1.- Introducción a CSS.

Caso práctico

Carlos sale de la reunión muy intrigado pensando en todo lo que le contó Ada con respecto a las hojas de estilo en cascada. Así que, al llegar a casa, decide buscar información en la web sobre los beneficios de su utilización a la hora de mejorar la presentación de las páginas web.

Carlos aprovecha para ir confeccionándose un pequeño manual de CSS en el wiki de BK Programación que todos podrán ir editando y completando.

Las hojas de estilo en cascada (CSS - Cascading Style Sheets) son un estándar W3C que define la presentación de los documentos Web, es decir, el modo en el que se muestra un documento en pantalla o se suministra al usuario, ya sea por el monitor, en la pantalla del teléfono móvil o leído por un lector de pantalla. Lo más importante es que con CSS se mantienen las instrucciones de presentación separadas del contenido del documento XHTML.
Las hojas de estilo en cascada como hoy las conocemos, comenzaron cuando Håkon Lie publicó su primer borrador de hojas de estilo HTML en cascada, al que pronto se le unió Bert Bos, gran impulsor de este estándar.

- **CSS Nivel 2, Revisión 1** que es simplemente una recomendación que realiza unos ajustes menores a CSS2 basándose en la experiencia de trabajo con él entre 1998 y 2004.
- **CSS Nivel 3,** basada en módulos, añade soporte para texto flotante vertical, mejor manejo de tablas, idiomas internacionales y una mejor integración con otras tecnologías XML como SVG, Math ML y SMIL.

El CSS WG también está trabajando en conjuntos CSS especiales orientados a medios específicos como: CSS Mobile, CSS Print y CSS TV.

Para saber más

En el enlace siguiente puedes encontrar información actualizada sobre las actividades del W3C WG.

www.w3.org/Style/CSS

Los navegadores actuales implementan bastante bien CSS1 desde 1999 (tres años después de su lanzamiento) aunque, dependiendo de la marca y versión del navegador, hay algunas pequeñas diferencias de implementación. El primer navegador en dar soporte completo al CSS1 ha sido Internet Explorer 5.0 aunque hasta ese momento fue el Opera el que soportaba mejor CSS1. Después otros navegadores también lo han ido implementando.

Sin embargo, en los navegadores más recientes hay diferentes niveles de implementación de CSS2.1.

1.1.- Añadir estilos a un documento con CSS.

Sin duda, no existe ninguna desventaja por utilizar CSS en la maquetación de páginas web, son todo ventajas y, entre ellas, podemos destacar las siguientes:

- **Mayor control** en el diseño de las páginas: Se puede llegar a diseños fuera del alcance de HTML.
- **Menos trabajo**: Se puede cambiar el estilo de todo un sitio con la modificación de un único archivo.
- **Documentos más pequeños**: Las etiquetas `<font>` y la gran cantidad de tablas empleadas para dar una buena apariencia a los sitios web desaparecen ahora, por lo que se ahorra código en la configuración de la presentación del sitio.
- **Documentos mucho más estructurados**: Los documentos bien estructurados son accesibles a más dispositivos y usuarios.
- **El HTML de presentación está en vías de desaparecer**: Todos los elementos y atributos de presentación de las especificaciones HTMLy XHTML fueron declarados obsoletos por el W3C.
- **Tiene buen soporte**: En este momento, casi todos los navegadores soportan casi toda la especificación CSS1 y la mayoría también las recomendaciones de nivel 2 y 2.1.

Pero, ¿cómo funciona CSS? El proceso de funcionamiento de las hojas de estilo en cascada podemos resumirlo en tres pasos:

1. Hay que comenzar con un documento XHTML (o HTML). En teoría, el documento tendrá una estructura lógica y un significado semántico a través de los elementos XHTML adecuados. Con XHTML se crea **la estructura de la página web.**
2. Luego hay que **escribir las reglas de estilo para definir el aspecto ideal de todos los elementos.** Las reglas seleccionan el elemento en cuestión por su nombre y, a continuación, listan las propiedades (fuente, color, etc.) y los valores que se le van a aplicar.
3. Por último, hay que **vincular los estilos al documento**. Las reglas de estilo pueden reunirse en un documento independiente y aplicarse a todo el sitio, o pueden aparecer en la cabecera y aplicarse solo a ese documento.

Lo primero que deberías saber es que, las hojas de estilo consisten en una o más reglas que describen cómo debería mostrarse en pantalla un elemento.

A la hora de aplicar las reglas de estilo a un documento (X)HTML, debes tener en cuenta que existen tres modos distintos:

- Estilos en línea.
- Hojas de estilos incrustados.
- Hojas de estilos externas: vinculadas o importadas.

Debes conocer

En el enlace siguiente verás una presentación donde se verán las principales diferencias que existen entre los modos de aplicar las reglas de estilo y su sintaxis.

1.2.- Hojas de estilo externas.

En la presentación del apartado anterior ya has podido ver el modo de emplear las hojas de estilo externas: importándolas o enlazándolas. También has visto cómo se crea una regla de estilo y sus componentes: selector, propiedad y valor. Pero hay algunas cosas que nos quedan por comentar.

Las hojas de estilo son documentos de texto con, por lo menos, una regla. Estos archivos no contienen ninguna etiqueta HTML, ¿para qué?. Al igual que en los documentos HTML, en las hojas de estilo se pueden incluir comentarios pero, en este caso, se escriben del siguiente modo: **`/* Este es un comentario */`**

CSS2 introduce la posibilidad de orientar las hojas de estilo a medios de presentación específicos. Para ello se emplea el atributo **`media`** del elemento **`link`** del cual ya viste un ejemplo en la presentación del apartado anterior.

La siguiente tabla muestra los valores que puede tomar el atributo **`media`**:

Valores del atributo media.

Medios de presentación	Valor atributo media
`all`	Todos los medios definidos.
`braille`	Dispositivos táctiles que emplean el sistema Braille.
`embosed`	Impresoras que emplean el sistema Braille.
`handheld`	Dispositivos de mano: móviles, PDA, etcétera.
`print`	Impresoras y navegadores en el modo "vista previa para imprimir".
`projection`	Proyectores y dispositivos para presentaciones.
`screen`	Pantallas de ordenador.
`speech`	Sintetizadores para navegadores de voz empleados por personas discapacitadas.
`tty`	Dispositivos textuales limitados, como teletipos y terminales de texto.
`tv`	Televisores y dispositivos con resolución baja.

En el ejemplo siguiente se muestra cómo se pueden emplear en la función **`@media`** de la misma forma que hacíamos con la función **`@import`**.

Ejemplo:

```
<style type="text/css">
<!--
@import url(http://estilos/miestilo.css);
@media print {
body { font-size: 10pt; }     /* Establece el tamaño de fuente para impresión */
}
@media screen {
body { font-size: 13px }  /* Establece el tamaño de fuente para visualización */
}
p {font-face: Verdana;}
-->
</style>
```

1.3.- Conceptos clave de CSS.

Para que te puedas familiarizar con el comportamiento de CSS, es importante comprender una serie de conceptos clave.

Estructura y herencia.

Un documento (X)HTML tiene una estructura determinada que es equivalente a un árbol genealógico cuando se hace referencia a la relación entre elementos:

- Se dice que un elemento es "hijo" de otro si está contenido directamente en él y este último pasa a ser su "padre". Por ejemplo: el elemento `p` es hijo del elemento `body` y elelemento `body` es padre del elemento `p`.
- Los elementos que tienen el mismo padre son "hermanos". Por ejemplo: un elemento `p` puede ser hermano de otro elemento `p` si ambos son hijos directos del elemento `body`.

Controlar la relación padre-hijo es fundamental para el funcionamiento de CSS. Un hijo puede "heredar" valores de propiedad de su padre. Con una buena planificación, la herencia puede emplearse para hacer más eficiente la especificación de los estilos.

Este principio por el que algunas reglas se ignoran y otras se heredan nos introducen un concepto muy importante: "**la cascada**".

Reglas de estilo en conflicto: la "cascada".

La "cascada", de las hojas de estilo en cascada, se refiere a lo que ocurre si varias fuentes de información de estilo quieren dar formato al mismo elemento de una página. Cuando un navegador encuentra un elemento para el cual hay varias declaraciones de estilo, las ordena de acuerdo al origen de la hoja de estilo, la especificidad de los selectores y el orden de la regla para poder determinar cuál aplicar.

Origen de la hoja de estilo.

Los navegadores otorgan un peso distinto a las hojas de estilo que, ordenadas de menor a mayor peso, son:

- Hojas de estilo del navegador.
- Hojas de estilo del lector.
- Hojas de estilo de la persona que ha diseñado la página web.
- Declaraciones de estilo `!important` del lector.

Además de este orden, existe otra jerarquía de pesos que se aplican a las hojas de estilo creadas por la persona que ha diseñado la página web. Es importante entender esta jerarquía y tener en cuenta que las reglas de estilo que están al final de la lista ignorarán a las primeras. La siguiente lista, que como la anterior está ordenada de menor a mayor peso, muestra esta otra jerarquía:

- Hojas de estilo externas vinculadas (empleando el elemento `link` en la cabecera del documento).
- Hojas de estilo externas importadas (empleando el elemento `@import` dentro del elemento `style` en la cabecera del documento).
- Hojas de estilo incrustadas (empleando el elemento `style` en la cabecera del documento).
- Estilos en línea (empleando el atributo `style` en la etiqueta del elemento).
- Declaraciones de estilo marcadas como `!important`.

