1 {  
    width: 70%;  
    background-color: lightgreen;  
}  
  
.celda2 {  
    width: 30%;  
    background-color: lightblue;  
}
```

5.2. Diseño con una tabla más estructurada

Veamos ahora una tabla más estructurada con un encabezado y un pie de página. Le añadiremos el encabezado con la propiedad `display: table-header-group`; y el pie con `display: table-footer-group`. Este es el código:

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
    <title>Mi diseño</title>  
    <meta charset="UTF-8" />  
    <style>  
        #tabla {  
            display: table;  
        }  
        #encabezado {  
            display: table-header-group;  
            background-color: lightyellow;  
        }  
        #pie {  
            display: table-footer-group;
```

```

        background-color: lightyellow;
    }
    .celda1, .celda2 {
        display: table-cell;
    }
    .celda1 {
        width: 500px;
        background-color: lightgreen;
    }
    .celda2 {
        width: 300px;
        background-color: lightblue;
    }
</style>
</head>
<body>
<div id="tabla">
    <div class="fila">
        <div class="celda1">
            <p>Prima luce...</p>
        </div>
        <div class="celda2">
            <p>Lorem ipsum...</p>
        </div>
    </div>
    <div id="encabezado">
        <p>Encabezado. Immensae subtilitatis...</p>
    </div>

```

```
<div id="pie">
  <p>Pie de página. Quis aute iure...</p>
</div>
</div>
</body>
</html>
```

El orden de los elementos no altera la visualización. En este ejemplo hemos declarado el encabezado y el pie de página y después las celdas, pero quizá prefiera una estructura más semántica y fácil de asimilar: encabezado, celdas y pie.

¿Qué obtenemos?

Como puede ver, solo hemos establecido el ancho de dos celdas, es más que suficiente para que se vean correctamente. Una vez más, le corresponde decidir si prefiere imponer todos los anchos de los elementos de la tabla.

5.3. Otras propiedades de diseño

Veamos algunas propiedades específicas para el diseño en forma de tabla que influyen directamente en el resultado en el navegador.

5.3.1. Propiedad **table-layout**

La propiedad **table-layout** determina cómo se va a calcular el ancho de las celdas de una tabla que tenga un ancho fijo. Esta propiedad acepta dos valores:

- **table-layout: auto:** es el valor por defecto. En este caso se calcula el ancho de la celda para que la que tenga más contenido sirva como base para el cálculo de la altura de las celdas, para que sea igual en todas.
- **table-layout: fixed:** en este caso el ancho de las celdas es igual y su suma es igual al ancho fijo de la tabla. La altura de las celdas la determina la que tenga más texto.

Estas son las propiedades de la tabla del primer ejemplo:

```
#tabla {
  display: table;
  width: 800px;
```



```
table-layout: auto;
}
```

Y obtenemos esto:

Prima luce, cum quibus mons aliud consensu ab eo. Tu quoque, Brute, fili mi, nihil timor populi, nihil! Quae vero auctorem tractata ab fiducia dicuntur. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Plura mihi bona sunt, inclinet, amari petere vellent.	Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ut enim ad minim veniam, quis nostrud exercitation. Quam diu etiam furor iste tuus nos eludet? Me non paenitet nullum festiviorem excogitasse ad hoc. Morbi odio eros, volutpat ut pharetra vitae, lobortis sed nibh. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Ut enim ad minim veniam, quis nostrud exercitation. Quisque ut dolor gravida, placerat libero vel, euismod. Plura mihi bona sunt, inclinet, amari petere vellent. Ut enim ad minim veniam, quis nostrud exercitation. Sed haec quis possit intrepidus aestimare tellus.
--	---

Y estas las del segundo ejemplo:

```
#tabla {
  display: table;
  width: 800px;
  table-layout: fixed;
}
```

Y obtenemos esto:

