CSS Básico

octubre 2018

Prof. Ainhoa Iglesias

ÍNDICE:

CSS

1. INTRODUCCIÓN:

CSS[1] es un lenguaje de hoja de estilo que permite que los autores y los usuarios asocien un estilo (por ejemplo, fuentes, espaciado, color...) a los documentos estructurados (por ejemplo, documentos HTML y aplicaciones XML).

Separando el estilo de presentación del contenido de los documentos, CSS simplifica la creación y mantenimiento de los sitios Web.

Estado actual de la especificación CSS:

La evolución de la especificación, está dividida en módulos, para permitir que cada uno pueda ir a ritmos diferentes:[2]

Cada módulo tiene sus propios editores y su propio ritmo, por lo que algunos módulos ya han pasado a ser candidatos a recomendación, y otros acaban de empezar a trabajar:

Soporte actual:

Cada motor de renderizado está adoptando las nuevas propiedades a su ritmo. Antes de dar soporte completo, cada motor implementa una versión "beta" de dicha propiedad.

La implementación “beta” de una propiedad CSS se realiza utilizando unos **prefijos**[3] específicos (en función de cada motor de renderizado) delante de la propiedad CSS a utilizar:

prefix	organization
`-ms-`, `mso-`	Microsoft
`-moz-`	Mozilla
`-o-`, `-xv-`	Opera Software
`-khtml-`	KDE
`-webkit-`	Apple
`-rim-`	Research In Motion

Se pueden utilizar soluciones[4] [5] para no tener que re-escribir propiedades continuamente, o consultar qué propiedades necesitan prefijos o no[6].

No obstante, en octubre de 2015, el W3C instó a los desarrolladores y navegadores a no utilizar estos prefijos[7].

[1] http://www.w3.org/Style/CSS/
[2] http://www.w3.org/Style/CSS/current-work
[3] http://www.w3.org/TR/CSS21/syndata.html#vendor-keywords
[4] http://leaverou.github.io/prefixfree/
[5] https://autoprefixer.github.io/
[6] http://shouldiprefix.com/
[7] https://drafts.csswg.org/css-2015/#future-proofing

Soporte actual en los navegadores:

- Can I Use?[8]
- HTML5 please[9]

Edición en tiempo real

- Codepen[10]
- Dabblet[11]
- CSS3 please[12]
- JSFiddle[13]

Generadores automáticos generales:

- CSS3 Generator[14]
- CSS3 Click Chart[15]
- CSS3 Maker[16]

[8] http://caniuse.com/
[9] http://html5please.com/
[10] http://codepen.io/
[11] http://dabblet.com/
[12] http://css3please.com/
[13] http://jsfiddle.net/
[14] http://www.css3generator.com/
[15] http://www.impressivewebs.com/css3-click-chart/
[16] http://www.css3maker.com/

2. ANTES DE EMPEZAR:

Antes de empezar un proyecto, es recomendable implementar varias soluciones generales:

Normalización de propiedades CSS:

Es recomendable normalizar las propiedades CSS de los elementos HTML que por defecto implementan los navegadores.

Por ejemplo, ante una lista desordenada (<ul>), algunos navegadores añaden un margen lateral izquierdo en el elemento <ul> para simular la sangría; mientras que otros navegadores añaden ese margen en el elemento <li>.

A lo largo de los años, han existido varios métodos: Normalize[17], Reset[18]..., siendo a día de hoy el más recomendable una combinación de ambos, personalizada al gusto del maquetador.

Comentarios condicionales[19]

Es un método que se utiliza para filtrar código para diferentes versiones de Internet Explorer (hasta IE9 inclusive).