Especificidad del selector.

Hasta ahora se tuvieron en cuenta las distintas fuentes de la información del estilo. Pero aún puede existir algún conflicto a nivel de reglas. Por esa razón, "la cascada" continúa a nivel de reglas. Lo verás mejor con el el siguiente ejemplo, que podría estar en una hoja de estilo externa o incrustada. En él se muestran dos reglas que hacen referencia al elemento **`strong`**.

Ejemplo:

```
strong {color: red;} h1 strong {color: blue;}
```

En el ejemplo anterior, todo el texto del documento (X)HTML marcado con la etiqueta **strong** aparecerá en color rojo. Sin embargo, si el texto marcado con la etiqueta **strong** aparece dentro de una cabecera de primer nivel (**h1**), su color será azul. Esto ocurre porque un elemento en un contexto determinado es más específico que en un contexto general y, por lo tanto, tiene más peso. Debes tener en claro que, cuanto más específico sea el selector se le dará más peso para ignorar las declaraciones en conflicto.

Orden de las reglas.

Cuando una hoja de estilo contiene varias reglas en conflicto de igual peso, sólo se tendrá en cuenta la que está en último lugar. En el siguiente ejemplo, todas las cabeceras de primer nivel del documento serían rojas porque se impone siempre la última regla:

Ejemplo:

```
h1 {color: green;} h1 {color: blue;} h1 {color: red;}
```

Tipos de elementos CSS. Elementos de bloque y en línea.

En CSS, la noción de "elemento de bloque" y "en línea" es puramente de presentación. Un elemento de bloque de CSS siempre genera saltos de línea, antes y después de él, mientras que los elementos en línea de CSS no lo hacen, aparecen en el flujo de la línea y sólo pasarán a otra línea si no tienen espacio.

En (X)HTML, los párrafos (**p**), cabeceras (como **h1**), listas (**o**l, **ul**, **dl**) y contenedores (**div**) son los elementos de bloque más comunes, mientras que, el texto enfatizado (**em**), las anclas (**a**) y los elementos **span** son los elementos en línea más comunes.

Con CSS podrás indicarle al navegador cómo quieres que se vea en el documento empleando para ello los atributos **block** e **inline** de la propiedad **display**, independientemente de que un elemento sea de bloque o en línea.

Autoevaluación

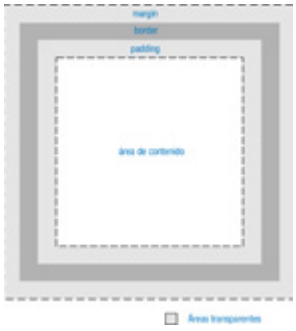
Relaciona los elementos XHTML con el tipo de elemento que son: de bloque o en línea.

Ejercicio de relacionar

Elemento XHTML	Relación	Tipo de elemento
p		1. De bloque.
div		
strong		2. En linea.
em		

Enviar

1.4.- El modelo de cajas de CSS.



El modelo de cajas es un concepto fundamental para comprender el funcionamiento de las hojas de estilo. Aquí podrás ver una introducción básica a este modelo. De acuerdo con este modelo, todos los elementos de una página web generan una caja rectangular alrededor llamada "caja del elemento".

En estas cajas se pueden configurar propiedades como bordes, márgenes y fondos (entre otras). Las cajas también se pueden emplear para posicionar los elementos y diseñar la página.

Las cajas de elementos, tal y como muestra la imagen, están hechas de cuatro componentes principales:

- **Contenido del elemento**: es lo que está en el núcleo de la caja está el.
- **Relleno** (`padding`): es el espacio que rodea al contenido.
- **Borde** (`border`): es la parte que perfila el relleno.
- **Margen** (`margin`): es el espacio que rodea al borde, la parte más externa del elemento.

Hay algunas características fundamentales del modelo de cajas que vale la pena destacar:

- El relleno, los bordes y los márgenes son opcionales, por lo que, si ajustas a cero sus valores se eliminarán de la caja.
- Cualquier color o imagen que apliques de fondo al elemento se extenderá por el relleno.
- Los bordes se generan con propiedades de estilo que especifican su estilo (por ejemplo: sólido), grosor y color. Cuando el borde tiene huecos, el color o imagen de fondo aparecerá a través de esos huecos.
- Los márgenes siempre son transparentes (el color del elemento padre se verá a través de ellos).
- Cuando definas el largo de un elemento estás definiendo el largo del área de contenido (los largos de relleno, de borde y de márgenes se sumarían a esta cantidad).
- Puedes cambiar el estilo de los lados superior, derecho, inferior e izquierdo de una caja de un elemento por separado.

Debes conocer

En el siguiente enlace de la web de la W3C encontrarás un resumen de las propiedades CSS más utilizadas en el modelo de cajas.

Enlace a la web de la W3C que habla sobre las propiedades CSS relacionadas con el modelo de cajas.

En los siguientes subapartados podrás ver con detalle las propiedades con las que podremos modificar la apariencia de las cajas

1.4.1.- Área de contenido y relleno.

Si comparamos el modelo de cajas con un Huevo Kinder, el área de contenido sería la sorpresa (en la imagen, el cochecito deportivo), mientras que el relleno sería la cápsula de plástico de color amarillo en la que viene la sorpresa. El chocolate sería el borde y el envoltorio de aluminio sería el margen.

Área de contenido.

Recuerda que el área de contenido es la parte más interna de la caja. En el ejemplo siguiente se muestra cómo se pueden modificar las propiedades que afectan al tamaño del área de contenido: su ancho (**width**) y su altura (**height**).

Ejemplo: **div {width:100px; height:200px; }**

Existen otras propiedades interesantes que nos permiten ajustar la altura y el ancho máximo y mínimo del área de contenido de las cajas. Estas propiedades son, respectivamente: **max-height, max-width, min-height y min-width.**

Relleno.

El relleno es una cantidad opcional de espacio existente entre el área de contenido de un elemento y su borde. Es conveniente que establezcas un valor de relleno cuando pones borde a un elemento.

Para establecer el relleno se emplea la propiedad **padding**. Esta propiedad, como muchas otras en CSS, obliga a configurar los valores en un orden determinado. Estos valores y su orden son: **top** (arriba), **right** derecha, **bottom** (debajo) e **left** (izquierda).

El ejemplo siguiente muestra una tabla que agrupa algunos ejemplos de la asignación de valores y su interpretación por CSS. En todos los ejemplos se ha empleado como selector el elemento **div**.

Ejemplos de asignación de valores a la propiedad padding.

EJEMPLO	INTERPRETACIÓN
`div {padding: 3px 20px 3px 20px; }`	Establece un relleno para todos los elementos div de 3 píxeles por encima del área de contenido, 20 píxeles a su derecha, 3 píxeles por debajo y 20 píxeles a su izquierda.
`div {padding: 3px 20px 3px; }`	Al omitir un valor, asume que el valor del relleno a la izquierda es el mismo que el de la derecha.
`div {padding: 3px 20px; }`	Al omitir dos valores, asume que el primer valor corresponde al relleno por encima y por debajo del área de contenido y, el segundo valor corresponde al relleno a la derecha y a la izquierda.
`div {padding: 3px; }`	Al omitir tres valores, asume que ese valor es el mismo para todos.

Otras características interesantes del relleno son:

- El valor del relleno se sumará al de `width` ya definido en el elemento.
- Su color es el mismo al del área de contenido.
- El relleno nunca se "colapsa". Esto lo entenderás cuando veas los márgenes que sí se colapsan.

1.4.2.- Bordes.

Un borde es una línea dibujada alrededor del área de contenido de un elemento y de su relleno (**padding**), aunque ya vimos que éste último era opcional.

> Los bordes funcionan, a la hora de establecer su valores, de la misma manera que el relleno visto anteriormente, siguiendo un orden: superior, derecho, inferior, izquierdo, siempre en el sentido de las agujas del reloj y comenzando en las 12. Es fácil de recordar.
>
> Se pueden establecer valores distintos para cada uno de los bordes y omitir valores, al igual que hacíamos con el relleno.

Podemos configurar el estilo del borde, su grosor y su color. Las propiedades que nos permiten hacerlo son:

- **Border-style**: con esta propiedad configuramos el estilo del borde. Esta propiedad es, sin duda, la más importante del borde, ya que, si no está presente el borde no existirá.
  La propiedad **border-style** puede tener los valores: **none**, **hidden**, **dotted**, **dashed**, **solid**, **double**, **groove**, **ridge**, **inset**, **outset** e **inherit**. En el siguiente ejemplo configuramos cada uno de los lados de la caja con un borde distinto.

  Ejemplo: **div {border-style: solid dashed dotted double; }div {border-style: solid dashed dotted double; }**

- **Border-width**: con esta propiedad configuramos el grosor del borde.
  Los valores que puede tomar esta propiedad son: **thin**, **medium**, **thick**, **inherit** o un valor concreto en píxeles. Si no se especifica esta propiedad tomará medium como valor por defecto. En el siguiente ejemplo configuramos un grosor distinto en cada uno de los lados del borde.

