**Módulo 1.Introducción a las tecnologías web. Lenguajes de construcción.**

**Introducción a las tecnologías web**

Las tecnologías web han evolucionado de manera espectacular en las últimas dos décadas.

Hoy en día son una gran mayoría de aplicaciones empresariales, de ocio como las redes sociales, webs de cines, tiendas ,de sectores como la banca, y un sinfín de sectores todos ellos inmersos en las tecnologías web.



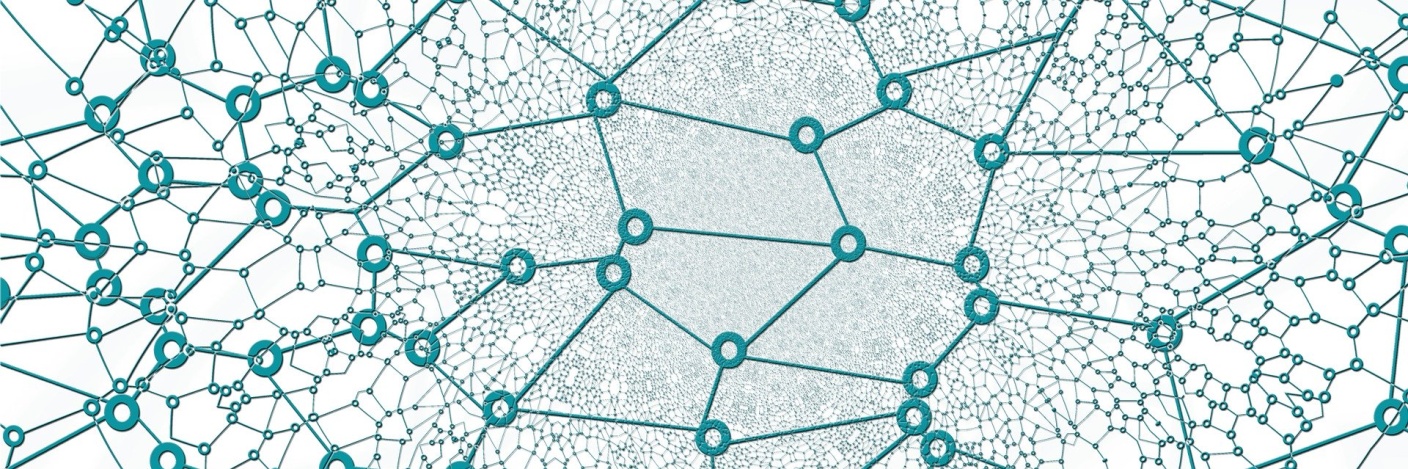
M1\_img\_1

Es también muy relevante su importancia dentro del mundo publicitario siendo una herramienta indispensable para mejorar no solo las tasas de ventas de productos sino para dar a conocer información a millones de usuarios de internet..

Lo que hoy conocemos como **internet** (la red de redes) es un conjunto de miles de redes conectadas entre ellas a través de elementos de red , tales como los routers que sirven para encaminar la información entre ordenadores de la red. Esta red tiene un alcance a nivel mundial y tiene sus orígenes en la red denominada ARPANET .

ARPANET es una red que fue diseñada en los Estados Unidos en los años 70, creada por encargo del Departamento de Defensa de los Estados Unidos (DOD) para usarla como medio de comunicación entre las diferentes instituciones académicas , estatales y gubernamentales.

La idea inicial es que si hubiese un conflicto bélico , no se destruyese de forma total las redes de comunicaciones.



M1\_img\_2

**Entorno de desarrollo web**

**Servidor WEB/Aplicaciones:**

-php

-java :

Servlets

JSP

Hibernate

Spring

-.net:

ASPX

**HTTP**

**Cliente WEB:**

-html

-html 5

-css (1,2,3)

-java script

En este esquema observamos las diferentes tecnologías tanto del lado del cliente como del lado del servidor.

## Internet y Extranet, conceptos.

Las intranets y extranets están de moda. Vamos a exponer algunos conceptos básicos. Las intranets y extranets comparten el mismo protocolo (TCP/IP) y las mismas tecnologías Web con Internet. Una **intranet** es un red cerrada limitada a la extensión de una empresa, pero que utiliza estándares abiertos como TCP/IP en lugar de los tradicionales protocolos propietarios utilizados en las LANs (redes de área local, normalmente cableadas) y en las WANs (redes de área extensa, normalmente LANs conectadas por cable, teléfono y conexiones sin cable).

Una **extranet** es una red WAN privada que utiliza protocolos públicos. Es decir, una extranet es una red virtual privada entre grupos privados basada en redes y protocolos abiertos. Para asegurar la seguridad y privacidad, una extranet se apoya en canales seguros utilizando protocolos especiales e identificadores digitales. Como idea y ejemplo, una extranet es una calle privada construida sobre suelo público (aunque los costes pueden ser adquiridos por grupos privados).

