Documentación técnica:

Conceptos básicos

Índice

1. Conceptos básicos.....5

1. Conceptos básicos

Los computadores tienen una gran repercusión en el mundo actual. No sólo por su capacidad de cómputo, sino porque sirven para almacenar y recuperar datos, para procesar información, para manejar redes de comunicación. Hoy en día hay computadoras integradas en los controles de navegación de losvehículos, en las televisiones, en los sistemas domóticos de las viviendas, en los electrodomésticos, en los puestos de control, en las fábricas y líneas de producción... Con la miniaturización de la electrónica, cualquier dispositivo puede acabar conectado a Internet gracias a un micro-procesador embebido, es el paradigma de Internet de las Cosas (IoT-Internet of Things).

Cualquier sistema informático está compuesto por tres partes principales:

- Hardware: son los componentes físicos, por ejemplo, las placas integradas, los micro-computadores, las pantallas, los periféricos, los cables, las baterías o fuentes de alimentación.
- Software: son los programas, las aplicaciones, y toda la información relacionada con ellos.
- Usuarios: son la componente humana del sistema. La interacción de las personas con los sistemas que llevan computadoras integradas es un aspecto fundamental en nuestra sociedad de hoy en día, por ello es tan importante realizar programas de calidad que sean fáciles de usar y accesibles para cualquier persona.

La palabra software se considera sinónimo de programa informático. El software de sistemas es el que permite a los micro-procesadores comunicarse con el resto del hardware, por ejemplo, los sistemas operativos. Así, por ejemplo, en un ordenador el sistema operativo es la conexión del hardware con el resto de programas instalados (para imprimir con la impresora, grabar vídeo con la cámara, mostrar imágenes y documentos en la pantalla). El software de aplicación es el que generalmente utiliza el usuario, como una hoja de cálculo, un procesador de textos, un programa de dibujo, un navegador de Internet o un programa de juegos.

La programación de computadoras consiste en, dado un problema, resolverlo fabricando una aplicación informática que consiga automatizar todos los procesos involucrados. Los aspectos más relevantes del desarrollo del software son:

- Portabilidad: el software debe funcionar independientemente de la máquina o sistema operativo en el que esté instalado.
- Productividad: es necesario simplificar los procesos de desarrollo y fabricación de nuevas aplicaciones. Ello se puede conseguir reutilizando código y componentes de la mejor forma posible.
- Mantenimiento: es la etapa generalmente más costosa del desarrollo de software. Es fundamental estructurar bien el programa en módulos para prever futuros cambios, modificaciones o revisión de errores para intentar optimizar su coste.
- Calidad: no hay que olvidar los aspectos de control de la calidad, control de versiones y programación a la defensiva para tratar los errores y excepciones de la mejor forma posible.

En el paradigma clásico, el de la programación Estructurada, el desarrollo de aplicaciones consiste en tomar un problema y descomponerlo en sub-problemas más pequeños hasta que éstos sean abordablesinformáticamente. Los principales problemas de esta forma de programar son:

- ☐ La creciente complejidad de las aplicaciones que produce descomposiciones en infinitas partes para programar y la dificultad de hacer un código cohesionado y desacoplado.
- ☐ Limitaciones para resolver problemas complejos no estructurados difíciles de descomponer y modularizar.
- ☐ Difícil reutilización del software porque la división en sub- problemas no conduce realmente a que cada módulo independiente sea reutilizable en sí mismo.
- ☐ Mantenimiento costoso y de difícil ejecución ya que a menudo se introducen ampliaciones y se desarrollan nuevos entornos en los que trabajar. Es frecuente encontrar programas mal estructurados en los cuales es difícil incorporar nuevos módulos o estructuras de datos.

Por ello, en los últimos 30 años otro paradigma, la programación Orientada a Objetos (OO), ha tomado el liderazgo en el desarrollo de aplicaciones. Generalmente se asocian a esta forma de programar las técnicas utilizadas para fabricar las interfaces gráficas de usuarios, con ventanas, iconos, formularios y botones. En parte es cierto, ya que se utilizan técnicas OO para construir estos entornos.

1. Conceptos básicos

La programación OO implica un cambio de filosofía en el planteamiento de la forma de resolver los problemas mediante la programación. Se basa en utilizar y definir clases y objetos que van a tener propiedades, comportamientos y que van a interrelacionar entre sí. Por ejemplo, si se quiere programar un juego basado en un deporte como el fútbol, la pelota podría ser un objeto que contiene variables que determinan su forma y posición en el campo, y habría que programar su comportamiento con algoritmos que la movieran y dibujaran en pantalla, calcularan sus desplazamientos, etc. Igualmente sucedería para los jugadores o incluso el árbitro. Serían objetos con sus atributos (nombre, equipo al que pertenecen, número de posición, altura, fuerza de patada) y también se podrían programar sus movimientos por el campo y la forma de interactuar con el balón y con otros jugadores (del mismo equipo o del contrario). Para ello habría que definir una serie de clases (clasePelota, claseJugadorFutbol o clase Árbitro) de las que se pudieran instanciar los objetos reales que son los que actuarán en el juego.

Documentación técnica:

Fundamentos de la programación

Índice

1. Programas, datos y algoritmos.....5
2. Ingeniería del software.....7

1. Programas, datos y algoritmos

Un programa informático consiste en una serie de instrucciones que indican al computador de forma precisa lo que debe hacer, consiguiendo que la máquina resuelva el problema que nos interesa. Para lograrlo, el programa debe estar escrito en un lenguaje que entienda el computador, y que denomina lenguaje-máquina. Sin embargo, esos códigos resultan muy crípticos y poco entendibles para los seres humanos. Por eso, los programadores escriben los programas en lenguaje pseudo-natural, que comprenden ellos, con lenguajes apropiados como Java, C, Python, PHP u otros muy conocidos, y luego utilizan herramientas que convierten las instrucciones en estos lenguajes al lenguaje que realmente entienden las máquinas. Estas herramientasse llaman compiladores.

A continuación, se presenta un pequeño trozo de código escrito en lenguaje Java que lee y reproduce en pantalla la frase introducida por el usuario a través del teclado de un ordenador.

```
class Ejemplo {

// Código de ejemplo de un programa escrito en lenguaje Java

        public static void main (String[] args) {

        // Declaración de variable para guardar la frase como cadena de caracteres

        String frase;

        // Mensaje de aviso al usuario

        System.out.println ("Introduzca una cadena de caracteres por teclado: ");

        // Detecta y almacena todos los caracteres introducidos por el teclado hasta el
        // fin de línea y se guardan en la variable mensaje

        frase = read.Line();

        System.out.println("Ha escrito lo siguiente: " + frase);

        □ // fin del método main (principal)

□ // fin de la clase Ejemplo
```

1. Programas, datos y algoritmos

Los programas suelen aceptar unos datos de entrada, por ejemplo, cuando seleccionamos una fecha y un destino para comprar un billete de avión, que luego procesan mediante unos algoritmos. Para una agencia de viajes on-line, los algoritmos serían la búsqueda de precios más baratos, las compañías aéreas más seguras, la disponibilidad de plazas libres suficientes en las fechas escogidas. Cuando el programa resuelve el algoritmo, se da una respuesta al usuario, siguiendo con el ejemplo, la posibilidad de comprar los billetes, o bien el computador proporciona unos datos de salida que a su vez podrán ser procesados de nuevo por otro programa, por ejemplo, para comprar con otra compañía un seguro de viajes.

Los algoritmos son parte fundamental de los programas. Cada uno de ellos está formado por un conjunto finito de líneas de código, de instrucciones, que tendrán que estar muy bien definidas y que deberán acabar en un tiempo también finito. No tiene sentido escribir algoritmos que duren una eternidad, la idea básica es que los algoritmos son como una receta práctica para resolver un problema concreto.

2. Ingeniería del software

En los inicios de los computadores, el hardware no era muy potente y por ello el software no era demasiado complejo. El desarrollo de software era casi un arte, y su calidad final dependía en gran medida de la experiencia y habilidad del programador. Poco a poco, las cosas fueron cambiando y los sistemas informáticos se hicieron grandes y complejos. Las planificaciones de tiempo y coste demostraron que las técnicas artesanales ya no servían y en los años 90's se vivió una gran crisis del software que finalizó incorporando al desarrollo del software parte de las técnicas ingenieriles utilizadas en las fábricas. Así, se definió el término Ingeniería del Software como la aplicación sistemática, disciplinada y cuantificable al desarrollo, operación y mantenimiento del software.

Con este nuevo punto de vista, el desarrollo de software trasciende de la mera programación. La ingeniería del software considera un ciclo de vida que va desde la detección y análisis del problema que se quiere informatizar, la concepción y diseño del programa, su implementación en código que entienda la máquina, la realización de pruebas y detección de errores, su uso y su mantenimiento, como se muestra en la siguiente figura.

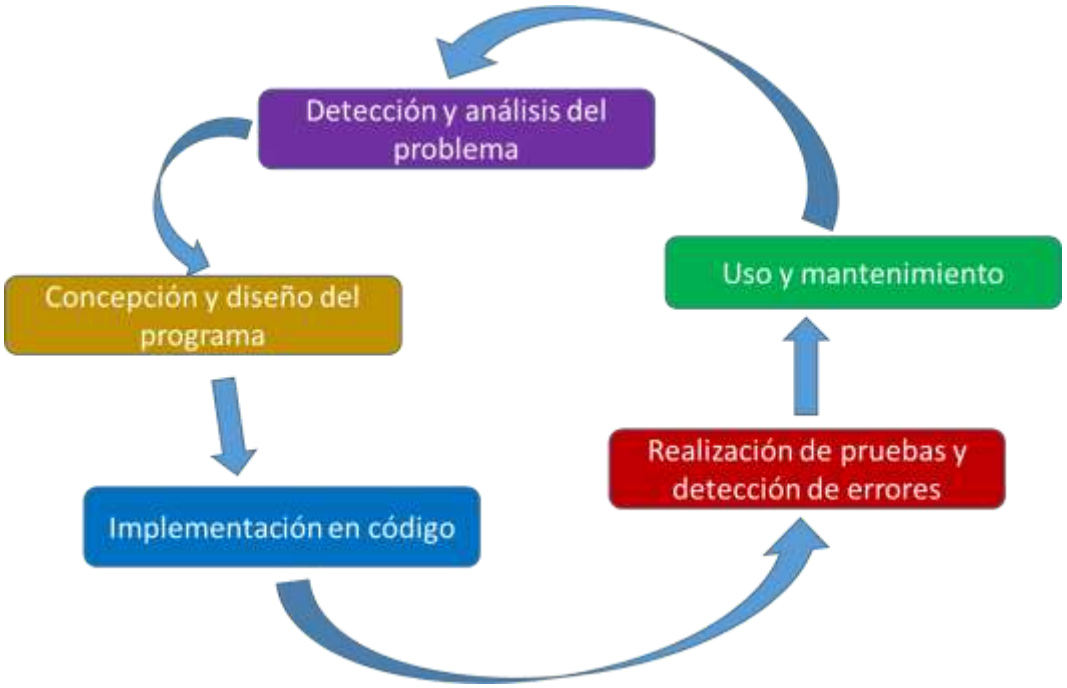


Figura 1. Fases dentro del ciclo de vida del software

2. Ingeniería del software

Un programa informático se considera eficaz si consigue los objetivos deseados y funciona correctamente. En cambio, un programa será eficiente si consigue los resultados consumiendo los menores recursos y tiempo posible.Un programa no solo es bueno por funcionar bien, sino que será mejor cuántos menos recursos utilice.

Al escribir un programa siempre se producen errores y nunca se pueden evitar del todo. Por ello, un proceso fundamental en la ingeniería del software es la depuración de los errores y considerar que hay que realizar operaciones de mantenimiento con una determinada frecuencia. Para facilitar el trabajo, se pueden incluir fases específicas que estén ya sistematizadas. Los errores se pueden clasificar en distintos tipos y es muy importante intentar eliminarlos en el proceso de desarrollo del software:

1. Errores de compilación: estos errores son los más fáciles de corregir ya que son detectados por los programas compiladores. Pueden ser errores de sintaxis en las instrucciones o uso de tipos de datos incompatibles.
2. Errores en tiempo de ejecución: aunque el código del programa compile bien y no dé problemas puede ocurrir que dé un error grave al ejecutarse y el programa se pare y deje de funcionar. Por ejemplo, que se produzca una división por 0 (el valor infinito no existe para las máquinas) o se escriba mal en un fichero del disco duro. Algunos de estos errores pueden ser recuperables en tiempo de ejecución y entonces se denominan excepciones. Para ello, los programas deben contener subrutinas específicamente diseñadas para atender cada tipo de excepción.
3. Errores lógicos: este caso se produce cuando no hay errores de compilación ni de ejecución, pero el programa produce un resultado incorrecto, que no es el esperado o incluso que es imposible. Por ejemplo, el precio de un billete de avión en cifras negativas (menor que cero). Estos errores son los más importantes pero los más difíciles de descubrir ya que habitualmente exigirá revisar con mucho cuidado grandes trozos de código de la aplicación contrastando resultados intermedios, que pueden suponer miles de líneas de programación.

Documentación técnica:

Estructuras de control y de datos

Índice

1. Tipos de datos e información estructurada.....5
2. Estructuras de control.....8

1. Tipos de datos e información estructurada

Cualquier programa puede considerarse como un conjunto de operaciones que se realizan en un orden determinado sobre una serie de datos. Cada lenguaje de programación define los tipos de datos permitidos y las operaciones disponibles sobre ellos. La idea de dato, por tanto, está muy ligada a las operaciones concretas que se pueden realizar sobre él. Un ejemplo sería un dato numérico que se puede representar como entero o real, internamente se guardan de forma diferente y las operaciones que se pueden ejecutar sobre cada tipo también son distintas.

Un dato es, por tanto, un conjunto de símbolos que representa un valor alfanumérico. Por ejemplo, un número de teléfono, la altura de una persona o el número del carnet de identidad. El término información se refiere a un conjunto de datos, ya procesados, organizados y que están relacionados. Por ejemplo, son datos individuales el nombre de una persona y su fecha de nacimiento, su salario y el nivel de estudios. Sin embargo, se pueden agrupar en una nómina y, organizados por fecha de nacimiento y salario, ofrecen la distribución de sueldos en función de la edad del trabajador. Organizando los mismos datos por nivel de estudios y salario, se obtendrá la distribución de salarios en función de la formación de los empleados.

La representación digital de los datos es necesaria para poder procesarlos y organizarlos. En las computadoras, la información se divide en trozos y cada trozo se representa numéricamente de forma individual, y lo que se maneja al final es ese conjunto de números. En los ordenadores toda la información está almacenada digitalmente: los números, los textos, los audios, los vídeos. Cada carácter de texto (cada letra o símbolo) se representa por un código numérico. Un conjunto de códigos muy conocido y utilizado es el código ASCII (American Standard Code for Information Interchange), pero tiene varios inconvenientes que, aunque codifica muy bien el idioma inglés, sin embargo, el conjunto de caracteres es muy limitado y le faltan muchos caracteres acentuados utilizados en otros idiomas. Por ello, más recientemente se utilizan otros códigos más completos y extensos como Unicode (UTF-8 y UTF-16) que pueden representar cualquier carácter incluso de los idiomas chino, japonés y coreano, así como cualquier mensaje codificado con ASCII.

Para representar números o colores, resulta conveniente por su simplicidad utilizar un sistema básico de 2 dígitos: 0 y 1, que se llaman bits. Estos valores se

representan de forma muy fácil con un circuito electrónico básico y por ello son la base de la representación de los datos en los sistemas informáticos como se verá más adelante.

Los tipos de datos primitivos son los mismos en todos los lenguajes de programación:

1. Numéricos: pueden ser enteros o reales. Los enteros se pueden representar a su vez de cuatro formas diferentes (byte, short, integer y long) según la cantidad de espacio en memoria que se quiera utilizar para guardarlos. El tipo escogido dependerá del valor máximo que necesitemos para ese dato. Los reales se representarán con coma flotante (para indicar la parte decimal) con 7 dígitos significativos (float) o 15 dígitos significativos (double).
2. No numéricos: serán de tipo carácter o lógicos (llamados booleanos). El tipo carácter se almacena en un símbolo alfanumérico y se identifica como char. Cada valor de tipo char almacena un carácter simple Unicode, y pueden ser letras minúsculas, letras mayúsculas, signos de puntuación, números, símbolos especiales y algún carácter de control (nueva línea o intro, por ejemplo). El tipo lógico se conocen también como booleanos por su término en inglés. Tienen dos posibles valores: verdadero (true) o falso (false).

Las variables, son nombres de huecos de memoria que se utilizan en programación para guardar datos y deben ser declaradas en el programa antes de que se utilicen. Habitualmente, al mismo tiempo que se declaran, se suelen inicializar a un valor determinado (por ejemplo, 0) para partir de un primer valor conocido y que sirva de referencia según el programa se va ejecutando. Un ejemplo de declaración e inicialización de tres variables numéricas distintas en Java sería:

```
int total=0, cuenta=20;

float precioItem = 25,4;
```

1. Tipos de datos e información estructurada

Las constantes son, en programación, entidades similares a las variables, con la diferencia de que su valor no cambia nunca y si se intentan modificar en el programa se produce un error.

2. Estructuras de control

Los programas resuelven problemas utilizando una serie de instrucciones organizadas en forma de algoritmos, de forma que le indican al computador qué hacer en cada momento.

Las instrucciones de asignación sirven para dar valores a las variables, colocando en su interior un valor determinado. Es importante notar que el signo = no representa una igualdad en programación, sino el efecto de colocar un valor en la variable que aparece a su izquierda. Así, la instrucción:

acumulado = 3;

Guarda un valor de 3 en la variable "acumulado". La sentencia:

acumulado = acumulado + 5;

produciría que la variable "acumulado" guardara un valor de 8.

Las instrucciones de entrada/ salida recogen datos desde el teclado o desde un fichero de la memoria y los muestran en la pantalla del ordenador.

Las instrucciones de control pueden variar el flujo normal de instrucciones, por medio de bucles, iteraciones o condiciones:

- Bucles: son útiles cuando se quiere repetir la misma instrucción o bloque de instrucciones varias veces en el mismo programa. Pueden repetirse varias veces mientras se cumpla una determinada condición.