  Ejemplo: **div {border-style: solid; border-width: thin medium thick 12px; }**

- **Border-color**: con esta propiedad configuramos el color del borde.
  Si no especificamos el color el elemento coge el del "primer plano", es decir que si, por ejemplo, tenemos una caja en cuyo interior hay texto, el color del borde será el color del texto.

  Existe el color **transparent** pero no está soportado por todos los navegadores. En el siguiente ejemplo especificamos un estilo y un ancho igual para todos los bordes y un color distinto para cada borde.

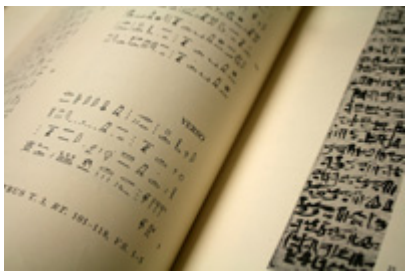
  Ejemplo: **div {border-style: solid; border-width: 4px; border-color: #333 #red rgb(0,0,255) #0044AC; }**

La propiedad `border` une todas las propiedades "`border`" vistas hasta ahora. En esta propiedad, a diferencia de las anteriores, no hay que colocar los valores en ningún orden concreto. La propiedad `border` se emplea cuando se quieren configurar los cuatro lados por igual.
También tenemos las propiedades: **border-top**, **border-right**, **border-bottom** y **border-left**.

Ejemplos:

```
h1 {border: .5em solid blue; }
h2 {border-left: solid blue .5em; }
h3 {border-right: solid .5em; }
```

1.4.3.- Márgenes.



El margen es la cantidad de espacio que se puede añadir alrededor del borde de un elemento.

Esta propiedad se configura con la propiedad **margin**. Al configurar esta propiedad debemos tener en cuenta que, a la hora de establecer los valores para los márgenes, hay que emplear la misma filosofía que con la propiedad **padding**.

Los márgenes **top** y **bottom** de dos elementos que van seguidos se " colapsan". Es decir, se asume como margen entre ambos elementos el mayor de ellos. El siguiente ejemplo muestra lo que que ocurre cuando tenemos dos elementos un **h1** y un **h2** colocados uno a continuación del otro.

Ejemplo:

```
h1 {margin: 10px 20px 10px 20px; }
h2 {margin: 20px; }
```

En el primer caso el margen superior e inferior es de 10px. En el segundo caso es de 20px. El espacio resultante entre los dos elementos será de 20px.

Por el contrario, si fuesen dos elementos colocados "uno al lado del otro" (como dos elementos **span**), deberíamos tener en cuenta que los márgenes **right** y **left** no se colapsan, sino que se suman.

Autoevaluación

¿En qué orden se dan las medidas de los márgenes en CSS?

- top bottom right left
- left right top bottom
- top left right bottom
- top right bottom left

2.- Selectores.

Caso práctico

Después de unos días estudiando CSS, Carlos ya tiene claro la estructura de las reglas de estilo que componen los archivos CSS.

Carlos también se da cuenta que una parte muy importante de estas reglas son los selectores y, como está empezando, decide pedir ayuda a Juan, que ya lleva más de cuatro años desarrollando aplicaciones web.

Juan le explicará cómo debe utilizar los selectores para aprovechar todo el potencial que tienen y, entre los dos, completarán el manual de CSS para poder consultarlo en cualquier momento que lo necesiten mientras realizan la aplicación web.

El selector es la parte de la regla de estilo que identifica el elemento concreto al que se aplicarán las instrucciones de presentación. CSS ofrece varios tipos de selectores que permiten mejorar la flexibilidad y la eficiencia en la creación de hojas de estilo.

Debes conocer

Debes tener muy claro cuál es la sintaxis de las reglas de estilo CSS: selector { regla1: valor; regla2: valor; ...}

En el siguiente enlace puedes encontrar una explicación interesante de cuál es la sintaxis de las reglas de estilo que complementará lo que ya viste anteriormente.

Reglas sintácticas de los estilos CSS.

Recomendación

En los siguientes puntos veremos los selectores más utilizados pero puedes consultar la web de la W3C donde encontrarás un resumen de los patrones de selectores CSS existentes.

Patrones de selectores CSS

2.1.- Selectores de elemento.

Recomendación

Los selectores de elemento son los más sencillos. Son aquellos que se aplican a un elemento (etiqueta) del lenguaje (X)HTML.

Ejemplos:

```
h1 {color: blue;}
h2 {color: blue;}
p {color: blue;}
```

Si te fijas en los ejemplos, verás que se está definiendo la misma propiedad (**color**) en todos los elementos e incluso se está asignando el mismo valor (blue). El ejemplo siguiente muestra como se puede escribir una única regla aplicada a varios selectores a la vez.

Ejemplo: **h1, h2, p {color: blue;}**

Cuando se realiza una declaración sobre varios selectores a la vez, éstos se separan por comas.

Existe un selector de elementos "universal" representado por el asterisco (*). El ejemplo siguiente muestra una regla que pondrá en gris todos los elementos del documento que no tengan especificado otro color.

Ejemplo:* **{color: grey;}**

2.2.- Selectores contextuales.

Como vimos, los **selectores de elemento** se aplican a todos los casos en los que se encuentre el elemento en el documento (X)HTML. En cambio, los selectores contextuales permiten aplicar estilos a los elementos basándose en su contexto o en su relación con otro elemento.

Hay varios tipos de selectores contextuales: **descendente, hijo y hermano.**

Los **selectores descendentes** hacen referencia a elementos que están contenidos en otro elemento. Un selector descendente se pone a continuación del selector en el que está contenido separado de él por un espacio en blanco. El siguiente ejemplo especifica que los elementos **em** deben tener color azul, pero sólo si son descendientes de un elemento de lista (**li**). El resto de los elementos em no se verán afectados.

Ejemplo: **`li em {color: blue;}`**

Los selectores descendientes también pueden estar anidados en varias capas de profundidad. El siguiente ejemplo pone de color amarillo sólo el texto enfatizado (**em**) de las anclas (**a**) que se encuentren en las listas ordenadas (**ol**).

Ejemplo: **`ol a em {color: yellow;}`**

Si se emplea el selector descendente combinado con el selector universal, se puede restringir el alcance de un selector descendente. El siguiente ejemplo muestra dos párrafos que contienen un hipervínculo. En el primer caso el elemento ancla es descendiente directo del elemento de párrafo y, en el segundo caso, es descendiente directo del elemento **span** que, a su vez, lo es del elemento de párrafo.

Ejemplo:

```
<p><a href="#">Enlace</a></p>
<p><span><a href="#">Enlace</a></span></p>
```

Examinemos las dos reglas de estilo siguientes:

```
p a { color: red; }
p * a { color: red; }
```

Con la primera se consigue que se muestren los dos enlaces de color rojo, mientras que con la segunda regla sólo se mostraría en rojo el segundo de los enlaces. La razón es que el selector **`p * a`** se traduce como "todos los elementos de tipo **<a>** que se encuentran dentro de cualquier elemento que, a su vez, se encuentra dentro de un elemento de tipo **<p>**". Como el primer elemento **<a>** está directamente incluido en un elemento **<p>**, no se cumple la condición del selector **`p * a`**.

Un **selector hijo** es un caso concreto de un selector descendente en el un selector está contenido directamente en otro, sin que existan niveles intermedios. Un selector hijo se escribe a continuación de su selector padre separándolo de él por el símbolo "mayor que" (**>**). En el siguiente ejemplo se pone en gris el fondo del texto enfatizado, pero sólo si es hijo directo de un párrafo.

Ejemplo: **`p > em {background-color: gray;}`**

El **selector adyacente** se utiliza para hacer referencia a un elemento que sigue inmediatamente a otro en el código, con el que comparte el mismo elemento padre. Un selector adyacente se escribe a continuación de otro selector separándolo de él por el símbolo de suma (**+**). El siguiente ejemplo pondría en color azul el primer párrafo que sigue a una cabecera de primer nivel.

Ejemplo: **`h1 + p {color: blue;}`**

2.3.- Selectores de clase e ID.

Para poder hacer uso de selectores más específicos, se hace necesario introducir los conceptos de identificador (**id**) y clase (**class**).

Identificadores (**id**).

Los elementos HTML disponen de un atributo llamado identificador (**id**), que tiene como finalidad identificar al de manera excluyente. De este modo, CSS u otro lenguaje podrá hacer referencia a él y distinguirlo del resto de los elementos del documento.

> Un **id** debe ser único en cada documento (X)HTML.

Ejemplo:** **

Se recomienda que el valor del id sea un nombre que caracterice o clarifique, de forma breve y esquemática al elemento y que, además, sea fácilmente reconocible por el programador. Se utilizan con frecuencia para identificar las secciones principales de las páginas: contenido, cabecera, pie, etcétera.

Para escribir una regla de estilo que se aplique a un determinado identificador hay que escribir el símbolo de la almohadilla (**#**) seguido del nombre del identificador. El ejemplo siguiente muestra algunas formas de establecer el tamaño de la fuente en 14 píxeles al elemento p identificado como "textocabecera" del ejemplo anterior:

Ejemplo:

```
p#textocabecera {font-size: 14px;}<br>
                #textocabecera {font-size: 14px;}</p>
```

Con el primera regla indicamos que se aplique el estilo a un párrafo cuyo identificador sea "textocabacera" pero, como el id es único en cada documento, realmente basta con la segunda forma para decir lo mismo, porque no va a haber otro elemento ** ** o diferente de ** ** que tenga ese mismo identificador.

