

Conferencia #2 HTML y CSS

Objetivos:

- Conocer el trabajo con los formularios, como otro elemento básico del lenguaje HTML.
- Saber utilizar las hojas de estilos para dotar a todas las páginas de un sitio Web, de una apariencia homogénea.
- Identificar y saber utilizar los diferentes estilos para separar el contenido de un sitio de su presentación.

Contenido:

- Introducción.
- Formularios HTML.
- Hojas de estilo. CSS.
 - Ventajas de los estilos CSS
 - Tipos de estilos
 - Estructura de un estilo
 - Definición de selectores y propiedades
 - Algunas aplicaciones de los CSS
- Conclusiones.
- Motivación para la próxima clase.

Bibliografía:

1. Deitel & Deitel. "e-Business & e-Commerce. How to program." Capítulo 19. p.589-619.
2. Hernán Ruiz, Marcelo. "Programación Web Avanzada". Capítulos 2 y 4. Colección: Manuales USERS. Editorial: MP Ediciones. Buenos Aires, Argentina. ISBN: 987-526-115-7.

Introducción

Los formularios serán de mucha utilidad cuando se comience el trabajo con PHP o alguna tecnología del lado del servidor, por lo que en la presente clase se estudiará con detalles todas sus características y propiedades.

Un formulario HTML por sí solo no puede hacer mucho. Simplemente se limita a enviar los valores al servidor pero, si allí no hay nada que los procese, será inútil.

Los formularios constituyen un mecanismo para recolectar información de los usuarios.

Formularios HTML

Definición de un formulario

Para definir un formulario primero hay que crearlo utilizando las etiquetas <FORM> y </FORM> y luego incluir entre ellas las etiquetas que lo componen. Estos elementos pueden ser campos de textos, listas de selección, casillas de verificación, botones, etc.

La etiqueta <FORM> tiene la propiedad ACTION, que indica la acción que se debe efectuar, es decir, a qué página se enviarán los datos para procesarlos o qué se hará con los datos una vez que se pulse el botón “enviar”. También tiene la propiedad METHOD que define la forma en que se enviarán los datos, los valores que puede tomar esta propiedad son:

- GET: Los datos del formulario son enviados dentro de la cadena de consulta de la URL del navegador, de ahí que la cantidad a enviar esté limitada a 4K. Se utiliza cuando los datos a enviar no provocarán cambios en los datos almacenados, por ejemplo, cuando se hace una consulta a una base de datos.
- POST: Los datos son enviados a través de las cabeceras HTTP y son invisibles para el usuario. Es más complejo que el método *GET*, pero permite el paso de datos más complejos. Se utiliza cuando los datos enviados provocan cambios del lado del servidor, por ejemplo, cuando se actualiza una base de datos.

Existen también la propiedad NAME en el formulario, con la cual se puede definir un nombre para el formulario (muy útil para aplicaciones con Java Script, en la próxima clase se verá su utilidad)

El atributo ENCTYPE del formulario se utiliza para indicar el método de codificación. Los valores que puede tomar son:

- "TEXT/PLAIN": No se codifican los datos.
- "X-WWW-FORM-URLENCODED": Se codifican los datos.

A continuación se muestra una declaración típica de formulario:

```
<FORM NAME="frmDatos" METHOD="POST" ACTION="processData.jsp">  
  <!-- Elementos del formulario -->  
  ...  
</FORM>
```

Campos de Entrada

Para la introducción de las variables se utiliza la etiqueta <INPUT>. Esta etiqueta tiene el parámetro TYPE que indica el tipo de variable a introducir y NAME que indica el nombre que se le dará al campo. Cada tipo de variable tiene sus propios parámetros.

TYPE= text NAME = campo

Indica que el campo a introducir será un texto. Sus parámetros son:

MAXLENGTH = número (número máximo de caracteres a introducir en el campo)

SIZE = número (Tamaño en caracteres que se mostrará en pantalla)

VALUE = "texto" (Valor inicial del campo. Normalmente será " ", o sea, vacío)

El valor inicial que aparezca en el campo puede ser borrado por el usuario para anotar sus propios datos.

Ejemplo. Con el siguiente código:

```
<INPUT NAME="txtName" TYPE="text" VALUE="Sandra" MAXLENGTH="15">
```

Se muestra:

TYPE = password NAME = campo

Indica que el campo será una palabra de paso. Mostrará asteriscos (*) en lugar de las letras escritas. Sus parámetros opcionales son los mismos que para text.

