UF3. Gestión de Estilos con CSS

Maquetar HTML con CSS3

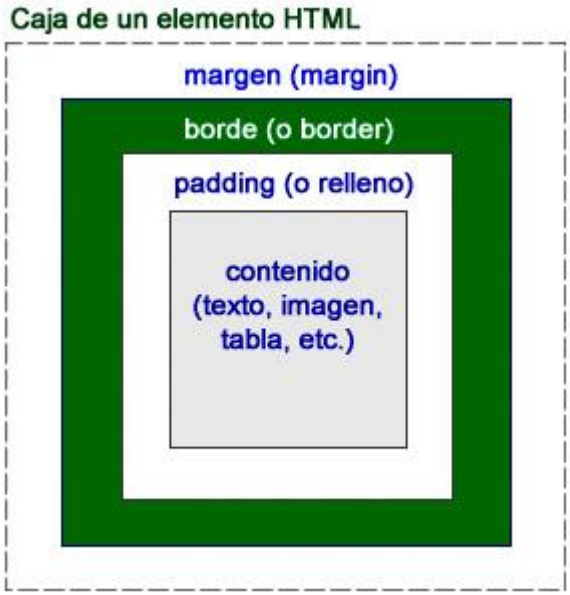
Índice

1. ¿Por qué maquetar con CSS?

- Idea principal: Con CSS separamos diseño de contenido.
- Menor tiempo de carga y menor tráfico en el servidor. Los navegadores guardan la hoja de estilos en la caché.
- Mayor posibilidades de diseño.
- Mantenimiento y flexibilidad. Cuando es necesario introducir un cambio en el diseño se modifica el CSS, sin tener que tocar las páginas html.
- Posicionamiento. Las páginas html diseñadas con CSS tienen un código más limpio porque no llevan diseño, sólo contenido. Esto es semánticamente más correcto y la página aparecerá mejor posicionada en los buscadores.

2. El "Modelo de Cajas" de HTML/CSS…

- Todos los elementos de una web (párrafos, enlaces, imágenes, tablas, etc.) son **cajas rectangulares.**
- Hay dos tipos de cajas: **block** e **inline**.
  - Los elementos **block** aparecen solos ocupando toda una fila, insertando "saltos de línea". Ejemplos: <h1>, <p>, <table>, , <ul>, <ol> y por supuesto <header> <nav> …
  - Los elementos **inline** siguen el flujo,y no "cortan" el texto donde está metido. Ejemplos: <a> <img> <video> <stong> …
- Hay que tener en mente que ambos tipos comparten el modelo de caja, que es el que aparece en la figura:



- Veremos que con CSS podremos posicionar estas "cajas" en el lugar que deseemos.

3. Contenedores de elementos HTML: Capas

- Tenemos en HTML la posibilidad de utilizar “contenedores de elementos” o capas.
- Tenemos el elemento **…** que es un contenedor de tipo “**inline**” (o sea, no incorpora salto de línea). Con él podemos agrupar o marcar elementos con el fin de darles formato.

```
<p> En un <span class=”highlight”>lugar de la <em>Mancha</em></span>
   cuyo nombre no quiero acordarme...</p>
```

- Tenemos el elemento ** ... ** que es un contenedor de tipo “**block**”. Esta etiqueta crea una capa, y sirve para contener a otros elementos:

```
<div id="header">
   <h1>Mi blog</h1>
   <ul id="menu">
       <li><a href="..." title="...">Principal</a></li>
       <li><a href="..." title="...">Acerca de</a></li>
   </ul>
</div>
```