Si tenemos varios elementos que necesitan un tratamiento similar, emplearemos el atributo **class**.

Clases (**class**).

Se emplea el atributo **class** para identificar distintos elementos como parte de un grupo conceptual. Así, los elementos de una clase pueden modificarse con una única regla de estilo.

En el siguiente ejemplo se muestra como dos elementos distintos se clasifican de la misma forma mediante la asignación del valor "especial" al atributo **class**.

Ejemplo:

```
<h1 class="especial">¡Atención!</h1>
<p class="especial">Hoy tenemos grandes rebajas.</p>
```

También se puede hacer que un elemento pertenezca a más de una clase separando sus nombres de clase con espacios. En el siguiente ejemplo el párrafo pertenece a dos clases: "textocabecera" y "especial".

Ejemplo:** Hoy tenemos grandes rebajas. **

Para escribir una regla de estilo que se aplique a todos los elementos de una determinada clase hay que escribir un punto seguido del nombre de la clase. Por ejemplo: **.especial {color: green;}**

El siguiente ejemplo muestra la forma de lograr que todos los elementos de la clase "especial" tengan un color verde a excepción de las cabeceras de primer nivel que tienen que ser rojas.

Ejemplo:

```
.especial {color: green;}<br />
h1.especial{color: red;}
```

Los nombres de clases y de identificadores no pueden contener espacios en blanco.

Recomendación

En el siguiente enlace podrás leer unas recomendaciones generales sobre CSS. Son especialmente interesantes aquellas que mencionan las reglas que deben cumplir los nombres de los identificadores y de las clases y cómo se debe estructurar una hoja de estilos.

Recomendaciones generales sobre CSS.

2.4.- Pseudoselectores.

Si queremos aplicar reglas de estilo a elementos especiales como: los vínculos visitados, la primera línea de un párrafo o su primera letra, emplearemos los **pseudoselectores**.

Hay dos tipos de pseudoelectores: **pseudoclases** y **pseudoelementos**.

Pseudoclases.

Clasifican a los elementos basándose en características que van más allá de su nombre, atributos o contenido. La mayoría de las pseudoclases afectan a los elementos ancla (**a**), pudiendo definir un estilo diferente en función del comportamiento del enlace: si todavía no ha sido visitado (**link**), si ya lo ha sido (**visited)**, mientras el ratón pasa por encima (**hover**) o justo cuando se pulsa el ratón sobre él (**active**).

Para emplear una pseudoclase se escribe la misma a continuación del selector separándola de éste por el símbolo de dos puntos (**:**). El ejemplo siguiente muestra cómo se distinguirían los cuatro estados posibles de un enlace mediante colores diferentes.

Ejemplo:

```
a:link {color: red;}
a:visited {color: blue;}
a:hover {color: fuchsia;}
a:active {color: maroon;}
```

Con el empleo de estas pseudoclases podemos quitar el subrayado de los hiperenlaces y hacer que aparezca sólo cuando pasamos el puntero por encima. Pel ejemplo siguiente muestra el empleo de la propiedad text-decoration para conseguir dicho objetivo.

Ejemplo:

```
a:link {color: red; text-decoration: none;}
a:visited {color: blue; text-decoration: none;}
a:hover {color: red; text-decoration: underline;}
```

> Debes tener en cuenta que las pseudoclases ancla deben aparecer siempre en un determinado orden. Este orden es: :**link**, **:visited**, **:hover** y **:active.**
>
> Por si te ayuda, para recordarlo, se emplean las iniciales: LVHA.

Existen otras pseudoclases, que se emplean menos:

- `:focus` hace referencia a los elementos que tienen el foco, como ocurre, por ejemplo, en los elementos de un formulario. Un ejemplo sería: `input:focus {background-color: yellow;}`
- `:first-child` hace referencia el primer hijo de un elemento padre. En el siguiente ejemplo se aplica el estilo al primer elemento de una lista desordenada: `ul li:first-child {font-weight: bold;}`
- `:lang`(idioma) hace referencia al idioma en el que está un determinado elemento. En el siguiente ejemplo se aplica el estilo a cualquier párrafo que esté escrito en inglés:`p:lang(en) {color:red}`

Cuidado! Muchos navegadores como Explorer 6 y anteriores, IE5 para Macintosh, Netscape 6+ y Opera 7+ no tienen soporte para las pseudoclases **`:focus`** y **`:first-child`**.

Pseudoelementos.

Estos pseudoelementos suelen ser partes de un elemento ya existente, como puede ser su primera línea (**`:first-line`**) o su primera letra (**`:first-letter`**), aunque también nos permite hacer referencia a elementos sin concretar en la estructura del documento porque dependen de la estructura del documento (**`:before`** y **`:after`**).

Ejemplos:

```
p:first-line {letter-spacing: 6pt;}
p.definicion:first-letter {font-size: 300%; color: red;}
p.incompleto:after { content: " continuará ..."; }
```

En el primer ejemplo añadimos espacio extra a la primera línea del texto de cada párrafo, en el segundo modificamos el estilo (tamaño y color) de la primera letra de los párrafos pertenecientes a la clase "definicion" y, en el tercero, añadimos el texto " continuará ..." al final de cada párrafo perteneciente a la clase "incompleto".

Recomendación

En el siguiente enlace podrás hacer pruebas con los distintos tipos de selectores y familiarizarte con ellos. Podrás escribir selectores descendentes, hijos y adyacentes, utilizar selectores de clase e ID, los pseudoelementos y las pseudoclases y comprobar en el lado derecho de la página cuáles son los elementos que se verían afectados por una reglas de estilo aplicada a esos selectores.

Esta página consta de tres bloques titulados: Selector, DOM Sample y HTML for DOM Sample.

Para saber qué selectores, clases e identificadores puedes comprobar, tendrás que mirar el código que se encuentra en el tercer bloque. Escribiendo los selectores que quieras en el primer bloque y pulsando en el botón Apply, verás que se enmarcan en rojo dichos elementos en el segundo bloque.

Simulador de selectores

2.5. La palabra clave !important en CSS.

!important se utiliza para dar prioridad a ciertas reglas. Cualquier definición de estilo que vaya acompañada de un **!important** tendrá prioridad sobre cualquier otra.

Como ya sabes, cuando tenemos una propiedad aplicada dos veces, el navegador hará caso a la última. En el ejemplo que se muestra a continuación está claro que se le asignarán 1200 píxeles al ancho del elemento identificado con el **id="principal"**.

Ejemplo:

```
#principal {  width: 800 px;
              width: 1200px; }
```

Sin embargo, en el siguiente ejemplo se muestra como dando prioridad a la primera declaración con la palabra **!important**, podemos tener ese elemento con un ancho de 800 píxeles.

Ejemplo:

```
#principal { width: 800px !important;
             width: 1200px;  }
```

También en este caso debemos tener en cuenta la compatibilidad con los navegadores. Así, Internet Explorer 6 y versiones anteriores ignoran esta palabra clave mientras que IE7 la soporta sin problemas.

> Las declaraciones acompañadas de la palabra **!important**!important tienen prioridad sobre otras declaraciones que afecten al mismo elemento.

Con esto de los selectores descendentes, hijos y adyacentes, se puede dar el caso de que, sin querer, estemos aplicando un estilo que no queremos a un elemento porque se ve indirectamente afectado por alguna de las reglas y no sabemos cuál. Por ejemplo: sale un párrafo en color rojo y no nos acordamos dónde lo configuramos porque realmente no tenemos ninguna regla que diga **p {color:red}**. En ese caso, y para no descerebrarnos buscando dónde está el fallo, si sabemos que los párrafos de texto emplean la letra de color negro, crearíamos la regla **p { color: green !important; }** ¡y solucionado!

Autoevaluación

¿Cuál de los siguientes es el selector CSS que selecciona a todos los objetos con class="nuevo" que están contenidos en un tabla?

- table .nuevo
- table.nuevo
- table#nuevo

3.- Propiedades de fuente y texto.

Caso práctico

La web de “Migas Amigas” ofrecerá a sus visitantes varias páginas donde se relatará su historia, la descripción del entorno donde se encuentra ubicada y, además, algunas recetas de los productos que en ella se venden.

Todo esto hace necesario que Carlos tenga los conocimientos necesarios para configurar la apariencia de todos estos textos.

Carlos se pone de nuevo en contacto con Juan para que éste le guíe en el aprendizaje de las propiedades y técnicas que necesita.

Las hojas de estilo nos van a permitir un control total sobre el formato del texto en las páginas web. Veamos las propiedades de CSS más utilizadas para formatear el texto.

Recomendación

A medida que vayas estudiando estos puntos también puedes visitar el siguiente enlace de la web de la W3C donde encontrarás un resumen de las propiedades CSS más utilizadas para formatear las fuentes y el texto.

Propiedades CSS para fuentes y texto.

3.1.- Propiedades de fuente.

Las propiedades de las fuentes en CSS son usadas para configurar la apariencia deseada para el texto de un documento. Veamos las más empleadas:

font-family

Nos permite especificar un nombre de fuente en concreto o bien una familia genérica de fuentes. Se puede especificar una lista de fuentes separadas por comas teniendo en cuenta que si el nombre de la fuente o familia etiene algún espacio en blanco intercalado habrá que encerrarlo entre comillas.