  Con el bucle tipo while, se repite un bloque de sentencias mientras se cumple una determinada condición, es decir, mientras sea verdadera. En el momento en que la condición se vuelve falsa, ya no se repite el bloque de sentencias y el programa avanza hasta la siguiente instrucción.

  while (condicion){

  //Bloque de sentencias

  }

  Con el bucle tipo do …while, se asegura que el bloque de sentencias se ejecuta al menos una vez, puesto que la condición de mantenimiento del bucle se evalúa al final de la primera ejecución. Su sintaxis sería:

```
do {
//Bloque de sentencias
□ while (condicion);
```

- Iteraciones: el bloque de sentencias se ejecuta un número fijo de veces. Es un tipo de bucle que está controlado por un contador, en cada vuelta se incrementa el contador de uno en uno, y el bucle se para cuando el contador llega a su valor límite. Este tipo de iteración se utiliza cuando se conoce con precisión el número de veces que debe ejecutarse el bloque de sentencias.

```
for (inicialización; condición; incremento) {
// Bloque de sentencias
}
```

- Condiciones: se llaman también sentencias de selección o decisiones ya que evalúan una condición y luego, según el resultado, deciden qué bloque de sentencias de entre varias ejecutan. La palabra reservada if (si condicional, en inglés) se utiliza de forma conjunta con else, de la siguiente forma:

```
if (condicion) {
// Primer bloque de sentencias
□ else {
// Segundo bloque de sentencias
}
```

Si la variable condición es verdadera se ejecuta el primer bloque de sentencias, si es falsa, el segundo.

Documentación técnica:

Funciones y programas básicos

Índice

1. Funciones.....5
2. Programas Básicos.....8

1. Funciones

Hay tres acciones elementales en programación: secuencia, selección y bucle.

Secuencia es una lista simple, consecutiva de acciones. El orden está implícito en el orden de las instrucciones individuales. Por ejemplo:

```
int contador=0; // inicialización

contador = contador +1;

System.out.println (contador); // escribe en pantalla el valor de contador
```

Las tres instrucciones se ejecutarán una tras otra empezando por la inicialización, luego el incremento de 1 en la variable contador y luego se escribirá en pantalla.

El mecanismo de selección básico es la sentencia if- else, que actúa según el valor de una condición. Se pueden encadenar varias sentencias de selección de dos formas distintas:

- Por concatenación: una a continuación de la otra. Por ejemplo, una óptica hace un descuento del 10 % si el gasto por la compra de gafas es mayor que 100 euros. Si el cliente compra más de dos gafas, se le regala las fundas. El código en Java correspondiente sería algo así:

```
if (precio>100) {
        precio=precio*0.9;
        }
if (cantidad>2) {
        regaloFundas;
        }
```

- Por anidamiento: una dentro de otra. Siguiendo con el ejemplo anterior, la óptica hace un descuento del 10 % si el gasto por la compra de gafas es mayor que 100 euros. Pero si una de las gafas compradas es de sol, la tienda le añade otro 10% más de descuento, porque están en promoción. El código en Java correspondiente sería algo así:

```
if (precio>100) {
        precio=precio*0.9;
        if (gafasSol==true){
                precio=precio*0.9;
                }
        }
```

En los lenguajes modernos existe una posibilidad de realizar selección múltiple. En función del valor de una variable seguiremos un camino u otro, el comportamiento es equivalente a una serie de if-anidados. Para ello, existe una sentencia especial que se llamaswitch. Su sintaxis es:

```
switch (expresion) {
        case Valor1:
                // Bloque sentencias 1
                break;
        case Valor2:
                // Bloque sentencias 2
                break;
        case Valor3:
                // Bloque sentencias 3
                break;
        ....
        default:
                // Bloque sentencias por defecto
        }
```

1. Funciones

Otra forma de alterar el flujo secuencial de instrucciones es utilizando bucles, los cuales permiten repetir las mismas instrucciones varias veces. En los programas es muy habitual encontrar estos bloques de iteración o repetición. El bucle más directo se implementa con la palabra while, que repite un bloque de sentencias mientras se mantiene cierta condición. Si la condición es falsa desde el principio, es decir, que no va a suceder desde el principio, no se ejecutará ninguna instrucción de todo el bloque contenido dentro del while. El otro tipo de bucle, el que comprueba la condición al final del bloque de sentencias, es el tipo do- while. Este tipo de bucle se ejecuta al menos una vez ya que la condición se ejecuta al final del bloque de sentencias.

Por último, se encuentran los bucles controlados por un contador. El bucle for se utiliza cuando se conoce el número de veces que se tiene que repetir el bucle. En un bucle for, se ejecuta el bloque de sentencias y luego se incrementa en uno el contador. Habitualmente se coloca en la cabecera del for los tres componentes necesarios sobre el contador: inicialización, límite para continuar en el bucle e incremento. Sin embargo, si falta alguna de ellas, el comportamiento por defecto es:

1. Si no se inicializa el contador, éste deberá haberse inicializado en el programa antes del bucle for.
2. Si no existe un valor límite para el contador, el bucle será infinito y no terminará nunca.
3. Si no se especifica un incremento, no se incrementará el contador.

2. Programas básicos

A continuación, se presentan una serie de ejemplos de programas básicos que utilizan las estructuras de acciones: secuencia, selección y bucle.

Un programa que escribe en pantalla una serie de números correspondientes a la secuencia de Fibonacci, que es en sí una sucesión matemática infinita. Se van sumando los números consecutivos de dos en dos, comenzando con 0 y 1 de la siguiente manera:

0,1,1,2,3,5,8,13,21,34...

(0, 1, 0+1=1, 1+1=2, 1+2=3, 2+3=5, 3+5=8, 5+8=13, 8+13=21, 13+21=34...)

Así sucesivamente, hasta el infinito. Utilizando un bucle for, el código en lenguaje Java podría ser:

```
// Programa escrito en lenguaje Javapara escribir en pantalla la serie de Fibonacci
public static void main (String[] args) {
        //Declaración de variables
        int n1 =1;
        int n2;
        int max = 0; // guarda la cantidad de números que se van a escribir en pantalla
        int temp = 0; // variable auxiliar para guardar datos temporalmente
        //La variable leer guarda los datos introducidos por el usuario con el teclado
        Scanner leer = new Scanner (System.in);
        // Mensaje de pregunta al usuario
        System.out.println ("¿Cuántos números de la serie Fibonacci desea crear >2? ");
        //Guarda la cantidad de números de la serie que se van a escribir en pantalla max =
leer.nextInt();
        // Un bucle para calcular y escribir los números de la secuencia de Fibonacci
        for (i=1; i<=max; i++){
                n2 = temp;
                temp = n1 + temp;
                n1 = n2;
                System.out.print (n1); System.out.print (",");
                }
□ // fin del método main (principal)
```

2. Programas básicos

Un programa que controla las ventas de gafas en una tienda de óptica, que hace descuentos según el volumen de compra del cliente y le hace regalos si compra gafas de sol, sería el siguiente en código Java:

```
// Programa escrito en lenguaje Java para gestionar las ventas de la tienda de gafas
public static void main (String[] args) {
        // Declaración de variables
        boolean gafasSol=false;
        int cantidad=0; // variable que guarda la cantidad de gafas que se van a comprar
        int total = 0; // variable que guarda la cantidad de dinero a gastar
        // La variable leer guarda los datos introducidos por el usuario con el teclado
        Scanner leer = new Scanner (System.in);
        // Mensaje de pregunta al usuario
        System.out.println ("¿Cuántas gafas quiere comprar?");
        // Guarda la cantidad de gafas que el cliente va a comprar
        cantidad = leer.nextInt();
        total = cantidad * 50; // cada par de gafas cuesta 50 euros
        if (total>=100) {total = total *0.9;}
        if (cantidad>=2) {regaloFundas;}
        // Mensaje de pregunta al usuario
        System.out.println ("¿Quiere unas gafas de sol? S=Si, N=no");
        char caracter;
        caracter = leer.next().charAt(0);
        if (caracter == S) {gafasSol==true;}
        if (gafasSol==true) {total = total *0.9;}
        System.out.println ("Muchas gracias. Tiene que pagar: " + total);
□ // fin del método principal
```

Documentación técnica:

Introducción al desarrollo web

Índice

1. Historia de Internet 5
2. Estructura de Internet y protocolo TCP/IP 7
3. Direcciones URL y protocolo HTTP 12

1. Historia de internet

A finales de la década de los sesenta, en el siglo XX, el Departamento de Defensa de los EEUU promovió la creación de una red de comunicación que permitiera la interconexión de varios proveedores y centros de investigación localizados geográficamente muy distantes entre sí. Dicha red fue evolucionando en los siguientes veinte años, desarrollándose un protocolo estándar (denominado TCP/IP) que permitió la conexión de cualquier ordenador a esa red de forma interoperable, fuera cual fuera su tecnología. Sin embargo, Internet seguía siendo una red de uso complejo sólo apta para investigadores y técnicos. Su interfaz era textual y la mayoría de las operaciones se realizaba mediante correo electrónico, programas de transferencia de ficheros y conexión con sistemas remotos. Con el crecimiento en el uso de la red, se hicieron necesarias nuevas formas de navegar a través de la red y en 1992 surgió la Web como ahora la conocemos.

Internet ha supuesto una revolución para la sociedad actual, incluso mayor en algunos aspectos a la producida por la Revolución Industrial del siglo XIX. La red es un foro de intercambio de información sin límites y un mercado en crecimiento, todo ello accesible a cualquier persona independientemente de sus conocimientos técnicos. Internet ha ocupado su lugar gracias a la mejora de las comunicaciones que permiten acceder a la red de redes desde cualquier lugar y a cualquier persona, a la aparición y mejora de las tecnologías que proporcionan los recursos accesibles en Internet y a la mejora en la presentación de la información en las páginas web, de forma mucho más comprensible para la mayor parte de los usuarios.

Los recursos principales a los que se puede acceder a través de Internet son:

- Páginas web: alojan los recursos asociados a los sitios web, tanto corporativos como de índole pública.
- Aplicaciones que se ejecutan en la red: hoy en día coexisten diferentes aplicaciones a las que se puede acceder, como son tiendas virtuales (ropa, viajes, supermercados con servicios de comercio electrónico), banca electrónica, juegos y deportes (e-sports, apuestas y juegos on-line) que requieren a las computadoras y servidores ejecutar programas que están basados en diferentes tecnologías.

1. Historia de internet

- Gestión de recursos y seguridad: el acceso a muchos de los recursos en la red requiere el envío seguro de información privada del usuario como datos personales, bancarios o de una tarjeta de crédito.

2. Estructura de Internet y protocolo TCP/IP

La red Internet recibe también el nombre de World Wide Web (WWW o la Web a secas). Este interfaz permite un acceso sencillo y comprensible a la información disponible en Internet, que se organiza en forma de páginas web. Estas páginas pueden contener combinaciones de texto, imágenes, sonido, vídeos y animaciones, de forma atractiva y completa para los usuarios. Una de las ventajas de esta forma de presentar la información es el uso de los enlaces o hiperenlaces. Un enlace permite referenciar una página web desde otra, y cualquier página puede incluir referencias automáticas a otras páginas relacionadas.

Los navegadores, son los programas que permiten acceder a la Web y manejar páginas escritas en lenguaje HTML, incluyendo programas visualizadores para las imágenes, vídeos y animaciones.Los navegadores permiten acceder a las páginas locales o remotas y almacenan un historial de las páginas accedidas más recientemente. Se han añadido varios mecanismos que permiten ejecutar programas de forma que las páginas web no sean solamente estáticas sino que puedan presentar información de forma interactiva y dinámica. La página que muestra estas características debe utilizar uno de los lenguajes disponibles (JavaScript, por ejemplo) y el navegador debe estar preparado para interpretarlo.

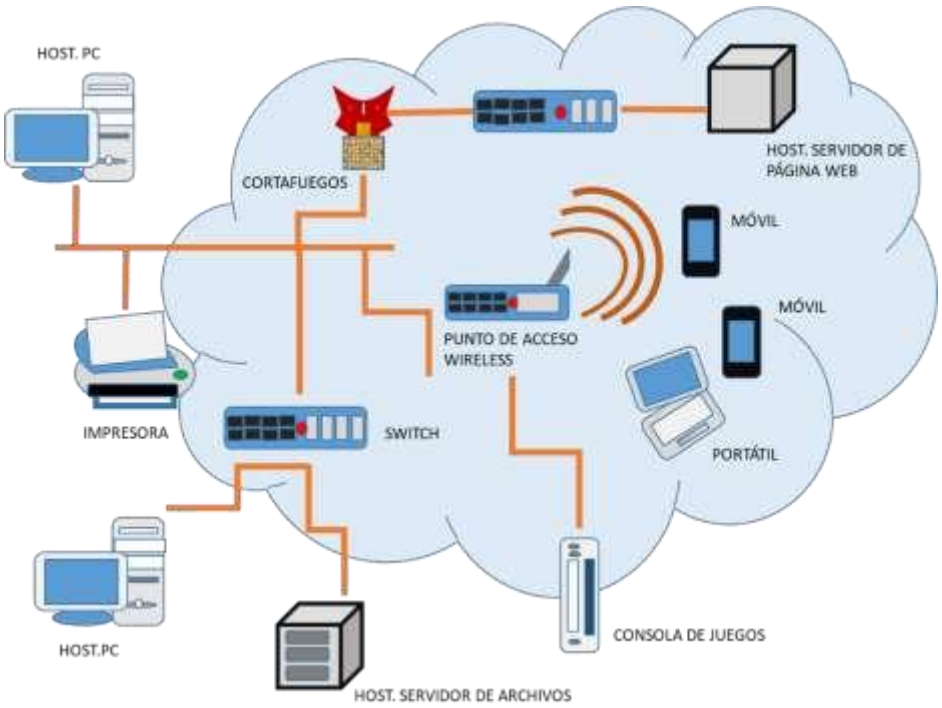


Figura 1. Ejemplo de elementos interconectados a través de Internet

2. Estructura de Internet y protocolo TCP/IP

Para que dos computadoras remotas puedan hablar entre sí y se entiendan, es necesario que hablen el mismo lenguaje. Es lo que se denomina un protocolo, es decir, un conjunto de reglas y mensajes estructurados que ambas máquinas deben entender y compartir para entenderse mutuamente. En Internet, el protocolo por excelencia de las comunicaciones se denomina TCP/IPen referencia a los dos protocolos más importantes que la componen, que fueron de los primeros en definirse:

- TCP: protocolo de control de transmisión.
- IP: protocolo de internet.

Además, la dirección IP en la que están conectadas a la red es única para cada máquina y le permite identificarse de forma inequívoca cuando se comunica con otra máquina a través de la red. Cada dirección tiene un formato con varios números separados por puntos, por ejemplo: xxx.xxx.xxx.xxx en la que cada campo indica una jerarquía de dominio en Internet(como se indica en la Figura 2). Para mejor entendimiento de los humanos, los números se sustituyen por alias en lenguaje natural, así por ejemplo el subdominio: .bejob.es se refiere al sitio web de la empresa BeJob en el país España.

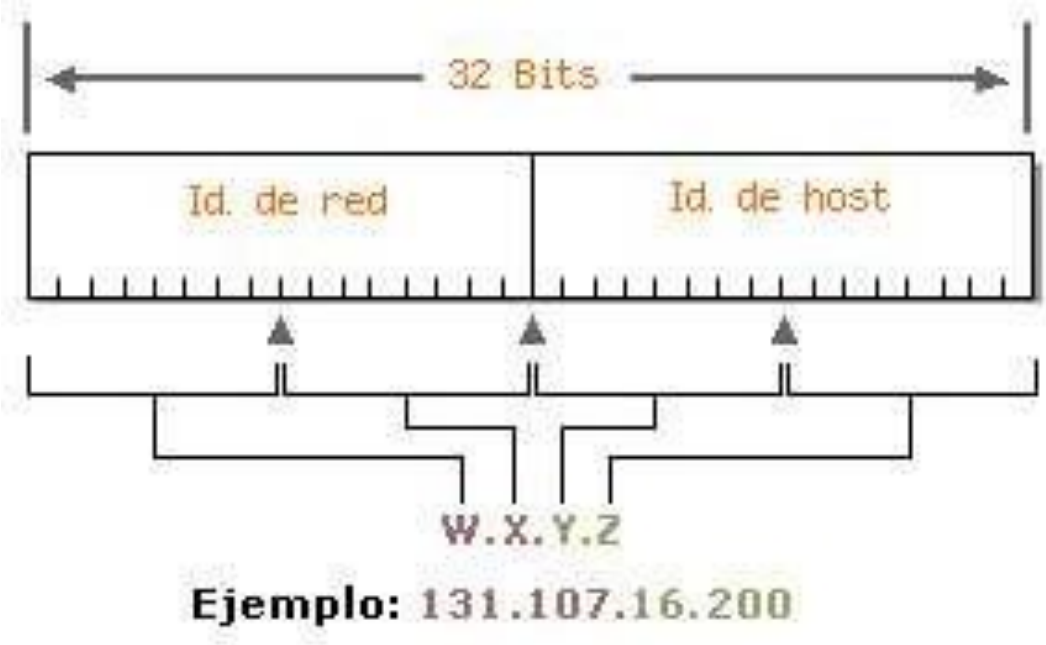


Figura 2. Direccionamiento IP

2. Estructura de Internet y protocolo TCP/IP

En realidad, existe toda una familia completa de protocolos de Internet que permiten la transmisión de datos entre nodos de la red. Existen tantos protocolos en este conjunto que llegan a ser más de cien diferentes, entre ellos se encuentran algunos de los más conocidos:

- ARP: protocolo de resolución de direcciones, para encontrar la dirección física (MAC) correspondiente a una determinadadirección IP.
- FTP: protocolo de transferencia de archivos, popular en la transferencia de archivos.
- HTTP: protocolo de transferencia de hipertexto, muy conocido porque se utiliza para fabricar páginas web.
- POP: protocolo de oficina de correo, para correo electrónico.
- SMTP: protocolo para transferencia simple de correo, para el correo electrónico.
- Telnet: protocolo para acceder a equipos remotos.

Por ejemplo, el protocolo HTTP (Hyper Text Transfer Protocol) mencionado antes, es el utilizado para programar aplicaciones en la web. Las aplicaciones desarrolladas con este protocolo aseguran que la información viaja de forma segura y confiable a través de la red utilizando el protocolo de transmisión TCP, situado en un nivel inferior. Otro de los conceptos importantes para entender la comunicación de datos a través de Internet es el concepto de puerto. Un puerto es un número que identifica a la aplicación que está preparada para participar en una comunicación de tipo TCP. Esta numeración está acordada previamente, es un estándar, de forma que cada número de puerto está asociado a un servicio determinado, y así, las computadoras pueden establecer múltiples conexiones TCP entre ellas, cada una con un puerto diferente para cada interlocutor. En el caso del protocolo HTTP, el número de puerto usado más comúnmente es el 80.