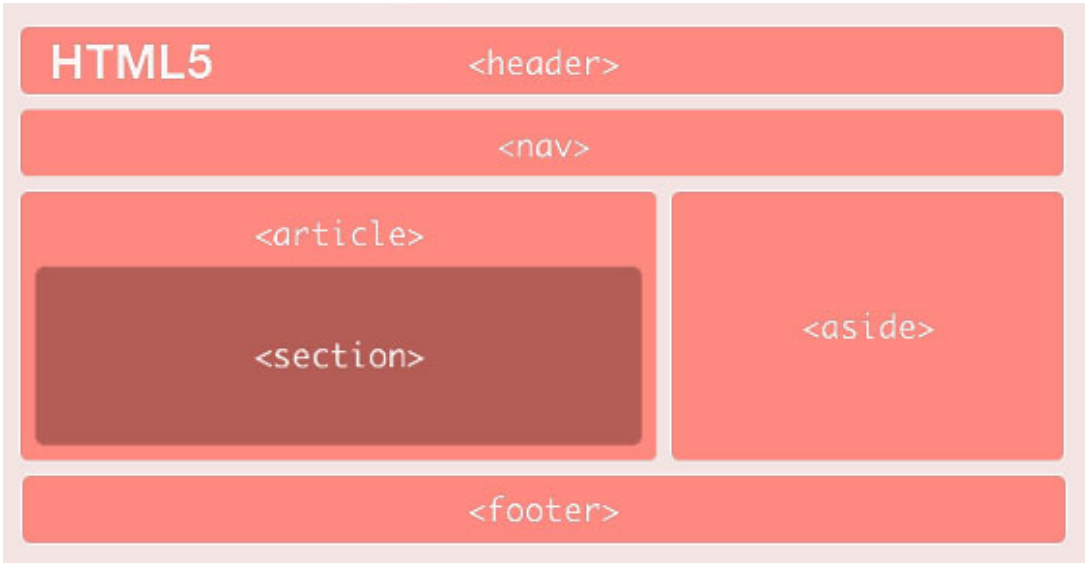
- Con las etiquetas ** ** y su atributo **id**, podemos organizamos semánticamente la página, y luego con CSS podemos maquetarla, posicionando cada “capa” donde deseemos.
- No obstante en HTML5, tenemos ya capas con nombre: **<header> <nav> <section>**...

Nota: Los nuevos contenedores HTML5 se comportan como **div**’s, o sea, son tipo “block”.

Recordatorio: Mediante CSS podemos decidir como se muestre cada capa: “block”, “inline”, “inline-block”

En HTML 4.01 utilizábamos siempre div's:

Ahora en HTML5 tenemos etiquetas exclusivas:



No obstante, HTML5 no implica que ya no se usen , todavía son imprescindibles

3.1. Modificando el modelo cajas por defecto

- A cada "caja" podemos especificarle espaciados ***margin*** y ***padding***, *y también un* ***border*** *visible, así como un* ***outline*** *(elemento exterior a border,no consume espacio)*
  - *Por defecto: ancho de un elemento se altera si aplicamos* ***border*** *y/o* ***padding****!! → La anchura del elemento que se especifica con CSS, por defecto no incluye borde ni padding.* – Esto traía de cabeza a los diseñadores
  - CSS3 soluciona esto…
    - → Con **box-sizing: border-box;** // hacemos que el ancho especificado sea el equivalente al ancho total.
      - Nota: box-sizing: content-box; // es el valor por defecto, o sea anchura no incluye ni borde, ni padding.

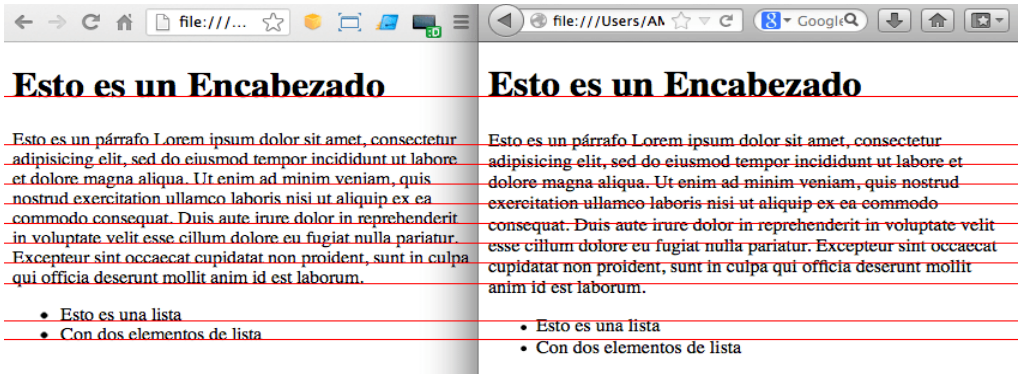
Importante

+info: https://emiliocobos.net/la-propiedad-box-sizing/
https://www.w3schools.com/cssref/css3_pr_box-sizing.asp

