Tema 6

Hojas de estilo CSS (II)

Objetivos

- Aplicar las características de las hojas de estilo en los documentos HTML.
- Conocer las técnicas básicas del diseño visual de páginas y sitios Web.
- Reconocer las ventajas que aporta la utilización del modelo de cajas.
- Conocer las características avanzadas de las hojas de estilo y aplicarlas al diseño visual.
- Reconocer las ventajas que aporta la utilización de elementos de diseño visual accesible y adaptativo *(responsive)* a las características de cada dispositivo.

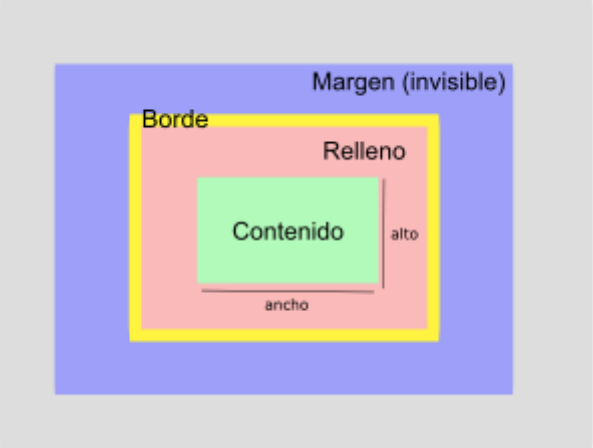
Introducción

En el tema anterior se estudiaron las características más básicas de las hojas de estilo en cascada. El uso de las hojas de estilo permite mejorar la apariencia de la presentación de los distintos elementos que componen un documento HTML. En este tema se aborda el estudio de aspectos de diseño visual utilizando para ello las características avanzadas de las hojas de estilo en cascada. El diseño visual de las páginas y sitios Web tiene su importancia en un entorno de información multimedia, como es el servicio Web. La utilización de las técnicas de diseño visual, que se estudian en este tema, permite al desarrollador de páginas y sitio Web obtener resultados que utilizan la potencialidad de las técnicas de desarrollo Web. El estudio de las hojas de estilo en cascada se ha dividido en varios temas debido a su amplitud, facilitándose de este modo su aprendizaje de una forma más detallada y ordenada. Estos temas se han planificado de forma consecutiva dentro de la periodificación de contenidos del módulo profesional.

Índice

6.1 MODELO DE CAJAS

El diseño visual de páginas Web se basa en la utilización de las hojas de estilo y, más concretamente, en el denominado modelo de cajas. El modelo de cajas es un modelo de formato que se basa en que cada elemento HTML produce una caja o zona rectangular a su alrededor. Así, cualquier elemento HTML (tabla, fila, párrafo, división, etc.) es en realidad una caja con la siguiente estructura:



Así, cada caja tiene un **contenido** que puede ser un texto en el caso de un párrafo, una imagen, el contenido de una celda, otras capas, etc. Alrededor del contenido, hay un espacio de **relleno** *(padding)*, que es un espacio entre el contenido y el borde. El **borde** *(border)*, que puede estar visible o no, corresponde con una línea que encierra completamente el contenido y su relleno. Y, por último, el **margen** *(margin)*, que es el espacio libre existente entre la caja y aquello que le rodea, es decir, las otras cajas adyacentes. La apariencia de la presentación visual de cada una de las partes que constituyen una caja puede establecerse mediante propiedades de estilo:

- Relleno

Propiedad	Descripción	Valores
padding-top **padding-right** **padding-bottom** **padding-left**	Ancho del relleno superior, derecho, inferior e izquierdo	[<longitud> | <porcentaje> | auto]
padding	Tamaños para rellenos	[<longitud> | <porcentaje> | auto] {1,4}

- Márgenes

Propiedad	Descripción	Valores
margin-top **margin-right** **margin-bottom** **margin-left**	Tamaño del margen superior, derecho, inferior e izquierdo	[<longitud> | <porcentaje> | auto]
margin	Ancho para márgenes	[<longitud> | <porcentaje> | auto]{1,4}

- Bordes

Propiedad	Descripción	Valores
border-top-width **border-right-width** **border-bottom-width** **border-left-width**	Anchura del borde superior, derecho, inferior o izquierdo	[thin | **medium** | thick | <longitud>]
border-width	Anchos de varios bordes individuales	[thin | medium | thick | < longitud>] {1,4}
border-top-color **border-right-color** **border-bottom-color** **border-left-color**	Color del borde superior, derecho, inferior o izquierdo	[<color> | transparent]
border-color	Colores de varios bordes individuales	[<color> | transparent] {1,4}
border-top-style **border-right-style** **border-bottom-style** **border-left-style**	Estilo del borde superior, derecho, inferior o izquierdo	[**none** | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset]
border-style	Estilos de varios bordes individuales	[none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset] {1,4}
border-top **border-right** **border-bottom** **border-left**	Ancho, estilo y el color para el borde superior, derecho, inferior o izquierdo	[<border-top-width> || <border-top-style> || <border-top-color>]
border	Ancho, el estilo y el color para los 4 bordes	[<border-top-width> || <border-top-style> || <border-top-color>]

- Contenido

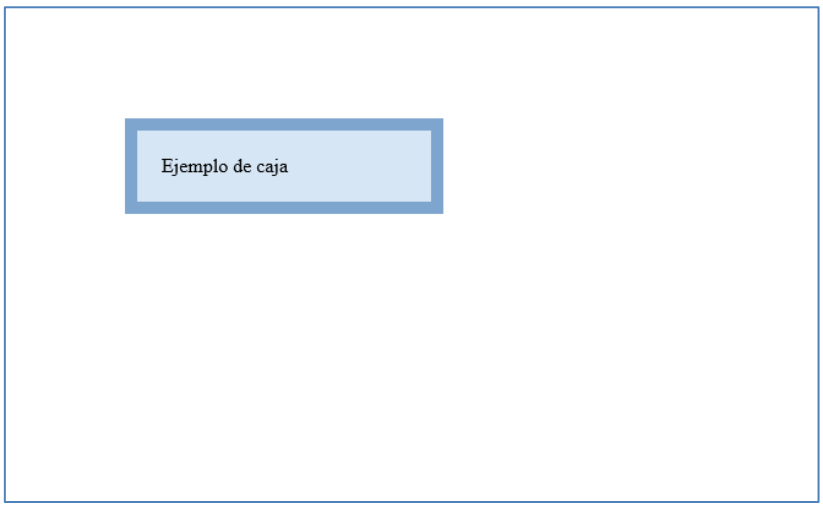
Propiedad	Descripción	Valores
width	Ancho	[<longitud> | <porcentaje> | **auto**]
min-width	Ancho mínimo	[<longitud> | <porcentaje>]
max-width	Ancho máximo	[<longitud> | <porcentaje> | **none**]
height	Alto	[<longitud> | <porcentaje> | **auto**]
min-height	Alto mínimo	[<longitud> | <porcentaje>]
max-height	Alto máximo	[<longitud> | <porcentaje> | **none**]
line-height	Altura entre las bases del texto	[**normal** | <número> | <longitud> | <porcentaje>]
vertical-align	Alineación vertical del texto	[**baseline** | sub | super | top | text-top | middle | bottom | text-bottom | <porcentaje> | <longitud>]
text-align	Alineación horizontal del texto	[left | right | center | justify]

En las tablas anteriores, se incluyen algunas de las propiedades de estilo empleadas para definir el estilo de la presentación cuando se emplea el modelo de cajas, puede accederse a una referencia más amplia en: http://www.w3c.es/Divulgacion/GuiasReferencia/CSS21/#modeloCajas. Al definir las dimensiones de la caja, hay que tener siempre en cuenta que las propiedades **width** y **heigh** definen el ancho y el alto del contenido, sin relleno, sin márgenes y sin bores. La dimensión total de la caja se calcula sumando las dimensiones del contenido, del relleno, del borde y del margen.

Una caja suele recibir otros nombres como son: contenedor, bloque, capa, etc. Según el organismo W3C, una caja es una zona rectangular constituida por un contenido, un relleno, un borde y un margen que puede ser tratada de forma independiente, como un elemento único. De manera que puede ser alterada de diversas formas, por ejemplo: puede posicionarse o incluso moverse en la ventana de visualización de la página, puede especificarse una imagen o un color para el fondo, se puede escribir encima o debajo de ella, puede hacerse transparente u opaca, etc. Por ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS - Ejemplo de caja</title>
    <meta charset="utf-8" />
    <style type="text/css">
        p {
            width: 200px;
            border: 10px solid rgb(125,165,205);
            padding: 20px;
            margin: 100px;
            background-color: rgb(215,230,245);
        }
    </style>
</head>
<body>
    <p> Ejemplo de caja </p>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador:



Estructura y composición de páginas Web

La utilización del modelo de cajas ofrece múltiples ventajas. La principal ventaja es que permite establecer la estructura visual de las páginas Web. En la actualidad, el modelo de cajas es muy ampliamente utilizado y constituye una característica fundamental del diseño visual de las páginas y de los sitios Web. Este modelo de formato visual se utiliza para definir la composición visual de los elementos HTML que forman una página Web. Para ello, se suele fraccionar la página en secciones utilizando divisiones, mediante etiquetas . A cada etiqueta se le asigna un identificador, de manera que es posible referirse a una división concreta desde una hoja de estilo y así, poder aplicarle las características de estilo específicas que sean apropiadas en cada caso: el color de fondo, el estilo de los bordes, el tipo de letra, la posición en la página, etc. Por ejemplo:

El resultado de la presentación de la página Web anterior se ha obtenido estructurando adecuadamente los contenidos mediante el uso de divisiones. En el código HTML correspondiente, que se muestra a continuación, puede apreciarse el uso que se hace de las divisiones: *contenedor*, *cabecera*, *menu*, *contenido* y *pie* para estructurar los contenidos sobre la página Web.