Internet es una gran colección de millones de computadoras independientes que están conectadas entre sí. Cada una de ellas proporciona sus propios servicios y ofrece sus recursos al resto de máquinas conectadas a la red. Cada una de ellas recibe el nombre de host o servidor.

El protocolo HTTP sigue el modelo cliente- servidor. El servidor es el host que permanece a la escucha en unpuerto de comunicaciones y recibe peticiones por parte de nodos clientes. Los clientes son los que llevan la iniciativa de las

comunicaciones y ejecutan las peticiones en el protocolo de aplicación que envían a través de la Web. Los servidores se limitan a ejecutar el código necesario para atender a las peticiones que le llegan y responder con los resultados al cliente (puede ser, por ejemplo, visualizar una página web).Este proceso se muestra en la figura 3.

PETICIÓN	DESCRIPCIÓN
GET	Solicita leer una página web
HEAD	Solicita leerla cabecera de una página web
PUT	Solicita almacenar una página web
POST	Envía datos a una aplicación web
DELETE	Borra la página web
LINK	Conecta dos recursos existentes entre sí
UNLINK	Rompe la conexión existente entre dos recursos

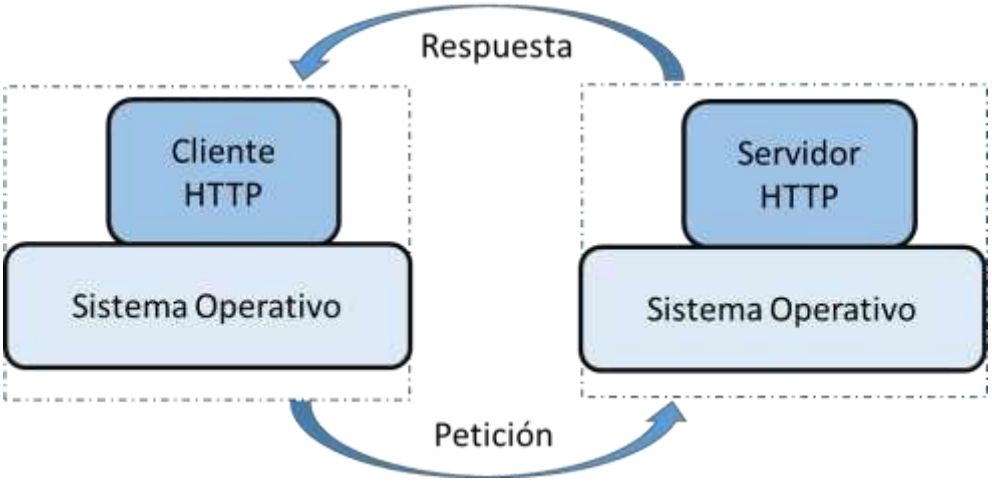


Figura 3. Modelo cliente-servidor

Normalmente, un servidor puede atender a múltiples clientes. Por ejemplo, un servidor de correo electrónico, como Gmail o Yahoo o Hotmail, que proporcionan cuentas de correo electrónico gratuito para cualquier usuario que lo solicite. Estas computadoras deben ser muy potentes, de hecho, se agrupan en grandes superficies, especialmente construidas y preparadas, denominadas Centros de Proceso de Datos (CPD).

Los servicios web, básicamente, consisten en establecer una conexión para obtener un documento, cerrar la conexión y visualizar el documento mediante un navegador. Si la página web, por ejemplo, es un sitio de comercio electrónico, cada

vez que el usuario añade nuevos artículos a la cesta de la compra habría que realizar una nueva conexión, y en cada conexión se perdería el estado previo de la cesta de la compra. Esto es así porque en la arquitectura cliente-servidor todo el estado de la conexión se mantiene en el servidor, por lo que al cerrarla se pierde toda la información. La solución que se utiliza es hacer que sea el cliente el encargado de guardar la información del estado de las conexiones. A este mecanismo se le denomina cookie, el cual almacena en modo texto toda la información acerca de las conexiones previas del usuario en el mismo sitio web.

3. Direcciones URL y protocolo HTTP

El protocolo de transferencia de hipertexto (HTTP Hyper Text Transfer Protocol) es un protocolo cliente-servidor que se usan para el intercambio de información entre los clientes y servidores de la Web, es decir, entre los programas navegadores y los servidores HTTP. Este es un protocolo de aplicación que funciona sobre los servicios de TCP/IP siguiendo el mismo esquema que otros servicios de red. En cada máquina conectada a Internet, hay un servicio HTTP escuchando habitualmente en el puerto 80 y esperando las peticiones de servicio de los clientes. Una vez que se establece la conexión, es el protocolo TCP el que se encarga de mantener la comunicación abierta y garantizar un intercambio de datos entre las máquinas que esté siempre libre de errores.

La Web utiliza una forma de direccionamiento que se denomina URL (Uniform Resource Locator) para localizar documentos o páginas web en Internet. Las URL pueden indicar el nombre del host que lo aloja, el puerto TCP/IP desde el que se va a conectar, así como el documento en sí. Por ejemplo:

```
http://www.bejob.es:212/docencia/aplicacionesweb.pdf
```

indica que:

- el protocolo que se va a utilizar para recuperar el documento es HTTP,
- que el host que lo alberga se llama www.bejob.es
- la conexión se debe establecer con el puerto número 212,
- el documento que se va a recuperar se llama aplicacionesweb.pdf y está alojado en la carpeta /docencia del host.

Para cada tipo de servicio existe un puerto predefinido, por eso, la mayoría de las veces, no hay que especificarlo. Así, por ejemplo, el puerto predefinido para el protocolo HTTP es el 80 y no suele aparecer.

Documentación técnica:

Estructura básica de un sitio web

Índice

1. Áreas de una página web……. 5

2. Partes principales de una página web …….. 6

3. Archivos de una página web…….. 7

1. Áreas de una página web

En una página web existen diferentes zonas o áreas, entre las que podemos destacar las siguientes:

- Encabezado:Es la parte superior de la página web en la que, normalmente, aparece un logo, un título y/o un menú con accesos a otras partes de la web.
- Pie: Es la parte inferior de la página en la que, normalmente, aparece la información de contacto, la información sobre redes sociales y/o enlaces a otras webs.
- Cuerpo: Es la parte de la web en la que aparece la información específica de los contenidos de la página. En esta parte se presenta a los usuarios la información concreta que queremos mostrar en nuestra web.

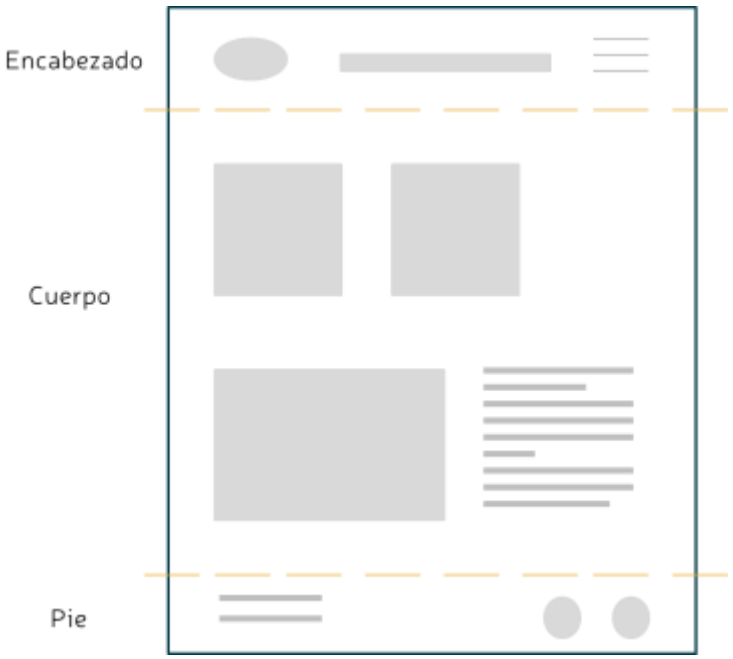


Figura 1.Áreas de una web

2. Partes principales de una página web

En una página web podemos encontrar cuatro partes principales diferenciadas que son las siguientes:

- Estructura: aquí se definen las áreas que existen en nuestra página web y que hemos visto en el punto anterior. El lenguaje HTML se utiliza para definir esta parte de la web.
- Contenido: esta es la información que queremos mostrar en nuestra web. Esta información pueden ser textos, enlaces a otras web, imágenes, así como cualquier otro tipo de información. También utilizaremos HTML para mostrar esta información.
- Apariencia: esta es la forma en la que se muestra la información en nuestra página web. Una misma información se puede ver de forma diferente dependiendo de quién acceda a ella o del dispositivo que se utilice, es decir, si un usuario tiene algún tipo de discapacidad visual, por ejemplo, puede que no vea colores u objetos de forma diferente a otro que no tenga esa discapacidad. También hay páginas que se muestran de forma diferente si las ves en un ordenador o en teléfono móvil. En la apariencia se tienen en cuenta diferentes tamaños, disposiciones, colores, etcétera. La apariencia de una web se controla mediante el uso de hojas de estilo en cascada o CSS.
- Comportamiento: es la forma en la que reacciona una web frente a una acción. Una web puede reaccionar frente a la interacción de un usuario de manera personalizada. Hoy en día las webs son, en su mayoría, dinámicas y pueden incluso cambiar de aspecto. Normalmente se utilizan lenguajes script para programar el comportamiento.

3. Archivos de una página web

Una página web estándar tendrá tres archivos diferenciados, que son los siguientes:

- .html: El archivo con extensión .html contendrá la estructura de nuestra página web así como toda la información que queremos que sea visible para los usuarios. Este archivo estará en lenguaje HTML y contendrá uno o varios enlaces a archivos con extensión .css y uno o varios enlaces a archivos con extensión.js.
- .css: El archivo con extensión .css contendrá las reglas devisualización del contenido de nuestra web. Estas reglas estarán escritas en CSS. El contenido de nuestra web es el que hemos creado en el archivo .html.
- .js: El archivo con extensión .js contendrá todos los elementos que harán que nuestra web sea dinámica, así como la programación de las funcionalidades que queremos que tenga nuestra página. Este archivo se escribirá utilizando el lenguaje JavaScript.

En resumen, la estructura y el contenido de nuestra web estarán escritos en HTML e irán definidas en el archivo .html. La apariencia de nuestra web se describirá en uno o varios archivos .css escritos en CSS y el comportamiento de nuestra web se definirá en uno o varios archivos .js que se escribirán en Java Script. Los archivos .css y .js serán llamados mediante enlaces que se incluirán en el archivo .html.

Documentación técnica:

Qué es HTML

Índice

1. Lenguajes de marcado 5

1.1. Características 5

2. Historia de los lenguajes de marcado 6

2.1. GML (Generalized Markup Language) 6

2.2. SGML (Standard Generalized Markup Language) 6

2.3. HTML (HyperText Markup Language) 6

2.4. XML (eXtensible Markup Language) 6

2.5. XHTML (eXtensible HyperText Markup Language) 7

3. HTML 8

3.1. HTML5 (HyperText Markup Language, versión 5) 8

1. Lenguajes de marcado

Los lenguajes de marcado son lenguajes que utilizan texto y etiquetas. Las etiquetas contienen información que definirá tanto la estructura como la presentación de dichos textos. Las etiquetas están predefinidas en el propio lenguaje y contienen información útil para las plataformas que leen estos lenguajes sobre la estructura de los textos.

La función de estos lenguajes es principalmente descriptiva.

Los lenguajes de marcado no son lenguajes de programación como tal, ya que estos últimos contienen funciones aritméticas y variables y los lenguajes de marcado no.

1.1. Características

Las principales características de los lenguajes de marcado son:

- Describen elementos mediante etiquetas específicas y aclaratorias.
- Las etiquetas van entre los símbolos < (menor que) y > (mayor que). Un ejemplo de etiqueta sería <HTML>.
- Habitualmente se utiliza un par de etiquetas, una de inicio <> y otra de cierre </>, aunque hay algunas etiquetas que van solas.
- Suelen guardarse en ficheros de texto plano.

2. Historia de los lenguajes de marcado

2.1. GML (Generalized Markup Language)

En informática, uno de los problemas que existen es la falta de estándares en los formatos que usan los distintos programas.

Para resolver este problema, en la década de 1960, Charles Goldfarb, investigador de la compañía IBM, empezó a desarrollar la idea de separar la presentación de la estructura, creando así el lenguaje GML (Generalized Markyp Language) cuyo objetivo era describir los documentos para que fuesen independientes de la plataforma y la aplicación utilizada.

2.2. SGML (Standard Generalized Markup Language)

Debido al éxito de GML, el ANSI, Instituto Americano de Normativas, empieza a fomentar el desarrollo de este tipo de lenguajes, llegando a desarrollar el lenguaje SGML (Standard Generalized Markup Language) que en 1986 dio lugar al estándar ISO 8879 . Después de esto, SGML pasa a ser la base del diseño de los nuevos lenguajes de marcado.

2.3. HTML (HyperText Markup Language)

A principios de los noventa, Tim Berners-Lee, creador del World Wide Web, tuvo la necesidad de organizar, enlazar y compatibilizar gran cantidad de información procedente de diversos sistemas y para ello creó un lenguaje de etiquetas de hipertexto llamado HTML.

HTML es una versión simplificada de SGML, ya que sólo utiliza instrucciones imprescindibles. Este lenguaje tuvo una gran acogida y hoy en día es el estándar general para la creación de páginas web.

HTML tiene algunas de sus desventajas ya que no es flexibley no permite mostrar contenido dinámico.

2.4. XML (eXtensible Markup Language)

Para resolver las desventajas de HTML, la W3C, fundada por Tim Berners-Lee entre otros, desarrolla el estándar internacional XML, un lenguaje de marcado que no incluye información relativa al diseño. Este lenguaje permite, entre otras cosas, definir etiquetas propias, asignarle atributos a esas etiquetas, manteniendo una independencia entre el diseño y la estructura.

2.5. XHTML (eXtensible HyperText Markup Language)

En el año 2000 surge XHTML (eXtensible HyperText Markup Language) para expresar el lenguaje HTML como un lenguaje XML válido.

3. HTML

HTML es el lenguaje de marcado más común que se utiliza para escribir una página web. Este lenguaje, como lenguaje de marcado, está compuesto por etiquetas que identifican el principio y el final de los elementos de un documento.

Los documentos HTML tienen extensión .html, extensión que reconoce un navegador para mostrar una página web. Estos documentos pueden estar formados, además de por texto, por imágenes, sonidos, vídeos, etc.

Un usuario puede ver una página web porque el navegador interpreta el código escrito en el documento .html y lo muestra visualmente en la pantalla.

3.1. HTML5 (HyperText Markup Language, versión 5)

Después de evolucionar en diferentes versiones, la versión actual más popular y utilizada es HTML5 (HyperText Markup Language, versión 5) publicada en Octubre de 2014. HTML5 especifica dos variantes de sintaxis para HTML: una clásica, HTML (text/html), que es la conocida como HTML5, y una variante XHTML, denominada XHTML5 que deberá servirse con sintaxis XML (application/xhtml+xml). Es la primera vez que HTML y XHTML se han desarrollado en paralelo.

HTML5 no puede ser reconocido por las versiones antiguas de los navegadores, debido a sus nuevas etiquetas, por ello se recomienda a los usuarios que deben actualizar su navegador a la versión más nueva, para poder disfrutar de todo el potencial que provee HTML5. De forma general, establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos (como <figure>, <audio>, <video>, <canvas> o <dialog>), modifica otras (<hr>, <i> o <input>) y también elimina varias etiquetas antiguas (por ejemplo <frame>, <font>, <dir>).

HTML5 ofrece versatilidad en el manejo y animación de objetos simples y contenido multimedia. Incorpora para ello nuevas etiquetas relacionadas con las imágenes y recursos multimedia (canvas 2D y 3D, audio, vídeo) así como con los códecs necesarios para su visualización más óptima en los navegadores y la posibilidad de arrastrar objetos como imágenes (Drag & Drop). También hay nuevos visores para fórmulas matemáticas (MathML) y para gráficos vectoriales (SVG).

También se han añadido nuevas etiquetas para manejar grandes conjuntos de datos: Datagrid, Details, Menu y Command, permitiendo generar tablas dinámicas que pueden filtrar, ordenar y ocultar información. Asimismo, se han incorporado mejoras en el manejo de los formularios añadiendo nuevos tipos de datos (como eMail, number, url, datetime) y facilidades para validar el contenido sin tener que utilizar otros lenguajes, como Javascript.

En relación a la potencialidad de la web semántica (Web 3.0) se añaden las nuevas etiquetas que permiten describir cuál es el significado del contenido, su importancia, su finalidad y las relaciones que existen entre diferentes partes de la página. No tienen especial impacto en la visualización sino que más bien se orientan a su interpretación por parte de los buscadores, de forma que podrán indexar e interpretar esta meta información para no buscar simplemente apariciones de palabras en el texto de la página.

Algunas de las etiquetas son técnicamente similares a las etiquetas y , pero tienen un significado semántico, como por ejemplo <nav> (bloque de navegación del sitio web), <header> y <footer>, <time> para indicar la fecha del contenido o link rel=" para referenciar el tipo de contenido que se enlaza.

Asimismo, permite incorporar a las páginas ficheros con meta-información (RDF / OWL) para describir relaciones entre los términos utilizados.

Por último, incorpora nuevas APIs y eventos Javascript, para trabajar en local, geolocalizar dispositivos, añadir facilidad de almacenamiento persistente en local, dispone de una base de datos con la posibilidad de hacer consultas SQL y acceso a dispositivos hardware de bajo nivel como elementos de red, ficheros, CPU, memoria, puertos USB, cámaras o micrófonos.

Documentación técnica:

Crear archivosHTML

Índice

1. Crear archivos HTML.....5

1. Crear archivos HTML

Un archivo HTML, en síntesis, es un documento con una estructura jerárquica donde cada uno de sus nodos será un elemento definido mediante una etiqueta de apertura (y generalmente, su correspondiente etiqueta de cierre). Entre estos dos marcadores puede ir contenido el valor de dicho elemento, y dentro de la etiqueta de apertura puede incluirse un conjunto de atributos con sus correspondientes valores.

Una de las principales ventajas de los archivos HTML es que se generan en formato "texto plano", o lo que es lo mismo, no presentan ningún tipo de codificación o similar. Por lo tanto, empleando cualquier herramienta de edición de texto se podrá crear un archivo HTML (en el caso de Windows, el Notepad o bloc de notas, o en el caso de Linux el editor Vi).

Es importante señalar que los archivos de este tipo requieren una extensión determinada, deben acabar con la extensión **.html**, o con **.htm**, para que el sistema operativo sepa que se trata de un archivo con unas características específicas y lo abra con la aplicación correspondiente (en este caso el navegador web).