Nos valemos de un simple comentario HTML para implementar este filtro, por ejemplo:

```
<!--[if IE]>
Código que queramos filtrar
<![endif]-->
```

Es necesario mantener los corchetes y el texto "if". A continuación, en función de la versión que queramos filtrar, podemos realizar diferentes combinaciones:

```
[if lt IE version]: versión menor que la indicada.
[if lte IE version]: versión menor o igual que la indicada.
[if IE version]: sólo la versión indicada.
[if gte IE version]: versión mayor o igual que la indicada.
[if gt IE version]: version mayor que la indicada.
[if ¡IE version]: ignora el contenido para la versión indicada.
[if ¡IE]: ignora el contenido para cualquier versión de Internet
Explorer.
```

Dentro de estos comentarios condicionales podemos poner cualquier tipo de contenido: código HTML, CSS, Javascript…:

```
<!--[if IE]>
<link rel="stylesheet" type="text/css" href="styleIE.css" />
<![endif]-->
```

[17] http://necolas.github.io/normalize.css/
[18] http://meyerweb.com/eric/tools/css/reset/
[19] http://msdn2.microsoft.com/en-us/library/ms537512.aspx

Soporte parcial automático para Internet Explorer:

Debido a la baja compatibilidad de Internet Explorer (y sus múltiples versiones) con gran parte de las propiedades CSS, es recomendable utilizar una serie de librerías Javascript que, aunque no dan un soporte completo, sí que ayudan a dar un soporte mínimo.

Las más recomendables son:

- HTML5 Shiv [20]
- IE9.js [21] [22]
- Respond[23]

Detectar si existe o no soporte

- Modernizr[24]

Saber diferenciar elementos en bloque y elementos en línea

Los elementos en bloque son aquellos que, por defecto, tienen una anchura del 100% de su elemento padre, como los encabezados, listas...

Los elementos en línea tendrán como anchura lo que ocupe su contenido (enlaces, palabras, imágenes, campos de formulario...).

[20] https://github.com/afarkas/html5shiv
[21] https://code.google.com/p/ie7-js/
[22] https://github.com/mylovecompany/ie9-js
[23] https://github.com/scottjehl/Respond
[24] http://www.modernizr.com/

3. VALIDACIÓN:

Para prevenir errores a la hora de implementar la CSS, es recomendable validar el código periódicamente.

Para ello, podemos utilizar el servicio gratuíto del W3C:

https://jigsaw.w3.org/css-validator/

4. INSERTAR CSS:

Hoja de estilo externa:

```
<link rel="stylesheet" type="text/css" href="estilos.css"
media="all" />
```

En la cabecera (<head>...</head>):

```
<style type="text/css" media="all">
    a {text-decoration:none;}
</style>
```

En línea, como atributo:

```
<a href="index.html" style="text-decoration:none">Inicio</a>
```

La regla @import: (desaconsejada)

La regla @import permite importar hojas de estilo desde otras hojas de estilo. Se puede usar de las siguientes formas:

```
@import "hoja_de_estilo.css";
@import url("hoja_de_estilo.css");
```

Esta declaración deberá ir antes que cualquier otra regla en la hoja de estilo.

5. HERENCIA:

Cada página HTML está compuesta por una serie de elementos (títulos, párrafos, listas, tablas, etc.) organizados en una estructura donde cada elemento está contenido por otro elemento, que a su vez puede estar contenido por otro.

En esta estructura existe un elemento **raíz** que es el que actúa de contenedor de todos los demás elementos. En HTML se puede considerar como elemento raíz al elemento <body> o al elemento <html>.

Cada elemento *hereda* las propiedades del elemento que lo contiene (elemento **padre**). Si, por ejemplo, indicamos la propiedad negrita para el elemento <body>, todos los elementos de la página heredarán este atributo, sin necesidad de indicarlo en cada uno de los elementos.

Al especificar una propiedad a un elemento determinado, esta propiedad reemplazará al valor heredado.

Todas las propiedades tienen siempre un valor asignado. Puede ser el especificado en la CSS o, en su defecto, el valor heredado. En ausencia de los dos anteriores, se usa el valor inicial (el valor predeterminado para cada propiedad).

