**Css**

**Display**

display es la propiedad más importante para controlar estructuras. Cada elemento tiene un valor de display por defecto dependiendo de qué tipo de elemento sea. El valor por defecto para la mayoría de los elementos es usualmente block (de bloque) o inline (en línea). Un elemento que es block es comunmente llamado elemento block-level. Un elemento inline siempre es llamado elemento inline.

**block**



**inline**

El span es el elemento inline estándar. Un elemento inline puede contener algo de texto dentro de un párrafo <span> como esto </span> sin interrumpir el flujo del párrafo. El elemento a es el elemento inline más común, ya que se usa para links.

**none**

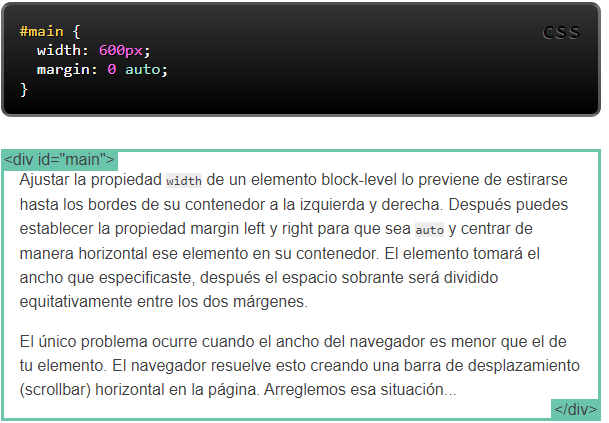
Otro valor común de display es none. Algunos elementos especializados como script usan este por defecto. Es comúnmente usado en JavaScript para ocultar o mostrar elementos sin eliminarlos ni recrearlos.

Esto es diferente de visibility. Usar display: none no dejará espacio donde el elemento se encontraba, pero visibility: hidden; dejará un espacio vacío.

**otros valores de display**

Hay bastantes valores exóticos de display, como list-item y table. [Aquí hay una lista exhaustiva](https://developer.mozilla.org/es/docs/CSS/display). Discutiremos inline-block y flex después.

|  |
| --- |
| / \* Valores <display-outside> \* /  display: block;  display: inline;  display: run-in;  / \* Valores <display-inside> \* /  display: flow;  display: flow-root;  display: table;  display: flex;  display: grid;  display: ruby;  display: subgrid;  / \* Valores <display-outside> más valores <display-inside> \* /  display: block flow;  display: inline table;  display: flex run-in;  / \* Valores <display-listitem> \* /  display: list-item;  display: list-item block;  display: list-item inline;  display: list-item flow;  display: list-item flow-root;  display: list-item block flow;  display: list-item block flow-root;  display: flow list-item block;  / \* Valores <display-internal> \* /  display: table-row-group;  display: table-header-group;  display: table-footer-group;  display: table-row;  display: table-cell;  display: table-column-group;  display: table-column;  display: table-caption;  display: ruby-base;  display: ruby-text;  display: ruby-base-container;  display: ruby-text-container;  / \* Valores <display-box> \* /  display: contents;  display: none;  / \* Valores <display-legacy> \* /  display: inline-block;  display: inline-table;  display: inline-flex;  display: inline-grid;  / \* Valores globales \* /  display: heredar;  display: initial;  display: unset; |