Emplear un editor de texto presenta un problema: puesto que hay una sintaxis específica que respetar es asumible que se pueden cometer errores en la escritura del fichero, por lo que el resultado puede no ser el deseado.

Para facilitar la ayuda del programador de HTML existen numerosas aplicaciones denominadas *Integrated Development Environment* (IDE), Entorno de Desarrollo Integrado en español, que permiten entre otras cosas predecir el código que ha de insertar el usuario (en función de lo ya escrito), indicar los lugares en los que se producen errores o permitir la previsualización de lo que se está desarrollando.

Son numerosas las aplicaciones de este tipo, cabe destacar las siguientes (la mayoría gratuitas):

- Notepad++.
- Apache Netbeans.
- Eclipse.
- Kompozer.
- Aptana Studio.
- BlueGriffon.
- Sublime Text.

1. Crear archivos HTML

Una vez que el código está desarrollado la página web estará preparada para ser visualizada. La visualización del contenido (con sus formatos y demás aspectos asociados) se hará a través del navegador web. En este caso, lo que se consigue es que cada elemento utilizado en el documento HTML aparecerá en el navegador con la funcionalidad que tienen asociada.

Así, si por ejemplo se emplea un elemento <img>, que permite insertar una imagen en un documento, lo que aparecerá reflejado en el navegador no será el texto "<img>", sino que se mostrará la imagen indicada en dicho elemento. Ten en cuenta que para insertar una imagen sin definir la ruta en la que está guardada, la imagen deberá estar en la misma carpeta en la que se encuentra el archivo .html.

Aunque el código HTML emplea etiquetas estandarizadas cada navegador puede hacer una representación personalizada del contenido, de modo que pueden existir ligeras diferencias entre el resultado obtenido en uno u otro. Incluso puede ocurrir que algunas funcionalidades estén implementadas parcialmente, o incluso no estar implementadas, en ciertos navegadores y completamente desarrolladas en otros. Esto provoca que muchas veces haya que desarrollar código HTML que se ejecute de manera específica según el tipo de navegador que se quiera utilizar.

Aunque el número de navegadores es elevado existe un conjunto de navegadores que están ampliamente extendidos y su uso es generalizado:

- Mozilla Firefox.
- Microsoft Edge/Internet Explorer (solo para sistemas operativos Windows).
- Google Chrome.
- Apple Safari.

Documentación técnica:

Primeros pasos: encabezado, párrafos y saltos de lineas

Índice

1. Encabezado 5

2. Saltos de línea 7

3. Párrafos 9

1. Encabezado

A la hora de diseñar una página web es importante saber que existen dos zonas, una de ellas es la cabecera de la página web y la otra es el cuerpo de la página web.

```html
<!DOCTYPE html>
<html lang="es">
    <!-- Comienzo de la cabecera de la página WEB -->
    <head>
        <meta charset="utf-8">
        <title>Cabecera fija</title>
        <link rel="stylesheet" href="style.css">
    </head>
    <!-- Fin de la cabecera de la página WEB -->
    <!-- Comienzo del cuerpo de la página WEB -->
    <body>
        <p>Párrafo de prueba</p>
    </body>
    <!-- Fin del cuerpo de la página WEB -->
</html>
```

La cabecera es la parte superior de la página web donde se agrupan los metadatos, datos que no se muestran al usuario pero que le son de utilidad el navegador.

La cabecera se establece mediante las etiquetas **<head>** y **</head>** que se sitúan justo debajo de la etiqueta <**html**>.

Dentro de la encabecera, hay información del documento que se está creando pero que no es visible en la pantalla. Por ejemplo el título. Esta información se sitúa entre las etiquetas **<title>** y **</title>**. Title indica el título del documento. El título debe describir de una manera breve el contenido del texto que queremos plasmar, ya que cuando los usuarios añadan nuestra página a su barra de marcadores (bookmark o agenda de direcciones), la dirección debe caber en un pequeño espacio.

En cuanto al cuerpo, es decir, la parte visible de nuestra página WEB, el texto que queramos poner en nuestra página puede tener hasta 6 títulos o encabezados definidos por las etiquetas **<h1>** y **</h1>**, **<h2>** y **</h2>**, etc. Este orden determina el tamaño del texto, es decir, **<h1>** y **</h1>** tendrá un tamaño mayor que **<h2>** y **</h2>**.

1. Encabezado

Si quieres poner un párrafo, una imagen, etc., en tu página web, una etiqueta que puede ayudarte es la etiqueta y </ center> que sirve para centrar todo lo que pongas dentro de ella. Es bueno que conozcas esta etiqueta, pero debes saber que en HTML 5 no se utiliza, sino que se hace directamente con CSS.

2.Saltos de línea

A la hora de estructurar un texto en la página web debes saber que, para escribir espacios en blanco, introducir saltos de página o tabuladores no se puede hacer tal cual, igual que se hace en cualquier documento, sino que se tienen que añadir etiquetas concretas.

Etiqueta br: Esta etiqueta provoca un salto de línea simple, es decir, como si pretases "enter" una sola vez en otro documento. Para introducir un salto de línea simple una vez que se haya definido un bloque de texto, deberás utilizar
.

Etiqueta p: también denominada etiqueta de párrafo es una de las etiquetas más utilizadas para estructurar un texto en HTML. Esta etiqueta introduce un párrafo. Los navegadores agregan automáticamente una sola línea en blanco antes y después de cada elemento <p>.

A la hora de maquetar y crear una página web se pueden añadir varios párrafos de texto, pero para darle forma debemos utilizar la etiqueta <p>. Esta etiqueta dejará una línea en blanco entre los párrafos. Si queremos varias líneas en blanco no es suficiente con añadir la etiqueta <p> varias veces, sino que deberás combinarla con la etiqueta
(break o romper).

La etiqueta
 permite poner párrafos en los textos, pero sin que aparezca una línea en blanco.

Ninguna de las dos etiquetas anteriores tiene etiqueta de cierre.

A continuación, te mostramos un ejemplo de cómo deberías poner estas etiquetas en tu texto para que tenga los párrafos que quieres:

```
<html>
    <head>
        <title>Consejos de HTML </title>
    </ head>
    <body>
        <p> Cuando quieras poner párrafos en los textos de tu página deberás utilizar la etiqueta < p>
        <br>
        <br>
```

2.Saltos de línea

<p> Si quieres que entre los párrafos aparezcan líneas blancas deberás utilizar la etiqueta
.

</ body>

</html>

A continuación, te mostramos como se vería en una página de HTML:

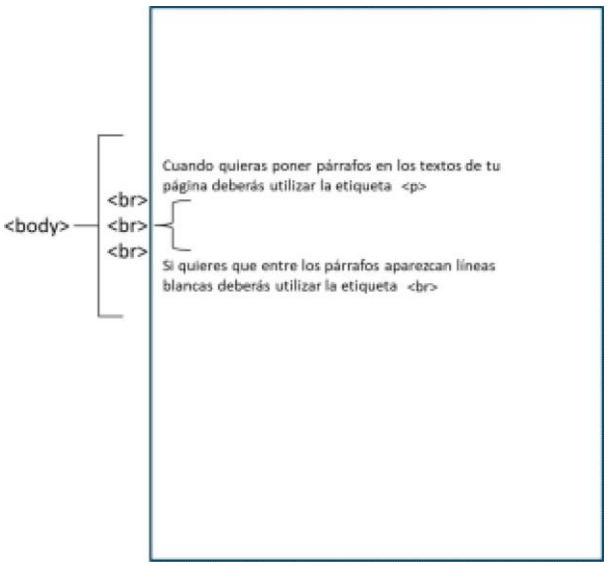


Figura 1. Ejemplo visualización

3. Párrafos

En algunos textos se desea poner más de un espacio entre dos palabras, pero el navegador sólo acepta crear el hueco de un espacio. Mediante el código " : (non-breakingspace) se puede forzar a que acepte más espacios.

Si estamos, por ejemplo, escribiendo un texto con muchos párrafos, quizá queramos destacar algunas palabras, entonces podemos utilizar:

<del> y </del> para tachar una palabra o texto.

<b> y <b/> para poner en negrita una palabra o un texto (bold).

<u> y </ u> para subrayar una palabra o un texto.

<i> y </ i> para poner una palabra o texto en cursiva (italic).

Otras etiquetas que podemos utilizar para cambiar la apariencia del texto son <pre> y </ pre >. Esta etiqueta hará que tu texto aparezca en formato preformateado, es decir, aparecerá como si se hubiera escrito a máquina con una fuente de espaciado, tipo Courier, ya establecido.

También podemos utilizar la etiqueta <tt> y </ tt > para conseguir que los caracteres tengan apariencia de haberse escrito en una máquina de escribir y también para que el texto tenga un tamaño menor. Puede confundirse con la etiqueta <pre> pero la etiqueta <tt > no preformatea el texto, sino que cambia su apariencia. Esta etiqueta nos permite tener un espaciado simple ya que la etiqueta
 no nos sirve.

```
<html>
    <body>
        <tt>
        <p> Cuando quieras poner párrafos en los textos de tu página
        deberás utilizar la etiqueta
        </p>
        </tt>

    </body>
</html>
```

Para poder dar formato de sangría a un párrafo de HTML y para definir citas largas, podemos utilizar la etiqueta <blockquote> y </ blockquote>.

Documentación técnica:

Estructura base de HTML

Índice

1. Estructura de HTML.. 5

2. Elementos de HTML 6

2.1. Etiqueta <head></head> 8

2.2. Etiqueta <body></body> 8

1. Estructura de HTML

En 1980 Tim Bernes-Lee, uno de los padres de la World Wide Web, junto con su equipo del CERN crearon el lenguaje HTML (HyperText Markup Language) o lenguaje de etiquetas de hipertexto.

HTML tiene una estructura determinada, siendo los bloques <head> y <body> los más importantes y principales. Combinando estos bloques con otras etiquetas se podrán conseguir otras funcionalidades y/o permitirá relacionar el archivo .html con otros archivos HTML, JS o CSS.

HTML está formado por una serie de elementos que son los componentes fundamentales. Estos componentes tienen a su vez dos propiedades básicas, contenido y atributo.

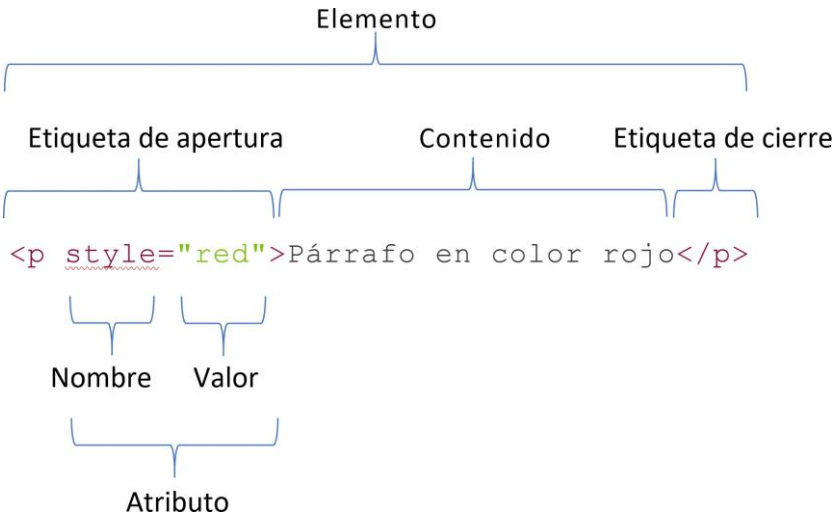
Tanto el contenido como el atributo tienen una estructura definida y específica. El contenido estará comprendido entre la etiqueta de apertura y la etiqueta de cierre. Los atributos se colocan únicamente dentro de la etiqueta de apertura. A la hora de escribir un atributo debes saber que son pares de nombres y valores que se representan entre comillas además de estar separados por un "=".

```
<p color="A001" >CursoHTML</p>

<img src="https://images.unsplash.com/photo-1535378917042-10a22c95931a">

<a href="http://www.bejob.es">Acceso a Bejob</a>
```

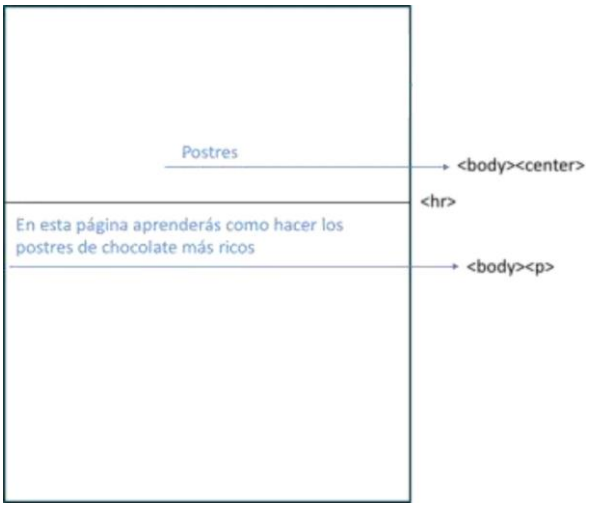
A continuación, te mostramos un ejemplo de cómo se articula un elemento:



2.Elementos de HTML

A continuación, te mostramos un ejemplo de código HTML sencillo en el cuál verás cuales son las partes de una página en lenguaje HTML:

```
<html>
<head>
        <title> Mis clases de cocina
        </title>
</head>
<body>
        <center>
                <h1>Postres</h1>
        </center>
        <hr>
        <p>En esta página aprenderás
como hacer los postres de chocolate
más ricos.
</body>
</html>
```



Ahora te explicaremos uno a uno los elementos que han aparecido en el ejemplo anterior:

<html>…</html>:nos indica que el texto escrito está en un formato de lenguaje HTML. Como se ve en la imagen anterior, un documento HTML posee dos partes: la cabecera (head) y el cuerpo (body).

<head><title>…</title></head>: la etiqueta <head> marca el inicio de la cabecera y la etiqueta </head> marca el fin de la cabecera. La etiqueta del título es imprescindible para el posicionamiento de nuestra página en internet, ya que define el título de la misma.

<body>…</body>: la primera etiqueta marca el inicio de del cuerpo del documento y la segunda etiqueta marca el fin del cuerpo del documento.

<h1>…</h1> : esta etiqueta determina que el texto se coloque centrado y en formato h1, es decir, a modo de título o encabezado.

2.Elementos de HTML

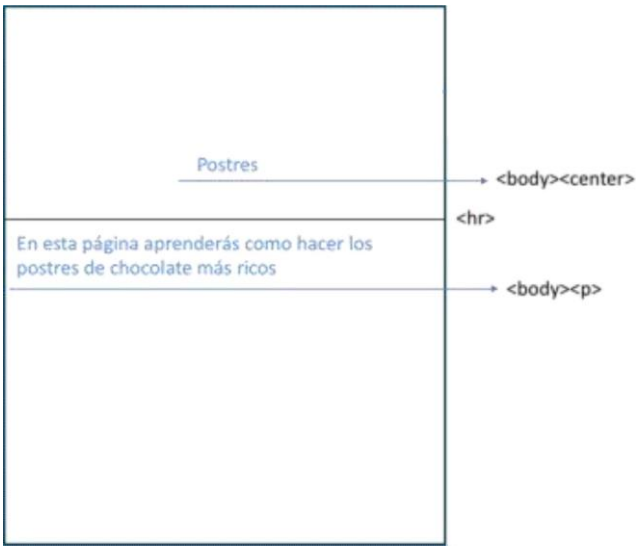
<hr>: esta etiqueta coloca una línea horizontal en la página, además no tiene etiqueta de cierre.

<p>: esta etiqueta se utiliza para marcar el inicio de un párrafo.

Algo que puede ser beneficioso es que cuando escribas el código en tu documento HTML, pongas comentarios sobre cómo escribiste este código por si tienes que modificar algo en un futuro. Estos comentarios no serán visibles en la pantalla.

Para incluirlos se utiliza el elemento <!- ..- >.

```
<html>
<head>
        <title> Mis clases de cocina
        </title>
</head>
<body>
        <! – aquí vamos a poner el
apartado postres-- >
        <center>
                <h1>Postres</h1>
        </center>
        <hr>
        <p>En esta página aprenderás
        como hacer los postres de
        chocolate más ricos.
</body>
</html>
```



2.Elementos de HTML

2.1. Etiqueta <head></head>

Como hemos explicado anteriormente los bloques más importantes a la hora de diseñar una página en formato.html son <head> y <body>.

En el bloque <head> se pueden incluir algunos elementos además de los que hemos comentado en el punto anterior.

El elemento <meta>…</meta> permite especificar información que puede ser de interés para el público como puede ser por ejemplo, saber quién es el autor del texto que estamos plasmando en nuestra página, la fecha de publicación de dicho texto, palabras claves para que ese texto o página pueda encontrarse más fácilmente en el buscador.También se puede incluir una breve descripción del contenido de la página para que el público pueda saber de qué trata la página de una manera rápida.

El elemento <base href=http:// www.lablibliotecа.com / publicaciones> permite especificar la URL directa a un apartado de una página web.

El elemento <style>…</style>permite especificar las reglas de estilo CSS que se pueden incluir de manera directa, sin tener que usar un archivo en formato .css.

El elemento <script>…</script> permite incluir programas al documento, como por ejemplo, código JavaScript que está "incrustado" en el propio código HTML (no en un fichero aparte).

2.2. Etiqueta<body></body>

El bloque <body> permite utilizar y añadir más elementos, ya que en él se puede incluir más texto que en el apartado <head>.

Algunos de los elementos, además de los mencionados anteriormente, que se pueden añadir son:

Elemento <address>…</address>: este elemento permite indicar direcciones, no direcciones electrónicas sino direcciones físicas.

Elemento <abbr>…</abbr>: este elemento se utiliza para representar abreviaturas.

2.Elementos de HTML

Elemento … se utiliza para darle énfasis a una frase.

Elemento <p>: Este elemento se utiliza para delimitar un párrafo e inserta una línea en blanco antes del texto.

Elemento <h1>…</h1>// <h6>…</h6>: Se utiliza para establecer títulos. Se pueden poner hasta 6 niveles distintos.

Elemento <tt>…</tt>: este elemento se utiliza para representar el texto marcado con una fuente de ancho fijo.

Elemento <blockquote>…</blockquote>: este elemento se utiliza para citar un texto ajeno a nuestro documento. Es muy utilizado ya que a la hora de implantarlo en nuestro documento HTML se suelen dejar márgenes tanto a la derecha como a la izquierda.

Elemento <cite>…</cite>: este elemento también se utiliza para citar texto ajeno, pero no a nuestro documento.