Si el navegador no aplica un estilo a un elemento, puede ser por un problema de CSS Specifity [25] [26] [27] [28]

[25] http://css-tricks.com/specifics-on-css-specificity/
[26] http://www.smashingmagazine.com/2007/07/27/css-specificity-things-you-should-know/
[27] http://cssspecificity.com/
[28] http://www.standardista.com/css3/css-specificity/

6. SINTAXIS BÁSICA:

Selector { propiedad: valor;}

```
a{
    text-decoration :none;
    color: #FF0000;
    font-family: Arial, Verdana, sans-serif;
}
```

Cada declaración deberá terminar con un punto y coma ;

Se pueden poner todas las declaraciones seguidas:

```
a{ text-decoration :none; color: #FF0000;}
```

Comentarios:

```
/*    comentarios .... */
```

Los comentarios abarcan todo el bloque.

7. SELECTORES BÁSICOS:

Los utilizamos para elegir a qué elementos aplicaremos las propiedades. Hay varios tipos de selectores.

Tipo (elementos):

```
a{ ..... }
p{ ..... }
...
```

Elementos únicos (IDs):

```
#cabecera{ ..... }
#menu{ ..... }
...
```

Clases:

```
.aviso{ ..... }
.error{ ..... }
...
```

Descendentes:

```
ul li{ ..... }
ol li{ ..... }
form dl dt input{ ..... }
...
```

Universal:

Se aplica a todos los elementos

```
* {font-weight: bold;}
```

*Hijos *:*

Estos selectores se utilizan para aplicar propiedades a un elemento que es un hijo directo del elemento padre:

```
div > em { color: #FFFF00; }
div>em { color: #FFFF00; }
```

```
<body>
<h1>Texto de <em>cabecera</em></h1>
<div>
Ejemplo de <em>texto</em>                          <- texto afectado.
<p>Texto dentro de un <em>párrafo</em></p>          <- no afectado.
</div>
</body>
```

*Adyacentes *:*

Se aplica para elementos que están uno detrás del otro:

```
em + strong { color: #FFFF00;}
<p>Hay un <em>texto</em> y <strong>otro texto</strong></p>.
```

*De Atributos *:*

```
input[type="text"]{ ..... }
input[type="submit"]{ ..... }
<input type="text" name="nombre" />
<input type="submit" value="enviar" />
```

input[type="text"] todos los elementos <input> de tipo "text"

`p[lang]` todos los elementos `<p>` con el atributo `"lang"`

`p[lang="en"]` todos los elementos `<p>` con el atributo `lang="en"` (valor exacto)

Pseudo-clases:

```
:link
:visited
:hover
:active
:focus *
:first-child *
```

Pseudo-elementos *:

```
:first-line
:first-letter
:before **
:after **
```

* Estos selectores no tienen soporte en IE 6 pero sí en IE 7 (salvo los marcados con **).

8. FORMAS DE APLICAR ESTILOS:

Los utilizamos para elegir a qué elementos aplicaremos las propiedades. Hay varios tipos de técnicas.

Técnica chapucera

Si queremos que nuestra página tenga un fondo gris y un color de texto negro, hasta ahora hacemos lo siguiente:

```
<body bgcolor="#E6E7E6" color="000000">
```

o si lo refinamos un poco más:

```
<body class="pagina">

.pagina{
    background-color:#E6E7E6;
    color:#000000;
}
```

Pero, ¿no sería más fácil decirle al navegador que cambiara las propiedades de `body`, sin necesidad de crear estilos, indicándoselo una u otra vez?

Lo conseguimos mediante la redefinición de elementos.

La "técnica": redefinición de elementos

En nuestra hoja de estilos, en vez de crear una clase a la que referenciaremos, llamaremos directamente al elemento que queramos cambiarle las propiedades.