Hay que darse cuenta de que el tipo de letra elegido debe estar instalado en el equipo cliente. Por lo que, si escogemos un tipo de letra "poco habitual", corremos el riesgo de que el usuario no vea la página tal y como la hemos diseñado.

Recomendación

En el siguiente enlace puedes acceder a un recurso en línea de Microsoft. En él puedes encontrar las fuentes que se instalan por defecto con cualquier tipo de programa de Microsoft y las que se instalan con el sistema UNIX o en un Macintosh OS X.

Recurso en línea de Microsoft sobre fuentes que se instalan por defecto.

font-size

Nos permite configurar el tamaño del texto. Mientras que el HTML estándar prevé sólo 7 niveles predefinidos para el tamaño del texto, las hojas de estilo CSS permiten un control mucho más preciso y elástico sin, prácticamente, limitaciones.

Podemos establecer tamaños de forma absoluta, de forma relativa, con un valor numérico o en forma de porcentaje.

font-weight

Nos permite establecer el espesor o intensidad de las fuentes, como **<b>** para el HTML clásico. Es posible asignar hasta 7 valores diferentes: **normal**, **bold**, **bolder**, **lighter**, 100, 200, 300, 400, 500, 600, 700, 800 ó 900.

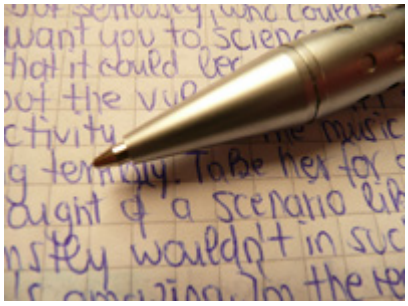
font-style

Nos permite configurar el "estilo" de la fuente. Hay tres valores posibles: normal que no configura ningún estilo en particular sino que toma el definido por defecto en el navegador, italic: que equivale al elemento del HTML clásico (**<i>**) que coloca el testo en cursiva y **oblique** que funciona, aparentemente, como "italic".

font-variant

Permite dos posibilidades: **normal** y **small-caps**. Con normal, el texto no cambia de apariencia y con small-caps el texto pasa a mostrarse en mayúsculas de un tamaño inferior.

3.2.- Propiedades de texto.



Las propiedades de texto permiten aplicar estilos a los textos espaciando sus palabras o sus letras, decorándolo, alineándolo, transformándolo, etcétera. Algunas de estas propiedades son:

text-decoration

Permite decorar el texto con subrayados y otros efectos. Los valores que puede tener son: **none** (ninguno), **underline** (subrayado), **overline** (con una línea por encima), **line-through** (tachado), **blink** (parpadeante) e **inherit** (heredado) .

text-transform

Controla la apariencia de las letras en un elemento. Los valores que puede tener son: **none** (texto normal, con mayúsculas y minúsculas), **capitalize** (cada palabra comienza con mayúsculas), **uppercase** (todo el texto aparece en mayúsculas) y **lowercase** (todo el texto aparece en minúsculas).

line-height

Establece el espacio que hay entre dos líneas consecutivas.

Ejemplos:

```
p { line-height: 1.4; } /* Se establece un factor de multiplicación respecto al tamaño
p { line-height: 14pt; } /* Establece una distancia fija entre las líneas de 14 puntos
p { line-height: 140%; } /* Establece una distancia proporcional (%) respecto a font-si
```

text-indent

Sangra la primera línea de texto de un párrafo.

Ejemplos:

```
p { text-indent: 20px; } /* Define un sangrado positivo. El valor por defecto es 0. */
p { text-indent: -12px; } /* Éste es negativo (hacia la izquierda). */
p { text-indent: 10%; } /* Aquí está como un porcentaje respecto al ancho del elemento
```

letter-spacing

Configura sobre el espacio que hay entre los caracteres. Este valor puede aumentar o disminuir ya que, al igual que **text-indent** y otras propiedades, admite valores positivos y negativos.

Ejemplos:

```
p { letter-spacing: 12px; } /* Expande los caracteres. Los separa */
p { letter-spacing: -0.5px; } /* Contrae los caracteres. Los junta */
```

white-space

Permite establecer cómo se gestionan los espacios en blanco en un elemento. Los valores que puede tener son: **normal** (los espacios en blanco adicionales son ignorados por el navegador), **pre** (los espacios en blanco adicionales son utilizados como cuando se emplea la etiqueta pre en HTML), **nowrap** (no se produce el ajuste de línea automático por lo que el texto permanecerá en la misma línea hasta que encuentre una etiqueta **
**).

4.- Los colores y los fondos.

Caso práctico

Ni que decir tiene que el equipo de BK intentará que la web de “Migas Amigas” adquiera una apariencia lo más elegante y vistosa posible. Para que esto sea así, un factor muy importante será configurar los colores y los fondos adecuados.

Los colores y las imágenes de fondo ya fueron seleccionados por todo el equipo de diseño. Ahora, Carlos debe descubrir cómo aplicar éstos a la web sin que la programación de CSS sea un obstáculo para lograr justo el diseño deseado.

Esta vez será Ada la que se ponga en contacto con Juán para guiarlo en el aprendizaje de las propiedades y técnicas necesarias.

CSS permite controlar el color y los fondos con unas posibilidades que están a años luz de los efectos que podemos alcanzar empleando sólo HTML.

Recomendación

A medida que vayas estudiando estos puntos también puedes visitar el siguiente enlace de la web de la W3C donde encontrarás un resumen de las propiedades CSS más utilizadas en el modelo de cajas.

Propiedades CSS para colores y fondo.

Debes conocer

Al utilizar estilos de forma habitual tendrás que establecer el color del los diferentes elementos, por lo que debes conocer el modo de hacerlo. Los colores en CSS se pueden indicar de cinco formas diferentes: con palabras clave (nombres propios de los colores), con colores del sistema, empleando el sistema RGB con numeración hexadecimal normal o simplificada, decimal o porcentual.

4.1.- Color del primer plano y el fondo

Para establecer los colores de primer plano y de fondo existen dos propiedades distintas. La propiedad **color** es la que debes utilizar para configurar el color del primer plano, es decir, el color del texto y el color por defecto del borde de un elemento. Mientras que para configurar el color de fondo deberán emplear la propiedad **background-color**.

color

CSS también reconoce hasta 16 nombres de color válidos: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white y yellow.

Con respecto al color del primer plano deberás tener en cuenta que:

Si añades color al primer plano de una imagen, ésta seguirá viéndose pero el color se aplicará al borde de la imagen.

La propiedad **border-color** ignora la propiedad **color**.

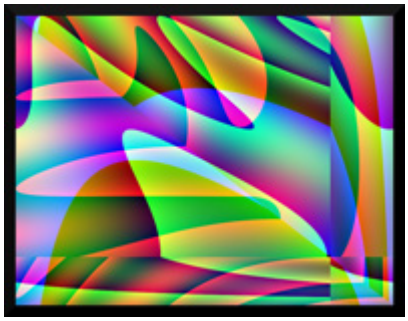
Para configurar el color de todo un documento debemos escribir una regla con esta propiedad color para el selector **body**.

La aplicación de color a los elementos de los formularios no funciona bien en todos los navegadores.

Los ejemplos siguientes muestran diversas formas de configurar el color del texto de los párrafos.

Ejemplos:

```
p {color: #0000FF; }
p {color: #00F; }
p {color: rgb(0,0,255); }
p {color: rgb(0%, 0%, 255%); }
p {color: gray; }
```



background-color

Con CSS no sólo se puede proporcionar un color de fondo a toda la página, también se puede configurar el color de fondo de cualquier elemento del documento, tanto si son elementos de bloque como de línea.

Con la aparición de CSS, se recomienda emplear "cajas de color" en sustitución de las tablas.

> Las propiedades relativas al fondo no se heredan pero, como el valor predeterminado de esta propiedad es **transparent**, salvo que se especifique un color concreto, el color de fondo del elemento padre aparecerá a través de sus elementos hijos.

Ejemplo:**p {padding: 5px; background-color: #ccc; }**

4.2.- Imágenes de fondo.

Las imágenes las podrás configurar de forma muy completa ya que, además de poner una imagen como fondo de una página, podrás ajustar su punto de partida, su patrón de repetición, su posición dentro de un elemento cualquiera y lograr que permanezca fija en esa posición aunque se mueva el resto del documento. Para realizar esta configuración detallada se emplean las propiedades: **background-image**, **background-repeat**, **background-position**, **background-attachment** y **background**.

background-image

Esta propiedad sirve para configurar la imagen de fondo de cualquier elemento.

> La propiedad **background-image** prevalece sobre la propiedad **background-color** por lo que si con un elemento realizas declaraciones con estas dos propiedades ignorará la declaración de **background-color**.

En el siguiente ejemplo se muestra la configuración de la imagen de fondo, usando un archivo de imagen de nombre “fondo.gif”, en todo el documento excepto para el párrafo que tiene un identificador “cabecera” que tendrá como imagen de fondo un archivo de imagen de nombre “fondo-cuerpo.gif”.

Ejemplo:

```
body {background-image: url(fondo.gif); }
p#cabecera {background-image: url(fondo-cuerpo.gif); }
```

background-repeat

Permite configurar a tu gusto la forma en la que se repetirá la imagen cuando su tamaño sea más pequeño que la ventana del navegador. También permite evitar que la imagen se repita.