Prima luce, cum quibus mons aliud consensu ab eo. Tu quoque, Brute, fili mi, nihil timor populi, nihil! Quae vero auctorem tractata ab fiducia dicuntur. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Plura mihi bona sunt, inclinet, amari petere vellent.	Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ut enim ad minim veniam, quis nostrud exercitation. Quam diu etiam furor iste tuus nos eludet? Me non paenitet nullum festiviorem excogitasse ad hoc. Morbi odio eros, volutpat ut pharetra vitae, lobortis sed nibh. Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Ut enim ad minim veniam, quis nostrud exercitation. Quisque ut dolor gravida, placerat libero vel, euismod. Plura mihi bona sunt, inclinet, amari petere vellent. Ut enim ad minim veniam, quis nostrud exercitation. Sed haec quis possit intrepidus aestimare tellus.
--	---

5.3.2. Propiedad border-collapse

Tiene la posibilidad de aplicar bordes a la tabla o a las celdas. Si todas las celdas de la tabla tienen bordes, no es necesario aplicárselo a la tabla. Por defecto, los bordes de las celdas se aplican a cada celda individualmente. Eso significa que los bordes de cada celda son visibles.

Veamos el código de este ejemplo, una tabla con dos filas y dos celdas en cada fila.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <Title>Mi página web</title>
  <meta charset="UTF-8" />
  <style>
    #tabla {
      display: table;
      width: 600px;
    }
    .fila {
      display: table-row;
    }
    .celda {
      display: table-cell;
      width: 300px;
    }
  </style>
</head>
<body>
<div id="tabla">
  <div class="fila">
    <div class="celda">
      <p>Prima luce...</p>
    </div>
    <div class="celda">
      <p>Lorem ipsum dolor sit amet...</p>
    </div>
  </div>
  <div class="fila">
    <div class="celda">
      <p>Prima luce...</p>
    </div>
    <div class="celda">
      <p>Lorem ipsum dolor sit amet...</p>
    </div>
  </div>
</div>
```

```

    </div>
</div>
<div class="fila">
    <div class="celda">
        <p>Quis aute iure reprehenderit...</p>
    </div>
    <div class="celda">
        <p>Gallia est omnis divisa in partes tres...</p>
    </div>
</div>
</div>
</body>
</html>

```

Añade a la clase `.celda` borde simple y a cada uno de los párrafos más texto

¿Cual es el resultado?

Como puede comprobar, pueden verse los bordes alrededor de las celdas. Es el comportamiento normal, por defecto.

Utilice la propiedad `border-collapse` para obtener bordes superpuestos, que se fusionen. Los dos valores posibles son:

- **border-collapse: separate:** los bordes se separan, es el valor por defecto.
- **border-collapse: collapse:** los bordes se combinan.

Si al anterior ejemplo se le aplicara la opción **collapse**. ¿Cómo se vería?

Si al anterior ejemplo se le aplicara la opción **separate**. ¿Cómo se vería?

Evidentemente, puede aplicar un borde solo a un lado de la celda para tener así una separación única entre dos columnas. Independientemente de la cantidad de contenido que haya en las dos celdas, en las dos columnas, el borde se ajustará siempre a la que tenga más contenido.

5.3.3. Propiedad border-spacing

Si no combina los bordes de las celdas (border-collapse: separate), puede especificar el espacio entre los bordes de las celdas. Esta propiedad se aplica a la tabla de esta manera:

Si al anterior ejemplo se le aplicara un espacio de 5 píxeles entre los bordes de las celdas ¿Cómo se vería?

Si se introduce un único valor, el espacio se aplica horizontal y verticalmente. Si indica dos valores, el primero se aplicará horizontal y el segundo verticalmente:

Si al anterior ejemplo se le aplicara un espacio de 5 píxeles horizontal y 20 píxeles vertical. ¿Cómo se vería?

5.3.4. Propiedad empty-cells

La propiedad empty-cells define el comportamiento de las celdas vacías con bordes, en el contexto de una tabla con bordes separados. El valor hide oculta las celdas vacías y show las muestra.

Veamos el primer ejemplo.

Estas son las propiedades de la tabla:

```
#tabla {  
  display: table;  
  width: 600px;  
  border-spacing: 10px;  
}
```

Estas son las propiedades aplicadas a las celdas:

```
.celda {  
  display: table-cell;  
  width: 300px;  
  border: 1px solid #333;  
  empty-cells: hide;  
}
```

La última celda de la tabla carece de contenido: `<div class="celda"></div>`.

¿cómo se verá?

Si utilizamos el valor `empty-cells: show;`, ¿que pasan con los bordes?:

5.3.5. Propiedad caption-side

El elemento `table-caption` añade una leyenda en la parte superior de la tabla.

El código de este ejemplo es el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <Title>Mi página web</title>
  <meta charset="UTF-8" />
  <style>
    #tabla {
      display: table;
      width: 600px;
      border-collapse: collapse;
    }
    .fila {
      display: table-row;
    }
    .celda {
      display: table-cell;
      width: 300px;
      border: 1px solid #333;
    }
    #leyenda {
```

```

        display: table-caption;
    }
</style>
</head>
<body>
<div id="tabla">
    <div class="fila">
        <div class="celda">
            <p>Prima luce...</p>
        </div>
        <div class="celda">
            <p>Lorem ipsum dolor sit amet...</p>
        </div>
    </div>
    <div class="fila">
        <div class="celda">
            <p>Quis aute iure reprehenderit...</p>
        </div>
        <div class="celda">
            <p>Quisque placerat facilisis...</p>
        </div>
    </div>
    <p id="leyenda">Quis aute iure reprehenderit in voluptate
    velit esse.</p>
</div>
</body>
</html>

```

¿Cómo se verá?

La propiedad `caption-side` establece la ubicación de la leyenda. Puede utilizar los valores `top`, `right`, `bottom` y `left`.

¿Cómo podemos colocar la leyenda debajo de la tabla?

5.4. Alineación vertical

La alineación vertical en las cajas `<div>` siempre ha supuesto un "problema" para los maquetadores. La propiedad `vertical-align` definida en las celdas es nativa en el diseño con tablas. Sirve para centrar el contenido en una celda.

Veamos un ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <Title>Mi página web</title>
  <meta charset="UTF-8" />
  <style>
    #tabla {
      display: table;
    }
    .celda1, .celda2 {
      display: table-cell;
      height: 200px;
      vertical-align: middle;
    }
    .celda1 {
      background-color: lightgreen;
    }
    .celda2 {
      background-color: lightblue;
```

```

    }
  </style>
</head>
<body>
<div id="tabla">
  <div class="celda1">
    <p>Prima luce, cum quibus mons aliud consensu ab
eo.</p>
  </div>
  <div class="celda2">
    <p>Lorem ipsum dolor sit amet, consectetur adipisici
elit.</p>
  </div>
</div>
</body>
</html>

```

¿Cómo se verá?

6. Conclusión

El diseño con tablas con la propiedad `display` y el resto de valores son una verdadera ventaja para los diseñadores.

Veamos algunas:

- La altura de las celdas es siempre la misma. Dicho de forma más técnica, la altura de los elementos hermanos es idéntica. Con otros métodos, la altura de las celdas depende de la cantidad de contenido que tengan. Con este tipo de tablas no nos tenemos que preocupar por este aspecto.
- La propiedad `vertical-align` gestiona de forma nativa la alineación vertical. Una vez más, es una buena manera de aplicar un centrado vertical a un elemento en una celda.

- No hay problema de diferentes alturas entre celdas con colores de fondo. Como la altura de las celdas es igual, no suponen ningún problema para que los colores del fondo se vean adecuadamente.
- No hay problemas a la hora de generar bordes en un solo lado de las celdas con contenidos distintos, ya que la altura de estas será siempre igual. De nuevo una ventaja para los maquettadores.
- Puede utilizar unidades distintas, fijas en píxeles y relativas en porcentaje, para gestionar el ancho y el alto de las celdas.
- Se acabaron los problemas de flotación de los elementos adyacentes. Todos los elementos de las tablas están en el flujo "normal".
- Código limpio, semántico, ligero y no verbal.