Esta "técnica" no es nueva, ya la aplicamos en nuestras hojas de estilo, lo que ocurre es que sólo la aplicamos en el elemento a:

```
a{ text-decoration: none;}
a:link{ font-weight: bold;}
......
```

Una buena noticia: esta técnica se puede aplicar a cualquier elemento html.

```
body{
    background-color:#E6E7E6;
    color:#000000;
}
```

De esta forma, cada vez que el navegador encuentre el elemento `body` en nuestras páginas, aplicará los estilos indicados en la hoja de estilos.

Otras propiedades que podemos aplicar a `body`, por ejemplo: tipo de fuente y tamaño:

```
body{
    background-color:#E6E7E6;
    color:#000000;
    font-size:80%;
    font-family: Verdana, Arial, sans-serif;
}
```

Pensemos ahora en todos los `border="0"` que aplicamos en todas nuestras imágenes que son enlaces. ¿no sería más cómodo que las imágenes, simplemente, no tuvieran borde por defecto?

```
img a { border: 0px; }
```

9. CONSEJOS Y BUENAS PRÁCTICAS:

Nombrar las clases y los identificadores por su función y no por su aspecto.

Si creamos una clase llamada ".textonaranja", que aplicamos a textos que queremos que estén en color naranja, y más tarde nos piden que los cambiemos a color verde, tendríamos una clase llamada "textonaranja" que pone los textos en verde.

Para evitar estas incongruencias, debemos llamar a las clases por la función que van a realizar (importante, menú, nota…).

Aplicar varias clases a la vez.

Podemos aplicar dos o más clases a un elemento, separándolas con un espacio:

```
<h1 class="titular empresas">Título del contenido</h1>
```

Elimina el borde punteado de los enlaces.

Para evitar este efecto, basta con aplicar la siguiente propiedad a los enlaces:

```
a{outline:none;}
```

Sin embargo, esto hará que una persona que utilice el teclado para desplazarse por pantalla no pueda saber por dónde va el tabulador. Por lo tanto, si se utiliza, será necesario implementar estilo la pseudo-clase :focus del enlace:

```
a:hover, a:focus{color:#F00;}
```

Hacer que salga el icono de enlace en un elemento.

Si queremos hacer que aparezca el icono típico de los enlaces (la mano) en otros elementos (por ejemplo, botones de formulario), la propiedad CSS a utilizar será:

```
.boton {cursor:pointer;}
```

Lo más habitual es pensar en la propiedad "hand", pero esta propiedad no existe en la especificación CSS.

No asociar hojas de estilo vacías.

Si tenemos pensado implementar varias hojas de estilo y ponemos las llamadas en la cabecera, mejor comentar la llamada si una de las CSS está vacía, ya que algunos navegadores pueden tener problemas con las hojas de estilo en blanco y el tiempo de carga de la página será mayor.

Ordenar correctamente las pseudoclases.

Para que Internet Explorer implemente correctamente las pseudoclases, es necesario establecer un orden obligatorio:

a:link, a:visited, a:hover, a:active

(recordar **LVHA** (Love & Hate).

Si utilizamos también la pseudoclase :focus, entonces la regla es la siguiente: **LVHFA** (o "Lord Vader's Handle Formerly Anakin", tal y como sugiere Matt Haughey).

PROPIEDADES BÁSICAS

10. PROPIEDADES BÁSICAS:

Tipografía

FONT-FAMILY

Tipo de fuente.

- Arial; Helvetiva; sans-serif; serif; Courier;...

FONT-SIZE

Tamaño de la fuente.

- valor relativo (% o em)

- valor absoluto (puntos o pixels)

FONT-STYLE

Estilo o efecto tipográfico de la fuente.

- italic; normal

FONT-WEIGHT

Enfatizado (negrita).

- bold; normal

TEXT-DECORATION

Efectos tipográficos.

- blinks (parpadeante); line-through (tachado); underline (subrayado); overline ('sobrerayado')

TEXT-TRANSFORM

Transformación del texto.