Ejemplos:

```
<div#cuerpo {background-image: url(fondo.gif); background-repeat: no-repeat; }
div.horiz {background-image: url(imagen.gif); background-repeat: repeat-x; }
div.vert {background-image: url(imagen.gif); background-repeat: repeat-y; }
```

Con el primer ejemplo, consigues que la imagen aparezca sólo una vez, con el segundo ejemplo la imagen se repetirá a lo ancho (horizontalmente) tantas veces como necesite y, con el tercer ejemplo la imagen de fondo se repetirá a lo largo del documento (verticalmente).

background-position

Especifica la posición de la primera imagen que cubrirá el fondo del elemento en el que esté definida esta propiedad. Al posicionamiento podemos asignarles los valores: left, right, top, bottom y center los cuales se usan, normalmente, de dos en dos sin importar el orden (uno indica su posición horizontal y el otro indica su posición vertical). Si sólo se indica una, se supone que la otra es center.

En el siguiente ejemplo, se configura el fondo del elemento body con una imagen llamada “fondo.gif” que aparecerá sólo una vez en la parte superior central del cuerpo del documento.

Ejemplo:

```
body {background-image: url(fondo.gif);
background-position: top center;
background-repeat: no-repeat; }
```

Para el posicionamiento se pueden emplear también las medidas de longitud vistas anteriormente. En este caso, las medidas son relativas al extremo superior izquierdo del elemento. En el siguiente ejemplo, se configura el mismo fondo de antes que aparecerá a 150 píxeles de la esquina superior izquierda del cuerpo del elemento (horizontal y verticalmente).

Ejemplo:

```
body {background-image: url(fondo.gif);
background-position: 150px 150px;
background-repeat: no-repeat; }
```

También puedes utilizar los valores porcentuales. Como seguro supones, si indicas sólo un valor se asume que el otro es un 50%. Debes tener en cuenta que el valor porcentual se aplica “al contenedor y a la imagen en sí”. En el siguiente ejemplo se muestra la regla de estilo que logra que la imagen quede colocada en el centro del elemento **body**.

Ejemplo:

```
body {background-image: url(imagen.gif);
background-position: 50% 50%;
background-repeat: no-repeat; }
```

background-attachment

Con esta propiedad puedes fijar la imagen en una posición concreta. Se le pueden asignar los valores: **scroll**, **fixed** e **inherit**, siendo **scroll** el valor por defecto. Se suele emplear el valor **fixed** para conseguir que la imagen no se desplace con el documento.

background.

Esta propiedad permite configurar todas las propiedades de fondo vistas anteriormente usando una única declaración, de forma similar a lo que ocurría con la propiedad **font**, pero a diferencia de ésta, no tiene ninguna propiedad obligatoria y sus valores pueden aparecer en cualquier orden.

Sólo debes tener una restricción: la posición se indica con dos valores que deben aparecer juntos, primero el horizontal seguido, inmediatamente después, por el vertical ya que si sólo aplicamos un valor el otro se configura por defecto a **center**.

Ejemplos:

```
body {background: url(fondo.gif) fixed top center no-repeat; }
div.cabecera {background: repeat-x url(fondo.gif) red; }
p {background: #ccc; }
#contenido span.imagen {background: url("imagenes/imagen.gif") 0 50% no-repeat; }
```

4.3.- Opacidades.

Reflexiona

Si te fijas en las ventanas, persianas y cortinas de una casa, verás que algunas ventanas como las de los cuartos de baño son de cristal pero no son transparentes, dejar ver el exterior pero no de forma nítida. Cuando se bajan las persianas, no se ve absolutamente nada del exterior, y las cortinas, dependiendo de su tejido, pueden dejar pasar la claridad en mayor o menor medida.

La opacidad es una característica de los elementos que nos permite mostrar o no otros elementos que tengan por debajo. Para conseguir efectos de transparencia en algunos elementos tienes las siguientes propiedades: **opacity**, **moz-opacity** y **filter**.

opacity

Esta propiedad, que es compatible con todos los navegadores que soporten CSS3, permite asignar valores comprendidos entre 0 (invisible o totalmente transparente) y 1 (totalmente opaco).

-moz-opacity

Esta propiedad, permite asignar los mismo valores que la propiedad anterior. La diferencia está en que sólo es compatible con versiones anteriores del Firefox 0.9.

filter

Esta propiedad, de IE (5.5 y siguientes), tiene varios efectos: degradaciones, desenfocados, sombras, etcétera. Para lograr la transparencia hay que aplicar el filtro alpha, con valores entre 0 y 100.

En el ejemplo siguiente tienes el código donde se configura la opacidad de algunos elemento para que funciones en todos los navegadores.

Ejemplo:

```
<html>
        <head>
            <style type="text/css">
                body {background: #00f; color: black}
                #saludo {     background: white;
                                width: 200px; height: 200px;
                                filter: alpha(opacity=50);
                               -moz-opacity: 0.5;
                                opacity: 0.5; }
                #frase {          background: white;
                                width: 200px; height: 200px;
                                filter: alpha(opacity=100);
                               -moz-opacity: 1;
                                opacity: 1; }
                #despedida {     background: white;
                                width: 200px; height: 200px;
                                filter: alpha(opacity=0);
                               -moz-opacity: 0;
                                opacity: 0; }
```

```
        </style>
    </head>
    <body>
        <div id="saludo"><h1>¡Hola!</h1></div>
        <div id="frase">Este es un ejemplo de opacidad</div>
        <div id="despedida"><h2>¡Adiós!></h2></div>
    </body>
</html>
```

Autoevaluación

A la vista del ejemplo anterior ¿Cuáles de las siguientes afirmaciones son correctas?

- [ ] El color del fondo de toda la página es blanco.
- [ ] El color de fondo del saludo es blanco.
- [ ] El color de fondo de la frase es blanco
- [ ] El color de fondo de la despedida es blanco.
- [ ] El saludo no se ve.
- [ ] La frase no se ve.
- [ ] La despedida no se ve.
- [ ] El fondo del saludo es el mismo que el el fondo de la página.
- [ ] El fondo del texto es el mismo que el de la página.
- [ ] El fondo de la despedida es el mismo que el de la página.

Mostrar Información

5.- Flotar y posicionar.

Caso práctico

Para la web de "Migas Amigas" el equipo de BK programación decidió al final seleccionar un diseño de dos columnas clásico, para toda la página a excepción de la página de portada.

Para realizar este diseño será necesario que Carlos comprenda cómo se flotan y se posicionan cada uno de los elementos que aparecen en las webs.

A medida que Juan le explica las propiedades y técnicas para flotar y posicionar los elementos, Carlos va tomando conciencia de que esto es, sin duda, lo más complicado a lo que se ha enfrentado hasta ahora en lo que respecta a la tecnología CSS.

Carlos piensa: "Sin duda, me voy a tener que dedicar a fondo".

CSS utiliza el flotado y el posicionamiento para tener el máximo control sobre el lugar que ocupa cada elemento en una página web, sus condiciones de visibilidad y "flotabilidad", así como controlar el manejo de capas.

En los siguientes apartados veremos algunas propiedades de CSS 2.1 que se utilizan para controlar el posicionamiento de los elementos. Estas son: **`float`**, **`clear`**, **`position`**, **`bottom`**, **`top`**, **`left`**, **`right`**, **`overflow`**, **`clip`**, **`visibility`** y z-**`index`**.

Recomendación

A medida que vayas estudiando de los apartados siguientes también puedes visitar el siguiente enlace de la web de la W3C donde encontrarás un resumen de las propiedades CSS más utilizadas para dar forma a la apariencia de las webs.

Propiedades CSS del modelo de formato visual.

Un término que aparecerá a menudo es “flujo normal”. Cuando hablamos de que los objetos de una página siguen el flujo normal del documento, queremos indicar que la forma en la que se disponen en la ventana del navegador coincide con el lugar que ocupan en el documento escrito (en el código (X)HTML), donde el orden de lectura es de arriba a abajo y de izquierda a derecha.

Flotando y posicionando con CSS conseguimos que los elementos abandonen su flujo normal. De esta forma un elemento que este en el documento escrito más abajo que otro en el documento puede verse en el navegador por encima de él.

5.1.- Flotar.

Flotar sirve para mover una caja a la izquierda o a la derecha hasta que su borde exterior toque el borde de la caja que lo contiene o toque otra caja flotante.

Para que un elemento pueda flotar debe tener definido implícita o explícitamente su tamaño.

Las cajas flotantes no se encuentran en el "flujo normal" del documento por lo que las cajas que sí siguen el flujo normal se comportan como si las flotantes no estuviesen ahí.

`float`

Flotar es algo más que mover una imagen. Sirve para crear diseños multicolumna, barras de navegación de listas no numeradas, poner contenido en forma tabular pero sin emplear tablas y mucho más.

La propiedad **`float`** puede tener los siguientes valores:

- **none** hará que el objeto no sea flotante.
- **left** hace que el elemento flote izquierda.
- **right** hace que el elemento flote a la derecha.
- **inherit** hará que el elemento tome el valor de esta propiedad de su elemento padre.

`clear`

Sirve para mantener limpia el área que está al lado del elemento flotante y que el siguiente elemento comience en su posición normal dentro del bloque que lo contiene.