Elemento <**div aling** = **x**>**…**</**div**>: este elemento te permite justificar el texto del párrafo cambiando el valor x por left (izquierda), right (derecha), justify (a ambos márgenes) o center (centrado).

Elemento <big>…</big>// <small>…</small>:big te permite aumentar el tamaño de la letra del texto y small te permite disminuir el tamaño del texto.

Elemento <samp>…</samp>: este elemento se utiliza para representar caracteres literales.

Elemento <pre width=x>…</pre width=>: este elemento tiene dos funcionalidades. Por un lado, width especifica el número máximo de caracteres que puede haber en una línea y este atributo en su conjunto presenta el texto encerrado con un tipo de fuente de ancho fijo.

Elemento <blink>…</blink>: este elemento se utiliza para hacer parpadear el texto.

Elemento <var>…</var>: este elemento te permite representar variantes de un código.

Elemento <code>…</code>: este elemento se utiliza para escribir un código fuente.

2.Elementos de HTML

Elemento
//<hr> : el elemento
 se utiliza para dar un salto de línea y el elemento <hr> se utiliza para insertar una línea horizontal.

Elemento …: este elemento se utiliza para remarcar datos importantes.

Elemento <u>…</u>: este elemento se utiliza para subrayar un texto o una palabra.

Elemento …: este elemento se utiliza para poner un texto o palabra en negrita.

Elemento <s>…</s>: este elemento se utiliza para tachar una palabra o un texto.

Elemento …: este elemento se utiliza para poner un texto o una palabra en cursiva.

Elemento<kbd>…</ kbd>: este elemento permite indicar en el texto que debe ser tecleado por el usuario.

Elemento _…//[…]: el elemento <sub> permite poner textos en formato subíndice y el elemento <sup> poner textos en formato superíndice.

Como hemos visto la etiqueta <body> tiene una gran cantidad de elementos, pero se pueden complementar con una serie de atributos como son:

Atributo vlink="color" o vlink= "#XXYYZZ" este atributo define el color de los enlaces que han sido visitados.

Atributo alink= "color" o alink= #XXYYZZ": este atributo define el color del enlace que se está pulsando. Si no se define un color, su color por defecto será el rojo.

Atributo link="color" o link="#XXYYZZ": este atributo define el color del link o enlace que no se ha visitado.

Atributobackground="fichero": este atributo se utiliza para establecer una imagen como fondo de la página que se está maquetando.

Atributo bgcolor="color" o bgcolor="#XXYYZZ": este atributo se utilizar para definir el color de fondo de la página.

Atributo leftmargin / topmargin: este atributo especifica el número de píxeles de margen entre el borde de la ventana y el contenido de la página. Si no se especifica su valor por defecto, se inicializa a 0.

Atributo text="color" o text="#XXYYZZ": este atributo se utiliza para definir el color del texto.

Documentación técnica:

Etiquetas en HTML

Índice

1. Qué son las etiquetas 5
2. Principales etiquetas en HTML 6
2.1.Etiquetas para la estructura básica 6
2.2. Etiquetas para definir elementos de una página 6
2.3. Etiquetas de encabezado y párrafo 6
2.4. Etiquetas de enlaces 7
2.5. Etiquetas de listas 7
2.6. Etiquetas de tablas 7
2.7. Etiquetas de formularios 7

1. Qué son las etiquetas

Los lenguajes de marcado, como HTML, se caracterizan por el uso de etiquetas.

Una etiqueta es una marca que nos permite indicar cómo se van a visualizar los elementos que aparecen en nuestra web y cómo se van a comportar los elementos que no son visibles en nuestra web.

Habitualmente hay una etiqueta de inicio que se representa entre los signos menor qué < y mayor qué >, por ejemplo, <etiqueta>, y una etiqueta de cierre que se representa con el signo menor que < seguido de la barra /, el texto de la etiqueta y el signo mayor que <. Para el ejemplo anterior la etiqueta de cierre sería </etiqueta>.

Los nombres de las etiquetas pueden estar tanto en mayúsculas como en minúsculas, por ejemplo, <etiqueta> o <ETIQUETA>.

Todo lo que situemos entre la etiqueta de inicio y la etiqueta de cierre estará influenciado por dicha etiqueta.

2. Principales etiquetas en HTML

HTML tiene diferentes etiquetas, pero vamos a destacar algunas de las más utilizadas, según su uso, desarrollándolas más en detalle durante todo este curso.

2.1. Etiquetas para la estructura básica

Entre las etiquetas que utilizamos para crear la estructura básica de una página web podemos destacar:

- <html>, que se utiliza para abrir la raíz del documento. Todo el documento estará contenido entre <html> y </html >.
- <head>, que se utiliza para definir la información acerca del documento, incluyendo enlaces, hojas de estilo, etc.
- <body>, que se utiliza para definir el cuerpo del documento. Esta etiqueta solo aparece una vez en todo el documento.

2.2. Etiquetas para definir elementos en una página

- <title>, que se utiliza para definir el título de la página o documento.
- , que se utiliza para definir una sección en una página o documento.
- <img>, que se utiliza para insertar imágenes en una página o documento.
- <link>, que se utiliza para definir la relación entre una página o documento con un elemento externo, generalmente un elemento de JavaScript o CSS.
- <style>, que se utiliza para definir el estilo de la información de una página o documento. Con esta etiqueta se puede escribir código en CSS directamente dentro del <head> de nuestro documento.

2.3. Etiquetas de encabezado y párrafo

Estas etiquetas son las más utilizadas para dar formato a párrafos y encabezados.

- <h1>, <h2> ... <h6>, que se utilizan para definir encabezados o títulos.
- <p>, que se utiliza para definir un párrafo.
-
, que se utiliza para definir un salto de línea.

2. Principales etiquetas en HTML

- <b>, que se utiliza para resaltar texto en negrita.
- <i>, que se utiliza para escribir texto en cursiva.

2.4. Etiquetas de enlaces

La etiqueta <a> es la utilizada para definir hipervínculos dentro de nuestra página o documento.

2.5. Etiquetas de listas

Las etiquetas más utilizadas para trabajar con listas son:

- , que se utiliza para definir un elemento u objeto dentro de una lista.
- , que se utiliza para definir una lista ordenada de elementos.
- , que se utiliza para definir una lista desordenada de elementos.

2.6. Etiquetas de tablas

Las etiquetas más utilizadas para trabajar con tablas son:

- <table>, que se utiliza para definir una tabla.
- <tr>, que se utiliza para representar una fila en una tabla.
- <td>, que se utiliza para representar una celda en una tabla.

2.7. Etiquetas de formularios

La etiqueta <form> se utiliza para definir formularios dentro de nuestra página web.

Documentación técnica:

Atributos en HTML

Índice

1. ¿Qué son los atributos?..... 5

2. Tipos de atributos..... 6

2.1. Atributos básicos..... 6

2.2. Atributos de internacionalización..... 6

2.3. Atributos de foco..... 7

2.4. Atributos de evento..... 7

1. ¿Qué son los atributos?

Los atributos en HTML son valores que configuran elementos o que definen detalles de comportamiento para cumplir un criterio específico. Los atributos se sitúan dentro de las etiquetas para proveer de mayor información a las mismas.

Los atributos son pares formados por un nombre de atributo y un valor, separados ambos por un signo igual =. El campo valor de un atributo siempre irá entre comillas.

Un ejemplo de atributo en una etiqueta sería:

<htmlla ng="ES">

En el ejemplo el atributo es lang="ES" donde el atributo lang se utiliza para identificar un idioma y el valor ES identifica el idioma español. Este atributo en la etiqueta html quiere decir que el idioma de la página web que vamos a escribir será español.

2. Tipos de atributos

En general, cada etiqueta define sus propios atributos, es decir, los atributos que se pueden utilizar con ellas, pero, hay algunos atributos que se pueden utilizar con la mayoría de las etiquetas.

Vamos a ver algunos atributos que, según su funcionalidad, se puede clasificar en atributos básicos, atributos de internacionalización, atributos de foco y atributos de eventos.

2.1. Atributos básicos

Estos atributos son globales y se utilizan en la mayoríade las etiquetas de HTML.

- Class: Normalmente se utiliza con CSS para aplicar estilos a los elementos que tienen propiedades en común.

Ejemplo: <p class="texto">. Esta sentencia aplica al párrafo definido por la etiqueta p el estilo definido en el atributo con valor texto. El valor a texto se incluirá en el archivo .css.

- Id: Establece un indicador único para cada elemento para que se pueda acceder a él desde CSS o JavaScript.
- Style: Aplica los estilos CSS a un elemento anulando los estilos establecidos previamente.

Ejemplo: <p style="color:green">. Esta sentencia pondrá un color verde al texto del párrafo definido por la etiqueta p.

- Title: Establece los títulos de los elementos. El título de un elemento concreto se mostrará cuando el cursor esté sobre él. Este atributo mejorar la accesibilidad de nuestra página.

Ejemplo: <p title="Prueba">. Esta sentencia resaltará la palabra Prueba cuando pongamos el cursor del ratón sobre el párrafo definido por la etiqueta p.

2.2. Atributos de internacionalización

Estos atributos también son globales y se utilizan en las páginas que tienen su contenido en variosidiomas.

- Lang: Indica el lenguaje o idioma utilizado en el elemento.

2. Tipos de atributos

- Dir: Indica la dirección del texto. Los valores permitidos en este atributo son ltr, de izquierda a derecha, o rtl, de derecha a izquierda.

2.3. Atributos de foco

Estos atributos globales se utilizan para el control de los elementos que han sido seleccionados en una web. El elemento seleccionado es el que tiene el foco, cuando deja de estar seleccionado y se selecciona otro elemento el foco pasa a este segundo elemento.

- Accesskey: Establece una tecla de acceso rápido que permite activar, o poner el foco, en un elemento.

Ejemplo: HTML5

Esta sentencia quiere decir que hemos activado la letra h para acceder directamente al hipervínculo que está sobre la palabra HTML5.

- Tabindex: Sobrescribe el orden de tabulación que tiene predeterminado el navegador y usa el que le especifiquemos. Su valorestará entre 0 y 32.767. Es muy importante definirlo para determinar el orden de navegación de los campos que componen un formulario, por ejemplo.

2.4. Atributos de eventos

Los atributos de eventos se utilizan en las páginas que incluyen código en JavaScript y permiten asociar acciones dinámicas a eventos relacionados con elementos de la web.

Algunos de estos atributos son globales para todos los elementos, pero algunos son específicos para algunas etiquetas.

Eventos de ratón

Estos atributos son globales para todos los elementos.

- Onclick: Ejecuta la acción cuando se haceclic sobre el elemento.
- Ondblclick: Ejecuta la acción cuando se hace dobleclic sobre el elemento.
- Onmousedown: Ejecuta la acción cuando se pulsa un botón del ratón.

2. Tipos de atributos

- Onmouseup: Ejecuta laacción cuando se suelta el botón del ratón.
- Onmousemove: Ejecuta la acción cuando se mueve el ratón.
- Onmouseover: Ejecuta la acción cuando el ratón se sitúa sobre un elemento.
- Onmouseout: Ejecuta la acción cuando el ratón deja de situarse sobre un elemento.

Eventos de documento

Estos atributos son utilizados con la etiqueta <body>.

- Onload: Ejecuta una acción cuando se carga el documento.
- Onunload: Ejecuta una acción cuando se abandona el documento.
- Onresize: Ejecuta la acción cuando se modifica el tamaño dela ventana del navegador.

Eventos de teclado

Estos atributos son utilizados por la etiqueta <body> y también por los elementos de los formularios.

- Onkeydown: Ejecuta una acción cuando se pulsa una tecla.
- Onkeyup: Ejecuta una acción cuando se deja depulsar una tecla.
- Onkeypress: Ejecuta una acción cuando se pulsa y se despulsa una tecla.

Eventos de formulario

Estos atributos se pueden utilizar en diferentes etiquetas que indicaremos en cada uno de ellos.

- Onblur: Ejecuta una acción cuando el elemento deja de ser el foco. Lo utilizan las etiquetas <button>, <input>, <label>, <select>, <textarea> y <body>.
- Onfocus: Ejecuta una acción cuando el elemento obtiene el foco. Lo utilizan las etiquetas <button>, <input>, <label>, <select>, <textarea> y <body>.

2. Tipos de atributos

- Onchange: Ejecuta una acción cuando cambia el valor del elemento. Lo utilizan las etiquetas <input>, <select> y <textarea>.
- Onreset: Ejecuta una acción cuando se reinicia un formulario a sus valores por defecto. Lo utiliza la etiqueta <form>.
- Onselect: Ejecuta una acción cuando se selecciona un elemento. Lo utilizan las etiquetas <input> y <textarea>.
- Onsubmit: Ejecuta una acción cuando se envía un formulario. Lo utiliza la etiqueta <form>.

Documentación técnica:

Enlaces en HTML

Índice

1 Enlaces en HTML 5

1.1 Enlaces con una dirección decorreo electrónico 5

1.2. Enlaces dentro de la misma página 6

1.3. Enlaces con una página fuera de nuestra web 6

1.4. Enlaces con otra página nuestra 7

2. Imagen como enlace 8

2.1. Atributos etiqueta <img> 9

1.Enlaces en HTML

Son muchos los complementos que podemos añadir a nuestra página web, pero en este caso hablaremos de los enlaces.

Los enlaces son textos que al pulsar sobre ellos con el ratón nos pueden llevar a un documento externo o a un apartado dentro de nuestro propio texto.

Existen al menos 4 tipos de enlaces:

- Enlaces con una dirección de correo electrónico
- Enlaces dentro de la misma página
- Enlaces con una página fuera de nuestra web
- Enlaces con otra página nuestra

Para crear enlaces en nuestra página html debemos utilizar la etiqueta <a>. Como con etiquetas anteriores, la etiqueta enlace tiene apertura <a> y tiene cierre y todo lo que pongamos entre estas dos etiquetas, ya sean imágenes o texto serán considerados parte del enlace.

Estas dos opciones se representan de forma distinta, es decir, las imágenes aparecerán delimitadas por un borde que indicará que es un enlace y los textos aparecerán subrayados con diferentes colores, uno cuando no haya sido pulsado, otro color cuando el usuario tenga el cursor encima y otro color cuando el enlace ya haya sido visitado.

La estructura base es <a href= "…XXXX". href es un atributo propio de la etiqueta <a> que indica la URL del recurso que se quiere enlazar. Las "XXXX" tendrán que ser sustituidas por la localización del documento o texto de destino.

A continuación, te mostramos ejemplos de cómo se tienen que explicar los enlaces, en lenguaje html, dependiendo de dónde esté el texto destino:

1.1. Enlaces con una dirección de correo electrónico

Para escribir un enlace con una dirección de correo electrónico en lenguaje HTML debes escribir la estructura base y añadir el texto de enlace y la etiqueta de cierre. Sería así:

.

Si por ejemplo la página tiene muchos departamentos podrás añadir un texto de enlace como,por ejemplo:

préstamos.

1.2. Enlaces dentro de la misma página

Si tu página es muy extensa, puede ser beneficioso para el público poner enlaces que te lleven a otra parte del documento. En estos ejemplos verás cómo poner un enlace para ir al principio y al final de tu documento:

Principio de la página

final de la página

Si tu página tiene varios apartados también puedes poner enlaces para ir a cada uno de esos apartados, por ejemplo:

Donde id es el atributo que sirve de ancla hacia la ubicación "préstamos" en nuestra propia página.

1.3. Enlaces con una página fuera de nuestra web

Este tipo de enlaces se utilizan cuando queremos poner la dirección de una página de otro servidor como destino. Es importante que pongas la dirección completa, ya que, si no es así, el enlace no funcionará. A continuación, te mostramos un ejemplo:

Estas direcciones se incluyen en formato URL (Uniform Resource Locator) y además se puede incluir la ruta que se va a seguir para llegar al apartado deseado. Este formato puede ser utilizado en telnet, foros de debate, mailto, ftp y, las más conocidas, http y https, por ejemplo:

Enviar correo electrónico

1.4. Enlaces con otra página nuestra

Si tu documento tiene más de una página podrás poner enlaces que te lleven a otra página dentro del mismo documento. Te mostramos a continuación como deberías poner esto en lenguaje html.

Anexos3</ a>

A esto también podemos añadirle un enlace directo a un apartado, por ejemplo, al apartado alinear de Anexos3.

 Anexos3

2. Imagen como enlace

También podemos utilizar imágenes como enlace a otra página. Pues bien, para usar estos enlaces tienes que utilizar la etiqueta <img> y se escribiría así, por ejemplo:

<img scr="caratula1jpg " alt="caratulas ">

Donde el atributo src se utiliza para indicar el nombre del fichero gráfico que vamos a integrar y el atributo alt se utiliza para indicar el texto alternativo que se debería mostrar en nuestra web en caso de la que la imagen no se pudiera mostrar por algún motivo. Estos textos alternativos son muy útiles para personas con discapacidad visual, ya que es la única información que reciben desde su lector de pantalla. Lo que esté escrito en ese texto alternativo, por ejemplo, "imagen de persona triste" es la información que recibirán.

Las imágenes que añadas pueden ser también enlaces a otras imágenes,

<imgsrc="autores.gif">

También puedes añadir un texto que sirva como enlace a una imagen.

<a href="ganadores.gif">Ganadores premio slibris</ a>

Las imágenes que añadas a tu documento pueden ser tuyas o pueden ser imágenes que estén en otro servidor. Para añadir una imagen de otro servidor tendrías que añadir su URL o su dirección completa, pero esto puede ralentizar mucho tu página y empeorar la experiencia del usuario. Las imágenes que utilices puedes almacenarlas pulsando el botón derecho sobre la imagen y darle a "Guardar imagen como".

Seguro que has visitado muchas páginas web que, aparte de imágenes,tienen iconos, barras, bocadillos de comentarios, etc. Estos iconos son también considerados imágenes por lo que tendrás que guardarlos en tus carpetas como imágenes para poder utilizarlas como te hemos explicado anteriormente.

2. Imagen como enlace

2.1 Atributos etiqueta <img>

Al igual que el texto que ponemos en nuestro documento html, las imágenes pueden tener atributos para darle el aspecto y dimensiones que queramos. Vamos a ver algunos de ellos.

- Atributo height: este atributo te permite indicar el alto en píxeles de la imagen.Las imágenes que añadas pueden tener el tamaño que quieras, pero cuanto más grande sea la imagen, mayor tendrá que ser el fichero y podría ralentizar mucho tu página.

- Atributo src: este atributo es el que especifica el nombre del fichero gráfico. Los más utilizados son los formatos GIF, JPG y PNG.