- capitalize (convierte la primera letra de cada palabra a mayúsculas); uppercase (mayúsculas); lowercase (minúsculas)

FONT-VARIANT

Efecto versales: small-caps; normal

LETTER-SPACING

Espacio entre letras. En unidades

WORD-SPACING

Espacio entre palabras. En unidades

LINE-HEIGHT

Espacio entre renglones. En unidades

TEXT-INDENT

Sangrado. En unidades

Colores

BACKGROUND-COLOR

Color de fondo.

- nombre del color

- valor del color

- transparent

BORDER-COLOR

Color del borde de la caja.

- nombre del color

- valor del color

COLOR

Color del elemento.

- nombre del color

- valor del color

Bordes

BORDER-STYLE

Tipo de borde de la caja.

- solid (solido); dashed (lineado); dotted (punteado); double (dos líneas); inset (efecto 3D); none (sin borde)

BORDER-WIDTH

Ancho del borde px

BORDER-LEFT, BORDER-RIGHT, BORDER-BOTTOM y BORDER-TOP

Tipo, ancho, color... de los diferentes bordes.

- unidades

- color

- solid (solido); dashed (lineado); dotted (punteado); ...

Alineación de textos

VERTICAL-ALIGN

Alineación vertical respecto al contenedor (solo en elementos inline/inline-block).

- baseline; bottom; middle; sub; super; text-top; text-bottom; top

TEXT-ALIGN

Alineación horizontal del texto.

- left; rigth; center; justify

Dimensiones y Márgenes

WIDTH y HEIGHT

Ancho y alto de la caja.

- unidades

- porcentaje

PADDING

Margen interno de la caja.

- unidades

- porcentaje

PADDING-TOP, PADDING-BOTTOM, PADDING-LEFT y PADDING-RIGHT

Márgenes internos superior, inferior, izquierdo y derecho.

- unidades

- porcentaje

MARGIN

Margen externo de la caja.

- unidades

- porcentaje

MARGIN-TOP, MARGIN-BOTTOM, MARGIN-LEFT y MARGIN-RIGHT

Márgenes externos superior, inferior, izquierdo y derecho.

- unidades

- porcentaje

Listas

LIST-STYLE-TYPE

Define el símbolo o tipo de numeración que precede a cada elemento de una lista.

- circle; disc; square; decimal; lower-alpha; lower-roman; upper-alpha; upper-roman; none

LIST-STYLE-IMAGE

Define la utilización de una imagen como marcador o símbolo de elementos de una lista.

- url(localización de la imagen)

- none

LIST-STYLE-POSITION

Determina la forma de sangrado de aquellas líneas que no son la primera.

- inside; outside

Visualización

DISPLAY

- inline; block; inline-block; list-item; none

Posicionamiento

POSITION

Posicionamiento con respecto a la página o elemento padre.

- absolute / relative / fixed

- top / right / bottom / left

FLOAT

Alineación del elemento respecto a elementos contiguos.