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS - Página personal</title>
    <meta charset="utf-8" />
```

```
    <link rel="stylesheet" type="text/css" href="./PaginaPersonalEstilo.css">
</head>
<body>
    <div id="contenedor">
        <div id="cabecera">
            <h2>MI <span class="orange">P</span>ÁGINA</h2>
            <h3><span class="green">P</span>ERSONAL</h3>
        </div>
        <div id="menu">
            <a class="orange" href="">Inicio</a> <br />
            <a class="orange" href="">Aceca de mí</a> <br />
            <a class="orange" href="">Curriculum</a> <br />
            <a class="orange" href="">Contactar</a> <br />
            <div class="noticias">
                <h2 class="grey">Noticias</h2>
                <span class="green">01.02.2009</span>
                <p class="grey">
                    Lorem ipsum dolor sit amet igitur lex in amicitia. Praesent dignissim,
                    lectus pharetra blandit cursus.
                    <a class="orange" href="">learn more...</a>
                </p>
            </div>
        </div>
        <div id="contenido">
            <img src="image4.jpg" alt="Mi viaje" title="MY VOYAGE" id="imagen" />
            <h2>ALGUNAS PALABRAS SOBRE MI</h2>
            <h4 class="green">Lorem ipsum amet</h4>
            <p>Praesent dignissim, lectus pharetra blandit cursus, urna nibh sagittis
               nulla, et pellentesque lorem sapien vel.</p>
            <p>Praesent dignissim, lectus pharetra blandit cursus, urna nibh sagittis
               nulla, et <a class="orange underline" href="">pellentesque lorem</a>.</p>
            <p>Praesent dignissim, nibh lectus pharetra blandit cursus, urna nibh sagittis
               nibh nulla.</p>
        </div>
        <div id="pie">
            <p class="copyright">Copyright &#169; 2016 TU NOMBRE</p>
        </div>
    </div>
</body>
</html>
```

En el archivo de hoja de estilo externa CSS que se utiliza en el ejemplo anterior, que se muestra a continuación, puede apreciarse la aplicación de las propiedades de estilo correspondientes.

```
body{
    background: rgb(0,0,0);
    color: #a9a9a9;
    font-family: Verdana,Arial,sans-serif;
    font-size: 11px;
    font-weight: 300;
    line-height: 15px;
}
#contenedor {
    width: 80%;
    margin-top: 5px;
    margin-left: auto;
    margin-right: auto;
    padding: 30px;
}
```

```
#cabecera {
    font-size: 15px;
    letter-spacing: 12px;
}
#menu {
    float: left;
    width: 15%;
    height: 400px;
    margin: 5px;
    margin-top: 20px;
}
#contenido {
    float: right;
    width: 75%;
    text-align: justify;
    margin: 20px;
}
#pie {
    clear: both;
    float: none;
    padding: 5px;
    font-size: 0.9em;
    border-top: solid 1px;
    text-align: center;
}
.orange {
    color: #f47b33;
}
.green {
    color: #74be37;
}
.grey {
    color: #898989;
    font-style: italic;
}
.noticias {
    text-align: left;
    font-size: 0.8em;
    line-height: 1.5em;
    margin-top: 120px;
}
a { text-decoration: none; }
a:hover { color: #fff; text-decoration: none; }
#imagen {
    width: 100%;
    max-width: 611px;
    max-height:400px;
    border-bottom: solid 2px;
}
```

Definir la estructura visual de una página Web es una tarea compleja, es necesario seguir una serie de pasos. El primer paso es planificar la estructura de la página, para ello suele realizarse un boceto del resultado final que se pretende conseguir. A partir del boceto, se escribe el código HTML de la página, utilizando etiquetas para dividir los elementos de contenido en partes. El tercer paso, consiste en añadir los contenidos (información y otros elementos HTML) a cada una de las divisiones. Finalmente, se crea una hoja de estilo para establecer el posicionamiento cada una de las divisiones y definir el formato visual de los elementos de la página y poder alcanzar el resultado deseado.

En los apartados siguientes se estudian las propiedades de posicionamiento y de control de la visualización de las cajas, que permiten definir la estructura o composición visual de los elementos de las páginas y sitios Web. En el ámbito del diseño visual, la acción de definir la composición o estructura visual de una página Web suele recibir el nombre de maquetación.

6.2 POSICIONAMIENTO

Por regla general, el posicionamiento exacto de los elementos de HTML en una página Web es desaconsejable. Ello es debido a las características de visualización diferentes que presentan los diversos tipos de dispositivos que, en la actualidad, pueden emplearse para visitar las páginas y los sitios Web. Los contenidos que presenta cualquier página Web deberían poder verse de una manera adecuada, independientemente de los dispositivos empleados para su consulta: monitor de ordenador de sobremesa, monitor de ordenador portátil, Tablet, Smartphone, etc. En la actualidad, las hojas de estilo en cascada CSS proporcionan diversas herramientas para posicionar, de manera adecuada en cada caso, cualquier elemento en su documento HTML. En general, para posicionar los elementos HTML en una página suele recurrirse a crear un "puzzle" de elementos anidados en divisiones. Las propiedades de posicionamiento se incluyen en la especificación CSS2.

Las hojas de estilo CSS permiten utilizar cinco métodos distintos para establecer el posicionamiento de una caja en la ventana de visualización del navegador: estático, relativo, absoluto, fijo y flotante. Las propiedades de estilo que pueden intervenir para definir el posicionamiento de elementos son las siguientes: position, top, bottom, right, left, float y clear.

Propiedad	Descripción	Valores
position	Forma o esquema de posicionamiento de la caja	[**static** | relative | absolute | fixed]
top **right** **bottom** **left**	Desplazamiento vertical y horizontal de la caja (respecto al límite superior, derecho, inferior o izquierdo del contenedor)	[<longitud> | <porcentaje> | **auto**]
float	Posicionamiento flotante	[left | right | **none**]
clear	Control de cajas adyacentes con posicionamiento flotante	[none | left | right | both]

Posicionamiento estático

Es el valor por defecto establecido para establecer el posicionamiento de todos los elementos HTML en la página. Los elementos posicionados de forma estática aparecen en pantalla según el flujo normal de representación de HTML y, por tanto, no se ven afectados por las propiedades top, bottom, left y right. Además, el elemento afectado no puede posicionarse ni reposicionarse y su visibilidad no puede modificarse. El posicionamiento estático se especifica mediante la propiedad:

```
position: static;
```

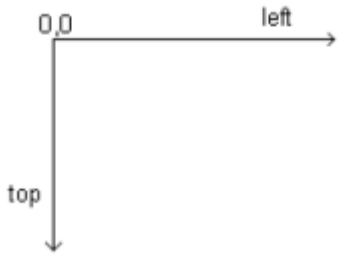
Posicionamiento relativo

Consiste en desplazar la caja un cierto valor en pixels o en % respecto a su posición normal o estática. Por tanto, el elemento permanece en el flujo de los datos, aunque está, en una cierta medida, desplazado en relación con lo que sería su posición estática o normal. El posicionamiento relativo actúa juntamente con las propiedades: top, bottom, right y left. La posición relativa se especifica mediante las coordenadas (x,y) donde:

- **x**, es la distancia respecto al borde izquierdo del elemento padre o de la ventana del navegador (eje horizontal). De este modo, la propiedad left determina la distancia entre la parte izquierda del elemento y el borde izquierdo de la página y, right la distancia entre la parte derecha del elemento y el borde derecho de la página.

- **y**, es la distancia respecto al borde superior del elemento padre o de la ventana del navegador (eje vertical). De este modo, la propiedad top determina la distancia entre el borde superior del elemento y el borde superior de la página y, bottom la distancia entre el borde inferior del elemento y el borde inferior de la página.

La posición de coordenadas (0,0) corresponde con la esquina superior izquierda del elemento contenedor, es decir, del elemento padre, o bien, de la ventana de visualización del navegador:



El posicionamiento relativo se especifica mediante la propiedad:

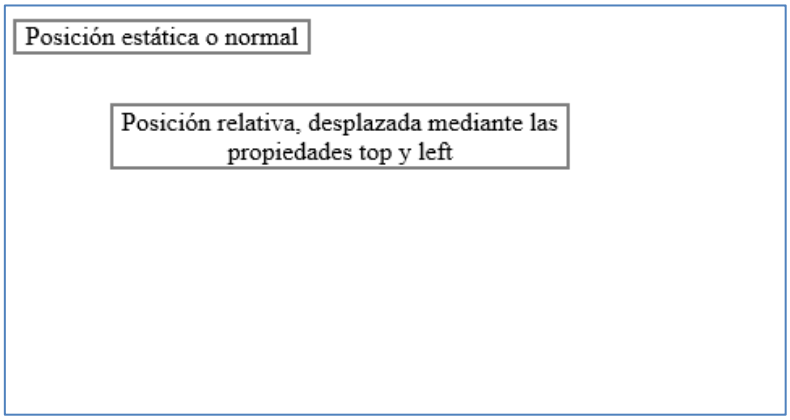
```
position: relative;
```

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS - Posicionamiento Estático y Relativo</title>
    <meta charset="utf-8" />
    <style type="text/css">
        .normal {
            width: 180px;
            border: 2px solid gray;
            text-align: center;
        }
        .relativa {
            position: relative;
            top: 30px;
```

```
            left: 60px;
            width: 280px;
            border: 2px solid gray;
            text-align: center;
        }
    </style>
</head>
<body>
    <div class="normal">
        Posición estática o normal
    </div>
    <div class="relativa">
        Posición relativa, desplazada mediante las propiedades top y left
    </div>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador:



Posicionamiento absoluto

La posición de la caja vendrá dada por las coordenadas que se definan. Así, la caja no estará dentro del flujo estático o normal del documento, por lo que se podrá solapar o superponer con otros elementos de la página. El resto de los elementos se comportan como si no existieran los elementos posicionados de forma absoluta. El elemento contenedor o padre del elemento que se posiciona de forma absoluta debe estar posicionado, aunque sea con un posicionamiento relativo. El posicionamiento absoluto actúa juntamente con las propiedades: top, bottom, right y left.

El posicionamiento absoluto se especifica mediante la propiedad:

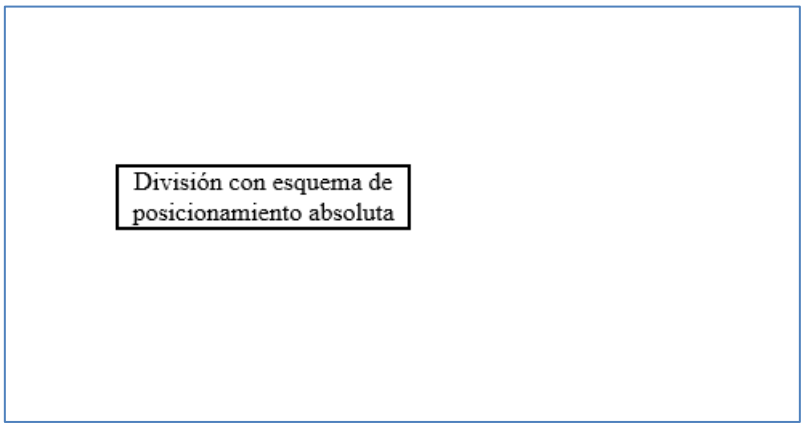
```
position: absolute;
```

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>Posicionamiento absoluto</title>
    <meta charset="utf-8" />
```

```
    <style type="text/css">
        .absoluta {
            position: absolute;
            left: 75px;
            top: 100px;
            width: 180px;
            border: 2px solid black;
            text-align: center;
        }
    </style>
</head>
<body>
    <div class="absoluta">
        División con esquema de posicionamiento absoluta
    </div>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador:



Cuando se utiliza el posicionamiento absoluto se corre el riesgo de que no se muestre la página correctamente si se emplean resoluciones de pantalla diferentes a la utilizada durante el tiempo de diseño. Ello es debido a que los elementos que se hayan posicionado de forma absoluta pueden superponerse sobre los elementos que se posicionan siguiendo el flujo normal de la página. Dado que, en la actualidad, se emplean multitud de dispositivos para utilizar el servicio de información Web, se recomienda no utilizar el posicionamiento absoluto. Conviene, además, recordar que los navegadores Web agregan, de manera predeterminada, un margen al cuerpo de la página y que este margen varía de un navegador a otro. Por ello, es prudente, en caso de usar posicionamiento absoluto, especificar los márgenes de la etiqueta <body>.

Posicionamiento fijo

Implica un posicionamiento absoluto, pero manteniendo la posición de la caja incluso cuando el usuario desplaza la página mediante las barras de desplazamiento. De esta manera, la caja estará siempre visible. Resulta útil para encabezados o pies de páginas, así como para los índices de las páginas que se mantendrán inalterables en un documento extenso. El posicionamiento fijo actúa juntamente con las propiedades: top, bottom, right y left.

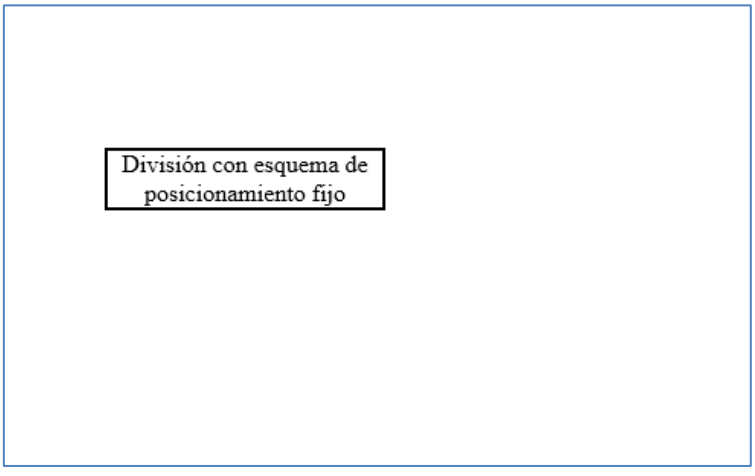
El posicionamiento fijo se especifica mediante la propiedad:

```
position: fixed;
```

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>Posicionamiento fijo</title>
       <meta charset="utf-8" />
    <style type="text/css">
        .fija {
            position: fixed;
            left: 75px;
            top: 100px;
            width: 180px;
            border: 2px solid black;
            text-align: center;
        }
    </style>
</head>
<body>
    <div class="fija">
        División con esquema de posicionamiento fijo
    </div>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador:



Posicionamiento flotante

El posicionamiento flotante es muy utilizado en la actualidad. Desplaza horizontalmente la caja todo lo que sea posible hacia la izquierda o hacia la derecha dentro contenedor en el que se encuentra. El resto de las cajas ocupan el lugar dejado por la caja flotante. Los elementos que hubiera antes del elemento float no se ven afectados. Utiliza las propiedades float y clear.

La propiedad **float** permite situar una caja lo más a la derecha o lo más a la izquierda posible en la misma línea, dentro de su contenedor. Esta propiedad puede adoptar los siguientes valores:

- right. Alinea a la derecha el elemento indicado, empujando a los demás elementos hacia la izquierda.
- left. Alinea a la izquierda el elemento indicado, empujando a los demás elementos hacia la derecha.
- none. No especifica nada y deja la gestión al navegador para situar la caja según el flujo normal o estático de presentación del documento.

Con el posicionamiento flotante, no se produce superposición o solapamiento de cajas porque la propiedad float tiene en cuenta el resto de las cajas flotantes. Una caja flotante debe tener definida su anchura, de manera implícita o explícita, ya que de lo contrario será tratada como no flotante.

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>Posicionamiento flotante</title>
       <meta charset="utf-8" />
    <style type="text/css">
        .derecha {
            float: right;
        }
        .izquierda {
            float: left;
        }
    </style>
</head>
<body>
    <div>
        <div class="derecha">
            <img src="coche.png">
        </div>
        <p>
            Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
            non risus. Suspendisse lectus tortor, dignissim sit amet,
            adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
            diam. Maecenas ligula massa, varius a, semper congue, euismod non,
            mi.
        </p>
    </div>
</body>
</html>
```

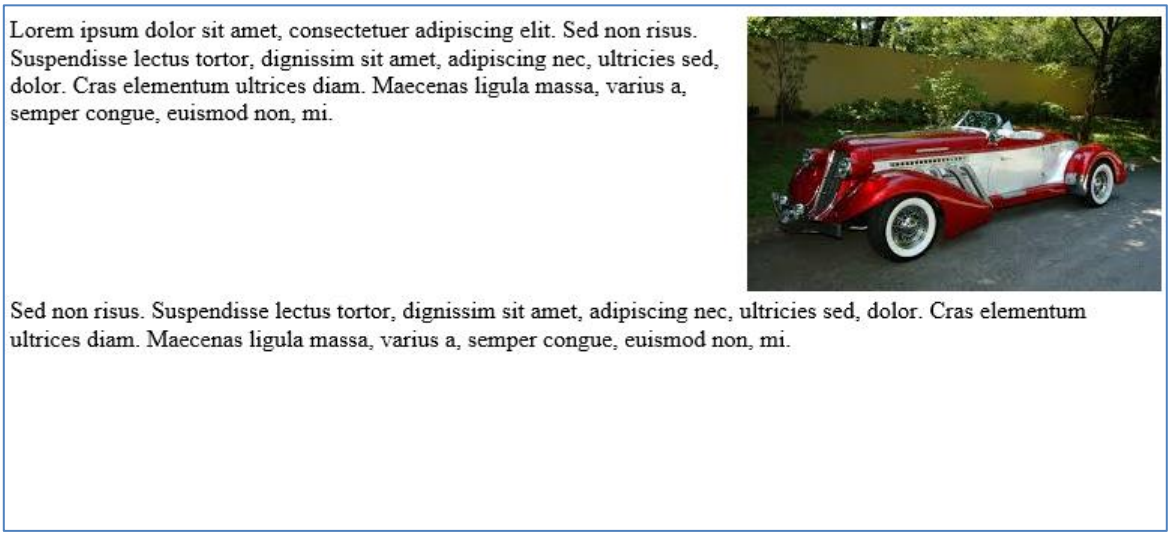
En el código anterior puede apreciarse que el uso de la declaración float: right; fuerza a la imagen a posicionarse en la parte derecha del contenedor. Además, el texto del párrafo se distribuye a la izquierda. Este mismo ejemplo puede modificarse para aplicar a la imagen la clase de estilo denominada izquierda. El ejemplo anterior, produce el siguiente resultado en el navegador.