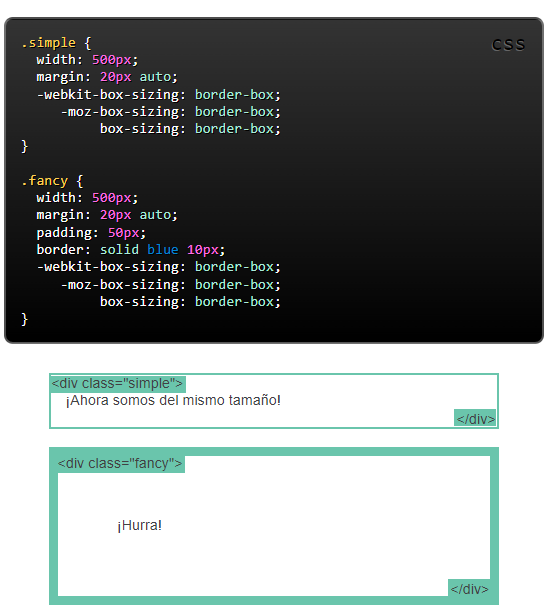
****

**Border**

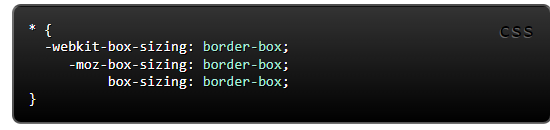
Ya que estamos hablando de la propiedad width, deberíamos hablar del *box model* (modelo de caja). Cuando ajustas la propiedad width de un elemento, este puede parecer mas grande de lo que es: la propiedad border (borde) del elemento y padding (relleno) estirarán el elemento más allá del ancho especificado. Mira el siguiente ejemplo, donde dos elementos con el mismo valor width terminan teniendo diferente tamaño al final.



Durante generaciones, la gente se ha dado cuenta de que las matemáticas no son divertidas, así que una nueva propiedad CSS llamada box-sizing fue creada. Cuando ajustas un elemento con box-sizing: border-box;, el padding y border de ese elemento no incrementan su ancho. Aquí está el mismo ejemplo de la página anterior, pero usando box-sizing: border-box; en los dos elementos:



Ya que esto es mucho mejor, algunos autores quieren que todos los elementos de sus páginas trabajen de la misma manera. Estos autores ponen lo siguiente en sus páginas:



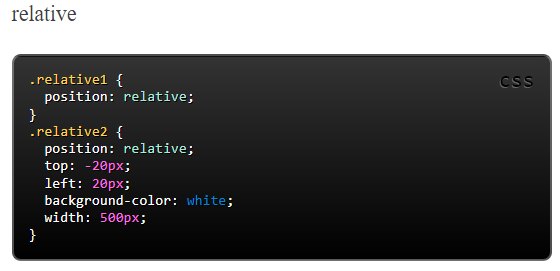
Esto asegura que el tamaño de todos los elementos siempre será modificado de la manera más intuitiva.

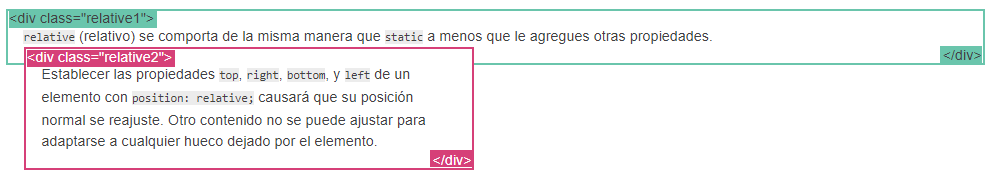
Ya que box-sizing es bastante nuevo, te recomiendo usar los prefijos -webkit- y -moz- por ahora, como yo los uso en el ejemplo. Esta técnica permite funciones experimentales en navegadores específicos. También, ten en cuenta que es compatible con [IE8+](http://caniuse.com/#search=box-sizing).

Para poder tener estructuras mas complejas, tenemos que discutir la propiedad position (posición). Esta propiedad tiene un montón de posibles valores, sus nombres no tienen sentido y son casi imposibles de memorizar. Veámoslos uno a uno, pero antes deberías guardar esta página en tus marcadores para futuras referencias.

**Static**







**Fixed**

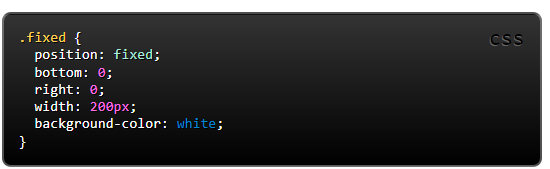
<div class="fixed">

Hola! No me prestes atención todavía

</div>

Un elemento fixed (fijo) se posiciona a la ventana del navegador de manera relativa, lo que significa que se mantendrá en el mismo lugar incluso después de hacer scroll en la página. Al igual que con relative, las propiedades top, right, bottom, y left también son usadas.