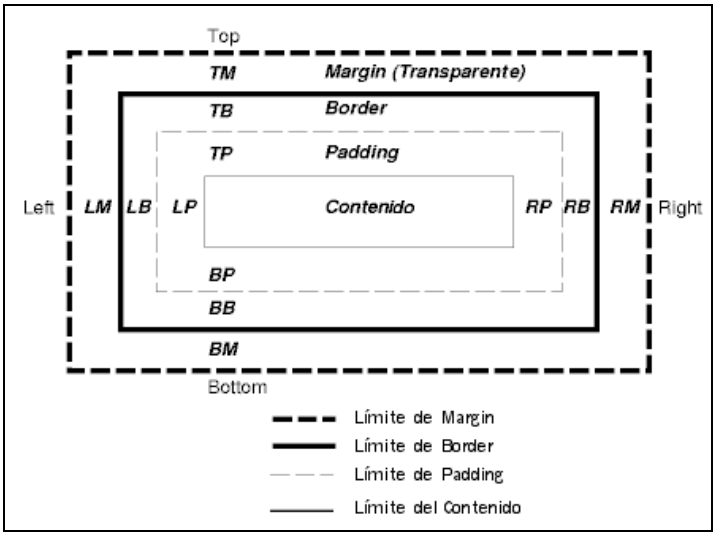
-left; right; none

POSICIONAMIENTO BÁSICO

1. EL MODELO DE CAJA:

Cada caja[29] tiene un área de contenido (ej., texto, una imagen, etc.) y las áreas circundantes opcionales de `padding`, `border` y `margin`; el tamaño de cada área es especificado por las propiedades que se definen abajo.

El siguiente diagrama muestra cómo se relacionan estas áreas y la terminología usada para referirse a las partes de `margin`, `border` y `padding`:



El ancho de la caja está dado por la suma de los márgenes, bordes y rellenos izquierdos y derechos, y el ancho del contenido. La altura está dada por la suma de los márgenes, bordes y rellenos superiores e inferiores, y la altura del contenido.

Por lo tanto si tenemos un elemento con las siguientes propiedades:

```
div {
    width: 100px;
    padding: 10px;
    border: 5px solid black;
    margin: 10px;
}
```

Aunque inicialmente nuestro bloque tiene una anchura de 100 px, en realidad, nuestro elemento tiene una anchura de 150 px:

```
100 px de anchura
10 px de padding izquierdo      10 px de margin izquierdo
10 px de padding derecho        10 px de margin derecho.
5 px de borde izquierdo         5 px de borde derecho
```

Este es el comportamiento del modelo de caja.

[29] http://www.w3.org/TR/CSS21/box.html

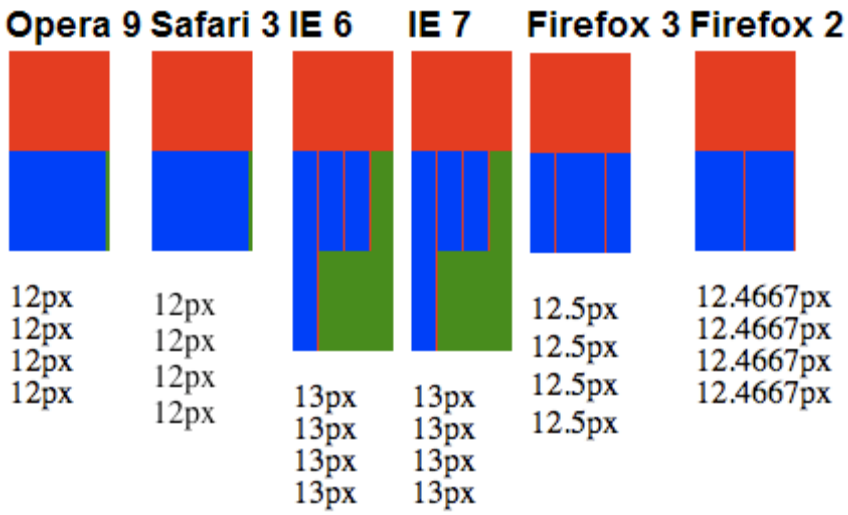
Márgenes colapsados:

Cuando dos elementos en bloque tienen márgenes superior y/o inferior, estos márgenes se fusionan entre sí. Es lo que se conoce como "márgenes colapsados"[30].

Renderizado Subpixel:

A priori, teniendo en cuenta cómo funciona el modelo de caja, la maquetación sería fácil de realizar. Sin embargo, existe un problema: cómo renderizan los navegadores los tamaños en porcentajes[31].

```
.padre{width:50px;}
.hijo{width:25%;} /* 50px / 4 (25%) = 12,5px */
```



Modelo de caja alterado:

En algunos navegadores (IE 5.5 y anteriores), este comportamiento es incorrecto y en vez de añadir las anchuras del padding, border y margin, lo que hace es restarlas de la anchura del bloque.