Las compañías que trabajan entorno al comercio electrónico comenzaron hace más de dos décadas con la introducción del intercambio electrónico de datos (EDI) entre firmas comerciales (envío y recibo de pedidos, información de reparto y pago, etc.) Incluso el comercio electrónico orientado al cliente consumidor tiene también una larga historia: cada vez que utiliza un cajero automático o presenta una tarjeta de crédito, está efectuando una transacción electrónica. EDI y ATM, sin embargo, operan en un sistema cerrado; son un medio de comunicación más conveniente, estrictamente entre las partes involucradas.

Internet y más concretamente todo lo referente a páginas web (WWW), la parte cliente-servidor de Internet, ha abierto una nueva era combinando el carácter abierto de Internet con un interfaz de usuario sencillo. La WWW fue desarrollada en el **Laboratorio de física de partículas CERN** en Ginebra en 1991 (con Mosaic, el predecesor de Netscape y actual google Chrome). Le llevó dos años a Mosaic penetrar en Internet, y otros dos años antes de que las empresas y el público en general se dieran cuenta de su potencial.

Recordemos que Mosaic fue de los primeros navegadores en internet, después vino NetScape, muchos de vosotros habréis navegado con este browser perteneciente a una de las grandes compañías que más aportó al mundo web y que posteriormente fue adquirida por la compañía Google.

En estos navegadores más antiguos había que configurar absolutamente todo, hasta los ficheros de sonido y video en caso de querer escucharlos a través de la página web. Hoy en día todo esto que lo hace el browser de forma automática era una gran problemática hace pocos años.

#### Correo electrónico en la web (MIME)

El mecanismo básico de transporte de correo a través de Internet queda definido por el protocolo SMTP (Simple Mail Transfer Protocol). El RFC 822 describe el formato de los mensajes intercambiados. Según este estándar, cada mensaje consta de dos partes:

* header (cabecera): contiene la información necesaria para que el mensaje llegue a su destino. El formato de la cabecera es definido ambién por el estándar y está formada por un conjunto de pares clave/valor estructurados.
* body (cuerpo o contenido): la información que debe recibir el destinatario. Esta información estará siempre en formato de texto de siete bits (US-ASCII).

**MIME** es un protocolo de intercambio de objetos a través de Internet. Se creó inicialmente para enriquecer los intercambios de correo electrónico, limitados según el RFC 822 a ASCII de siete bits, habiéndose extendido a otros muchos protocolos. El formato MIME permite el envío de texto enriquecido (8 bits), gráficos, ficheros de audio, vídeo.

En **MIME**, cada objeto de mensajería se encapsula en una especie de envoltorio que especifica tanto su semántica como el medio de codificación utilizado. La caracterización semántica hecha en la cabecera permite asociar los datos con su mecanismo de transporte (codificación) y con su significado, de forma que el remitente y el destinatario utilicen coordinadamente los datos intercambiados.

#### Privacy Enhanced Mail (PEM)

PEM es un sistema similar a MIME y desarrollado en paralelo con éste para crear objetos de mensajería seguros. Incluye cifrado, autenticación y gestión de claves y permite el uso de algoritmos simétricos y de clave pública. Con el desarrollo de MIME, **PEM** es, de alguna forma, ciclico, por lo que se ha visto desplazado por S/MIME. De hecho, lleva algunos años en estado de borrador.

#### MIME Object Security Objects (MOSS)

MOSS (también conocido como PEM-MIME) es una extensión de MIME derivada de PEM que aporta exclusivamente lo que le falta a MIME para obtener las garantías deseadas: claves, firmas digitales, certificados, etc. De este modo se consigue un intercambio seguro de objetos. De acuerdo con la propia naturaleza de MIME, es posible aplicar diferentes servicios de seguridad a cada parte del mensaje.

Este estándar ha sufrido críticas debido a sus requerimientos, que son muy flexibles. Puede ocurrir que dos agentes de correo diferentes, ambos soportando MOSS, sean incompatibles. Esta flexibilidad es, en parte, una reacción a la rigidez de PEM, la cual le hacía impopular entre los usuarios.

#### Secure-Multipurpose Internet Mail Extensions (S/MIME)

S/MIME es un protocolo que añade firmas digitales y cifrado a los mensajes MIME. MIME, en sí mismo, no proporciona ningún servicio de seguridad. S/MIME define esos servicios, utilizando criptografía de clave pública, siguiendo la sintaxis definida en el [PKCS#7](file:///D:\Archivos%20de%20programa\WebCopier\Download\Critpo_1\16.html#7).

S/MIME ha sido adoptado recientemente por un gran número de compañías como FTP Software, Qualcomm, Microsoft, Lotus, VeriSign, Netscape o Novell.

#### Secure HTTP (S-HTTP)

Secure HTTP es un protocolo propuesto por Enterprise Integration Technologies (EII), patrocinado por el consorcio CommerceNet y desarrollado en la actualidad por Terisa Systems. Constituye una extensión del protocolo HTTP, incorporando cabeceras MIME para aportar confidencialidad, autenticación, integridad e irrenunciabilidad de las transacciones.