Estoy seguro que ya notaste el elemento fijo en la parte de abajo a la derecha de la página. Te doy permiso de que lo veas ahora. Aquí está el CSS que lo posiciona donde está:

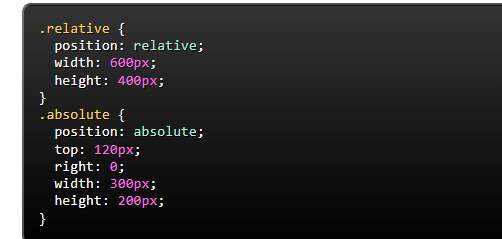


Un elemento con valor fixed no deja espacio en el lugar de la página donde estaba ubicado normalmente.

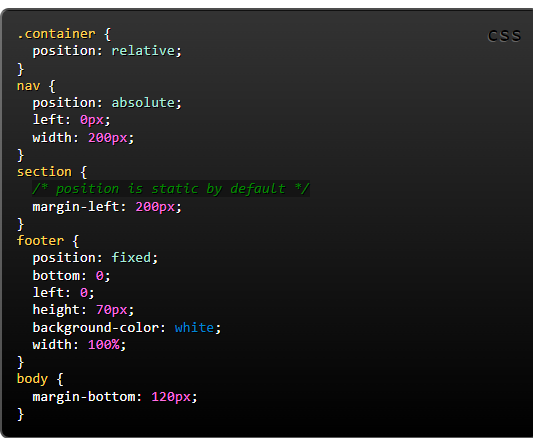
Los buscadores móviles, sorprendentemente, no tienen muy buen soporte para el valor fixed. [Aprende más al respecto aquí](http://bradfrostweb.com/blog/mobile/fixed-position/).

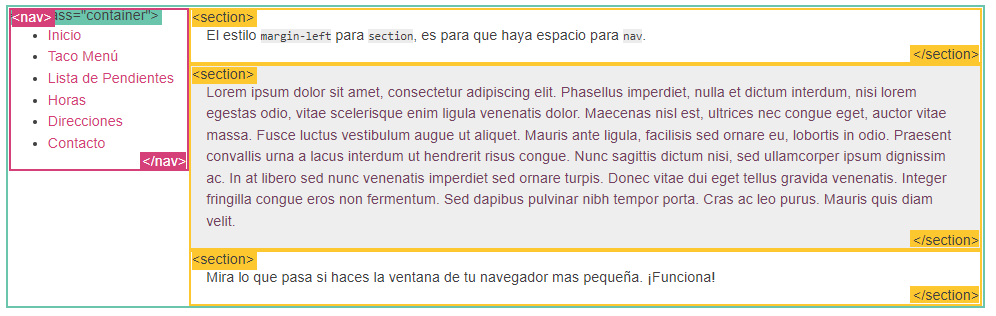
**absolute**

absolute (absoluto) es el valor más mañoso. absolute se comporta como fixed pero es relativo a *su ancestro posicionado más cercano* en lugar de ser relativo a la ventana del navegador. Si un elemento con position: absolute; no tiene ancestros posicionados, usará el elemento body del documento, y se seguirá moviendo al hacer scroll en la página. Recuerda, un elemento "posicionado" es aquel cuyo valor es cualquiera excepto static.



Todo esto de position tal vez tenga un poco más de sentido con un ejemplo práctico. Abajo está la estructura real de una página.





Este ejemplo funciona porque el contenedor es más alto que el nav. Si no lo fuera, el nav se saldría de su contenedor. En las páginas que vienen a continuación discutiremos otras técnicas para estructurar y hablaremos de sus pros y sus contras.