Por lo tanto, aunque inicialmente nuestro bloque tiene una anchura de 100 px, para estos navegadores, en realidad, nuestro elemento tiene una anchura de 50 px (si restamos los demás valores).

Era necesario utilizar hacks para que la página se renderizase bien[32], como por ejemplo, el de la “caja dentro de una caja” explicado a continuación.

[30] http://www.complexspiral.com/publications/uncollapsing-margins
[31] http://ejohn.org/blog/sub-pixel-problems-in-css/
[32] http://css-discuss.incutio.com/wiki/Box_Model_Hack

Caja dentro de otra caja

Esta "técnica" era considerada uno de los hacks más fáciles de entender y de utilizar, aunque requería añadir un `<div>` extra en nuestro código por cada bloque. Sin embargo, era el único que permitía el uso de unidades relativas y sin tener que hacer cálculos.

Englobamos nuestro bloque dentro de otro. Al bloque exterior le aplicamos la anchura que realmente queremos y al bloque interior, le aplicamos el `padding, border` y `margin`.

```
div {
    width: 100px;
}
div .i {
    padding: 1em;
}

<div>
    <div class="i">
       Text
    </div>
</div>
```

En realidad, lo que estamos consiguiendo es simular el comportamiento erróneo del modelo de caja.

Box-sizing[33] [34]:

Este comportamiento "erróneo", en realidad, es mucho más práctico a la hora de maquetar.

Es por ello, que en CSS3 podemos conseguir que un bloque no renderice siguiendo el modelo de caja, sino que renderize con el comportamiento "erróneo".

Por lo tanto, el tamaño del bloque permanecerá inalterable aunque implementemos bordes y/o padding (el margen sí que lo tiene en cuenta).

```
box-sizing: border-box;
```

Para que vuelva a renderizar siguiendo el modelo de caja:

```
box-sizing: content-box;
```

Esta propiedad se puede aplicar a nivel global a todos los elementos. Es especialmente útil en campos de formularios y bloques de maquetación.

[33] http://www.paulirish.com/2012/box-sizing-border-box-ftw/
[34] https://css-tricks.com/box-sizing/

2. PROPIEDADES DE MAQUETACIÓN:

Posicionamiento

POSITION

Posicionamiento con respecto a la página o elemento padre.

- static (valor genérico)

Posicionamiento con respecto al primer pariente superior que tenga establecido unposition diferente a static.

- absolute
- relative

Posicionamiento con respecto a la página:

- fixed

Coordenadas:

- top / bottom
- right / left

FLOAT

Alineación del elemento respecto al flujo de elementos de la página (lado izquierdo o derecho[35].

-left; right; none

(nota: float:center no existe ni va a funcionar)

CLEAR

Despeja y modifica el flujo del elemento teniendo en cuenta si hay otros elementos a la izquierda o derecha.

- left; right; both; none

[35] https://css-tricks.com/all-about-floats/

Visualización

DISPLAY

`- none; block; inline; inline-block; table; table-cell; list-item; flex;`

`display: flex;` necesita más propiedades adicionales para poder funcionar adecuadamente[36].

Podemos ver varios ejemplos[37] y herramientas de simulación[38] [39] [40].

GRID

Es una nueva propuesta de maquetación[41] [42] que aún está en fase inicial.

[36] https://css-tricks.com/snippets/css/a-guide-to-flexbox/
[37] http://philipwalton.github.io/solved-by-flexbox/
[38] http://the-echoplex.net/flexyboxes/
[39] http://bennettfeely.com/flexplorer/
[40] http://demo.agektmr.com/flexbox/
[41] https://www.w3.org/TR/css3-grid-layout/
[42] https://css-tricks.com/snippets/css/complete-guide-grid/