Ejemplo. Con el siguiente código:

```
<INPUT NAME="txtPass" TYPE="password" VALUE="miClave">
```

Se muestra:



TYPE = checkbox NAME = campo

El campo se elegirá marcando una casilla. Se permite marcar varias casillas. Los valores de las casillas serán indicados por:

VALUE = "valor"

CHECKED (La casilla aparecerá marcada por definición)

Esta opción se utiliza para que el usuario seleccione distintas opciones, pueden ser una o más. Se suelen utilizar para la elección de productos a comprar, listas de correos a suscribirse, etc.

Ejemplo. Con el siguiente código:

```
<INPUT NAME="check" TYPE="checkbox" VALUE="1" CHECKED="checked">
```

Se muestra:



TYPE = radio NAME = campo

El campo se elegirá marcando una casilla. Sólo permite marcar una sola de las casillas. Los valores de las casillas serán indicados por:

VALUE = "valor"

Su uso es similar a los de tipo checkbox, con la diferencia de que se usan para elegir sólo una de las opciones del listado.

Ejemplo. Con el siguiente código:

```
<INPUT NAME="rdo" TYPE="radio" VALUE="1" CHECKED>
```

Se muestra:



TYPE = image NAME = campo

Se utiliza para crear un botón a partir de una imagen, una práctica muy habitual para dar a los botones un aspecto más vivo. Debe indicarse el camino de la imagen con el parámetro:

src = "fichero de imagen".

Ejemplo. Con el siguiente código:

```
<INPUT TYPE="image" NAME="btnAceptar" SRC="aceptar.gif">
```

Se muestra:



TYPE = hidden NAME = campo

El usuario no puede modificar su valor, ya que el campo no es visible, se envía siempre con el valor indicado por el parámetro:

VALUE = "valor"

Ejemplo: `<INPUT TYPE="hidden" NAME="hdn" VALUE="35">`

No se muestra nada en el navegador, dado que se trata de un campo oculto.

TYPE = submit

Representa un botón. Al pulsar este botón la información de todos los campos se envía a la página indicada en el ACTION del formulario `<FORM>`. Tiene el parámetro VALUE = "texto" que indica el texto que aparecerá en el botón.

Ejemplo. Con el siguiente código:

`<INPUT TYPE="submit" NAME="btnEntrar" VALUE="Entrar Datos">`

Se muestra:

TYPE = reset

Representa un botón. Al pulsar este botón se limpia el contenido de todos los campos y opciones que se hayan ingresado, dejando el formulario tan como estaba al comienzo. El parámetro VALUE = "texto" indica el texto que aparecerá en el botón.

Ejemplo. Con el siguiente código:

`<INPUT TYPE="reset" NAME="btnLimpiar" VALUE="Limpiar">`

Se muestra:

Campos de Selección

Este tipo de campo despliega una lista de opciones, entre las que se puede escoger una o varias. Básicamente su función es muy similar a la del checkbox y el radio, con la diferencia de que en el caso de los menús es posible ocupar poco espacio y colocar un mayor número de opciones. Es posible crear menús de selecciones simples o múltiples. Los de selección múltiple permiten elegir varias opciones a la vez utilizando la tecla CTRL. Se utilizan para crear este menú las etiquetas `<SELECT>` `</SELECT>`. Sus parámetros son:

NAME = campo (nombre del campo)

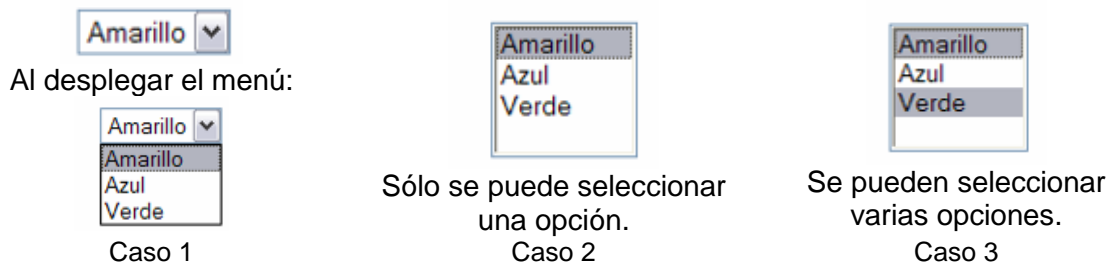
SIZE = numero

Número de opciones visibles. Si se indica 1 se presenta como un menú desplegable, si se indica más de uno se presenta como una lista con barra de desplazamiento.