- Por otra parte, podemos decidir nosotros la forma de visualizar cada elemento:
  - **`display: none;` //oculta la "caja"**
  - **`display: block;` //muestra la "caja" como block**
  - **`display: inline;` //muestra la "caja" como in-line**
  - **`display: inline-block;` //muy utilizado hoy en día, lo veremos**
  - **`display: table table-row table-cell;`**

+info: https://uniwebsidad.com/libros/css-avanzado/capitulo-4/propiedad-display

3.2 Inicialización de estilos del navegador



- **PROBLEMA: Diferentes navegadores => diferentes estilos por defecto**
- **SOLUCIÓN: Inicializar los valores de estilo a 0**

Importante

```
* { padding: 0px; margin: 0px; box-sizing: border-box;
    list-style: none; text-decoration: none;
    border: none; outline: none; }
```

- **Hay hojas de estilo CSS ya creadas para realizar esto. Las + conocidas:**
  - **reset.css** de Eric Meyer http://meyerweb.com/eric/tools/css/reset/
  - **normalize.css** de Nicolas Gallagher http://necolas.github.io/normalize.css/

+ Información: https://octuweb.com/normalize-vs-reset-css/

https://franciscoamk.com/usar-reset-css-o-normalizar/

3.3. Centrando y Redimensionando contenedores

- Los navegadores ya aplican cierto “margin” y “padding” a ciertos elementos, como <h1>, <p>, <table>, … Es importante inicializar todos los elementos antes de empezar a maquetar:

```
* { margin: 0px; padding: 0px; box-sizing: border-box; … }
:root { font-size: 16px; } // Por si navegador tiene otro tamaño
```

- Podemos realizar centrado automático del contenedor principal:

```
body { margin: 0 auto; }
```

- A cada contenedor le podemos dar anchura y amplaria:

```
nav { width: 60rem; height: 8rem; }
section { width: auto; }
```

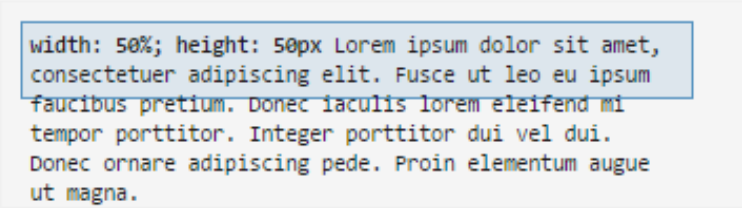
- Podemos también fijar los máximos y mínimos:

```
body {   width: 80%;
         max-width: 1000px;
         min-width: 400px;
         min-heigth: 800px;  }
```

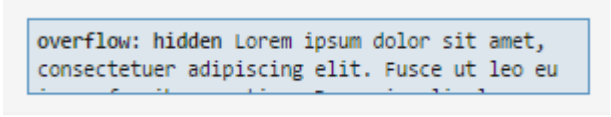
- Unidades pueden ser: **%**, **em, rem**, **ex** (altura x)**, px, in, cm, mm, pt**
  Otras que están por llegar: https://developer.mozilla.org/es/docs/Web/CSS/width

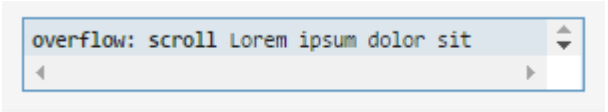
3.4. Propiedad overflow de los contenedores

- Si contenedor le asignamos height o max-height, en el caso de que sus contenidos de un elemento no quepan en su interior, el navegador debe mostrarlos aunque se salgan del elemento:



- Podemos cambiar este comportamiento con la propiedad **overflow**
  - **`overflow: visible;`** **//comportamiento por defecto**
  - **`overflow: hidden;`**
  - **`overflow: scroll;`**
  - **`overflow: auto;`** **Importante**





//Si no especificamos tamaño del contenedor, este se agranda tanto como su contenido.

* Si contenido se sale vertical u horizontalmente, se muestra la barra de scroll de ese lado.

* Si contenido se sale por todos los lados, se muestran las dos barras de scroll.