- Atributo align: este atributo define la manera en que se alinea una imagen respecto al texto que la acompaña. Puedes colocar la imagen a la derecha del texto utilizando la etiqueta right o a la izquierda del texto utilizando la etiqueta left. Y respecto a la posición de la imagen en la página, puede colocarse en el punto más alto para que coincida con la línea del texto actual utilizando la etiqueta Top, la imagen se puede alinear en el punto medio de la página con respecto al texto utilizando la etiqueta middle y, por último, la imagen puede ponerse en el punto más bajo de la imagen, con respecto al texto, utilizando la etiqueta bottom.
  <img src="libro.gif" align=bottom>

- Atributo width : este atributo te permite indicar el ancho en píxeles de la imagen. Las imágenes que añadas pueden tener el tamaño que quieras, pero cuanto más grande sea la imagen, mayor tendrá que ser el fichero y volvería a ralentizar mucho la carga de la página.

- Atributo alt: a la hora de añadir imágenes a nuestra web tenemos que tener en cuenta que hay personas que no pueden ver, por lo que es recomendable añadir una breve descripción de la imagen que queremos poner en el documento. Esta descripción de puede añadir mediante el atributo alt. A continuación, te mostramos como deberías utilizarlo:

  <img src="imagen.gif" alt="descripción">

  <img src="imagen.gif" alt="árbol pequeño con hojas amarillas">

Documentación técnica:

Introducción a las listas en HTML

Índice

1. ¿Qué son las listas? 5

2. Listas ordenadas y desordenadas 6

2.1. Listas ordenadas. 6

2.2. Listas no ordenadas o desordenadas 6

3. Listas de definición 8

4. Listas anidadas 9

1. ¿Qué son las listas?

Las listas en HTML son un medio que nos permite crear y presentar conjuntos de elementos en forma de lista en una página web.

Dentro de una lista podemos insertar textos, imágenes, tablas, inclusootras listas.

Hay tres tipos de listas:

- Listas ordenadas
- Listas no ordenadas o desordenadas
- Listas de definición

El formato de una lista es el siguiente, independientemente de si es una lista ordenada o desordenada:

<etiqueta del tipo de lista correspondiente>

 Elemento 1

 Elemento 1

.

.

.

 Elemento n

</etiqueta del tipo de lista correspondiente>

Las listas de definición tienen un formato diferente que veremos en el apartado correspondiente.

2. Listas ordenadas y desordenadas

2.1. Listas ordenadas

Las listas ordenadas son aquellas en las que los elementos que las forman se presentan de forma ordenada, habitualmente de menor a mayor. Los elementos de una lista ordenada están numerados, es decir, cada uno contendrá un número al inicio que nos indica el orden de ese elemento en la lista. Estos números no hay que introducirlos de manera específica, aparecerán de forma automática al utilizar la etiqueta <ol>, que es la etiqueta para generar listas ordenadas.

Un ejemplo de una lista ordenada sería:

```
<title>Ganadores</title>
 <ol>
        <li>Susana</li>
        <li>Luis</li>
        <li>Patricia</li>
</ol>
```

Y esto se mostraría en nuestra pantalla:

1. Susana
2. Luis
3. Patricia

2.2. Listas no ordenadas o desordenadas

Las listas no ordenadas o desordenadas son aquellas en las que los elementos que las forman no tienen ningún tipo de orden específico, es decir, todos los elementos tienen la misma relevancia y pueden leerse de forma aleatoria. Los elementos van precedidos de una viñeta que no hay que introducir de manera específica, sino que aparecen al utilizar la etiqueta <ul>, que es la etiqueta para generar listas desordenadas.

2. Listas ordenadas y desordenadas

Un ejemplo de lista desordenada sería:

```
<title>Lista de la compra</title>
<ul>
        <li>Azúcar</li>
        <li>Leche</li>
        <li>Café</li>
<ul>
```

Esto mostraría en nuestra pantalla:

- Azúcar
- Leche
- Café

Como podéis observar la diferencia entre ambas listas es la forma en la que se muestran en nuestra web, es decir, precedidas de números que indican un orden o de una viñeta que no indica ningún orden concreto.

3. Listas de definición

Las listas de definición son listados de un conjunto de elementos formados por un valor o término y una definición o descripción, es decir, es un listado de elementos con su definición o descripción.

Para definir una lista de definición utilizamos la etiqueta <dl>, para definir el término o valor utilizamos la etiqueta <dt> y para definir la definición o descripción utilizamos la etiqueta <dd>.

Las etiquetas <dt> y <dd> siempre van juntas, sin importar el orden en que aparezcan o el número de veces en que lo hagan, es decir, podemos tener varios términos para una sola definición o descripción, pero también podemos tener varias definiciones o descripciones para un solo término.

Un ejemplo de lista de definición sería:

```
<title>Páginas de consulta</title>
<dl>
        <dt>Educaciones</dt>
        <dd>Plataforma de contenidos formativos.</dd>
        <dd>Esta plataforma está estructurada en diferentes sectores.</dd>
        <dt>Fórmate</dt>
        <dd>Listado de cursos de formación.</dd>
</dl>
```

Que veríamos en nuestro web de la siguiente forma:

Educaciones

Plataforma de contenidos formativos.

Esta plataforma está estructurada en diferentes sectores.

Fórmate

Listado de cursos de formación.

4. Listas anidadas

Independientemente del tipo de listas que estemos utilizando en el desarrollo de nuestra página web, podemos anidar unas con otras haciendo que elemento de una lista sea, a su vez, una lista en sí mismo. No hay límite de anidación.

Para que quede más claro vamos a ver un ejemplo:

```
<ol>
    <li>Introducción</li>
    <li>Tema 1:
      <dl>
        <dt>Listas ordenadas.</dt>
          <dd>Estudiaremos lo que son las listas ordenadas</dd>
        <dt>Listas no ordenadas.</dt>
          <dd>Estudiaremos lo que son las listas no ordenadas.</dd>
      </dl>
    </li>
    <li>Tema 2:
      <ul>
        <li>Tablas</li>
        <li>Formularios</li>
      </ul>
    </li>
  </ol>
```

Esto es lo que veríamos en nuestra web:

1. Introducción
2. Tema 1:
   - Listas ordenadas.
     - Estudiaremos lo que son las listas ordenadas.
   - Listas no ordenadas.
     - Estudiaremos lo que son las listas no ordenadas.
3. Tema 2:
   - Tablas
   - Formularios

Documentación técnica:

Introducción a las tablas HTML

Índice

1. ¿Qué son las tablas? 5

2. Atributos de <table> 6

3. Elementos del atributo <tr>, atributo <td> y etiqueta <th> 9

3.1. Atributo <tr>: elementos 9

3.2. Atributo <td>: elementos 10

3.3. Etiqueta <th>: encabezado de columna 10

1. ¿Qué son las tablas?

A la hora de diseñar una página web existen diferentes herramientas que nos permiten organizar la información que queremos plasmar en la misma. Entre las diferentes herramientas, destacaremos las tablas, que nos permiten mostrarla información de una forma ordenada y clara.

Existen otras herramientas como los parámetros *aling*, pero sin embargo, la etiqueta específica de <table> de HTML te permitirá maquetar tu página web de una manera más controlada.

En primer lugar, se pueden definir las características generales de la tabla, por ejemplo, número de filas y columnas, el color del fondo y los bordes. Después, se puede definir cada fila con el elemento <tr> y cada celda en particular con el elemento <td>. Existirán tantos elementos <td> como columnas queramos definir.

Esto es así porque la etiqueta <table> encierra todas las filas y columnas de la tabla. Las etiquetas <tr> (del inglés "table row") definen cada fila de la tabla y encierran todas las columnas. Por último, la etiqueta <td> (del inglés "table data cell") define cada una de las columnas de las filas, aunque realmente HTML no define columnas sino celdas de datos.

Por ello, al definir una tabla, se debe pensar en primer lugar en las filas que la forman y a continuación en las columnas. El motivo es que HTML procesa primero las filas y por eso las etiquetas <tr> aparecen antes que las etiquetas <td>.

A continuación, te mostramos un ejemplo de cómo se crearía una tabla con 2 filas y 3 columnas:

```
<table>
<tr>
        <td>1,1</td><td> 1,2</td><td> 1,3</td>
</tr
>
<tr>    <td>2,1</td><td> 2,2</td><td> 2,3</td>

</tr
>

</table>
```

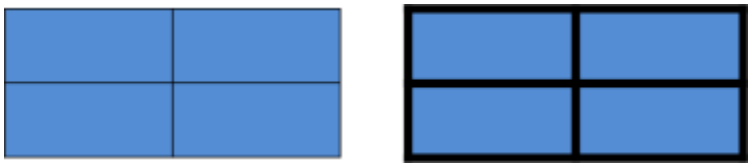
2. Atributos de<table>

Las tablas no siempre tienen que tener el mismo aspecto, por ello existen diferentes conjuntos de atributos del elemento <table> que nos ayudarán a caracterizar y diseñar nuestras tablas. Estos atributos son utilizados para diferentes temas como, por ejemplo, definir el grosor del borde de las celdas, determinará el tamaño de la tabla, etc. Vamos a ver cuáles son:

- Atributo Cellspacing: el atributo cellspacing te ayudará a definir el espacio entre las celdas. Este valor se expresa con un número, aunque ya no está soportado en HTML5.

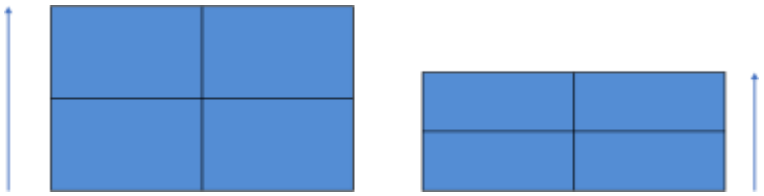
- Atributo Cellpadding: este atributo es diferente del anterior, ya que especifica el número de píxeles entre los bordes de una celda y el contenido de dicha celda.

- Atributo Border: podemos utilizar este atributo para definir el grosor del borde que define las celdas. Si no especificamos el grosor, su valor por defecto será 0, es decir no tendrá borde.



2. Atributos de<table>

- Atributo Height: el atributo height se utiliza para definir la altura de la tabla. La altura de la tabla se expresa con un número acompañado de un % cuando queramos que sea en porcentaje.



- Atributo Align: podemos utilizar este atributo para definir la alineación de la tabla en la página web, es decir, si queremos que la tabla estéa la derecha (right), a la izquierda (left) o en el centro (center).

- Atributo Width: el atributo width se utiliza para definir la anchura de la tabla. La anchura de la tabla se puede definir en porcentaje o píxeles, por ejemplo, 100% significa que queremos que la tabla ocupe todo el ancho de la ventana.

2. Atributos de<table>

Un ejemplo de código sería:

```
<table border="1" width="50%" align="center">
```

Este ejemplo representaría una tabla con grosor de borde 1, anchura al 50%, alineada al centro de la página.

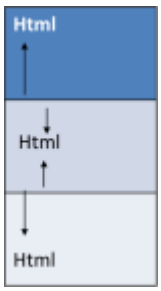
3. Elementos del atributo <tr> y <td>

Es importante tener en cuenta que los atributos que utilicemos tienen más prioridad cuando son establecidos para una celda que cuando se establecen para una fila completa, y a su vez, tienen más prioridad los atributos que se establecen para una fila que los atributos que se establecen para toda una tabla.

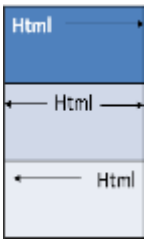
3.1 Atributo <tr>: elementos

Podemos utilizar el atributo <tr> para modificar todos los atributos de una fila. Aparte de las funcionalidades de los atributos que se han mencionado con anterioridad, el atributo<tr> tiene dos elementos que sirven para gestionar las filas:

- Atributo Valign: se utiliza para alinear el contenido de las celdas de la fila verticalmente, abajo (bottom), arriba (top) o al centro (center).



- Atributo Align: este atributo se utiliza para alinear el contenido de las celdas de la fila horizontalmente a la derecha (right), a la izquierda (left) o al centro (center).



3. Elementos del atributo <tr> y <td>

3.2 Atributo <td>: elementos

Por otro lado, tenemos el elemento <td> que sirve para gestionar cada una de las celdas de la tabla.

- Atributo width: con este atributo podrás definir la anchura de las celdas de tu tabla. Esta medida puede ser expresado en píxeles y en porcentajes.

- Atributo rowspan: este atributo te permiti rá determinar el número de celdas de una columna quieres unir para que pertenezcan a una misma fila, es decir, te permitirá combinar celdas.

- Atributo colspan: este atributo te permite especificar el número de celdas de la fila que se van a combinar en una misma columna. Es igual que el atributo rowspan, pero combinando celdas de diferentes columnas.

- Atributo align: este atributo te permite alinear el contenido de la celda horizontalmente a la derecha (right), a la izquierda (left) o al centro (center).

- Atributo valign: este atributo te permite alinear el contenido de la celda verticalmente hacia arriba (top), hacia el centro (center) o hacia abajo (bottom).

- Atributo nowrap: Este atributo no tiene valor en sí, si se quiere utilizar se tiene que añadir a la etiqueta <td>. Este atributo impide que el contenido de la celda no se ajuste de manera automática al ancho de la columna, sino que ocurra al revés, que el ancho de la celda se adapte al ancho del contenido. Esto permite que el contenido ocupe sólo una fila.

3.3 Etiqueta<th>: encabezado de columna

Ya hemos visto en el apartado anterior que la etiqueta <td></td> se utiliza para definir todas las celdas de cada una de las filas, pero a parte de este atributo, podemos utilizar el atributo <th> </th>.

3. Elementos del atributo <tr> y <td>

Las celdas escritas con la etiqueta <th> y su correspondiente cierre, admiten los mismos atributos que las etiquetas <td> y funcionan de la misma forma, salvo que el contenido que esté dentro de una etiqueta <th> está considerado como el encabezado de la tabla, por lo que se creará en negrita y centrado sin que se lo indiquemos explícitamente.

Además de los elementos anteriormente mencionados, es importante conocer el elemento <caption>. Con este elemento se puede añadir un titular o un título encima de nuestra tabla. Para utilizar esta etiqueta tendremos que utilizar <caption></caption >.

Documentación técnica:

Introducción a los formularios en HTML

Índice

1. ¿Quéson losformularios?..... 5

2. Atributos de <form>..... 7

1.¿Qué son los formularios?

Los formularios son herramientas que se incorporan a las páginas web para recoger datos introducidos por los usuarios.

Los datos recogidos pueden ser datos personales como el nombre, la edad, el sexo etc. Estos datos también pueden ser dudas u opiniones de los usuarios con respecto a cualquier temática que haya en nuestra web.

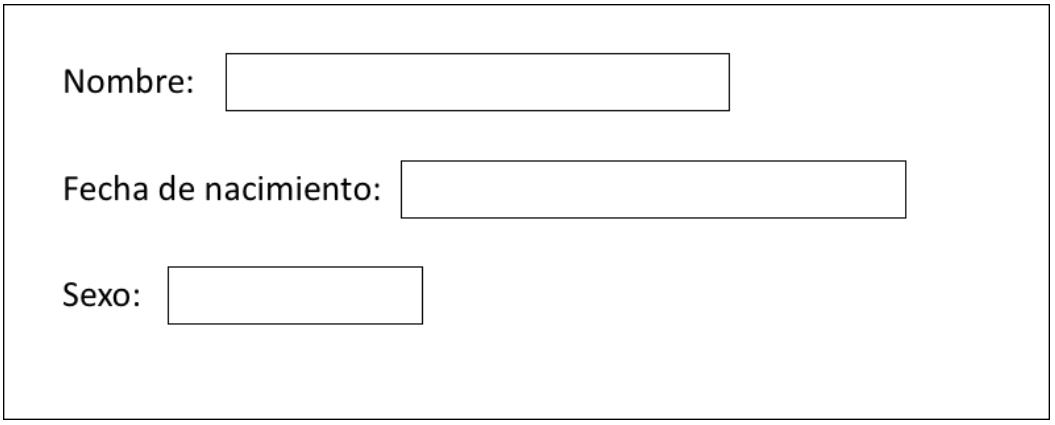


Figura 1.Ejemplo de formulario

A su vez, hay muchos tipos de formularios, por ejemplo, formularios simples en los que sólo hay dos campos de texto y formularios con botones y menús desplegables. Si quieres incrustar formularios con más de dos campos de texto y con menús desplegables terecomendamos que utilices tablas, ya que te ayudarán a organizar la información de una manera más sencilla así como a mejorar su apariencia.

1.¿Qué son los formularios?

DATO	VALOR
NOMBRE	LUCIA
SEXO	(•) MUJER ( ) HOMBRE
ALTURA	1,70
Describe que deportes practicas:	
SIGUIENTE →	

Figura 2. Ejemplo de formulariocon tabla

2. Atributos de <form>

Para incrustar un formulario en tu página web deberás hacerlo mediante la etiqueta FORM. Deberás poner <form> como etiqueta de apertura y </form> como etiqueta de cierre.

La etiqueta <form> tiene los siguientes atributos que te ayudarán a darle el diseño y la forma que quieras a tus formularios:

- Atributo Enctype: con este atributo establecerás el método de envío de la información de los formularios, es decir, con este atributo se elige de qué modo vas a recibir la información que los usuarios hayan añadido a tu formulario.

- Atributo Enctype="text/plain": este atributo permite que puedas recibir las respuestas a tus formularios de forma legible.

  Ejemplo de Enctype que permite subir archivos al servidor:
  <form enctype="multipart/form-data"></form>
  Significa que permite enviar archivos desde el formulario al servidor, subir ficheros…etc

- Atributo Action: este atributo se utiliza para indicar a qué sistema, programa o a qué correo se va a enviar la información de los formularios. Puede ser un correo personal en el que tendrás que ser tu quien analice la información.Otra opción es que la información pase a un sistema o un programa que analizará los datos, en función de lo que se haya programado o del objetivo que se quiera obtener de dichos datos y según esto, se podrá generar una respuesta al usuario que se mostrará en el servidor.

- Atributo Method: este atributo se utiliza para definir el formato en el que se enviará la información que se recaba a través del formulario. Existen dos valores posibles que son GET y POST. Si no se elige el formato, GET será el formato por defecto. GET tiene como límites que no permite enviar más de 500 bytes de información ni tampoco permite enviar documentos adjuntos, por lo que debes tener esto en cuenta si tu formulario tiene mucha información o los usuarios pueden o deben adjuntar algún documento o fotografía.