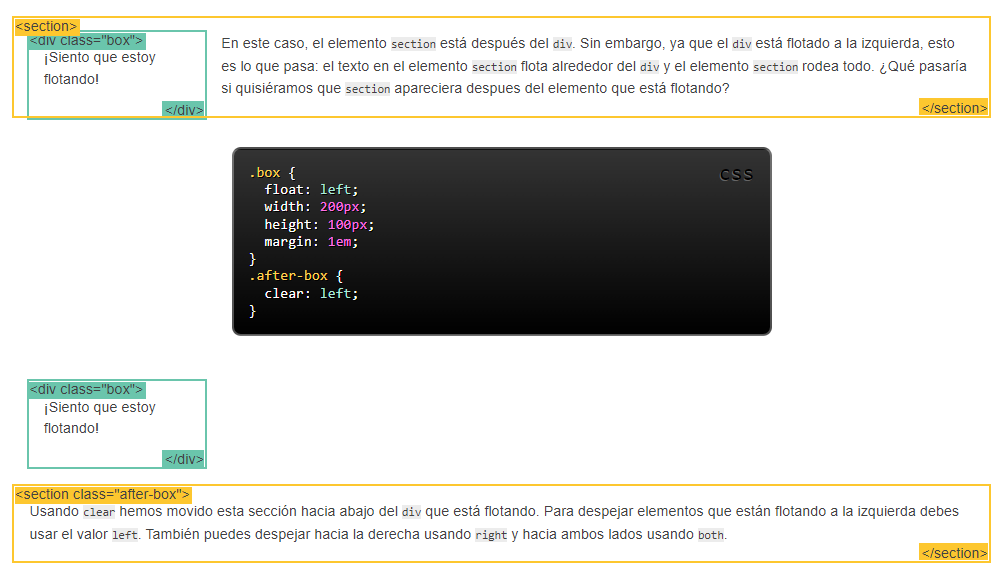
**Float**

Otra propiedad CSS para estructurar es float (flotar). Float se usa para envolver imágenes con texto, como aquí:



**Clear**

La propiedad clear (despejar) es importante para controlar el comportamiento de los floats. Compara estos dos ejemplos:



Es muy común hacer estructuras completas usando float. Aquí está la misma estructura que usamos con position hace rato, pero usando float.



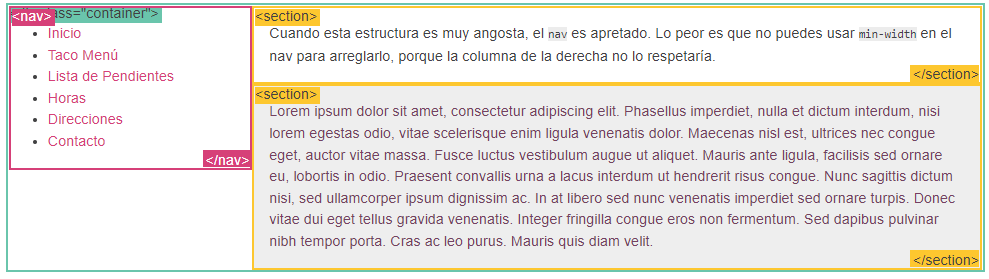
**Porcentaje**

Porcentaje es una unidad de medida relativa al bloque contenedor. Es muy útil para imágenes: aquí tenemos una imagen que ocupa el 50% de su contenedor en todo momento. ¡Haz la página mas pequeña para que veas lo que pasa!