La propiedad **clear** anula el efecto introducido por la propiedad float y afecta a los elementos que hubiera detrás del elemento "float". Se utiliza juntamente con la propiedad float para indicar si un elemento flotante permite otros elementos flotantes junto a él. El valor asignado establece el lado que no debe ser adyacente a ninguna caja flotante. Así, el valor right anula los elementos flotantes a la derecha. El valor left anula los elementos flotantes a la izquierda. Y, el valor both anula los elementos flotantes de ambos lados. Por ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>Posicionamiento flotante. Uso de la propiedad clear</title>
       <meta charset="utf-8" />
    <style type="text/css">
        .derecha {
            float: right;
        }
        .izquierda {
            float: left;
        }
        .despejar {
            clear: right;
        }
    </style>
</head>
<body>
    <div>
        <div class="derecha">
            <img src="coche.png">
        </div>
        <p>
            Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
            non risus. Suspendisse lectus tortor, dignissim sit amet,
            adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
            diam. Maecenas ligula massa, varius a, semper congue, euismod non,
            mi.
        </p>
        <p class="despejar">
            Sed non risus. Suspendisse lectus tortor, dignissim sit amet,
            adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
```

```
            diam. Maecenas ligula massa, varius a, semper congue, euismod
            non, mi.
        </p>
    </div>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador.



Si en el código HTML del ejemplo anterior, se eliminara la asignación de la clase de estilo *despejar* al último párrafo, el resultado en la ventana de visualización del navegador sería el siguiente.

Se recomienda realizar ejercicios para poder comprobar el funcionamiento de los distintos tipos de posicionamiento de las cajas y, de esa manera, tener la oportunidad de experimentar y poder comprender su comportamiento.

6.3 CONTROL DE LA VISUALIZACIÓN

Las propiedades de visualización permiten controlar la presentación de las cajas en la ventana del navegador. Estas propiedades de estilo pueden controlar efectos visuales como: la superposición, el desbordamiento, el recorte, la visibilidad y la visualización de las cajas. Las propiedades correspondientes a los efectos visuales relacionadas con el control de la visualización de cajas son, respectivamente, las siguientes: z-index, overflow, clip, visibility y display, respectivamente.

Superposición

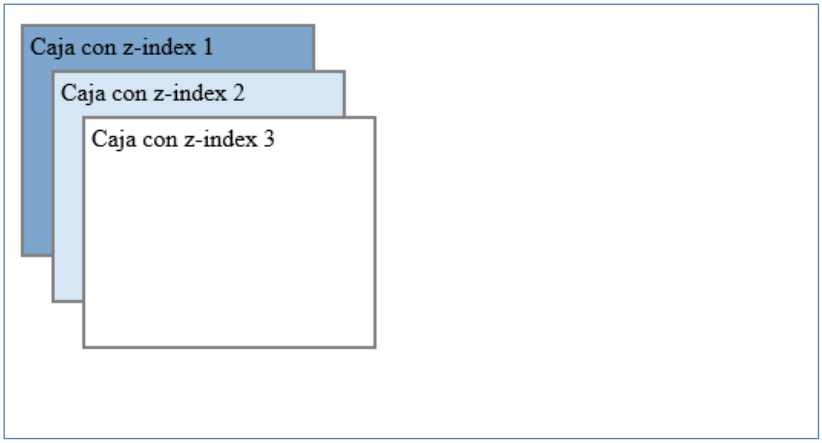
En el caso de que alguna caja se superponga o solape con otra, se puede determinar cuál estará delante utilizando la propiedad z-index. El valor de esta propiedad toma un valor entero para cada caja. La caja a la cual se le asigne un valor mayor será la que aparecerá por delante de las demás en la ventana de visualización. La propiedad z-index solo funciona con un posicionamiento absoluto.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Superposición (z-index)</title>
    <meta charset="utf-8" />
    <style type="text/css">
        .cajas {
            position: absolute;
            width: 180px;
            height: 140px;
            padding: 4px;
            border: 2px solid gray;
        }
        .cajal {
            left: 20px;
            top: 20px;
            background-color: rgb(125,165,205);
            z-index: 1;
        }

        .caja2 {
            left: 40px;
            top: 50px;
            background-color: rgb(215,230,245);
            z-index: 2;
        }

        .caja3 {
            left: 60px;
            top: 80px;
            background-color: white;
            z-index : 3;
        }
    </style>
</head>
<body>
    <div class="cajas cajal">
        Caja con z-index  1
    </div>
    <div class="cajas caja2">
```

```
        Caja con z-index 2
    </div>
    <div class="cajas caja3">
        Caja con z-index 3
    </div>
</body>
</html>
```

En el código anterior, puede apreciarse el uso que se hace de la propiedad z-index. Además, también puede apreciarse el uso del atributo *type* en la etiqueta <style>. El atributo type especifica el tipo de medio, anteriormente conocido como tipo MIME, que afecta al contenido de una etiqueta. Así, la etiqueta <style type=”text/css”> identifica que el contenido entre las etiquetas <style> y </ style> es código escrito mediante lenguaje de estilo CSS. En la especificación HTML5, el valor por defecto del atributo type de la etiqueta <style> es "text/css", por lo que ya no es necesario definirlo. Por otra parte, y como recordatorio, también puede apreciarse el uso que se hace del atributo lang de la etiqueta <html>. Este atributo especifica el idioma de los valores de los atributos y del texto contenido en un elemento. Así, la etiqueta <html lang=”es”> especifica que el idioma de los contenidos de los elementos de la página y de los valores de sus atributos es español. El objetivo del atributo lang es permitir a los navegadores representar el contenido de la página de forma más significativa, según la práctica cultural aceptada para un idioma dado. La funcionalidad que aporta este atributo depende de las prestaciones y configuración del navegador utilizado por el usuario.

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador.



Desbordamiento

La propiedad overflow determina las acciones que debe hacer el navegador cuando un elemento es más grande que su contenedor. Ha de recordarse que el contenedor de un elemento es el elemento padre que lo contiene. Los posibles valores de esta propiedad son:

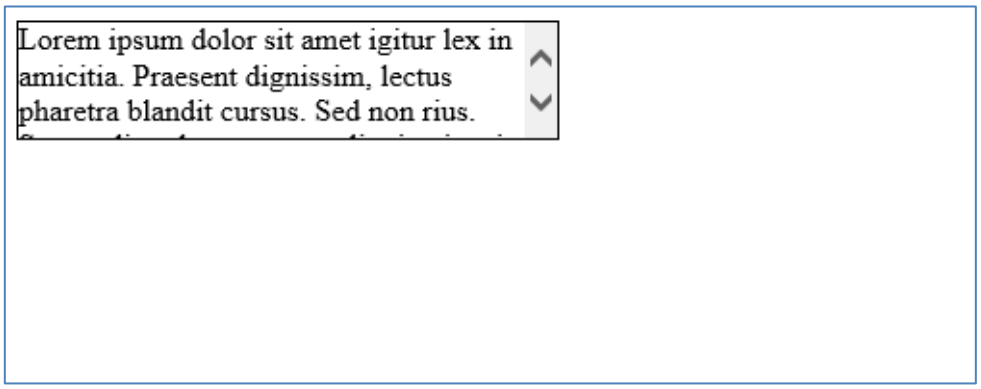
- hidden. Oculta la parte que se desborda, sin posibilidad de visualizar o acceder a ella.
- scroll. Oculta la parte que se desborda, aunque queda accesible mediante las barras de desplazamiento.

- visible. La parte que se desborda se muestra, ignorando la especificación del elemento padre que lo contiene.

- auto. Se deja la gestión del desbordamiento al navegador.

```
<!DOCTYPE html>
<html>
<head>
    <title>Desbordamiento (overflow)</title>
       <meta charset="utf-8" />
    <style type="text/css">
        .caja {
            width: 280px;
            height: 60px;
            border: 1px solid black;
            overflow: auto;
        }
    </style>
</head>
<body>
    <div class="caja">
        Lorem ipsum dolor sit amet igitur lex in amicitia. Praesent dignissim,
        lectus pharetra blandit cursus. Sed non rius. Suspendiese lectus tortor,
        digniessim sit amet, adipiscing nec, utricies sed, dolor. Gras elementum
        ultrices diam.
    </div>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador.