Documentación técnica:

Introducción a CSS

Índice

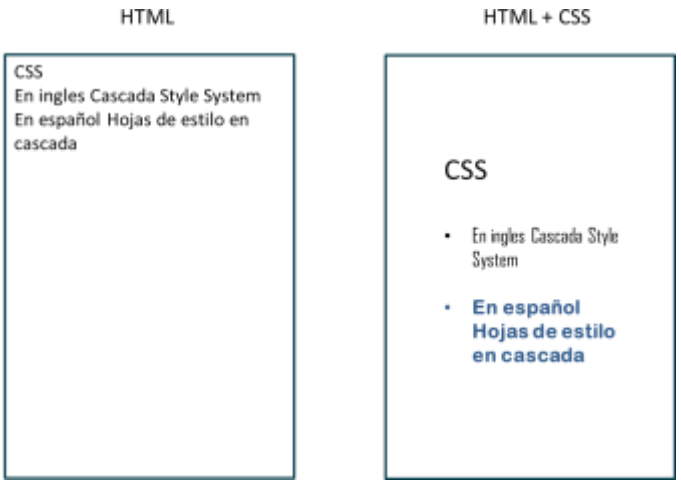
1. ¿Quées CSS?..... 5
2. ¿Cómose utiliza CSS?..... 7

1.¿Qué es CSS?

CSS responde a las siglasen inglés "Cascading Style Sheets " y en español "hojas de estilo en cascada". Es el lenguaje que define la apariencia de una página web, es decir, con las CSS puedes cambiar el color del fondo de la página, el color de la tipografía, el tamaño de los márgenes, el tamaño de las imágenes, etc. Y como dato interesante, el primer navegador que lo soportó fue Internet Explorer 3.0 de Microsoft.

Una hoja de estilo CSS es complementaria a otros documentos, por ejemplo, escritos en lenguaje HTML. De esta forma, son documentos independientes, que pueden modificarse sin alterar el otro, aunque también es posible crear un documento combinado HTML y CSS si así se desea. La combinación entre el lenguaje HMTL y CSSfacilitaen gran medida la navegación por las páginas web a las personas que tienen alguna discapacidad visual, ya que se pueden modificar los niveles de resolución o el tamaño de las letras sin tener que cambiar el contenido de la página web.

Para afianzar conceptos, la hoja HTML definiría el contenido de la web y la hoja de estilo CSS, definiría la apariencia de dicha página web.



1.¿Qué es CSS?

Se llama estilo en cascada porque se crea de arriba hacia abajo, y es un estándar que es aceptado por los principales navegadores como Google Chrome o Firefox.

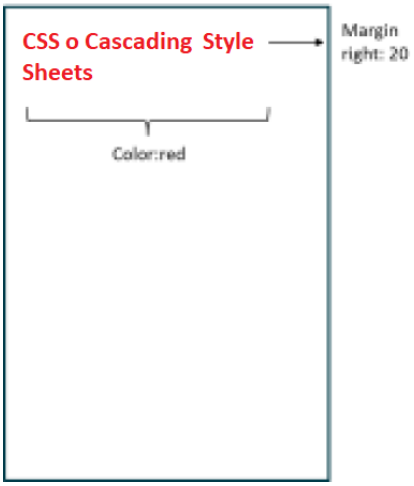
Otro de los beneficios de escribir el estilo de una página web en lenguaje CSS, es que, debido a la gran variedad de tamaños de pantallas de dispositivos móviles, se pueden crear diferentes estilos de apariencia que se adapten a cada tamaño y resolución sin tener que cambiar el contenido de la página.

2. ¿Cómo se utiliza CSS?

Como se ha indicado en el apartado anterior, el lenguaje CSS se utiliza para determinar la apariencia y el estilo de una página web. Para ello utilizaremos la etiqueta <style> y deberá ser colocada al principio de la página.

A continuación, te mostramos un ejemplo del formato que podría tener el lenguaje CSS para una página independiente de otra página escrita en HTML:

```
<style type="text/css">
        p {
                color: red; margin-right: 20px;
        }
</style>
```



<type> indica el tipo de medio que se va a utilizar, en este caso es textual y de tipo CSS, por tanto, se escribe:text/css.

p es la etiqueta que indica que ese párrafo se va a escribir en color rojo y con un margen a la derecha de 20 pixeles.

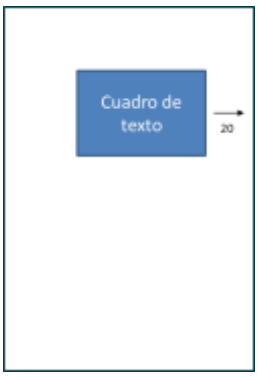
Si por ejemplo en un documento el lenguaje HTML y CSS están juntos, el lenguaje CSS se integrará con HTML.

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

Si por ejemplo, queremos utilizar el mismo estilo para una clase determinada, como puede ser el estilo de un cuadro de texto, podríamos utilizar:

2. ¿Cómo se utiliza CSS?

p.b lue{color:blue; margin-right:20px; }
El Código en HTML será: <p class="blue">



Todos los cuadros de texto que tengan este código serán de color azul y estarán a 20 píxeles a la derecha del margen. También se pueden poner las dimensiones de dichos cuadros.

Documentación técnica:

Sintaxis básica de CSS

Índice

1. ¿Cómo se aplica CSS? 5
2. Sintaxis básica de CSS 7

1. ¿Cómo se aplica CSS?

Con CSS vamos a tener la posibilidad, como se ha comentado con anterioridad, de ir aplicando diferentes propiedades o formatos a diferentes elementos contenidos en un documento HTML.

El objetivo de CSS es bastante sencillo de entender: separar todo lo que sería el contenido de un documento (que iría en un documento HTML, y que representa la información que nosotros queremos mostrar o simplemente aquella con la que queremos trabajar) de lo que sería la forma de representar esa información (que iría en un documento CSS, y cuyos cambios no afectarían a la información en sí contenida en el documento HTML). Esto nos permite, además, compartir fácilmente estilos entre múltiples documentos alojados en un mismo sitio web.

Las reglas de estilo se van a aplicar de forma jerárquica, o lo que es lo mismo, partiendo de los elementos superiores en un nivel de definición, hasta elementos inferiores en esos niveles de definición. Las reglas en cuestiónlas vamos a poder definir de tres maneras diferentes:

- En una hoja de estilo externa: dispondremos de un archivo de extensión .css que contendrá las reglas que queremos aplicar. Este archivo estará enlazado con el documento HTML mediante el uso de la etiqueta <link> que ya vimos con anterioridad. Tendríamos en el archivo .html lo siguiente:

  ```
  <link rel="stylesheet" href="ejemploHoja.css" type="text/css" />
  ```

  Y en el documento llamado ejemploHoja.css tendríamos lo siguiente:

  ```
  body {background-color: blue;}
  h1 { color: red;
        font- family: "Arial Narrow";}
  ```

- En una hoja de estilo interna: en este caso, no tenemos un archivo .css enlazado con un archivo .html, sino que las propias reglas van a estar colocadas en la cabecera (head) del documento html en cuestión. Un ejemplo sería:

  ```
  <head>
        <title>Ejemplo de documento</title>
  ```

1. ¿Cómo se aplica CSS?

```
        <style>
            body {background-color:blue;}
            h1 {color: red;
                font- family: “Arial Narrow”;}
            h2 {font-family: "Ubuntu Condensed"; }

            p {color: yellow; }
        </style>
    </head>
```

- Aplicando el estilo a un elemento en cuestión en lo que se llama un estilo en línea: aquí mediante el uso del atributo style, asociamos un estilo determinado a un elemento cualquiera. Podríamos tener el siguiente ejemplo:

```
    <body>
        <h1>Bienvenidos al siguiente ejemplo</h1>
        <p style="font-style: italic; text-align: center; color: pink;">Hola</p>
    </body>
```

Hay que tener en cuenta que, puesto que hay diferentes formas de aplicar las etiquetas, tenemos que saber si todas son tratadas de la misma manera o las hay algunas que se aplican con una mayor prioridad que otras, que es lo que verdaderamente ocurre. De este modo hay que saber que la prioridad de aplicación de las reglas de CSS es la siguiente:

1 Estilo en línea (regla que se define dentro del propio elemento en HTML).

2 Estilo que se define en una hoja de estilo interna (en el apartado <head> del documento HTML).

3 Estilo que se define en una hoja de estilo externa (que se ha enlazado mediante la etiqueta <link>).

4 Otros estilos, como, por ejemplo, los propios definidos por los navegadores.

2. Sintaxis básica de CSS

De forma general, las reglas que indicaremos en CSS siguen el siguiente formato:

selector { propiedad: valor }

Si lo que hacemos es aplicar una única regla a lo que llamamos un selector, o bien

selector { propiedad1: valor1;

propiedad2: valor2;

...

propiedadN: valorN }

si lo que vamos a aplicar son un conjunto de reglas sobre el selector en cuestión.

En cualquier caso, un aspecto importante a tener en cuenta es qué vamos a entender por un selector. Para nosotros un selector va a ser una descripción que nos va a permitir identificar sobre qué elemento o elementos vamos a aplicar las diferentes reglas que hemos definido. Como veremos con posterioridad, los selectores pueden ser básicos (por elemento, por clase o por identificador), o pueden ser algunos más complejos (por anidamiento, por posición según qué orden de hijo se posee, etc.).

Podemos combinar varios selectores (separados por comas) para indicar que las reglas se aplican a todos ellos. De este modo, a la hora de especificar las reglas, vamos a reducir el coste de escribirlas: todas las que sean iguales, se agruparán en una sola y proporcionaremos una lista de selectores que la cumplen:

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

Del mismo modo, es posible que encontremos más de una regla que se aplica sobre un mismo elemento dentro del archivo CSS (si estuvieran en diferentes formatos, se aplicaría la regla de prioridad que se mostró en el apartado anterior). Siempre que tengamos varias reglas que no se solapan entre sí, el resultado será

la suma de la aplicación de las mismas. En el caso de que se produzca un solapamiento, tendrá prioridad la que se definió en último lugar. Así, el color resultante de aplicar las siguientes reglas sobre el elemento p seríaun tipo de letra Courier New y un color amarillo:

```
p{font- family: “Courier New”;
        color : blue}

p {color :yellow}
```

Documentación técnica:

Selectores básicos en CSS

Índice

1. ¿Qué son los selectores?..... 5
2. Selector Universal y Selector tipo 6
3. Pseudoclases y pseudoelementos..... 7

1. ¿Qué son los selectores?

En CSS se utilizan selectores para elegir aquellos elementos de HTML a los que se le quiere dar un diseño de estilo.

Existen gran cantidad de selectores que te ayudan a ser precisos a la hora de utilizar CSS en tus documentos HTML. Combinando los diferentes selectores, conseguimos que para mismos elementos de HTML (por ejemplo, aquellos delimitados por un párrafo usando la etiqueta <p>), a unos se les aplique una regla, mientras que a otros no se les aplique la misma.

Es importante recordar que cuando hablamos de los diferentes elementos de HTML, en verdad estamos asumiendo que cada uno de los elementos se comporta como una caja (de ahí lo que se llama el modelo de cajas). Es por elloque, si cada elemento tiene una caja, vamos a poder definir dos características muy importantes relacionadas con la caja, a saber:

- El margen (llamado margin), que representa el espacio que hay alrededor del elemento. Por lo tanto, este espacio es lo que ocupa el elemento por fuera de su borde (o de la propia caja).
- El relleno (llamado padding), que representa, al contrario de lo que decíamos del margin, el espacio que hay dentro del elemento entre su borde y el propio contenido del elemento.

Por lo tanto, se va a poder definir características de estos espacios de una manera conjunta (por ejemplo, todos los márgenes son iguales), o diferenciada (el margen superior será diferente del inferior, por ejemplo).

2. Selector Universal y Selector de Tipo

El selector universal afecta a todos y cada uno de los elementos de la página en cuestión. Viene identificado por el asterisco (*), y, por lo tanto, si se define una regla con el mismo, se consigue aplicar dicha regla a cualquier elemento definido, sin necesidad de realizar una labor adicional. De este modo, si se quisiera que todos los elementos tuvieran un margen de 0px, un relleno de 0px y que el color del texto fuera azul, podríamos escribir la siguiente regla usando el selector universal:

```
* {
        margin: 0px;
        padding: 0px;
        color: blue;
}
```

Por otro lado, cuando se quiere definir reglas que afecten a un tipo de elemento en concreto, lo que se emplea es el llamado selector de tipo. Así, lo que se tiene es una regla en la que se indica a qué tipo (etiqueta) se quiere aplicar el conjunto de normas especificadas. De este modo, por ejemplo, si se quisiera que todos los párrafos tuvieran un relleno de 15px, se podría especificar así:

```
        p {
                padding: 15px;
}
```

A diferencia de lo que ocurría con el selector universal, el selector de tipo permite que la regla definida solo se aplique, en el caso del ejemplo, a los elementos <p> del documento HTML.

Existen otros muchos selectores, como puedan ser los selectores de clase, de identificador, y demás, que se tratarán en documentos posteriores.

3. Pseudoclases y pseudoelementos

Las pseudoclases son unos tipos de selectores muy específicos, que van a permitir, en lugar de aplicar un conjunto de reglas a un elemento, aplicar reglas al comportamiento de ciertos elementos. Emplean el operador ":". Hay varias, pero podemos destacar entre ellas:

- :link – Se aplica a los enlaces nunca visitados. Solo se puede aplicar a los enlaces (elemento <a>)
- :visited – Se aplica a los enlaces visitados (al menos, una vez). Solo se puede aplicar a los enlaces (elemento <a>)
- :hover – Se aplica a un elemento seleccionado por el usuario pero sin activarlo. Por ejemplo, colocar el ratón sobre un objeto sin pulsar el botón.
- :active – Se aplica a los elementos que están siendo activados, es decir, aquellos que están siendo seleccionados pulsando el botón izquierdo.
- :focus – Se aplica a un elemento que tiene el foco. Por ejemplo, el recuadro activo en un momento determinado en un formulario.
- :first-child – Esta pseudoclase selecciona el primer elemento hijo de un elemento.

Como ejemplo de esto indicado, si quisiéramos definir una regla que cambiara el fondo de una casilla de una tabla al color amarillo cuando pusiéramos el ratón sobre ella, podríamos escribir lo siguiente:

```
td:hover {
    background-color: yellow;
}
```

Los pseudoelementos son algo parecido a las pseudoclases A grandes rasgos, ser podría decir que un pseudoelemento permite modificar no un elemento en su totalidad, sino una parte de él. Emplean el operador "::". Entre los pseudoelementos más interesantes podemos encontrar:

- ::first- line – Se aplica a la primera línea de un texto. Sólo se puede utilizar con los elementos de bloque y las celdas de datos de las tablas.

3. Pseudoclases y pseudoelementos

- ::first- letter –Se aplica a la primera letra de un texto. Sólo se puede utilizar con los elementos de bloque y las celdas de datos de las tablas. Los signos de puntuación y los caracteres como las comillas que se encuentran antes y después de la primera letra también se ven afectados por este pseudo-elemento.
- ::before –Permite añadir contenido antes del contenido original de un elemento.
- ::after –Permite añadir contenido después del contenido original de un elemento.

Así, por ejemplo, si se quisiera que la primera letra de cada párrafo se colocase en cursiva, se podría establecer la siguiente regla:

```
p::first-letter {

    font-style: italic;

}
```

Documentación técnica:

Selectores ID

Índice

1 Selectores ID 5

1.1 Cómo usar los selectores ID 6

1. Selectores ID

Los selectores ID de CSS en un documento HTML, permiten buscar un elemento único que se base en el contenido del atributo id. El nombre del ID debe ser único en el documento.

Los selectores ID se utilizan cuando queremos modificar, con nuestras instrucciones, el estilo de un solo elemento o a un grupo de elementos específicos contenidos en un documento HTML, pero no al resto de elementos.

Con estos selectores podremos aplicar un estilo único a un elemento específico de forma independiente al estilo que se le haya aplicado al elemento de forma general.

Como comentábamos, los selectores ID son elementos únicos, es decir, que solo puede existir uno, en cada página de un sitio web, con el mismo nombre de atributo.

Los selectores ID se definen utilizando la siguiente estructura:

#valor_id {propiedad: estilo que se quiere dar al elemento;}

Por ejemplo, si tenemos el siguiente código en nuestro documento HTML:

<title id="especial">Este título tiene un ID definido</title>

Tendremos que definir en nuestro documento CSS lo siguiente:

```
#especial {
        font-weight: bold;
}
```

Esto quiere decir que el elemento con la etiqueta title tendrá la propiedad definida en el id con nombre especial. El id con nombre especial indica que el texto estará en negrita, a través de la propiedad font-weight que nos permite especificar el grueso de la letra. Por lo tanto veríamos:

Este título tiene un ID definido

Para los nombres de los atributos id podemos utilizar mayúsculas y minúsculas, además de algunos caracteres como el guión medio -. También podemos utilizar números, siempre que el primer elemento no lo sea, es decir, los nombres no pueden empezar por un número.

1. Selectores ID

Los nombres no pueden contener tildes, ni espacios en blanco y tampoco la letra ñ.

Si necesitamos utilizar palabras compuestas para un nombre podemos separarlas por un guión, por ejemplo, id="caso-especial" o podemos jugar con las mayúsculas y minúsculas, por ejemplo, id="casoEspecial".

1.1. Cómo usar los selectores ID

Para usar un selector ID tendremos que crear una regla en nuestro documento CSS como en el ejemplo que hemos visto anteriormente.

En una misma página podemos crear tantos selectores ID como necesitemos, pero teniendo en cuenta que cada uno de ellos debe tener un nombre único.

Vamos a ver un ejemplo de código HTML incluyendo un selector ID de CSS, pero utilizando la etiqueta style para incluir el código CSS directamente en el documento HTML.

En este ejemplo vemos como dependiendo del uso del selector ID diferentes párrafos se representan de diferente manera:

```
<head>
        <style type="text/css">
                p{
                        font-weight: normal;
                }
                #especial{
                        font-weight: bold;
                }
        </style>
</head>
<body>
        <p>A este párrafo le afecta el selector de etiqueta (p{}). .</p>
        <p id="especial">A este párrafo le afecta el selector ID.</p>
        <p>A este párrafo también le afecta el selector de etiqueta (p{}). .</p> </body>
```

1. Selectores ID

En este caso lo que veríamos en el navegador sería:

A este párrafo le afecta el selector único.

A este párrafo le afecta el selector ID.

A este párrafo también le afecta el selector único.

El selector ID asigna el estilo de una regla a un elemento concreto sin modificar el resto de elementos.