******

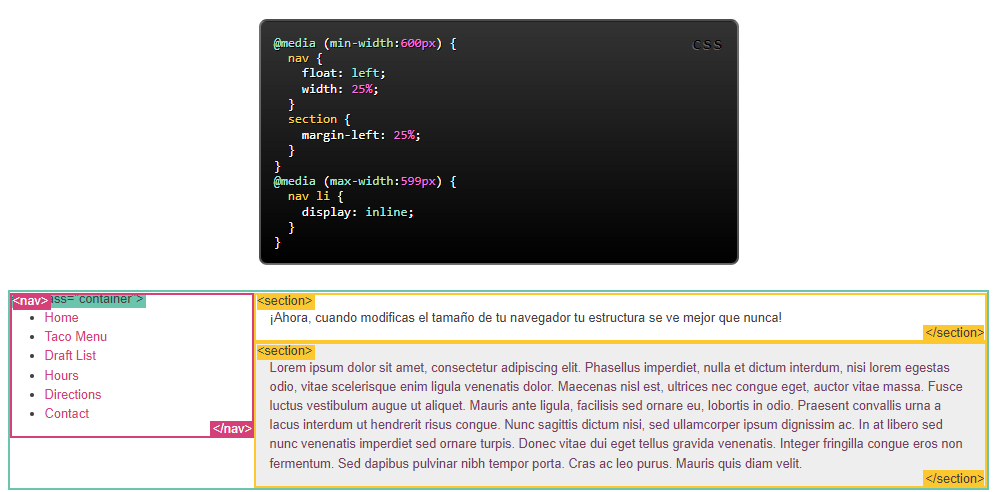
**estructuras con porcentaje de ancho**

Puedes usar porcentaje para estructuras, pero esto puede requerir más trabajo. En este ejemplo, el contendido de nav empieza a desenvolverse de una forma desagradable cuando la ventana es muy angosta. Así que todo se reduce a lo que funcione mejor en tu sitio.



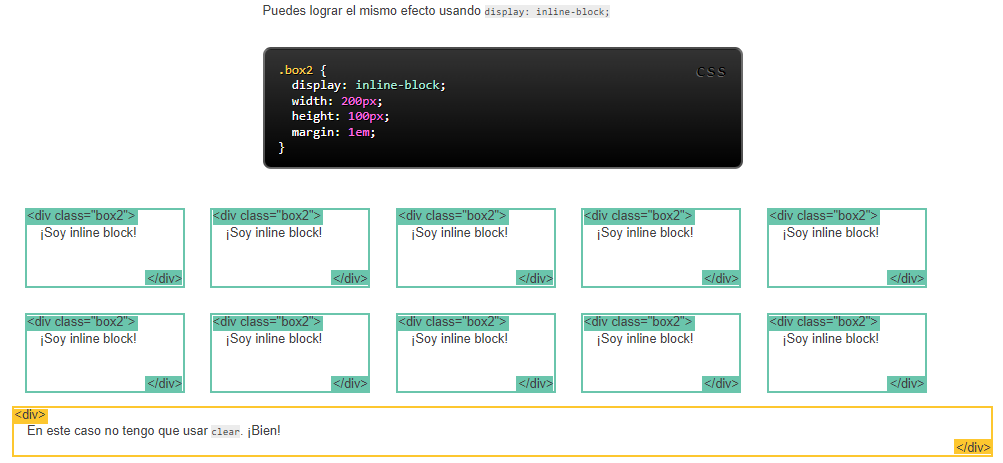
**"Responsive Design"** (Diseño Responsivo) es la estrategia para hacer que un sitio "responda" al navegador y dispositivo en el que se muestra... haciendo que se vea increible pase lo que pase.

Los media queries son la herramienta más poderosa para hacer esto. Tomemos la estructura donde usamos porcentajes y transformémosla en una columna cuando la ventana del navegador es muy pequeña para mantener el menú en la barra lateral:



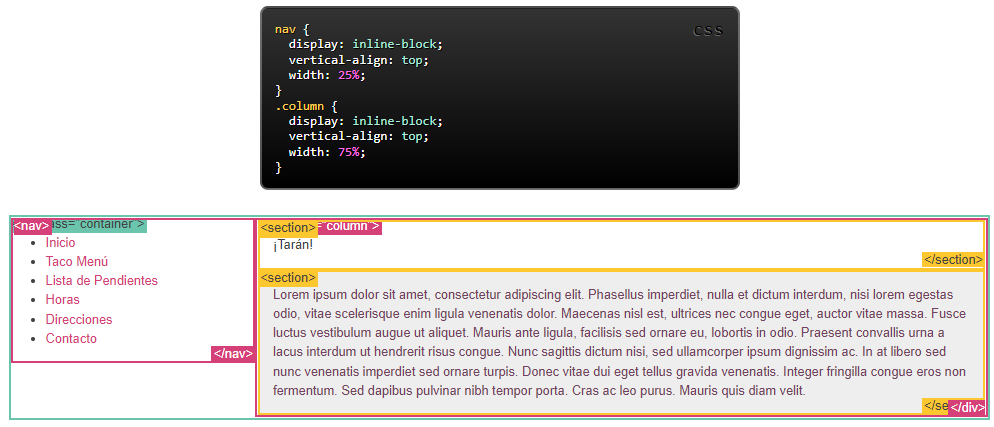
**Inline-Block**

Puedes crear una cuadrícula de cajas que llene el navegador armoniosamente. Esto ha sido posible por mucho tiempo usando float, pero ahora con inline-block es aún más fácil. Veamos ejemplos con las dos opciones.



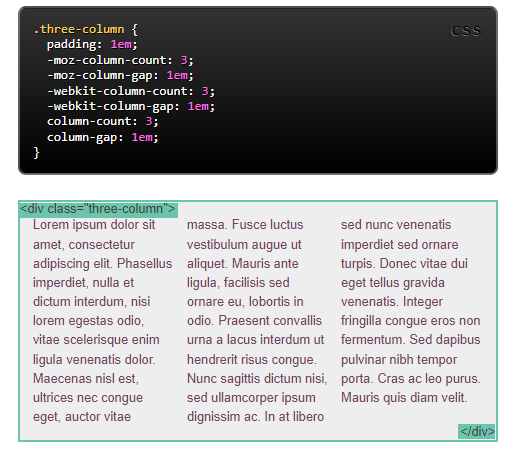
También puedes usar el valor inline-block para estructurar un sitio, pero hay algunas cosas que debes tener en cuenta:

* Los elementos inline-block son influenciados por la propiedad vertical-align, la cual quizas quieras establecer como top.
* Necesitas establecer el ancho de cada columna.
* Si hay un espacio en blanco entre columnas en tu HTML, habrá un vacío entre ellas.



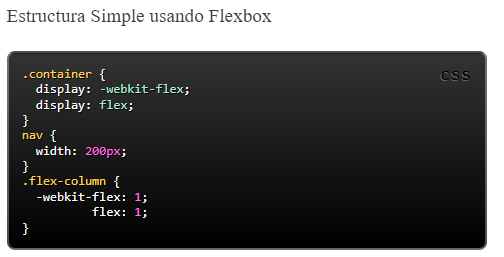
***Multiples Columnas***

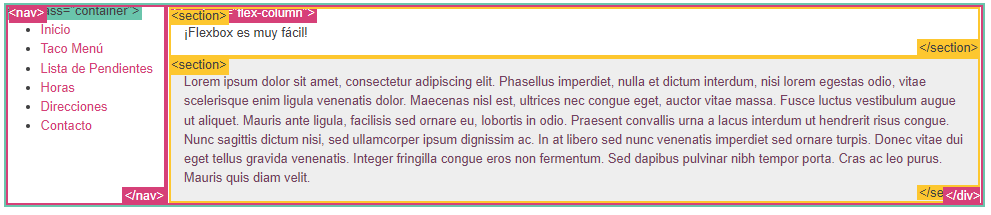
Hay un nuevo set de propiedades CSS que te permiten tener múltiples columnas de texto. Echa un vistazo:



***Flexbox***

La nueva estructura con flexbox está lista para redefinir la manera en la que hacemos estructuras con CSS. Desafortunadamente las especificaciones han cambiado bastante recientemente y su implementación es distinta dependiendo del navegador. De cualquier modo, me gustaría compartir algunos ejemplos, así podrás saber lo que está por venir. Estos ejemplos actualmente funcionan únicamente en Chrome, y se basan en la [última versión estandar](http://www.w3.org/TR/css3-flexbox/).





Estructura fantástica usando Flexbox



**Centrando con Flexbox**