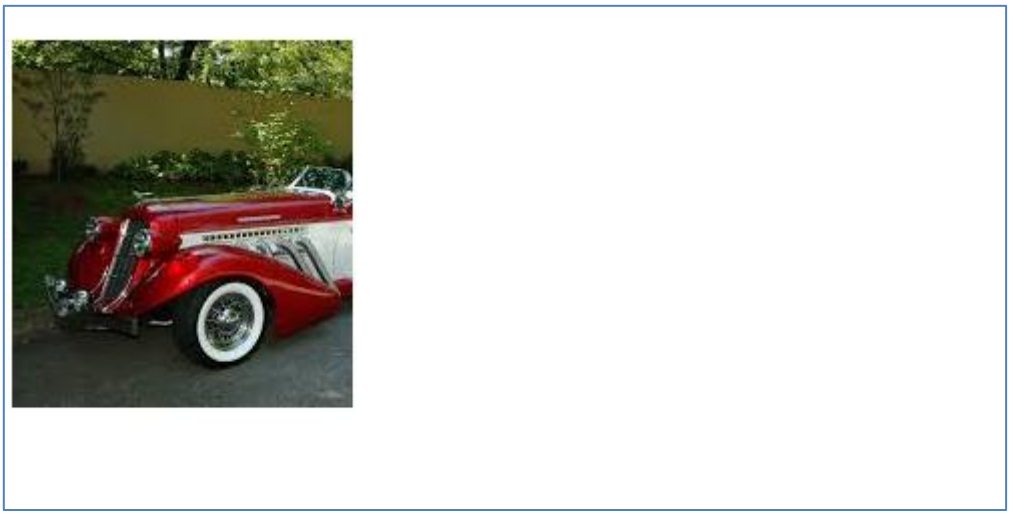


Recorte

La propiedad clip determina la parte visible de un elemento, generalmente una imagen. De modo que la imagen entera se incluirá en el documento, aunque solo una parte rectangular será visible. Se recomiendo incluir el elemento que se desea recortar dentro de una etiqueta . Esta propiedad solo funciona con un posicionamiento absoluto. Las coordenadas del rectángulo recortado vienen dadas por las esquinas superior-izquierda, superior-derecha, inferior-derecha e inferior-izquierda respecto de la imagen inicial.

```
<!DOCTYPE html>
<html>
<head>
    <title>Recorte (clip)</title>
       <meta charset="utf-8" />
    <style>
        .recorte {
            position: absolute;
            clip: rect(0px 170px 215px 0px);
        }
    </style>
</head>
<body>
    <div class="recorte">
        <p></p>
        <img src="coche.png" />
    </div>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador.



Visibilidad

La propiedad visibility determina si un elemento está visible u oculto. El valor visible muestra el elemento, mientras que el valor hidden oculta el elemento.

```
<!DOCTYPE html>
<html>
<head>
    <title>Visibilidad (visibility)</title>
       <meta charset="utf-8" />
    <style>
        .oculto {
            visibility: hidden;
        }
    </style>
</head>
```

```
<body>
    <p>
        Lorem ipsum dolor sit amet igitur lex in amicitia. Praesent dignissim,
        lectus pharetra blandit cursus.
    </p>
    <div class="oculto">
        <img src="coche.png" />
    </div>
    <p>
        Lorem ipsum dolor sit amet igitur lex in amicitia. Praesent dignissim,
        lectus pharetra blandit cursus. Sed non rius.
    </p>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador.

Lorem ipsum dolor sit amet igitur lex in amicitia. Praesent dignissim, lectus pharetra blandit cursus.

Lorem ipsum dolor sit amet igitur lex in amicitia. Praesent dignissim, lectus pharetra blandit cursus. Sed non rius.

Si en el código del ejemplo anterior se elimina la asignación a la clase de estilo oculto en la etiqueta correspondiente, se producirá el siguiente resultado.

Lorem ipsum dolor sit amet igitur lex in amicitia. Praesent dignissim, lectus pharetra blandit cursus.

Lorem ipsum dolor sit amet igitur lex in amicitia. Praesent dignissim, lectus pharetra blandit cursus. Sed non rius.

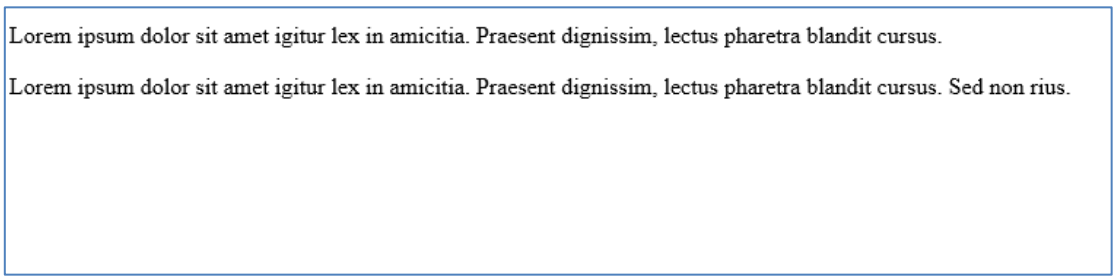
Visualización

La propiedad de estilo más característica que puede aplicarse para controlar la visualización de los elementos en la página es la propiedad **display**. Se trata de una propiedad muy potente cuyo uso está ampliamente extendido. Permite controlar gran diversidad de características de presentación de los elementos. Los valores más representativos de la propiedad display son: none, inline, block, run-in, list-item, inline-block, table, inline-table, table-row-group, table-row, table-column-group, table-column, table-cell y table-caption. Algunos de estos valores se estudian a continuación.

El valor **none** de la propiedad display, define un elemento que no se va a mostrar. El elemento afectado se retira del documento y de la visualización de la página en el navegador. Por ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>Visualización (display) - none</title>
       <meta charset="utf-8" />
    <style>
        .nomostrar {
            display: none;
        }
    </style>
</head>
<body>
    <p>
        Lorem ipsum dolor sit amet igitur lex in amicitia. Praesent dignissim,
        lectus pharetra blandit cursus.
    </p>
    <div class="nomostrar">
        <img src="coche.png" />
    </div>
    <p>
        Lorem ipsum dolor sit amet igitur lex in amicitia. Praesent dignissim,
        lectus pharetra blandit cursus. Sed non rius.
    </p>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador.

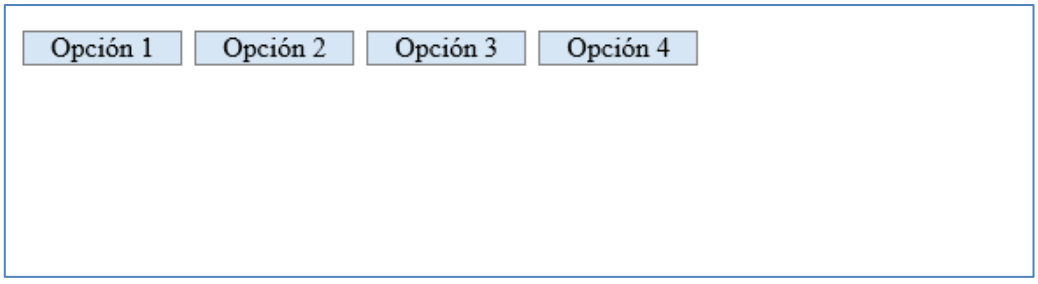


En el ejemplo anterior, puede apreciarse la diferencia existente respecto al uso de la declaración visibility: hidden; estudiada anteriormente. Además, puede apreciarse el efecto que produciría la eliminación de la asignación a la clase de estilo nomostrar en la etiqueta correspondiente.

Ya se han estudiado, los conceptos de elemento de bloque y elemento de línea. Los elementos de bloque crean bloques de contenido vertical en el contexto visual, generalmente usando saltos de línea antes y después de su contenido. Los elementos de línea se presentan horizontalmente, unos detrás de otros. La propiedad de estilo display permite redefinir un elemento de línea como un elemento de bloque y a la inversa. Para ello, se utilizan los valores **inline** y **block** de la propiedad de estilo display. Por ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>Visualización (display) - block/inline</title>
       <meta charset="utf-8" />
    <style>
        ul {
            list-style: none;
            padding: 5px;
        }
        li {
            display: inline;
            border: 1px solid gray;
            background-color: rgb(215,230,245);
            text-align:center;
            margin-left: 3px;
            margin-top: 6px;
            padding-right: 15px;
            padding-left: 15px;
            width: 80px;
        }
    </style>
</head>
<body>
    <ul>
        <li>Opción 1</li>
        <li>Opción 2</li>
        <li>Opción 3</li>
        <li>Opción 4</li>
    </ul>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador.



Los elementos de las listas son elementos de bloque, por lo que se presentan unos debajo de otros. En el ejemplo anterior, se presentan en una única línea horizontal al aplicar la declaración de estilo display: inline;. Puede comprobarse el efecto que se produciría al eliminar esta declaración.

El uso del elemento <table> de HTML presenta algunos problemas en la construcción de páginas Web. Estos problemas están causados por su falta de adaptación a la construcción de páginas Web que cumplan con las normas básicas de Accesibilidad Web y que se adapten a las características de visualización de algunos tipos de dispositivos. Por estos motivos, suele emplearse la propiedad de estilo display para crear tablas en páginas Web. Las tablas creadas con la propiedad display permiten presentar información en formato tabular de una forma que es compatible con las normas de Accesibilidad Web actuales y que queda adaptada a las características específicas de visualización de diversos tipos de dispositivos. Por tanto, la creación de tablas mediante la propiedad de estilo display permite crear tablas en páginas Web accesibles y adaptativas *(responsive)*.

Se utilizan varios valores de la propiedad de estilo display para construir y mostrar una tabla, empleado para ello divisiones. Algunos de estos valores son los siguientes: **table, inline-table, table-row, table-row-group, table-column, table-column-group, table-cell y table-caption**.