MULTIPLE

Permite seleccionar más de un valor para el campo.

Las diferentes opciones de la lista se indican con la etiqueta <OPTION>. Esta etiqueta puede incluir el parámetro SELECTED para indicar cual es la opción por definición. En caso de que no se especifique, se tomará la primera opción de la lista.



Para el Caso 1 se escribe el siguiente código:

```
<SELECT NAME="selColor">
  <OPTION VALUE="1">Amarillo</OPTION>
  <OPTION VALUE="2">Azul</OPTION>
  <OPTION VALUE="3">Verde</OPTION>
</SELECT>
```

Para el Caso 2 se escribe el siguiente código:

```
<SELECT NAME="selColor" SIZE="4">
  <OPTION VALUE="1">Amarillo</OPTION>
  <OPTION VALUE="2">Azul</OPTION>
  <OPTION VALUE="3">Verde</OPTION>
</SELECT>
```

Para el Caso 3 se escribe el siguiente código:

```
<SELECT NAME="selColor" SIZE="4" MULTIPLE="multiple">
  <OPTION VALUE="1">Amarillo</OPTION>
  <OPTION VALUE="2">Azul</OPTION>
  <OPTION VALUE="3">Verde</OPTION>
</SELECT>
```

Áreas de texto

Representa un campo de texto de múltiples líneas. Normalmente se utiliza para que el usuario introduzca textos largos, por ejemplo un mensaje de correo. Las etiquetas usadas son <TEXTAREA> </TEXTAREA>, y sus parámetros:

NAME = campo (nombre del campo)

COLS = num. (Número de columnas de texto visibles)

ROWS = num. (Número de filas de texto visibles)

WRAP = virtual / physical

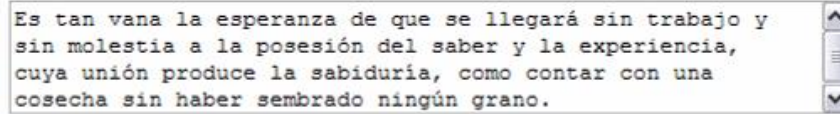
Justifica el texto automáticamente en el interior de la caja. La opción PHYSICAL envía las líneas de texto separadas en líneas físicas. La opción VIRTUAL envía todo el texto seguido.

Ejemplo. Con el siguiente código:

```
<TEXTAREA NAME="textarea" COLS="60" ROWS="4" WRAP="virtual">
```

</TEXTAREA>

Se muestra un área en la que se puede escribir:

A screenshot of a web browser's text area. The text inside is: "Es tan vana la esperanza de que se llegará sin trabajo y sin molestia a la posesión del saber y la experiencia, cuya unión produce la sabiduría, como contar con una cosecha sin haber sembrado ningún grano." The text is in a monospaced font and is right-aligned within the text area. There are scroll bars on the right side of the text area.

Hojas de estilo. CSS.

Las hojas de estilo en cascada (Cascading Style Sheet, CSS) se basan en un código estándar desarrollado por W3C mediante el cual es posible definir anticipadamente el estilo que tendrá cada etiqueta HTML en la página. Permite especificar el estilo de los elementos de una página, separado de la estructura del documento.

Ventajas de los estilos CSS

La utilización de los estilos CSS permite concentrar en un único lugar la definición de estilos de todos los elementos similares. Mediante un bloque de código situado dentro de la misma página o en un archivo separado, es posible controlar el diseño de cientos de elementos en lugar de dar formato a cada elemento de forma individual. De esta forma, cuando se decida realizar un cambio en el diseño y formato de la página bastará con hacer las modificaciones en un solo lugar, en vez de modificar cada elemento HTML por separado.

Tipos de estilos

1. Estilos “entre-líneas” (Inline)

Hay muchas maneras de declarar estilos para un documento. Una de ellas es a través del atributo STYLE. Este atributo permite especificar un estilo para un elemento determinado.