4. Métodos para posicionar las Capas con CSS

- Las etiquetas de capa <header> <nav> <aside> <section> <article> <footer> se comportan como , o sea son también "capas"
- Tenemos varias formas de maquetar las capas en CSS:
  - Métodos más rudimentarios, y poco aconsejables:
    - Posicionamientos estáticos, relativos, absolutos y fijos:

```
" position:static;  /* posicionamiento por defecto */ "
" position:relative; top:20px; left: 20px;"
" position:absolute; top:20px; left: 20px;"
" position:fixed; bottom:0px; left: 0px;"
Nueva → " position:sticky; top:0px;"
```

Interesantes para cosas concretas:
- banner de cookies
- headers/footers a los que no que no queremos que les afecte scroll

+info: http://www.lawebera.es/maquetacion-web/maquetar-paginas-web-posicion-capas-i.php
http://es.learnlayout.com/position.html https://developer.mozilla.org/es/docs/Web/CSS/position

    - Posicionamiento flotante:

propiedades "**float: left;**"
"**clear: both;**"

+info: http://www.comocrearunsitioweb.com/c10-maquetaci&oacuten-con-divs
asistente: http://www.pagecolumn.com/

4. Métodos para posicionar las Capas con CSS

- Métodos más modernos de **CSS3** :

Desaconsejado para Maquetar toda una página. Interesante para disponer texto de un párrafo a varias columnas.

- Posicionamiento con múltiples columnas:
  "`column-count: 3; column-gap: 1em; column-rule: 1px solid;`"
  +info: http://desieteados.xtreemhost.com/5180/multiples-columnas-en-css3/

- Posicionamiento con "display: table":
  +info: http://coolvillage.es/displaytable-cell-y-su-oportunidad-en-el-responsive-design/
  https://desarrolloweb.com/articulos/modelo-tabla-css-propiedad-display-explicaciones-ejemplos.html

- Posicionamiento con "inline-block" :
  +info: http://s1.accesoperu.com/wp6/wp6.php?p=15417

- Posicionamiento con "Flex Box" (display: flex):
  +info: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
  http://gnoma.es/blog/flexbox-interfaces-flexibles-css3/

- Posicionamiento con "Grid Layout" (display: grid):
  +info: https://css-tricks.com/snippets/css/complete-guide-grid/
  http://learncssgrid.com/
  https://lenguajecss.com/p/css/propiedades/grid-css
  https://developer.mozilla.org/es/docs/Web/CSS/CSS_Grid_Layout

Las tres últimas de las reseñadas, son especialmente interesantes para realizar *"**responsive web dessign**"* (web adaptable a diferentes formatos de pantalla con "media queries").

5. Posicionando con "display: table"

Propiedades CSS que utilizaremos

Se basa en la utilización de propiedades CSS, para que nuestros containers se comporten como celdas de una tabla:

- Donde tengamos más de una columna ponemos un div que será la "tabla": **`display:table`**
- Emplearemos **`display:table-row;`** o **`display:table-cell;`** en función de si queremos abrir una fila o una celda de nuestra tabla. (los **`table-row`** se puede omitir por comodidad)

Ejemplo:

```
<header>
     <img id="imglogo" src="img/logo.jpg" />
</header>

<main>
  <div id="tabular">
      <nav id="menuizq"> ... </nav>
      <article id="pancentral">
        <p>Lorem ipsum ...</p>
      </article>
      <aside id="panderecha">
        <p>Lorem ipsum ...</p>
      </aside>
  </div>
</main>
<footer><p>Copyrigh 2013</p></footer>
```

CSS: *la idea será maquetar el formato de 3 columnas, convirtiendo los div's en tabla, filas y celdas:*

```
div#tabular { display: table; }
nav#menuizq,
article#pancentral,
aside#panderecha { display: table-cell;
                   ...
                 }
```