El lenguaje CSS se lee en cascada, así que es normal que la regla con selector id sobrescriba a la del selector global porque está escrita después. Si tenemos el código CSS siguiente:

```
<style type="text/css">
    p{
    color: yellow;
    }
    #azul{
    color: blue;
    }
</style>
```

Y reordenamos las reglas y colocamos la del selector id la primera

```
<style type="text/css">
     #azul{
    color: blue;
    }
    p{
    color: yellow;
    }
    </style>
```

1. Selectores ID

El resultado visual será exactamente el mismo porque el selector id prioriza ante las reglas globales. El selector id hace una selección concreta para un elemento y le asigna el estilo de esa regla obviando al resto.

Documentación técnica:

Selectores Class

Índice

1. ¿Qué representa un selector class y por qué es importante? 5
2. Usando el selector class 6

1. ¿Qué representa un selector class y por qué es importante?

Es habitual querer agrupar un conjunto de elementos HTML por diferentes motivos (legibilidad, semántica, etc.). En el caso que ocupa, puede ser interesante agruparlos de alguna manera con vistas a poder aplicarle a esos elementos un conjunto de reglas comunes.

Por poner un ejemplo, imagínate un conjunto de párrafos definidos mediante etiquetas <p>, de los cuales se quiere que algunos se escriban en color rojo, y otros se escriban de color azul. Una forma simple (que no rápida) sería aplicar a cada uno de los párrafos una regla de CSS en línea (mediante el atributo style) que permitiría tomar el color correspondiente a cada uno de los párrafos afectados.

Esto a todas luces parece poco práctico, porque querer modificar de manera simple el código (por ejemplo, a otro tipo de tono rojo o de tono azul) forzaría el revisar todo el código HTML buscando párrafos afectados.

El atributo class nos permite que podamos ahorrarnos este trabajo. Con class vamos a agrupar elementos que queremos que pertenezcan a una misma clase. De esta manera se podrían definir un conjunto de reglas a aplicar a todos los elementos que tienen un class coincidente. De este modo, si con posterioridad se quiere cambiar esta regla para todos los elementos, sólo se debería actuar sobre un punto, ahorrándonos la labor de mantenimiento.

2. Usando el selector class

El formato general de uso del selector class es el siguiente:

```
.nombreClase {
        propiedad1: valor1; propiedad2: valor2;
        …
        propiedad: valorN;
}
```

Así, por ejemplo, si se quiere aplicar una regla a todos los elementos que tienen el atributo class="coche", poniendo el color correspondiente a azul, la regla quedaría definida de la siguiente manera:

```
.coche {
        color: blue;
}
```

Aquí se podría plantear la siguiente "problemática": hay diferentes elementos que llevan este valor de class="coche", y no todos ellos admiten todas las propiedades establecidas en la regla definida. En este caso lo que ocurrirá es que sólo se aplicaran a los elementos aquellas reglas que le son permitidas, ignorándose las restantes.

Del mismo modo, podría imaginarse que se tienen elementos p cuyo class="coche" y elementos h1 cuyo atributo class="coche", pero solo se quiere que a los elementos p se les aplique la regla de colocar eltexto en azul, no afectando a los h1. En ese caso, la regla anterior debería reescribirse del siguiente modo:

2. Usando el selector class

```
p.c oche {
        color: blue;
}
```

De este modo, lo que se consigue es que a todos los elementos p cuyo atributo class="coche" se les aplicará la regla de cambiar el color del texto a azul, quedando exentos de aplicación cualquier otro elemento que lleve el class="coche" pero que no cumplan con la restricción de ser elemento p.

Documentación técnica:

Combinadores y selectores

Índice

1. Combinando diferentes selectores 5

1. Combinando diferentes selectores

En la mayoría de los casos, no queremos aplicar un formato determinado a todos los elementos de tipo p, por ejemplo, sino que vamos a querer elegir un elemento p u otro en función de una serie de características (por ejemplo, el primer p que se encuentre debajo de la etiqueta body), para ello, CSS permite el empleo de un conjunto de reglas y combinaciones muy potentes que se van a adaptar a nuestras necesidades. Vamos a ver las principales, partiendo de algunas que ya conocemos, hasta otras muy específicas:

- Selector Universal: aplica a todos los elementos de la página. Se emplea el asterisco (*). Ejemplo:

  ```
  * {font- family: Courier; color: blue;}
  ```

- Selector de Tipo: permite escoger aquellos elementos cuya etiqueta coincide con la que se indica. Ejemplo:

  ```
  a {font-size: 30 px;}
  ```

- Selector Descendiente: un elemento es descendiente de otro si está contenido entro de él (entre su apertura y su cierre). Podemos definir que se aplique un formato a un elemento B descendiente de otro A en cualquier nivel. En el ejemplo, se aplica al elemento a que está bajo p:

  ```
  p a {font-family: Arial;}
  ```

- Selector Descendiente excepto Hijo: se aplica un formato a un elemento B descendiente de otro A excepto si B es hijo de A. Se usa combinado con el selector universal. Ejemplo:

  ```
  p * a {font- family: Arial;}
  ```

1. Combinando diferentes selectores

- Selector Hijo: el formato se aplica al primer descendiente de un elemento. Ejemplo:

```
p > a {Font-family: Arial;}
```

- Selector Adyacente: se aplica a un elemento M que va a continuación de un elemento E, teniendo E y M el mismo padre y yendo M a continuación justo de E. Se define como E + M {declaración;}. Ejemplo:

```
h1 + h2 {font- family: Courier; color: blue;}
```

- Selector Adyacente No Inmediato: se aplica a un elemento M que va a continuación de un elemento E, teniendo E y M el mismo padre y sin tener que ir M a continuación justo de E. Se define como E ~ M {declaración;}. Ejemplo:

```
h1 ~ h2 {font- family: Courier; color: blue;}
```

Documentación técnica:

Selector de atributos

Índice

1. Aplicando reglas según los atributos y sus valores 5

..

1. Aplicando reglas según los atributos y sus valores

El empleo de atributos en HTML es muy común. Aunque algunos atributos tienen un tratamiento especial (como es el caso del atributo id o del atributo class), lo cierto es sería interesante el poder aplicar un tipo de formato o generar una regla específica en función de los valores que pudiera tomar ese atributo. Y es algo que CSS permite de una manera sencilla.

- Selector [atributo]: la regla se aplicará (combinado con otro selector) a aquellos elementos que cumplan que tienen ese atributo definido, con independencia de su valor. El ejemplo siguiente aplica a los elementos h1 que tengan definido el atributo attr:

```
h1[attr] {font- family: Courier; color: blue;}
```

- Selector [atributo="valor"]: la regla se aplicará (combinado con otro selector) a aquellos elementos que cumplan que tienen ese atributo definido y cuyo valor sea el especificado. El ejemplo siguiente aplica a los elementos h1 que tengan definido el atributo attr y valga "hola":

```
h1[attr="hola"] {font-size: 30 px;}
```

- Selector [atributo ~="valor"]: la regla se aplicará (combinado con otro selector) a aquellos elementos que cumplan que tienen ese atributo definido y cuya valor sea el especificado, siendo posible que el atributo sea una lista de posibles valores separados por espacios (es decir, que "valor" sea uno de los posibles elementos de esa lista). El ejemplo siguiente aplica a los elementos h1 que tengan definido el atributo attr y valga "hola" (entre otros valores que pudiera tomar en un momento determinado):

```
h1[attr~="hola"]{Font-family:Arial;}
```

1.Aplicando reglas según los atributos y sus valores

- Selector [atributo |="valor"]: La regla se aplicará (combinado con otro selector) a aquellos elementos que cumplan que tienen ese atributo definido y cuyo valor tiene que ser una única palabra. El ejemplo siguiente aplica a los elementos h1 que tengan definido el atributo attr y tenga exactamente el valor "hola" o empiece por "hola "seguido de un guión -

h1[attr|=”hola”{Font-family:Arial}

- Selector [atributo ^=”valor”]: la regla se aplicará (combinado con otro selector) a aquellos elementos que cumplan que tienen ese atributo definido y cuya valor comience por lo especificado (el valor no tiene que ser una única palabra). El ejemplo siguiente aplica a los elementos h1 que tengan definido el atributo attr y cuyo valor comienza por “hola”:

```
h1[attr^=”hola”] {font-family: Arial;}
```

- Selector [atributo $=”valor”]: la regla se aplicará (combinado con otro selector) a aquellos elementos que cumplan que tienen ese atributo definido y cuya valor acabe por lo especificado (el valor no tiene que ser una única palabra). El ejemplo siguiente aplica a los elementos h1 que tengan definido el atributo attr y cuyo valor finalicen por “adios”:

```
h1[attr$=”adios”] {font-family: Arial;}
```

1. Aplicando reglas según los atributos y sus valores

- Selector [atributo *=”valor”]: la regla se aplicará (combinado con otro selector) a aquellos elementos que cumplan que tienen ese atributo definido y cuya valor contenga el valor especificado (el valor no tiene que ser una única palabra, y lo indicado está contenido dentro del valor). El ejemplo siguiente aplica a los elementos h1 que tengan definido el atributo attr y cuyo valor contenga la palabra “adios”:

```
h1[attr*=”adios”] {font family: Arial;}
```

Documentación técnica:

Estilos de texto en CSS

Índice

1. Propiedades de fuentes 5

2. Propiedades de texto 6

1. Propiedades de fuentes

Hay definidas un conjunto elevado de propiedades en CSS relacionadas con las fuentes (o tipografías) de modo que podemos variar sensiblemente qué fuentes se emplean y la forma en la que estas se usan. Podemos encontrar las siguientes propiedades:

- font- family (familia de fuente): permite cambiar la familia de fuentes que se usa. Así, podremos especificar que queremos usar la familia Times New Roman en los títulos h1 mediante la siguiente regla:

```
h1 { font-family: "Times New Roman";}
```

- font- style (Estilo de fuente): permite cambiar el estilo entre normal, italic (tipo cursiva) u oblique (un tipo diferente de cursiva). Siguiendo con el ejemplo anterior:

```
h1 { font-style: italic;}
```

- font- variant (Variante de fuente): indica si el texto en minúsculas ha de representarse con minúsculas (normal) o con mayúsculas pequeñas (small-caps).

```
h1 { font-variant: small-caps;}
```

- font- weight (Peso de fuente): permite indicar si la fuente es normal (normal), negrilla (bold) u otros valores más específicos (lighter, bolder, 100, 150, 200 ...).

```
h1 { font-weight: bold;}
```

- font- size (Tamaño de fuente): indica el tamaño de fuente que se empleará. Se puede especificar un tamaño relativo o absoluto de la fuente. Así el tamaño se podría representar en diferentes unidades (pt, in, cm, mm, em, ex, px), en un porcentaje respecto a lo definido (10%, por ejemplo) o bien usando unos ciertos valores ya definidos: xx-large, x- large, large, médium, small, x- small, xx- small, smaller, larger.

```
h1 { font-size: xx-large;}
```

2. Propiedades de textos

Del mismo modo que podíamos actuar mediante reglas de CSS sobre los diferentes tipos de fuentes y realizar variaciones sobre el mismo, vamos a poder usar reglas específicas para modificar el formato o la forma en la que los diferentes textos aparecen representados. Podemos encontrar las siguientes propiedades:

- word- spacing (espaciado de palabras): permite variar el espaciado que encontramos entre una palabra y otra de un texto. Se puede especificar el valor normal o preestablecido (normal), un valor determinado (por ejemplo, 15px o por ejemplo -10px) o un porcentaje (5%).

```
h1{ word-spacing: -10px;}
```

- letter- spacing (espaciado de letras): similar al anterior, pero en lugar de trabajar sobre el espacio entre dos palabras, estamos actuando sobre el espacio entre dos letras. Mismas características que las mostrada en el apartado anterior).

```
h1 { letter-spacing: 5px;}
```

- text- decoration (decorado de texto): permite eliminar variaciones y/o decoraciones de un determinado texto. Entre los permitidos, none (elimina decorados), underline (subrayado), overline (línea superior), line-through (tachado). Por ejemplo, el siguiente ejemplo permite quitar el subrayado de un determinado enlace.

```
a { text- decoration: none;}
```

- text- transformation (transformación de texto): permite convertir todas las letras del texto a mayúsculas (uppercase), minúsculas (lowercase) o permite que la primera de cada palabra se convierta a mayúsculas (capitalize). Poner a mayúsculas la primera letra de cada palabra de un párrafo sería así:

```
p { text- transformation: capitalize;}
```

2. Propiedades de textos

- text-align (alineado de texto): permite establecer el tipo de alineado horizontal que se va a emplear para el elemento en cuestión. Se puede elegir entre alineado a la izquierda (left), derecha (right), centrado (center) o justificado a ambos lados (justify).

```
p { text-align: justify;}
```

- text- indentation (sangría de texto): fija el tamaño o el porcentaje que se aplicará a la indentación (o sangría) de la primera línea del texto

```
p { text- indent: 10px;}
```

- line-height (altura de línea): fija la distancia que habrá entre dos líneas del mismo párrafo. Puede tomar un valor por defecto (normal) o se puede especificar un tamaño determinado o un porcentaje.

```
p { line-height: 15em;}
```

Documentación técnica:

Colores en CSS

Índice

1. Propiedades relacionadas con colores 5

2. Propiedades relacionadas con fondos 7

1. Propiedades relacionadas con colores

El empleo de una gama amplia de colores permite enriquecer de manera significativa los desarrollos realizados en HTML. En ese sentido, CSS permite establecer un conjunto considerable de propiedadesrelacionadas con aspectos de los elementos (líneas, espacios, rellenos, ...) que permiten cambiar el color de estos aspectos.

El color puede definirse de dos maneras diferentes:

- Empleando constantes definidas para los colores: así, existen valores definidos para el rojo (red), azul (blue), amarillo (yellow) y la mayoría de los colores básicos (black, green, pink, orange, magenta, white, cyan, ...)
- Empleando el código RGB del color: el código RGB se obtiene sumando un valor entre 0 y 255 (en hexadecimal o decimal) para los parámetros Red (rojo), Green (verde) y Blue (azul). Se escribirá con el formato #RRGGBB, donde RR es el valor del componente rojo en hexadecimal, GG el del verde en hexadecimal y BB el del azul en hexadecimal. También se puede emplear el formato rgb(rrr,ggg,bbb), donde rrr representa el valor decimal del rojo (entre 0 y 255), ggg el del verde y bbb el del azul. Así, el color rojo (red) se representaría cómo #FF0000 o rgb(255,0,0).

Aunque la lista es considerable, las siguientes propiedades sonlas más empleadas habitualmente en relación al color. Cabe recordar que no todas las propiedades están admitidas en todos los elementos, por lo que definir una propiedad en una regla que se aplica a un elemento que no tiene definida dicha propiedad, provocará que no surta efecto alguno:

- color: es la propiedad básica y más común. En aquellos elementos que la tienen definida, permite cambiar el color del contenido (texto, generalmente):

h1 { color: blue;}

1. Propiedades relacionadas con colores

- background-color: permite cambiar el color de fondo del elemento:

table { background-color: #AABBCC;}

- border-color: permite cambiar el color del borde del elemento. Si se quisiera cambiar uno de los bordes en lugar de todos, se podría emplear border- top-color, border- bottom-color, border- left-color o border-right-color.

td{ border-color: pink;}

- column-rule-color: cuando se disponen de varias columnas (por ejemplo, al estilo de las columnas periodísticas) se puede definir el color de la línea de separación.

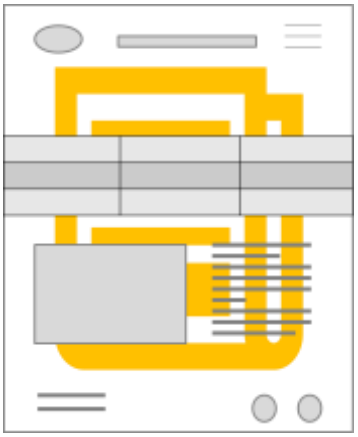
2. Propiedades relacionadas con fondos

El fondo que se aplica a un elemento o a un documento en su conjunto también dispone de un conjunto de propiedades que son interesantes de reseñar. De este modo, se conseguirá mejorar la riqueza multimedia del conjunto. Entre las propiedades más destacadas relacionadas con el fondo, se pueden destacar:

- background-color: permite cambiar el color de fondo del elemento:

table { background-color: #AABBCC;}

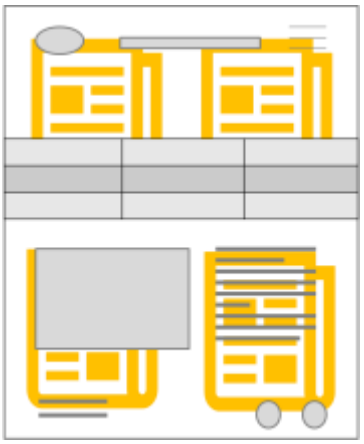
- background-image: permite que una imagen se coloque como fondo de un elemento (o incluso como fondo de toda la página web). Hay que especificar dónde está almacenada la imagen en cuestión (que puede estar en el equipo en cuestión o directamente puede ser una imagen que se encuentra en Internet, por ejemplo)



body { background-image: url("http://www.abc.com/img.jpg"); }

- background- repeat: permite que una imagen de fondo se repita en horizontal (repeat- x), en vertical (repeat- y), en horizontal y vertical (repeat) o que no se repita (norepeat). Continuando con el ejemplo anterior, una imagen que se repita en horizontal sería:

2. Propiedades relacionadas con fondos

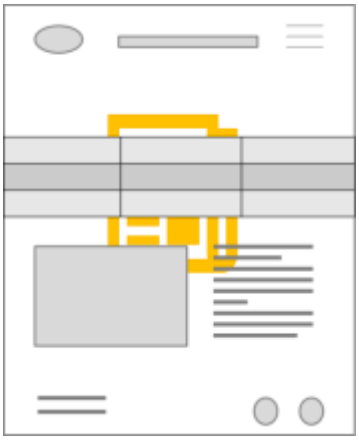


body background-image: url("http://www.abc.com/img.jpg");

background- repeat: repeat- x;}

- background- attachment: permite indicar si la imagen de fondo está fija (fixed) o se desplaza conforme desplazamos la página (scroll).

- background- position: permite colocar la imagen de fondo en una determinada posición (top left, top center, top right, center left, center center, center right, bottomleft, bottom center, bottomright), o bien especificar una posición fija mediante un valor concreto.



body { background- image: url("http://www.abc.com/img.jpg"); background- repeat: no- repeat; background-position: centercenter;}

- background: esta propiedad permite aglutinar todas las anteriores en una única regla. Su especificación sigue un formato determinado (el orden en el que tienen que especificarse cada una de las propiedades): background-color background- imagebackground- position background-size,background- repeatbackground- originbackground-clip background-attachment