- table. Sirve para crear una tabla que ocupa el ancho de su contenedor.
- Inline-table. Sirve para diseñar una tabla de tipo línea.
- table-row. Crea una fila de la tabla. Hace que cualquier elemento se muestre como una fila de una tabla.
- table-row-group. Reagrupa una o varias filas de la tabla.
- table-colum. Muestra una columna de la tabla.
- table-colum-group. Reagrupa una o varias columnas de la tabla.
- table-cell. Crea y muestra una celda de la tabla.
- table-caption. Crea y muestra la leyenda de la tabla. Por defecto, aparece en la parte superior de la tabla, pero su posición se puede modificar con la propiedad caption-side.

A continuación, se incluye un ejemplo de código para la presentación de información en formato tabular empleando la propiedad de estilo display.

```
<!DOCTYPE html>
<html>
<head>
    <title>Visualización - Display Table 1</title>
    <meta charset="utf-8" />
    <style>
        .contenedor{
            padding-top: 20px;
            font-family: Arial;
        }
        .tabla{
            display: table;
            width: 50%;
            margin-left: auto;
            margin-right: auto;
```

```
            border: 1px solid;
        }
        .fila{
            display: table-row;
        }
        .celda{
            display: table-cell;
            padding: 5px;
        }
        .titulo{
            display: table-caption;
            text-align: center;
            padding-bottom: 5px;
        }
        .agruparcolumnas{
            display: table-column-group;
        }
    </style>
</head>
<body>
    <div class="contenedor">
        <div class="tabla">
            <div class="titulo">CLIENTES</div>
            <div class="fila" style="background:#00ffff">
                <div class="celda">Nombre</div>
                <div class="celda">Dirección</div>
                <div class="celda">Ciudad</div>
                <div class="celda">Telefono</div>
            </div>
            <div class="fila">
                <div class="celda">Francisco Campos LLanda</div>
                <div class="celda">C/ La Rana, 4</div>
                <div class="celda">Almansa (Albacete)</div>
                <div class="celda">678 33 22 11</div>
            </div>
            <div class="fila">
                <div class="celda">María Fernandéz García</div>
                <div class="celda">C/ Mayor, 32</div>
                <div class="celda">Aspe (Alicante)</div>
                <div class="celda">643 77 88 99</div>
            </div>
            <div class="fila">
                <div class="celda">Jordi Moll Torrens</div>
                <div class="celda">C/ Baix, 2</div>
                <div class="celda">Xaló (Alacant)</div>
                <div class="celda">677 78 78 00</div>
            </div>
        </div>
    </div>
</body>
</html>
```

El ejemplo anterior produce el siguiente resultado de representación de información en formato tabular en la ventana de visualización del navegador.

CLIENTES

Nombre	Dirección	Ciudad	Telefono
Francisco Campos LLanda	C/ La Rana, 4	Almansa (Albacete)	678 33 22 11
María Fernandéz García	C/ Mayor, 32	Aspe (Alicante)	643 77 88 99
Jordi Moll Torrens	C/ Baix, 2	Xaló (Alacant)	677 78 78 00

La construcción de tablas mediante la propiedad de estilo display también se suele aplicar para resolver la composición o estructura visual y el posicionamiento de elementos complejos, como, por ejemplo, los controles y elementos que forman un formulario. Por ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
       <meta charset="utf-8" />
    <style>
        .cuerpo {
            display: table;
            margin-left: auto;
            margin-right: auto;
            padding: 10px;
            width: 60%;
        }
        .linea {
            display: table-row;
            height: 25px;
        }
        .textos {
            display: table-cell;
            text-align: right;
            width: 40%;
            padding-right: 10px;
        }
        .controles {
            display: table-cell;
            text-align: left;
            width: 50%;
        }
        .boton {
            text-align: center;
        }
    </style>
</head>
<body>
    <div style="text-align: center;">
        <h2>DATOS DE PRODUCTOS</h2>
    </div>
    <form id="form1" method="post" action="http://server/maneja.aspx">
        <div class="cuerpo">
```

```
            <div class="linea">
                <div class="textos">
                    Id. de Producto
                </div>
                <div class="controles">
                    <input type="text" name="Codigo" size="10"/>
                </div>
            </div>
            <div class="linea">
                <div class="textos">
                    Descripción
                </div>
                <div class="controles">
                    <input type="text" name="Descripcion" size="40" />
                </div>
            </div>
            <div class="linea">
                <div class="textos">
                    Precio
                </div>
                <div class="controles">
                    <input type="text" name="Precio" />
                </div>
            </div>
        </div>
        <div class="boton">
            <br />
            <input type="submit" name="Codigo" value="Enviar" />
        </div>
    </form>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador.

La composición y el posicionamiento de los controles y elementos que forman un formulario también suele resolverse empleando un posicionamiento flotante, mediante el uso de la propiedad de estilo float. El siguiente código produce una presentación idéntica a la del ejemplo anterior.

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
       <meta charset="utf-8" />
    <style>
        .cuerpo {
            display: table;
            margin-left: auto;
            margin-right: auto;
            padding: 10px;
            width: 60%;
        }
        .linea
        {
            height: 25px;
        }
        .textos
        {
            text-align: right;
            width: 40%;
            position: relative;
            float: left;
        }
        .controles
        {
            text-align: left;
            width: 58%;
            float: right;
        }
        .final
        {
            clear: both;
        }
        .boton {
            text-align: center;
        }
    </style>
</head>
<body>
    <div style="text-align: center;">
        <h2>DATOS DE PRODUCTOS</h2>
    </div>
    <form id="form1" method="post" action="http://server/maneja.aspx">
        <div class="cuerpo">
            <div class="linea">
                <div class="textos">
                    Id. de Producto
                </div>
                <div class="controles">
                    <input type="text" name="Codigo" size="10"/>
                </div>
            </div>
            <div class="linea">
                <div class="textos">
```

```
                    Descripción
                </div>
                <div class="controles">
                    <input type="text" name="Descripcion" size="40" />
                </div>
            </div>
            <div class="linea">
                <div class="textos">
                    Precio
                </div>
                <div class="controles">
                    <input type="text" name="Precio" />
                </div>
            </div>
        </div>
        <div class="final">

        </div>
        <div class="boton">
            <br />
            <input type="submit" name="Codigo" value="Enviar" />
        </div>
    </form>
</body>
</html>
```

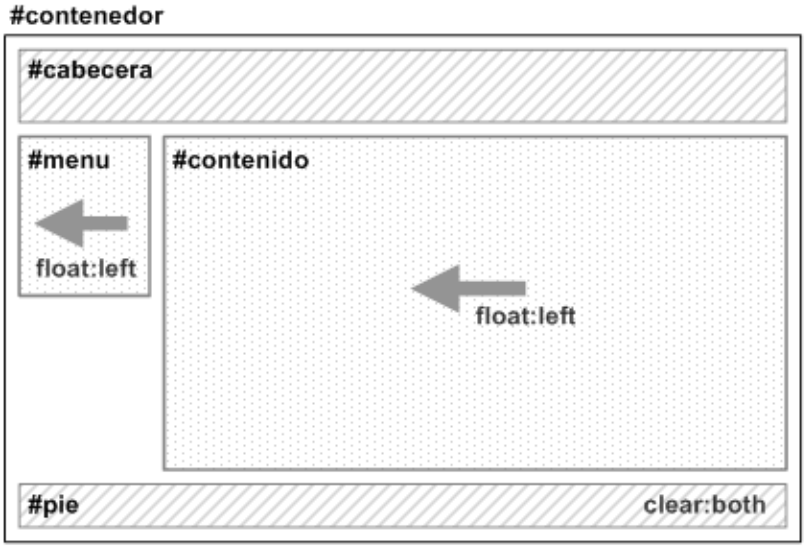
6.4 ESTRUCTURA Y LAYOUT

Se utiliza el término layout para referirse a la composición o estructura visual general de una página Web. La forma habitual de crear el layout de una página es utilizando divisiones, mediante el uso de la etiqueta . La acción de definir el layout de una página Web se conoce como maquetación.

Se pueden clasificar los diseños del layout de las páginas Web según diversos criterios: el número de columnas, la posición de los elementos de navegación, el ancho fijo o variable de los elementos, etc. Se utiliza el término **layout fluido o líquido** (*fluid layout* o *liquid layout*) para referirse a aquel tipo de layout que utiliza un ancho variable, mediante valores de porcentaje, para definir el ancho de las divisiones que lo componen. Así, una página web posee un diseño fluido cuando su tamaño se ajusta a la dimensión horizontal de la pantalla de forma automática, sin que haya necesidad de utilizar una barra de desplazamiento horizontal *(scroll)*. De esta forma, el diseño se expande al ancho disponible de la ventana de visualización o de la pantalla porque el tamaño de las divisiones que lo forman (elementos) es un porcentaje del total disponible (100%). El objetivo de este tipo de diseño de layout es conseguir que se produzca un ajuste automático del contenido de la página Web a las medidas y resolución del dispositivo, de la misma forma que un líquido se adapta a la forma del recipiente o contenedor donde se encuentra. El problema surge en pantallas muy grandes con resoluciones altas, porque suelen aparecer grandes espacios en blanco. Y también suelen aparecer problemas de visualización en pantallas pequeñas con resoluciones bajas inferiores a 1.024 px de ancho, por ejemplo, en *tablets* en orientación vertical *(portrait)* y en *smartphones* en orientación horizontal *(landscape)*, porque las imágenes se miniaturizan y los textos se vuelven ilegibles. Para evitar estos problemas se suele fijar un ancho mínimo a las divisiones que componen el layout, lo que hace que aparezcan las barras de desplazamiento horizontal cuando sea necesario.

Diseño fluido a dos columnas con cabecera y pie de página

Para obtener un diseño de layout fluido a dos columnas con cabecera y pie de página, tal como se muestra siguiente figura, se utiliza la propiedad **float**. Esta propiedad se aplicará a los elementos posicionados, como el elemento menú y el elemento contenido. Además, se aplicará la propiedad clear al elemento pie, para evitar los solapamientos ocasionados por los elementos posicionados con float. Y se aplicará una anchura relativa en % a los elementos de la página para que se adapten o fluyan sobre la ventana de visualización del navegador para diversos tipos de resoluciones.



Se utilizarán, por tanto, como mínimo las siguientes declaraciones de estilo CSS:

```
#menu {
    float: left;
    width: 15%;
}
#contenido {
    float: left;
    width: 85%;
}
#pie {
    clear: both;
}
```

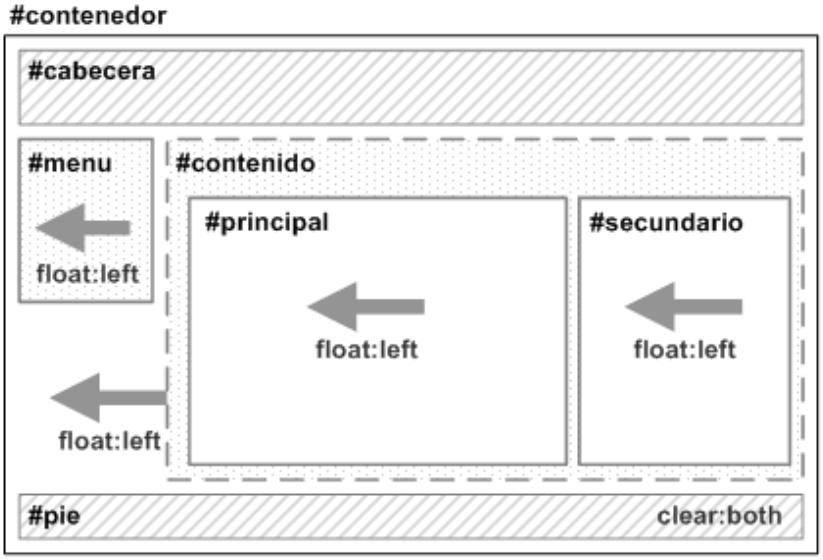
Las anteriores declaraciones se aplicarán a los elementos siguientes en el código HTML:

```
<body>
    <div id="contenedor">
        <div id="cabecera">
        </div>

        <div id="menu">
        </div>

        <div id="contenido">
```

```
        </div>

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

Diseño fluido a tres columnas con cabecera y pie de página

En la siguiente ilustración se muestra un caso de layout a 3 columnas con cabecera y pie de página.



El código CSS mínimo para definir la estructura de la página, sin aplicar ningún estilo adicional, es:

```
#menu {
  float: left;
  width: 15%;
}
#contenido {
  float: left;
  width: 85%;
}
#contenido #principal {
  float: left;
  width: 80%;
}
#contenido #secundario {
  float: left;
  width: 20%;
}

#pie {
  clear: both;
}
```

Y el código HTML para definir la estructura de la página es el siguiente:

```
<body>
    <div id="contenedor">
        <div id="cabecera">
        </div>

        <div id="menu">
        </div>

        <div id="contenido">
            <div id="principal">
            </div>

            <div id="secundario">
            </div>
        </div>

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

Ejemplo. A continuación, se incluye un ejemplo de código completo para obtener un diseño fluido de una página Web con layout a dos columnas con cabecera y pie de página.

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
       <meta charset="utf-8" />
    <style>
        body {
            font-family: Arial;
        }
        a:link {
            color:#069;
            text-decoration: none;
        }
        a:hover {
            text-decoration: underline;
        }
        #contendor {
            width: 100%;
            margin: 0px;
            padding: 0;
            min-width: 500px;
            overflow: auto;
        }
        #menu {
            float: left;
            width: 18%;
            margin: 1%;
            background-color: #ddd;
        }
        #contenido {
            float: right;
            width: 77%;
            margin: 1%;
```

```
            overflow: hidden;
            text-align: justify;
        }
        #pie {
            font-size: 80%;
            color: #aaa;
            background-color: #eee;
            height: 100px;
            overflow: auto;
            border-top: 1px solid #000;
            padding-top: 5px;
            text-align: center;
            clear: both;
        }
        ul {
            list-style: none;
            margin: 0px;
            padding-left: 5px;
        }
        #enlaces li{
            display: inline;
        }
        #enlaces a {
            text-align: center;
            margin: 1px;
            padding: 5px;
            width: 80px;
            font-weight: bold;
            background-color: #eee;
            border-top-right-radius: 8px;
        }
        #enlaces a:hover {
            color: white;
            font-weight: bold;
            background-color: #069;
            text-decoration: underline;
            border-top-right-radius: 8px;
        }
        #opciones li {
            border-top: 1px solid #ccc;
            font-size: 90%;
            margin-bottom: 5px;
        }
        #barra li {
            border-left: 5px solid rgb(192,192,192);
            font-size: 80%;
            padding-left: 15px;
            padding-top: 5px;
        }
    </style>
</head>
<body>
    <div id="cabecera">
        <h1>Lorem ipsum dolor sit amet</h1>
    </div>
    <div id="enlaces">
        <ul>
            <li><a href="#">Opción 1</a></li>
            <li><a href="#">Opción 2</a></li>
            <li><a href="#">Opción 3</a></li>
```

```
            <li><a href="#">Opción 4</a></li>
        </ul>
    </div>
    <hr />
    <div id="contendor">
        <div id="menu">
            <div id="opciones">
                <p>Opciones</p>
                <ul>
                    <li><a href="#">Opción 1</a></li>
                    <li><a href="#">Opción 2</a></li>
                    <li><a href="#">Opción 3</a></li>
                    <li><a href="#">Opción 4</a></li>
                </ul>
            </div>
            <br /> <br /> <br />
            <div id="barra">
                <p>Enlaces de interés</p>
                <ul>
                    <li><a href="#">Página 1</a></li>
                    <li><a href="#">Página 2</a></li>
                    <li><a href="#">Página 3</a></li>
                    <li><a href="#">Página 4</a></li>
                </ul>
            </div>
            <br />
        </div>
        <div id="contenido">
            <h3>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</h3>
            <p>
               Maecenas non tortor. Vestibulum ante ipsum primis in faucibus orci
               luctus et ultrices posuere cubilia Curae; Sed fermentum lorem a velit.
            </p>
            <p>
               Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Proin sodales,
               enim in volutpat vehicula, leo turpis vehicula magna, ut rutrum arcu lorem
               ac pede.
            </p>
            <p>
               Curabitur rutrum eros a risus. Cum sociis natoque penatibus et magnis dis
               parturient montes, nascetur ridiculus mus. In molestie suscipit libero.
               Cras sem. Nunc non tellus et urna mattis tempor. Nulla nec tellus a quam
               hendrerit venenatis. Suspendisse pellentesque odio et est. Morbi sed nisl
               sed dui consequat sodales. Donec porta porta ligula. Nam erat massa, blandit
               in, dapibus sit amet, vestibulum et, augue. Suspendisse ac risus. Duis
               semper fringilla sem. Praesent augue arcu, scelerisque nec, ornare
               malesuada, posuere a, neque. Nullam nulla nisi, ultrices quis, adipiscing
               non, varius ut, dui. Nulla viverra pellentesque sem.
            </p>
        </div>
        <div id="pie">
            Mi página Web - 2016
        </div>
    </div>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador.

El código del ejemplo anterior crea una página con anchura variable que se adapta a las dimensiones de la ventana de visualización del navegador. Puede probarse a restaurar la ventana del navegador y modificar su tamaño para comprobar su adaptación a las dimensiones. Para definir una página con anchuras fija, bastaría con sustituir las anchuras en porcentajes por anchuras en número de píxeles.

Diseño de layout con tablas

El diseño de layout de las páginas Web también puede resolverse mediante el uso de tablas, aplicando la propiedad de estilo **display** a los diversos elementos que la componen.

Tradicionalmente, los documentos en papel han utilizado las denominadas cuadrículas de diseño para situar con armonía los distintos bloques de lectura. Las cuadrículas están formadas por un conjunto de filas y columnas que permiten organizar el contenido y mejorar la comprensión del lector. El diseño del layout de páginas Web con la propiedad de estilo display y los valores table, table-row y table-cell, proporciona las siguientes ventajas a los diseñadores Web:

- La altura de todas las celdas de una fila es siempre la misma, lo que proporciona coherencia visual a la composición. Sin embargo, si se utilizan otros métodos de diseño de layout, entonces la altura de las divisiones de las filas depende del contenido que incluyan.

- La propiedad vertical-align gestiona perfectamente la alineación vertical sobre la celda.

- Pueden establecerse bordes en un solo lado de las celdas. O también, puede aplicarse estilos diferentes a cada uno de los bordes: superior, inferior, derecha e izquierda.

- Pueden utilizarse unidades distintas, fijas en pixeles y relativas en porcentaje, para establecer el ancho y alto de las celas.

- No se producen problemas de flotación de los elementos adyacentes, ya que todas las celdas, y por tanto todos los elementos de la composición, están el flujo normal de la página.

- Se genera un código limpio, semántico, ligero y claro.

Ejemplo. A continuación, se incluye un ejemplo de código completo para obtener un diseño de página Web mediante tabla con layout a tres columnas con cabecera y pie de página, empleando para ello la propiedad de estilo display y los valores: table, table-row y table-cell.