NOTA: Este método es apto para diseños simples, no adaptables a dispositivos

6. Posicionando con "display: inline-block" (1/5)

Propiedades CSS que utilizaremos

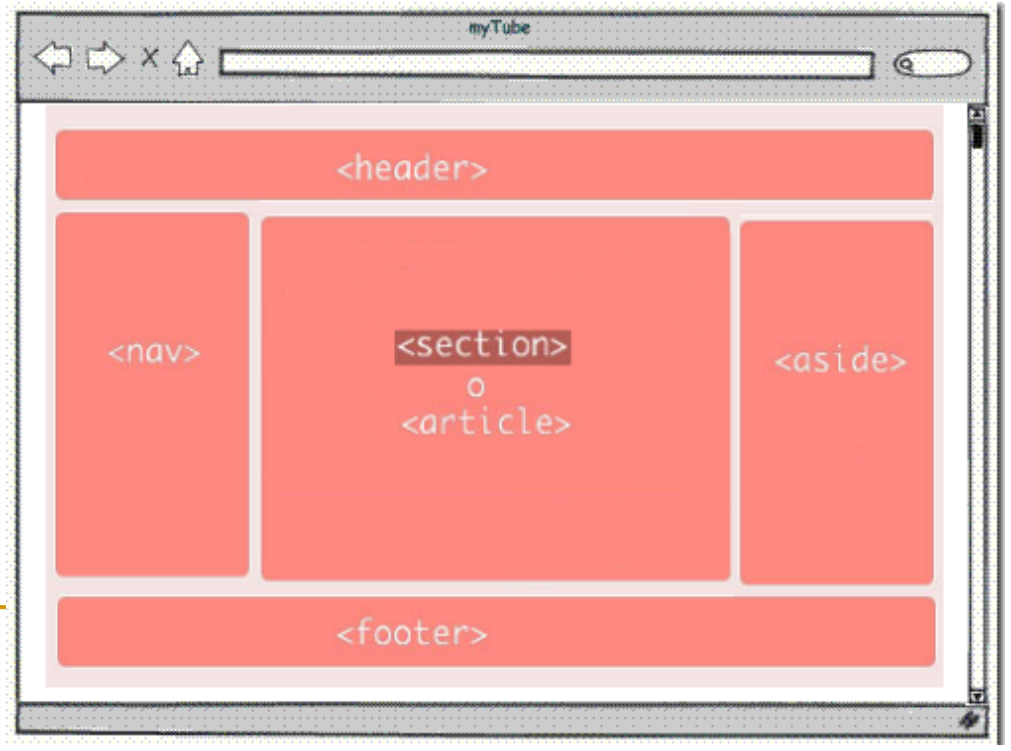
Se basa en la utilización de estas 4 propiedades:

- Poner siempre el ancho de las capas como porcentaje respecto al área que las contiene:
  Emplearemos la propiedad CSS `width:90%`
- Establecer en píxeles los anchos máximo y mínimo que cada capa va a permitir:
  Emplearemos las propiedades CSS `max-width:1200px; min-width:300px;`
- Emplear posicionamiento de capas de tipo ***block*** o de tipo ***inline-block*** en función de que porción ocupe cada capa respecto al área que la contiene:
  Emplearemos propiedades `display:block;` o bien `display:inline-block;`

Fase 0:

Diseñar el Layout de la página

- Número de capas
- Disposición: a 1 columnas,
  a 2 columnas,
  a 3 columnas …
- % que ocupa cada una de ellas dentro del área que las contiene
- Podemos usar herramientas de "wireframes" / "mock-ups" o directamente papel y lápiz.



6. Posicionando con "inline-block" (2 de 5)

Fase 1:
Maquetación HTML

- Pasar del diseño estático a la codificación HTML de la página.

- Establecemos los contenedores adecuado para cada sección: los propios del HTML5, y si es necesario DIV's (o TABLES)

- Creamos los vínculos a las páginas CSS externas, que es donde damos estilo y posicionamos las capas

```
<!DOCTYPE HTML>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <link rel="stylesheet" href="css/maquetar.css" type="text/css" />
    <link rel="stylesheet" href="css/estilos.css" type="text/css" />
</head>
<body>
    <header>
        <img id="imglogo" src="img/logo.jpg" width="100%">
    </header>
    <section id="tabular">
        <nav id="menuizquierda">
            <ul>
                <li><a href="pag1.html">Pagina 1</a></li>
                <li><a href="pag2.html">Pagina 2</a></li>
                    ...
            </ul>
        </nav>
        <main id="panelcentral">
            <p>Lorem ipsum ...</p>
        </main>
        <aside id="panelderecha">
            <p>Lorem ipsum ...</p>
        </aside>
    </section>
    <footer>
        <p>Copyrigh 2013</p>
    </footer>
</body>
</html>
```

6. Posicionando con "inline-block" (3 de 5)

Fase 2: Posicionamiento con CSS "inline-block"

Paso 1: Inicializar parámetros del modelo de cajas. Ponemos borde a las cajas temporalmente

```
/* Reseteamos los margin y padding de todas las cajas.*/
* {
   margin:0;
   padding:0;
   border: 1px dotted yellow; /* NO OLVIDAR BORRARLA */
}
```