Ejemplos:

```
<P STYLE = "font-size: 20pt">Here is some more text</P>  
<P STYLE = "font-size:20pt; color: #0000FF">Even more text</P>
```

En el primer ejemplo, cada propiedad de la hoja de estilo (en este caso font-size) está seguida de dos puntos y después, del valor del atributo. En este caso se especifica que el elemento P (de esa línea solamente) tendrá un tamaño de letra de 20 puntos.

En el segundo ejemplo se especifican dos propiedades separadas por punto y coma. En esta línea además se especifica el color del texto, en este caso el azul, a través de código hexadecimal #0000FF.

Los estilos entre – líneas sobrescriben cualquier otro estilo aplicado, de los que se verán a continuación.

2. Estilos definidos dentro del documento con la etiqueta STYLE

Para definir estilos se puede utilizar la etiqueta <STYLE> con el atributo TYPE="text/css", de la siguiente manera:

```
<STYLE TYPE = "text/css">
...
</STYLE>
```

El atributo TYPE especifica el tipo MIME del estilo. MIME es un estándar para especificar el formato del contenido. Otros tipos MIME son text/html, image/gif, y text/javascript. De estos el más usado es el primero.

Esta definición de estilos se realiza dentro del encabezamiento de la página HTML, es decir, dentro de la sección HEAD, aunque la mayoría de los navegadores la interpretará bien si se coloca en otro lado.

Ejemplo:

```
<HEAD>
<TITLE>Style Sheets</TITLE>
<STYLE TYPE = "text/css">
    EM {background-color: #8000FF;
        color:white}
    H1 {font-family: Arial, sans-serif}
    P  {font-size: 18pt}
    .blue {color: blue}
</STYLE>
</HEAD>
```

El fragmento de código anterior declara las reglas para esta hoja de estilo, específicamente para las etiquetas EM, H1 y P.

Notar que cada regla se define dentro de los paréntesis.

La manera de declarar una clase de estilo (style class) es poniendo un punto (.) delante del nombre de la clase. En este caso: .blue

Las reglas que se definan en cada clase sólo se aplicarán a los elementos específicos de esa clase.

Las reglas dentro de una hoja de estilo como esta utilizan el mismo formato que en los estilos entre líneas: la propiedad es seguida por dos puntos (:), y luego su valor. Cuando se define más de una propiedad, cada una se separa de la otra a través de un punto y coma (;) como se puede ver en este ejemplo en el caso de reglas asociadas a la etiqueta EM.

La propiedad color especifica el color del texto en un elemento, la propiedad background-color especifica el color del background del elemento (como el atributo bgcolor en HTML).

Con font-family se especifica el nombre de la letra que será mostrada. En este caso, se utiliza la letra Arial. El segundo valor: sans-serif, se refiere a una familia de letras, lo cual permite definir un tipo de letra. De esta manera, si la letra Arial no se encuentra en el

sistema del usuario que visita el sitio, el navegador puede en su lugar mostrar otra letra de la familia sans-serif como Helvetica o Verdana. Otras familias de letras son serif (por ejemplo Times New Roman o Georgia), cursive (por ejemplo Script), fantasy (por ejemplo Critter) y monospace (por ejemplo Courier o Fixedsys).

Por último, con font-size se especifica el tamaño de la letra. En este caso de 18 puntos.

Ejemplo

```
<H1 CLASS = "blue">A Heading</H1>  
<P CLASS = "blue">Here is some <EM> more</EM>text.</P>
```

En este otro ejemplo, el atributo CLASS aplica una clase de estilo, en este caso blue (como fue declarado con .blue). En la página de ejemplo podrán ver que en el primer caso, el texto se muestra con ambas propiedades: las que por definición tiene un elemento H1 y las definidas en .blue.

En el segundo caso, las propiedades de P y de la clase .blue se aplican a todo el texto. Todas las propiedades aplicadas a un elemento (elemento padre) también se aplican a sus elementos internos (elementos hijos). La palabra dentro de la etiqueta EM hereda el estilo de P (la letra de tamaño 18), sin embargo entra en conflicto con el atributo color de la clase blue, pues los estilos declarados en el elemento hijo tienen prioridad sobre los del elemento padre. Por tanto el estilo de EM sobrescribe al de la clase blue.