```
<!DOCTYPE html>
<html>
<head>
    <title>Estructura y Layout - Display Table 1</title>
       <meta charset="utf-8" />
    <style>
        #contenedor{
            width: 90%;
            margin: 10px auto;
            font-family: Arial;
        }
        #cabecera{
            padding-top: 30px;
        }
        #cuerpo{
            display: table;
            border-spacing: 5px;
            background: #ccc;
            border: 1px solid #333;
            margin-top: 20px;
        }
        #contenidos{
            display: table-row;
            background: #fcfcfc;
        }
```

```
        #secundario{
            width: 20%;
            display: table-cell;
            border: 1px solid #FFF;
            padding: .5em;
        }
        #principal{
            width: 60%;
            display: table-cell;
            border: 1px solid #FFF;
            padding: .5em;
        }
        #lateral{
            width: 20%;
            display: table-cell;
            border: 1px solid #FFF;
            padding: .5em;
        }
        #pie{
            padding-top: 50px;
            text-align: center;
        }
        ul {
            list-style: none;
            margin: 0px;
            padding-left: 5px;
            min-width: 500px;
        }
        li{
            display: inline;
        }
        #menu a {
            text-align: center;
            margin: 1px;
            padding: 5px;
            width: 90px;
            font-weight: bold;
            background-color: #ccc;
            text-decoration: none;
        }
        #menu a:hover {
            color: white;
            background-color: #069;
            text-decoration: none;
        }
    </style>
</head>
<body>
    <div id="contenedor">
        <div id="cabecera">
            <h1>LOREM IPSUM DOLOR</h1>
        </div>
        <div id="menu">
            <ul>
                <li><a href="#">Opción 1</a></li>
                <li><a href="#">Opción 2</a></li>
                <li><a href="#">Opción 3</a></li>
                <li><a href="#">Opción 4</a></li>
            </ul>
        </div>
```

```
        <div id="cuerpo">
            <div id="contenidos">

                <div id="secundario">
                    <h3>Curabitur rutrum eros a risus.</h3>
                    <p>
                       Cum sociis natoque penatibus et magnis dis parturient montes,
                       nascetur ridiculus mus. In molestie suscipit libero.
                    </p>
                    <p>
                       Cras sem. Nunc non tellus et urna mattis tempor. Nulla nec tellus a
                       quam hendrerit venenatis.
                    </p>
                    <p>Suspendisse pellentesque odio et est.</p>
                </div>

                <div id="principal">
                    <h3>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</h3>
                    <p>
                       Maecenas non tortor. Vestibulum ante ipsum primis in faucibus orci
                       luctus et ultrices posuere cubilia Curae; Sed fermentum lorem a
                       velit.
                    </p>
                    <p>
                       Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Proin
                       sodales, enim in volutpat vehicula, leo turpis vehicula magna, ut
                       rutrum arcu lorem ac pede.</p>
                    <p>
                       Curabitur rutrum eros a risus. Cum sociis natoque penatibus et
                       magnis dis parturient montes, nascetur ridiculus mus. In molestie
                       suscipit libero. Cras sem. Nunc non tellus et urna mattis tempor.
                       Nulla nec tellus a quam hendrerit venenatis. Suspendisse
                       pellentesque odio et est. Morbi sed nisl sed dui consequat sodales.
                       Donec porta porta ligula. Nam erat massa, blandit in, dapibus sit
                       amet, vestibulum et, augue. Suspendisse ac risus. Duis semper
                       fringilla sem. Praesent augue arcu, scelerisque nec, ornare
                       malesuada, posuere a, neque. Nullam nulla nisi, ultrices quis,
                       adipiscing non, varius ut, dui. Nulla viverra pellentesque sem.
                    </p>
                </div>

                <div id="lateral">
                    <p>
                       Nam erat massa, blandit in, dapibus sit amet, vestibulum et, augue.
                       Suspendisse ac risus.</p>
                    <img src="coche.png" />
                </div>

            </div>
        </div>
        <div id="pie">
            Mi página Web - 2016
        </div>
    </div>
</body>
</html>
```

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador.

6.5 BUENAS PRÁCTICAS

A continuación, se indican algunas recomendaciones o buenas prácticas en cuanto al uso de las hojas de estilo aplicadas al diseño visual de páginas Web:

- **Inicializar los valores de las propiedades de estilo**. Una hoja de estilo se aplica a las páginas Web a las que se ha asociado. Cuando una página no incluye la inicialización de los valores de las propiedades de estilo de manera expresa, el aspecto con el que se muestra en el navegador depende de los valores predeterminados de cada navegador.

- **Probar en los navegadores más utilizados**. Una práctica imprescindible de los diseñadores Web es probar su trabajo en varios navegadores distintos. De esta forma, el diseñador puede descubrir los errores propios de diseño y también las diferencias de visualización existentes en los distintos navegadores, debido a la hoja de estilos predeterminada del navegador.

- **Técnicas de depuración de errores**. Suelen cometerse errores al escribir código CSS para especificar las declaraciones de estilo. En primer lugar, cuando no se está seguro sobre si todas las reglas CSS están bien escritas, lo mejor es comprobar sintácticamente la hoja de estilos utilizando un validador de CSS, por ejemplo: http://jigsaw.w3.org/css-validator/. Una vez descartados los errores de sintaxis, el siguiente problema es detectar por qué una regla CSS no se aplica correctamente a un elemento. Una estrategia útil consiste en añadir alguna propiedad que sea visualmente significativa para comprobar si el selector es el correcto. Por ejemplo, aumentar mucho su tamaño de letra, cambiar el color del fondo, etc.

- **Utilizar el método de diseño de mejora progresiva**. La mejora progresiva propone que el diseño web se realice en dos pasos. En primer lugar, el diseño Web debe permitir el acceso completo y correcto a toda la información de la página, por lo que el diseño y la composición de la página incorpora solo las características más básicas. Y, en segundo lugar, después de cumplir el requisito anterior, se van añadiendo progresivamente las mejoras y características más avanzadas y específicas del diseño, al mismo tiempo que se van comprobando los resultados obtenidos en varios navegadores.

- **Tabular el código**. Esta técnica de escritura de código facilita significativamente su lectura, por lo que tabular el código CCS que especifica las propiedades y reglas de estilo es una de las prácticas recomendadas por los diseñadores Web profesionales. No es recomendable insertar tabuladores en el código, sino que es mejor insertar 2 o 4 espacios en blanco.

- **Mejora del rendimiento**. Diversas estrategias, que se enumeran a continuación, son útiles para mejorar el rendimiento de la aplicación de las hojas de estilo en cascada en las páginas Web: enlazar hojas de estilos externas en vez de incluir los estilos en la propia página, enlazar las hojas de estilos mediante <link> en vez de @import, reducir el número de archivos CSS externos de la página y combinar, si es posible, todos los archivos CSS externos individuales en un único archivo.

Glosario de términos

Accesibilidad Web. Un sitio web accesible se caracteriza por ser transformable, comprensible y navegable. El término transformable se aplica a que la información y los servicios del sitio Web deben ser accesibles para todas las personas y deben poder ser utilizados con todos los dispositivos de navegación. El término comprensible se refiere a que el sitio Web debe incluir contenidos claros y simples. El término navegable significa que el sitio Web debe incluir mecanismos sencillos de navegación.

Diseño Web *responsive* o adaptativo. El diseño web adaptativo, o adaptable, conocido por las siglas RWD del inglés, *Responsive Web Design*, es una filosofía de diseño y desarrollo de sitios Web cuyo objetivo es adaptar la apariencia de las páginas Web al dispositivo que se esté utilizando para visualizarlas. De manera que la presentación de la información sea independiente del dispositivo y, por tanto, la visualización de la información sea adecuada. Hoy día las páginas Web se presentan en multitud de dispositivos como ordenadores, tabletas, teléfonos móviles, libros electrónicos, ordenadores portátiles, etc. Además, aún dentro de cada tipo, cada dispositivo tiene sus características concretas: tamaño de pantalla, resolución, potencia de CPU, sistema operativo o capacidad de memoria entre otras. Esta tecnología pretende que, un único diseño Web se obtenga una visualización adecuada en cualquier dispositivo. El diseñador y autor norteamericano Ethan Marcotte creó y difundió esta técnica a partir de una serie de artículos en *A List Apart*, que es una publicación en línea especializada en diseño y desarrollo Web. Esta idea inicial la extendió en su libro *Responsive Web Design*, posteriormente. Actualmente, existe una variedad de *frameworks* de diseño Web que pueden ser utilizados para establecer las características de la presentación visual de la página incorporando diseño *responsive*. Uno de los *frameworks* más conocidos y empleados actualmente es Bootstrap.

Framework. Traducido al castellano como Entorno de trabajo o Marco de trabajo. Es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Landscape. Orientación horizontal de páginas. La página se presenta apaisada.

Portrait. Orientación vertical de páginas. La página gira por el lado más largo.