

MAQUETACIÓN CSS TRADICIONAL

Contenidos

CREAR BOCETO	1
PROPIEDAD DISPLAY	1
LAYOUTS.....	2
BOX-SIZING	2
CENTRADO	2
POSITION	3
COLUMNAS	4
ELEMENTOS FLOTANTES	4

CREAR BOCETO

Diseña siempre “**de lo grande a lo chico**”. El primer paso es crear un prototipo o boceto previo de lo que queremos:

- ▶ Con lápiz y papel
- ▶ mockflow.com
- ▶ moqups.com
- ▶ wireframe.cc
- ▶ balsamiq.com
- ▶ [ninemock](http://ninemock.com)

PROPIEDAD DISPLAY

- ▶ **inline**: no rompen el flujo de la línea y se van colocando uno detrás de otro mientras caben. Ignoran **width** y **height**.
- ▶ **inline-block**: como **inline** pero podemos asignarles **width** y **height**.
- ▶ **block**: rompen el flujo de línea provocando un salto de línea anterior y posterior. Por defecto ocupan todo el ancho de la etiqueta padre.
- ▶ **none**: el elemento desaparece sin dejar hueco vacío en su lugar.
- ▶ Relacionados con tablas: se comportan como el elemento de tabla indicado.
 - ✓ **table**
 - ✓ **table-row**
 - ✓ **table-cell**
 - ✓ **table-caption**
 - ✓ **table-column**
 - ✓ **table-colgroup**
 - ✓ **table-header-group**
 - ✓ **table-footer-group**
 - ✓ **table-row-group**
- ▶ Valores **flex** y **grid** (PRÓXIMA UNIDAD)

LAYOUTS

- ▶ **Fixed:** establece un tamaño fijo en **pixels** para la **anchura** de los elementos. Control total del tamaño de los elementos, pero en pantallas pequeñas muestra scroll horizontal y en grandes muchos espacios en blanco en márgenes del documento.
- ▶ **Elastic:** establece la **anchura** de los elementos en **em** (tamaño de letra por defecto) Al usar el zoom, los elementos se escalan con el texto, pero según cuanto escalemos los elementos pueden solaparse.
- ▶ **Fluid:** establece el **ancho** de los elementos en **%** (con respecto al padre). Los elementos mantienen sus proporciones independientemente de la pantalla, pero en pantallas pequeñas se pueden generar columnas muy estrechas y alargadas.
- ▶ **Híbrido:** establece el **ancho máximo y mínimo** que podrán crecer o encoger los elementos en **pixels** (**max-width** y **min-width**). Estos atributos suelen aplicarse combinados con los principios de alguno de los layouts anteriores, por eso se denomina híbrido
- ▶ **Responsive:** el layout **va cambiando** dependiendo generalmente de la anchura de la pantalla. Si uso **más de un layout** para la página, se denomina **adaptative**.

BOX-SIZING

Dimensiones por defecto de la caja de un elemento:

- ✓ Altura del elemento = altura del contenido + padding + borde
- ✓ Anchura del elemento = anchura del contenido + padding + borde

Para no echar tantas cuentas, se añade el siguiente código CSS indicando la propiedad **Border-Box**:

```
* {  
    -webkit-box-sizing: border-box;  
    -moz-box-sizing: border-box;  
    box-sizing: border-box;  
}
```

Tipos de box-sizing:

- ▶ **border-box:** el tamaño que demos al elemento será la dimensión de la caja.
- ▶ **border-content:** funcionamiento por defecto habría que realizar la suma comentada más arriba.
- ▶ **border-padding:** solo debo sumar el tamaño del borde y el contenido para calcular las dimensiones de la caja.

CENTRADO

- ▶ **Centrado Horizontal**
 - **Elementos en línea:** propiedad **text-align:center** al contenedor padre.
 - **Elementos en bloque:**
 - **Un elemento:** especificar anchura del elemento a centrar y usar la propiedad **margin: Xpx auto**
 - **Varios elementos:** propiedad **text-align:center** al contenedor y propiedad **display:inline-block** en los elementos a centrar.
 - **Contenedores flex:** (PROXIMA UNIDAD)

► Centrado Vertical

- **Elementos en línea:** indicando mismo padding arriba y abajo, o bien simulamos elemento tabla con display y le indicamos propiedad **vertical-align:middle** (en este caso el padre debe tener altura fija).
- **Elementos en bloque:** usando la propiedad **position** en el contenedor (padre) y elemento.
 - Si conocemos altura:

```
.contenedor {  
    position: relative;  
}  
  
.elemento_a_centrar {  
    height: 100px;  
    margin-top: -50px; /*mitad de la altura**/  
    position: absolute;  
    top: 50%;  
}
```
 - Si no conocemos altura:

```
.contenedor {  
    position: relative;  
}  
  
.elemento_a_centrar {  
    position: absolute;  
    top: 50%;  
    transform: translateY(-50%);  
}
```
- **Contenedores flex:** (PRÓXIMA UNIDAD)

POSITION

Permite romper el flujo normal de los elementos (en línea o en bloque) y colocarlos en lugares específicos. Sus valores pueden ser:

- **static:** valor por defecto. El elemento sigue el flujo que lo que corresponde y no se le aplica top, bottom, left, right o z-index.
- **relative:** es como **static** pero si atiende a los desplazamientos expresados en top, bottom, left, right o z-index.
- **fixed:** se le aplica top, bottom, right o z-index en **relación al documento**. Su posición **permanece fija** sin antender al scroll.
- **absolute:** se comporta como **fixed** pero **en relación con la primera etiqueta padre** que tenga **position:relative**.
- **sticky:** se comporta como **relative** hasta llegar a una posición de scroll y a partir de entonces **fixed**.

Propiedad **z-index**: sirve para establecer el orden en el que se verán los elementos cuando se solapan. Es como si estableciéramos “capas”, de modo que el elemento con un **z-index más alto** será el que se posicione **por delante**.

COLUMNAS

Podemos dividir el contenido que queremos mostrar en varias columnas, mediante las propiedades:

- **column-count**: especifica el número de columnas que tendrá el elemento contenedor.
- **column-width**: para fijar el ancho de las columnas del contenedor.
- **column-gap**: establece la distancia entre columnas.
- **column-rule**: similar a **border**, permite especificar el estilo, color y anchura de la línea que separa las columnas.
- **column-span**: con valores **all** o **none** para indicar si un elemento hijo en cuestión, sigue el flujo en columnas o no.
- **column-fill**: el contenedor debe tener altura y establece cómo se rellenan las columnas. Los valores son **auto** o **balance**.
- **break-inside**: con valor **void** si queremos que un elemento no quede roto de una columna a otra.

ELEMENTOS FLOTANTES

- ▶ **float**: originalmente pensada para indicar como se dispone un texto alrededor de una imagen. Hoy se usa para maquetar, flotando elementos y jugando con sus dimensiones. Sus valores son **left** o **right**.
- ▶ **clear**: para forzar que un elemento deje de flotar. Sus valores **left**, **right**, o **both**, si queremos asegurarnos.
- ▶ **clearfix hack**: si tenemos un contenedor que solo contenga elementos flotantes, su altura es cero por defecto, ya que al ser flotantes, el padre no tiene referencia de altura de los hijos. Para solucionarlo, podemos asignar al contenedor la clase clearfix, con dos posibles soluciones:

```
↳ .clearfix {  
    overflow-y: auto;  
}  
  
↳ .clearfix::after {  
    content: "";  
    clear: both;  
    display: table;  
}
```