Paso 2: Establecer parámetros del contenedor principal (<body>). Su % tamaño respecto al total, los márgenes de autocentrado, y establecemos su color y/o patrón de fondo:

```
/* html y body son capas contenedoras de todo. Vamos a prepararlas */
:root{ background-image:url(../img/patron.png);} /* Si lo deseamos */

body {
   width: 90%;          /* Diseño autoadaptable al 90%  peeero...*/
   max-width: 1000px;   /* Tamaño máximo aunque haga pantalla mayor*/
   min-width: 200px;    /* Tamaño mínimo aunque haga pantalla menor*/
   margin: 0 auto;      /* Truco para que siempre esté centrada */
   background-color:#CCF;
}
```

6. Posicionando con “inline-block” (4 de 5)

Paso 3: Posicionar las tres capas que ocupan filas enteras al 100%. Utilizamos **`display: block`**

- No hace falta especificarles el ancho, ya que las cajas tipo “block” ocupan todo el ancho disponible.

```
header {
    display: block;
    text-align: center;
    background-color:#999;
}
```

- La **section** de **id=“principal”** es el contenedor de las capas **menuizq**, **panelcentral**, **panelderecha**. Podríamos no haberla utilizado, pero en muchos casos es muy útil, por ejemplo, si deseamos establecer un color y/o patrón de fondo común a las 3 capas**:**

```
section#tabular {
    display: block; /* Este no es necesario */
    background-color:#1EE;
}
```

- El pié de página **footer** es similar al **header**, es también una caja tipo “block”.

```
footer {
    display: block;
    background-color:#99F;
}
```

6. Posicionando con "inline-block" (5 de 5)

Paso 4: Posicionar las capas que comparten fila. Utilizamos `display: inline-block`

- En este caso hay que especificar el `%` que ocupa cada capa respecto al 100% que sería todo el ancho
- Acotamos también el **ancho mínimo**, así como el **ancho máximo** para las capas laterales:

```
nav#menuizquierda {
    display:inline-block;
    width: 24%;
    min-width: 200px;
    max-width: 350px;
    vertical-align: top;
}
```

```
aside#panelderecha {
    display:inline-block;
    width: 24%;
    min-width: 200px;
    max-width: 350px;
    vertical-align: top;
}
```

- Normalmente la capa central es la más importante, por lo que a ésta no le acotamos el ancho máximo, la dejamos crecer todo lo que pueda. Podemos acotarle, si deseamos, un **alto** mínimo.

```
main#panelcentral {
    display:inline-block;
    width: 50%;
    min-width: 200px;
    min-height: 100px;
    background-color: white;
}
```

Nota: *la suma de los porcentajes de los anchos de todas las capas que compartan fila, debe sumar algo menos del 100%*

TRUCOS para poder sumar 100%:

1) **margin-right: -4px;**

2) ***font-size: 0;*** a su capa padre

7. Posicionando con "Flex Box" de CSS3

Posicionamiento con Flex Box de CSS3

Paso 1: Inicializar parámetros del modelo de cajas.
Paso 2: Establecer parámetros del contenedor principal (<body>).
Paso 3: Posicionar las capas que ocupan filas enteras al 100%

Igual que en ejemplo anterior

Paso 3.1: La capa que **contiene** a las otras 3 que comparten fila debe ser `display: flex`

```
div#tabular {
        display: flex;
        flex-flow: row;
        background-color: white;
}
```

Paso 4: Posicionar las capas que comparten fila.

```
div#tabular nav{
    flex: 25%; //ó px ó número
    …
}
```

```
div#tabular main{
    flex: 50%;
    …
}
```

```
div#tabular aside{
    flex: 25%;
    …
}
```

- Las combinaciones de flex son muchas… sobre todo tenemos comodidad:

`flex: 1;` ‖ `flex: 2;` ‖ `flex: 1;` => Obtendríamos lo mismo que antes

`flex: 100px;` ‖ `flex: 70%;` ‖ `flex: 30%;` => 1ª ancho fijo, 2ª y 3ª reparten el resto

- Podemos usar más propiedades de flex: `align-ítems, justify-content, order, align-self` …
- Podemos acotar también los **ancho mínimo** y **máximo** para capas que queramos:

```
min-width: 200px;
max-width: 350px;
```