3. Estilos definidos en un archivo externo

Para poder una hoja de estilo externa se puede usar cualquier editor de textos. El archivo deberá guardarse con extensión .css. Dentro del archivo se debe colocar solamente el código CSS, sin comentarios ni etiquetas HTML, por ejemplo:

```
body {background: #EEEEEE; color: green; font-family: verdana; font-size: 11px}
```

Las hojas de estilo externas permiten definir estilos para todo un sitio Web. Modificar el estilo de los diferentes elementos dentro de un sitio implica sólo cambiar el fichero .css.

El elemento LINK especifica la relación entre el documento en cuestión y otro documento, a través del atributo REL. En este caso, el documento que se relaciona es una hoja de estilo. El atributo TYPE especifica el tipo MIME como txt/css y el atributo HREF especifica la dirección URL donde se localiza el fichero css.

La etiqueta LINK sólo puede ubicarse en la cabecera de un documento HTML. A continuación se muestra un ejemplo:

```
<LINK REL = "stylesheet" TYPE = "text/css" HREF= "style.css">
```

El funcionamiento de este tipo de estilo será similar al anterior, sólo que este método es más recomendable cuando se tiene un sitio con varias páginas que comparten el mismo formato.

Estructura de un estilo

Para la creación de un estilo se debe seguir la siguiente estructura:


```
selector {propiedad_1: valor_1; propiedad_2: valor_2; propiedad_n: valor_n;}
```

Para mejor comprensión también se puede escribir de la siguiente manera:

```
selector {
  propiedad_1: valor_1;
  propiedad_2: valor_2;
  propiedad_n: valor_n;
}
```

Definición de selectores y propiedades

Los selectores pueden ser tanto etiquetas HTML como selectores propios creados por nosotros mismos. Para los nombres de los selectores se admiten caracteres dentro del rango de A – Z y 0 – 9.

En la tabla que se muestra a continuación se podrán encontrar algunos de los tipos de selectores que pueden crearse:

Selector	Significado
E	Afecta cualquier etiqueta de tipo E.
E F	Afecta cualquier etiqueta de tipo F que es descendiente de E.
E > F	Afecta a las etiquetas F que son hijas de E.
E:first-letter	Permite aplicar un estilo especial a la primera letra dentro del elemento
E[esto= "aquello"]	Afecta a las etiquetas E cuya propiedad "esto" tiene valor "aquello"
.algo	Para cualquier elemento que tenga su propiedad class = "algo"

A continuación se muestra un ejemplo. Si se desea modificar todos los párrafos, de manera que utilicen una fuente Arial con un tamaño de 12 px, se escribe:

```
P {font-family: Arial; font-size: 12px}
```

En el próximo ejemplo se verá el trabajo con etiquetas descendientes de otras. Una etiqueta es descendiente de otra si se encuentra en un nivel inferior, en el siguiente caso, la etiqueta <P> es descendiente de <DIV>, de y de :

```
<DIV>
  <UL>
    <LI><P>Esto es un texto.
    <LI><P>Esto es otro texto.
  </UL>
</DIV>
```

Si se desea que el estilo afecte solamente a las etiquetas <P> que descienden de se puede escribir lo siguiente:

```
UL P { /* Definición del estilo */ }
```

En la siguiente definición se especifica diferentes formatos para los hipervínculos en los estados normal (link), activo (active), con el mouse encima (hover) y visitado (visited).

A: link {color: blue; text-decoration:none;}

A: active {color:red}

A: hover {color:orange; text-decoration:underline;}

A: visited {color:#AAAAAA; text-decoration:none}

CSS define propiedades que pueden aplicarse mediante la sintaxis tradicional de HTML aunque otras sólo se pueden implementar mediante hojas de estilo. En CSS, cuando se especifica algún número como valor, es imprescindible especificar su unidad, de lo contrario, esa propiedad no será tomada en cuenta. Las unidades pueden ser relativas o absolutas. A continuación se ven las utilizadas en CSS2:

Relativas:

- em: tamaño relativo al de la fuente utilizada en el elemento en que se aplica.
- xm: tamaño relativo tomando como referencia la altura de las letras minúsculas de la fuente.
- px: píxeles, unidad relativa a la resolución de pantalla.

En el caso de em, si se usa para especificar el tamaño de una fuente (por medio de la propiedad font-size), se toma como referencia el tamaño de la fuente del elemento padre.