La propiedad **`clear`** puede tener los siguientes valores:

- **left** indica que el elemento comienza por debajo de cualquier otro elemento del bloque al que pertenece que estuviese flotando a la izquierda.
- **right** funciona como el left pero en este caso el elemento deberá estar flotando a la derecha.
- **both** mueve hacia abajo el elemento hasta que esté limpio de elementos flotantes a ambos lados.
- **none** permite elementos flotantes a ambos lados. Es el valor por defecto.
- **inherit** indica, al igual que en `float`, que heredará el valor de la propiedad `clear` de su elemento padre.

Debes conocer

Es importante saber aplicar las técnicas de flotado en contenedores y cajas en general. En la siguiente presentación veremos con detalle el funcionamiento de las propiedades **`float`** y **`clear`**.

5.2.- Posicionamiento.

Una vez te has familiarizado ya con el modelo de cajas y con el modo de flotarlas, se hace necesario estudiar los modelos de posicionamiento y del formato visual para poder tener una visión completa de cómo se organizan los elementos en una página. Para ello vamos a ver algunas de las propiedades que nos permiten organizar los elementos:

display

Esta propiedad permite al documento interpretar de otra forma los elementos de tipo bloque y los elementos de tipo línea. Para ello basta con que asignes a esta propiedad el valor **block** si quieres que un elemento "en línea" se comporte como un elemento de tipo bloque y **none**, si quieres que un elemento de bloque no genere caja, no muestre su contenido y no ocupe espacio en la página.

position

Permite posicionar los elementos en un documento. Esta propiedad admite cinco valores:

- static que permite colocar los elementos según el flujo normal. Es el valor que asumirá por defecto en todos los elementos HTML.
- relative que permite dejar el elemento exactamente donde está. Un elemento posicionado de esta forma se puede cambiar desde su punto de partida estableciendo para ello una distancia vertical y/o horizontal. En el siguiente ejemplo se desplaza la "caja2" 50px del extremo izquierdo y 50 px del extremo superior de su posición relativa.
  Ejemplo:

```
#caja2 {     width: 130px; height: 130px;
      background-color:#00FF00;
         border: solid 1px black;
         margin: 10px;
      /* Posicionamento relativo */
      position: relative;
      left: 50px; top: 50px; }
```

  Un elemento posicionado relativamente, que siga en el flujo normal del XHTML inmediatamente después a otro elemento posicionado también relativamente, calculará su origen de la forma siguiente:

  - Si el elemento es hijo del anterior, su origen estará en el final del anterior (su padre).
  - Si el elemento no es hijo del anterior, tendrá su origen donde el anterior tenga su final si no se fijaron valores distintos de cero en sus propiedades `top` y `left`.

- absolute permite abandonar el flujo normal del haciendo que el elemento no ocupe ningún espacio de forma que el resto de elementos del flujo normal actuarán como si el elemento no estuviese allí. El modo de determinar el origen de coordenadas de nuestro elemento será el siguiente:
  - Si el elemento posicionado de modo absoluto no está contenido dentro de ningún otro, su origen de coordenadas se mide respecto a la esquina superior izquierda del `body` (contenedor principal).
  - Si el elemento posicionado de modo absoluto está contenido dentro de otro elemento, el origen de coordenadas del elemento se calculan con respecto a la posición de la esquina superior izquierda del elemento que lo contiene.
- fixed funciona de forma parecida al posicionamiento absoluto pero posiciona con respecto a la ventana del navegador apareciendo en la misma posición aunque el usuario se desplace por la página con las barras de desplazamiento.

visibility

Esta propiedad controla si el elemento será visualizado según le asignes el valor visible o hidden. Debes tener en cuenta que, aunque un elemento no sea visible, éste continúa ocupando su espacio en el flujo normal del

documento al contrario de lo que ocurría con la propiedad **display** cuando se le asignaba el valor none.

z-index

Permite controlar el orden en el que se presentan los elementos que quedan solapados por efecto de otras propiedades. Si cuando definimos algún elemento con posición absoluta, éste tiene que visualizarse en el mismo lugar ocupado por otro elemento, se producirá una superposición de elementos visualizándose, en la parte coincidente, sólo el que está ocupando la "posición superior". La propiedad **z-index** permite especificar el orden en el eje z de los elementos, esto es, el orden de apilamiento en capas del documento.

Por defecto, los elementos se apilan en el orden en que aparecen: el elemento situado más abajo en el flujo normal del documento quedará encima. Si quieres que esta posición no sea tenida en cuenta, debes saber que los elementos con un valor mayor de la propiedad **z-index** son colocados encima de los que tienen un valor menor **z-index**, quedando estos últimos tapados por los primeros.

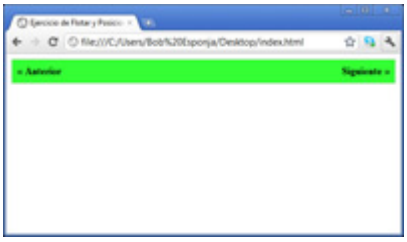
También debes saber que esta propiedad sólo se aplica a elementos que tengan la propiedad **position** absolute o en relative.

Ejercicio resuelto

A partir del siguiente código XHTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
<html xmlns="http://www.w3.org/1999/xhtml">
     <head>
          <meta http-equiv="Content-Type" content="text/html; charset=iso-8859
          <title>Ejercicio de Flotar y Posicionamiento</title>
          <style type="text/css">    </style>
     </head>
     <body>
          <div> &laquo; Anterior       Siguiente &raquo;  </div>
     </body>
</html>
```

Debes determinar las reglas CSS que pondrías en el elemento **style**, con las clases, identificadores y elementos que creas necesarios y modificar el código para que el resultado sea lo más parecido posible a la siguiente imagen:



6.- Cambiar la apariencia de otros elementos web.

Caso práctico

Como casi todas las web, la web de "Migas Amigas" tendrá imágenes y texto y otros elementos web como listas, tablas y formularios.

Para este tipo de elementos (X)HTML, el estándar CSS ofrece propiedades y técnicas específicas que permiten darles la apariencia más adecuada en cada momento.

Todo esto obliga a que Carlos posea los conocimientos necesarios para configurar la apariencia de todos estos elementos y en especial las listas, pues una parte importante de la web de "Migas Amigas" son los menús de navegación, un elemento al que han dedicado muchas hora durante la planificación del diseño.

Carlos se pone de nuevo en contacto con Ada para que ésta le guíe en el aprendizaje de las propiedades y técnicas necesarias.

En las páginas web existen muchos elementos específicos a los que debemos dar forma para así conseguir la apariencia que más nos interese.

Con la aparición de CSS, que permite la separación de la información de presentación del contenido, y con el modelo de cajas, que permite colocar cada elemento en el lugar deseado, te podrías preguntas ¿entonces por qué seguimos con las tablas?

Realmente con CSS, las tablas ya no son necesarias, al menos en lo que a la maquetación de los elementos principales se refiere. No son necesarias para indicar dónde queremos el encabezado o el pie del documento y tampoco son necesarias para hacer un diseño de dos o tres columnas. Pero, cuando se trata de colocar información de contenido en forma tabular, siguen siendo muy útiles.

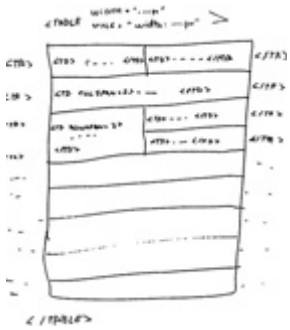
En los siguientes apartados verás aquellas propiedades que te permitirán conseguir personalizar elementos tan interesantes como las tablas, las listas y los formularios.

6.1- Las tablas con CSS.

Recomendación

A medida que vayas estudiando las propiedades de este apartado también puedes visitar el siguiente enlace de la web de la W3C donde encontrarás un resumen de las propiedades CSS más utilizadas para dar forma a las tablas.

Propiedades CSS para tablas



Para controlar la presentación de las tablas tenemos las propiedades:**caption-side**, **table-layout**, **border-collapse**, **border-spacing**, **empty-cells** y **display**.

caption-side

Esta propiedad sirve para indicar dónde se pone el título de la tabla. Puede tener los valores: top, bottom, left y right.

La recomendación CSS2.1 recoge la posibilidad de desplazar el contenido de la etiqueta **caption** a la izquierda o a la derecha con **text-align**, pero siempre manteniéndose por encima o por debajo de la tabla.

empty-cells

Esta propiedad soluciona la carencia del XHTML que, al no dibujar las celdas que estaban vacías, obligaba a poner un espacio en blanco usando el carácter ** ** . Los valores que admite son:

- show que permite mostrar los bordes y fondos como en las celdas con contenido.
- hide que permite ocultar los bordes y fondos de las celdas vacías.
- inherit que permite heredar el valor de `empty-cells` que tenga su elemento padre.

border-collapse

Permite establecer el modo en el que se dibujan los bordes de las tablas: separate (separados), collapse (juntos) e inherit. En el modo separate, cada celda está rodeada por su borde haciendo el efecto de un borde con una línea doble, mientras que, en el modo collapse las celdas contiguas comparten sus bordes.

border-spacing

Permite establecer la separación entre celdas contiguas. Para hacerlos se indica el valor del espaciado horizontal seguido del valor del espaciado vertical. Si se escribe un único valor, la separación horizontal y vertical serán iguales.

table-layout

Permite definir el modo en el que el navegador dibujará la tabla ya que puede hacerse de dos formas. Los valores que admite son:

- fixed dibuja la tabla basándose en las medidas establecidas en el código fuente. Con este valor se consigue que el sistema trabaje más rápido.
- auto dibuja la tablas basándose en el contenido de sus celdas. Es el valor por defecto.

Ejemplos:

```
table{     border:#000 solid thin;  /* Para que los bordes aparezcan */
          border-collapse: separate; /* Bordes separados */
            border-spacing: 0.5em 1em; /* Separación de los bordes */
           empty-cells: show;  /* Las celdas vacías se muestran */
             }
     table caption { caption-side: bottom;  /* El título de la tabla aparece debajo */ }
```

Recomendación

Una forma común de trabajar es usando plantillas que adaptamos a nuestro gusto. En el siguiente enlace accederás a la web de Christian Heilmann donde podrás encontrar una gran cantidad de plantillas de tablas creadas con CSS.

Galería de tablas formateadas con CSS.

Ejercicio resuelto

En el siguiente enlace podrás descargar el archivo que contiene el código fuente del que tendrás que partir para realizar el ejercicio propuesto.

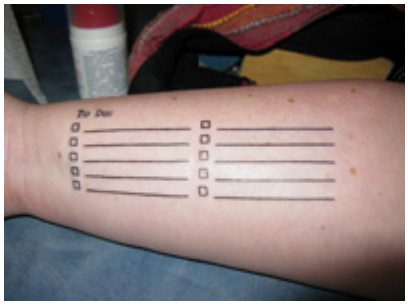
Código fuente (0.01 MB)

Una vez descargado el código XHTML puedes abrirlos con tu navegador. Verás que el resultado es el que se muestra en la imagen siguiente:

Tendrás que editar el código y determinar las reglas CSS necesarias para que el resultado sea lo más parecido posible al de la siguiente imagen (Nota: Aunque no sale en la imagen, el puntero está sobre la primera fila, por eso se ve de color amarillo):

6.2.- Las listas con CSS.

Las listas son un elemento muy utilizado en las páginas web. Hoy en día, su empleo no está limitado a la simple enumeración de elementos en el contenido, también se utilizan para crear barras de navegación verticales y horizontales.



Recomendación

A medida que vayas estudiando las propiedades de este apartado también puedes visitar el siguiente enlace de la web de la W3C donde encontrarás un resumen de las propiedades CSS más utilizadas para dar forma a las tablas.

Propiedades CSS para listas y contenido.

Para dar formato a las listas tenemos, entre otras, las propiedades: **`list-style-type`**, **`list-style-image`**, **`list-style-position`** y **`list-style`**.

`list-style-type`

Permite elegir el marcador visual de la lista asignando a la propiedad uno de los siguientes valores: none (eliminar el marcador), square (cuadrado), disc (círculo),circle (circunferencia), lower-roman (números romanos en minuscula), lower-alpha (letras en minúscula),upper-alpha (letras en mayúscula), etcétera.

`list-style-image`

Permite especificar una imagen como marcador. Para ello deberemos indicar la dirección o URL donde se encuentra la imagen.

Cuando se usa esta propiedad conviene declarar también la propiedad **`list-style-type`** en prevención de un fallo en la localización de la imagen.

Esto lo podríamos realizar, también empleando la propiedad background del elemento **`li`**. En este caso, mostrado en el ejemplo siguiente, debemos seguir los siguientes pasos:

- Eliminar previamente el marcador visual estableciendo none como valor de la propiedad `list-style-type`.
- Añadimos relleno a la izquierda de cada uno de los elementos de la lista.
- Colocamos de nuevo el marcador visual, declarando la propiedad `background` a la que asignaremos la URL de una imagen.
- Si cada elemento de la lista ocupa una sola línea, el marcador deberá centrarse verticalmente estableciendo su posición vertical al 50%.

Ejemplo:

```
ul {     margin: 0; padding: 0; list-style-type: none; }
     li {     background: url(boliche.gif) no-repeat 0 50%;
                    padding-left: 20px; }
```

list-style-position

Establece la posición del marcador de los elementos de la lista. Se puede colocar el marcador dentro del área de contenido con lo que todas las líneas de este elemento estarán alineadas por la izquierda (incluida la que lleva el marcador) o, se puede colocar fuera del área de contenido (como en una sangría francesa). Los valores que permiten posicionar el marcador son: **inside** (dentro) y **outside** (fuera).

list-style

Al igual que ocurría con otras propiedades que se vieron anteriormente, esta propiedad permite configurar las listas estableciendo, de forma abreviada y en cualquier orden, el valor de una o más de las propiedades individuales vistas en este apartado.

En el siguiente ejemplo se muestra el uso de las propiedades vistas hasta ahora.

Ejemplo:

```
ul {     list-style-image: url(boliche.gif);
                    list-style-type: disc;
                    list-style-position: outside; }
     ul.especial { list-style: outside circle url("imagenes/balon.png"); }
```

Un efecto muy utilizado para convertir una lista en una barra de menú de navegación es colocarla sus elementos dispuestos horizontalmente en la misma línea. El ejemplo siguiente muestra cómo hacerlo.

Ejemplo:

```
ul {      list-style-type: none;
          padding: 0;
               border: 1px solid #000;  }
li {      display: inline;
               border: 1px solid #00f; }
```

Si nos interesa eliminar los bordes simplemente debemos poner la propiedad **border** a 0.

Debes conocer

Es importante saber crear menús de navegación horizontales y verticales, en la siguiente presentación puedes ver con detalle cómo convertir una simple lista en un menú de navegación.

6.3.- Formularios con CSS

Los formularios son una parte esencial de toda página web en la que se busque interacción con los usuarios. Son muy parecidos a los formularios de papel tradicionales solo que en el caso de la web habrá que emplear el ratón o el teclado para rellenarlo en lugar de un simple bolígrafo.

Un formulario (elemento **`form`** del (X)HTML) pueden contener diferentes elementos:

- Elementos en los que el usuario tendrá que escribir: cajas de texto, áreas de texto.
- Elementos que el usuario podrá o no seleccionar: botones de opción, casillas de verificación, cuadros de listas.
- Elementos decorativos o descriptivos: etiquetas, textos.
- Elementos de agrupación de otros elementos.
- Elementos que permiten limpiar el formulario o enviar los datos para su procesamiento: botones de comando.

Con CSS puedes posicionar los elementos del formulario y configurar completamente su apariencia. Para ello, se emplean las propiedades vistas hasta ahora referentes al modelo de cajas, posicionamiento, flotado y las propiedades de fondo, texto, colores, etcétera.

Al igual que con las tablas, existen gran cantidad de plantillas para formularios que se pueden utilizar como base para nuestras webs. En el siguiente enlace puedes encontrar algunos ejemplos interesantes.

Ejemplos de formularios

Autoevaluación

¿Cuál de las siguiente reglas de estilo eliminan el marcador de los elementos de una lista?

- `ul {list-style-type: none; }`
- `ul li {list-style-type: none; }`
- `ul {list-style: none; }`
- `ul li {list-style: none; }`

Mostrar Información

7.- Introducción CSS3.

Caso práctico

Carlos está tan entusiasmado con la tecnología CSS que, ahora que ya conoce el estándar CSS2.1, el más compatible con los navegadores actuales, decide introducirse en el estudio de CSS3.

Carlos se da cuenta de que, con las novedades introducidas en CSS3, pronto podrá realizar en cualquier navegador algunos detalles que en estos momentos sólo se pueden realizar con la ayuda de algunos hacks de los navegadores y de los lenguajes de programación como JavaScript.

Carlos se ha dado cuenta de que la tecnología web evoluciona a un ritmo frenético y a medida que va aprendiendo más cosas se da cuenta de todo lo que le queda por aprender.

La especificación CSS3 trae grandes novedades para el diseño CSS.

Tal y como vimos en todo este tema, el objetivo principal de CSS es separar el contenido de la web de su apariencia. La novedad más importante que aporta CSS3, de cara a las personas que se dedican al desarrollo web, consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos o hacks, que a menudo complican el código de las web.

Ahora CSS3 va un paso más allá con el objetivo de ofrecer un control total sobre los elementos de la página y detalles como la oportunidad de crear diseños tan demandados como bordes redondeados o sombreado de elementos.

Algunas de las nuevas propiedades introducidas en CSS 3 podemos recogerlas en la siguiente lista:

- Para los bordes: `border-color`, `border-image`, `border-radius` y `box-shadow`.
- Para los fondos: `background-origin`, `background-clip` y `background-size`.
- Para el color en formatos HSL, HSLA y RGBA.
- Para el texto:`text-shadow` y `text-overflow` que permiten poner sombras y romper palabras largas al final de las líneas.
- Para las interfaces: `box-sizing`, `resize`, `outline`, `nav-top`, `nav-right`, `nav-bottom` y `nav-left`.
- Para el manejo de cajas: `overflow-x` y `overflow-y`.
- Nuevos tipos de selectores por atributos.
- Creación de múltiples columnas de texto.
- Propiedades orientadas al discurso o la lectura automática de páginas CSS. Muy importante para mejorar la accesibilidad de las páginas.

Para saber más

En el siguiente enlace encontrarás toda la información relevante referente a las novedades que tiene CSS3.

Novedades de CSS3.