8. Múltiples columnas con “column-count”

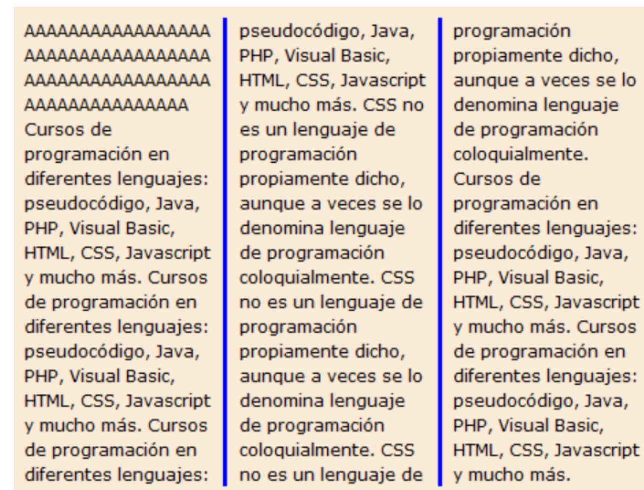
- **`column-count:`** *numero* CSS3 tiene una propiedad para formatear por columnas.

Además:

- Con **`column-gap:`** *medida*; podemos establecer la separación entre las columnas.
- Con **`column-rule:`** *formato*; podemos dibujar una barra de separación entre ellas.
- Podemos hacer que un elemento se expanda por más de una columna con **`column-span:`** *número*;

```
<html>
<head>
 <style type="text/css">
      .a3columnas{
         column-count: 3;
         column-gap: 20px;
         column-rule: 3px solid blue; }
</style>
</head>

<body>
    <div class=“a3columnas">
        AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA Cursos de programación en
diferentes lenguajes: pseudocódigo, Java, PHP, Visual
Basic, HTML, CSS, Javascript y mucho más. Cursos de
programación en diferentes lenguajes: pseudocódigo,
Java, PHP, Visual Basic, HTML, CSS, Javascript y mucho
más. Cursos de programación en diferentes lenguajes:
pseudocódigo, Java, PHP, Visual Basic, HTML, CSS,
Javascript y mucho más.
    </div>
</body>
```



9. Capa sobre capa con z-index

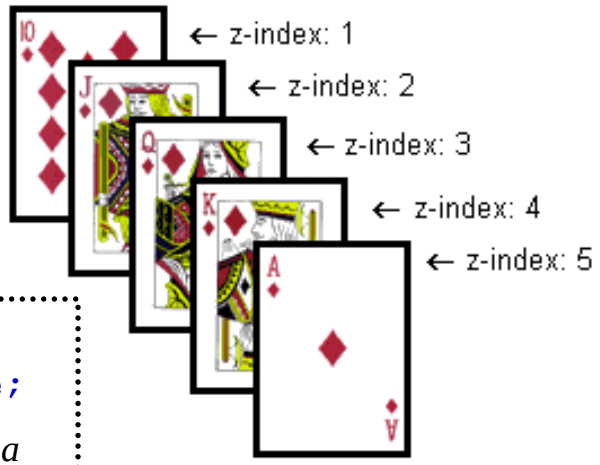
- CSS funciona sobre tres dimensiones: altura, anchura y profundidad.
- A parte de utilizar coordenadas **left** y **top**, tenemos también la posibilidad de asignar PROFUNDIDAD a cada una de las capas mediante la propiedad **z-index**.
- El elemento con **z-index** mayor, se superpone al elemento **z-index** menor.
- **z-index** solo funciona si **position no es static**
- Nos permite múltiples posibilidades de maquetación: texto sobre imagen, etc.

```
img#diez {
    position: absolute;
    left: 100px;
    top: 100px;
    z-index: 1;
}
img#jota {
    position: absolute;
    left: 115px;
    top: 115px;
    z-index: 2;
}
```

```
img#reina {
    position: absolute;
    left: 130px;
    top: 130px;
    z-index: 3;
}
```

etc, etc

Nota: *a partir de CSS2.1,* **z-index** *ya puede tomar también valores negativos.*



Nota: *Una buena práctica para usar* **z-index** *es utilizar* **position: absolute;**

Recordar, que si usamos **absolute**, *la capa que tomemos como referencia para posicionar debe tener* **position: relative;**