Absolutas:

- in: inches (pulgadas). 1 pulgada equivale a 25.4 milímetros.
- cm: centímetros
- mm: milímetros
- pt: points (puntos). En CSS2 equivale a 0.35 milímetros o 1/72 pulgadas.
- pc: picas. 1 pica equivale a 12 puntos, 4.23 milímetros o 1/6 de pulgada.

Algunas aplicaciones de los CSS

1. Elementos de posición

Controlar la posición de los elementos en HTML puede resultar difícil. En la clase pasada vimos cómo a través de tablas se puede alinear la información que se muestra en la página, de forma más conveniente para nosotros.

CSS introduce la propiedad POSITION, que puede darnos un mayor control sobre cómo ubicar los elementos dentro de una página. Los posibles valores pueden ser:

- Static: El bloque se comporta como una caja normal, siguiendo el flujo de contenido.
- Relative: La caja se ubica en relación con el flujo normal de contenido.
- Absolute: La posición es especificada utilizando las propiedades left, right, top, bottom. La caja se ubica en un lugar fijo dentro del documento.
- Fixed: la posición se especifica de igual forma que en el caso de absolute pero, adicionalmente, la caja queda fija aun al desplazar la página. La caja se ubica en relación a la ventana del navegador.

2. Cajas flotantes

Una caja flotante es un recurso muy interesante para crear recuadros de cualquier clase. Las mismas pueden ser colocadas a la derecha o a la izquierda y permitir, o evitar (mediante la propiedad CLEAR) que el contenido fluya por sus costados.

Una caja flotante se crea mediante la propiedad FLOAT y sus posibles valores son: LEFT (izquierda), RIGHT (derecha) y NONE (no flota).

Ejemplo:

```
<HTML>
  <HEAD>
    <TITLE>Revolución haitiana</TITLE>
    <STYLE TYPE = "text/css">
      DIV.figura {float: right; width: 25%; border: thin silver solid;
        margin: 0.5em; padding: 0.5em; text-align:center; }

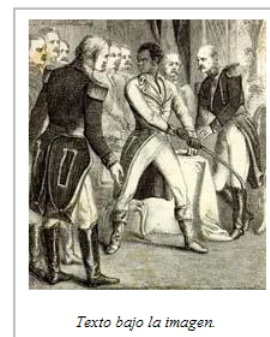
      P.epigrafe {font-style: italic; font-size:smaller; text-indent: 0; }
    </STYLE>
  </HEAD>

  <BODY>
    <DIV CLASS = "figura">
      <P></P>
      <P CLASS = "epigrafe">Texto bajo la imagen. </P>
    </DIV>

    <H1>Ejemplo de cajas flotantes</H1>
    <P>Hubiera querido escribir... </P>
  </BODY>
</HTML>
```

Ejemplo de cajas flotantes

Hubiera querido escribir la conferencia que voy a ofrecerles, pero la vida tuvo otros designios y, por tanto, prácticamente voy a improvisar a partir de algunas notas que he podido tomar y de muchísimas lecturas que hace años estoy haciendo, conmovido como estoy por la extraordinaria historia de Haití, país que visité en 1997, recorriendo el camino que iluminó a Alejo Carpentier cuando en su memorable viaje de 1943 tuvo la revelación —otra palabra no es posible— de muchos secretos y realidades de nuestra América. En rigor, como se ha dicho aquí, no vamos a conmemorar el Bicentenario de la Revolución de Haití (que comenzó en 1791, cuando el país se llamaba aún Saint-Domingue), sino su triunfo, el triunfo de esa Revolución, el cual hizo posible la independencia del país, proclamada el primero de enero de 1804, cuando sus libertadores, de la noche a la mañana, en un relámpago, le devolvieron su nombre aborígen. Creo que hasta ahora no se sabe de quién fue esta feliz idea, que se propuso borrar incluso verbalmente el atroz pasado colonial. Tales libertadores no eran aborígenes, pero tampoco europeos. Eran de procedencia africana, y decidieron, calibanescamente, hermanarse con la herencia de los primeros habitantes de su isla, los primeros humillados y ofendidos, los primeros oprimidos (hasta el exterminio), tras la segunda llegada a nuestras tierras de europeos: llegada que, absurdamente, fue llamada descubrimiento. En 1492, las dos ciudades más pobladas del mundo se llamaban Tenochtitlan y Pekín, y según lo que sé ninguna de ellas se encontraba en Europa. De manera que llamarle descubrimiento a la llegada de un grupo de europeos a un continente donde había millones de habitantes es una aberración. En realidad, merece ser llamada un cubrimiento de la historia verdadera. Sin embargo, aquella llegada tuvo, sin duda, trascendencia, ya que hizo posible lo que el gran historiador francés Fernand Braudel llamaría la mundialización, palabra que se hizo después muy conocida; hizo posible el nacimiento de la modernidad. Y esa llegada —aunque no se suele subrayar bastante— ocurrió en el Caribe, que devendría una de las grandes encrucijadas geográfico-históricas de la humanidad. La Revolución que condujo a la independencia de Haití, hará pronto doscientos años, fue el primer y magno acontecimiento en que el Caribe apareció del todo como actor en el planeta. Y fue el pórtico de la independencia de nuestra América.



Texto bajo la imagen.

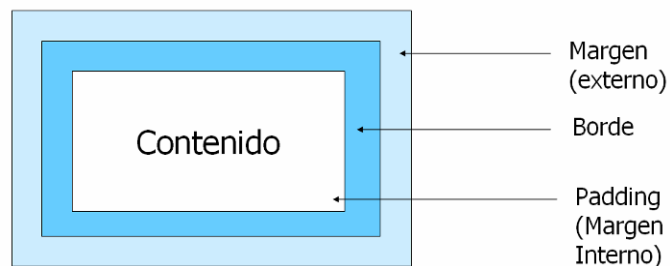
Extracto de "BICENTENARIO DE LA INDEPENDENCIA DE HAÏTÍ. Una esponja empapada en sangre."

De Roberto Fernández Retamar, extraído de *La jiribilla*.

Otra propiedad referente a las capas es z-index. Esta permite definir la “altura” de la capa sobre la página y con respecto a otras. Así, por ejemplo, una capa con z-index:1 estará por debajo de otra con z-index:2.

3. Modelo de cajas

Permite especificar atributos como el color de fondo del contenido, bordes y propiedades, margen interno (desde el contenido hasta el borde), externo (desde el borde hasta el próximo elemento). Las cajas pueden ser tanto celdas de tablas como texto o imágenes. La siguiente figura muestra los distintos elementos del modelo de caja:



Es posible determinar las propiedades de los bordes de una vez para todos los bordes por igual o especificar características diferentes para cada uno de los cuatro bordes.

Márgenes: Para especificar el margen de cada uno de los bordes se utiliza margin-top, margin-right, margin-bottom y margin-left. El valor puede ser un valor fijo o un porcentaje del ancho de la caja.

Margen interno: Se refiere a la distancia que existe desde el contenido hasta el borde de la caja. Si el borde es cero, entonces el límite del borde coincide con el límite del margen interno. Si el margen interno es cero, el límite del contenido coincide con el límite del margen interno, que es donde comienza el borde. Su declaración es similar al margen, pero utilizando las siguientes propiedades para definir cada lado independientemente: padding-top, padding-right, padding-bottom y padding-left.

Borde: Las propiedades del borde son: ancho (width), color (color) y estilo (style). Se debe tener cuidado y aplicar las propiedades en ese orden ya que el navegador así lo requiere.

Ejemplo:

```
<STYLE TYPE = "text/css">
  BODY {
    background-color:#CCFFCC
  }
  DIV {
    text-align: center;
    margin-bottom: 1em;
    padding: .5em
  }
  .thick {border-width: thick}
  .groove {border-style: groove}
```

</STYLE>

<DIV CLASS = "thick groove">This text has a thick and groove border</DIV>

Conclusiones

En la clase de hoy se estudiaron los formularios, los cuales constituyen un mecanismo para recolectar información de los usuarios. Se analizó con detalle cada uno de los elementos que componen un formulario, cómo se utilizan, y cómo deben lucir en el navegador.

También se estudiaron las hojas de estilo, las facilidades que incorporan: esencialmente la manejabilidad del estilo de los documentos y la gestión de cambios. Se estudiaron los distintos tipos de estilos que existen, cuál es la estructura general de un estilo, los selectores y algunas de las formas de utilizar los CSS.

Motivación para la próxima clase

En la próxima clase se verá de manera práctica el trabajo con formularios y la creación de estilos, tanto propios de un documento, como definidos en un archivo externo.