Utiliza un sistema inspirado en PEM, añadiendo suficientes cabeceras a cada transacción para lograr cada uno de los objetivos propuestos. Las transacciones HTTP constan simplemente de una petición de parte del cliente que produce una respuesta del servidor. **S-HTTP** especifica que el cliente envíe directamente toda la información pertinente: claves, certificados, códigos de integridad, etc.

El servidor responde siguiendo la misma filosofía PEM. El protocolo soporta varios mecanismos criptográficos y negocia los modos y opciones de estos mecanismos.

Las principales ventajas de **S-HTTP** son su flexibilidad y su integración dentro de HTML (extensiones al lenguaje similares a las introducidas periódicamente por Google en sus navegadores). .

#### Secure Sockets Layer (SSL)

Secure Sockets Layer (**SSL**) es un protocolo diseñado por Netscape Communications Co., que dispone un nivel seguro de transporte entre el servicio clásico de transporte en Internet (TCP) y las aplicaciones que se comunican a través de él. Proporciona conexiones seguras sobre una red insegura como es Internet, asegurando las siguientes características:

1. Conexión privada: la información se cifra utilizando criptografía de clave simétrica.
2. Autenticación: usando criptografía de clave pública.
3. integridad: la integridad de los mensajes se asegura usando firmas digitales.

**SSL** se compone de dos partes diferenciadas:

1. Handshake Protocol: se encarga de establecer la conexión y determinar los parámetros que se van a utilizar posteriormente (fundamentalmente se trata de establecer cual va a ser la clave simétrica que se utilizará para transmitir los datos durante esa conexión).
2. Record Protocol: comprime, cifra, descifra y verifica la información que se transmite.

Este sistema es transparente para las aplicaciones finales, que simplemente saben que el canal se encarga de proporcionarles confidencialidad entre extremos. Por tanto, podemos situar protocolos como HTTP, FTP, NNTP o Telnet.

**Conceptos fundamentales de internet. Protocolos TCP/IP. Dominios**

Una red es un sistema de ordenadores y otros dispositivos conectados por cables entre sí. La red más simple posible la forman dos ordenadores conectados mediante un cable. A partir de aquí su complejidad puede aumentar hasta conectar miles de ordenadores en todo el mundo. El ejemplo más conocido de este último caso es Internet. Las redes, en general, consisten en «compartir recursos», y uno de sus objetivos es hacer que todos los programas, datos y equipos estén disponibles para cualquiera de la red que así lo solicite.

Equipos

Router

INTERNET

Una red conectada en un área restringida se conoce como Red de área local (LAN). Podría ser una red dentro de nuestra propia oficina o en el aula donde damos clase etc…

Una LAN está contenida a menudo en una sola ubicación. En una LAN, los recursos o computadoras intercambian información entre sí, permitiendo compartirla. Lo compartido puede ser la información contenida en el disco, una impresora o un módem.

Una Red de área extensa (WAN) es un grupo de dispositivos, o varias LAN, conectados en una área geográficamente mayor, normalmentepor medio de líneas telefónicas u otro formato de cableado como puede ser una línea de alta velocidad, fibra o enlace vía satélite. Una de los mayores ejemplos de WAN es la propia Internet.

En una red se puede compartir la información y los recursos. Gracias a esta facilidad contamos con una serie de ventajas para nuestro trabajo en los centros:

Podemos compartir los periféricos, como pueden ser las impresoras, plotters, etc…En una red, todos los ordenadores pueden acceder a la misma impresora.

Podemos transferir datos entre los usuarios sin utilizar dispositivos de almacenamiento externo tipo pen drive o discos duros portátiles,

. La transferencia de archivos a través de la red elimina el tiempo que se pierde copiando archivos en disquete y luego en otro PC (por ejemplo, la actualización de nuestro antivirus u otro tipo de programas que requieran actualizaciones vía red). Además, hay menos restricciones en el tamaño del archivo que se transfiere a través de la red.

Se puede crear una copia de seguridad del archivo de forma automática y programada en el tiempo.. Se puede usar un programa para hacer copias de seguridad de archivos automáticamente, con lo que se ahorra tiempo y se garantiza que todo el trabajo ha quedado guardado.

Se puede enviar y recibir correos electrónicos a y desde cualquier punto del planeta, comunicar mensajes y avisos a mucha gente, en un sinfín de diferentes áreas, rápida y económicamente.

Se puede acceder a los vastos recursos de Internet y de la world wide web en general.

Los modelos más comunes son el Cliente-Servidor. En las oficinas, universidades, centros de estudio y de trabajo en general nos encontramos con el modelo Cliente-Servidor, en donde todas las estaciones de trabajo (equipos conectados a la red) pueden actuar como clientes conectadas a Ordenadores Centrales que actúan como Servidores.

**Cliente-Servidor**

Este es un modelo de proceso en el que las tareas se reparten entre programas que se ejecutan en el servidor y otros en la estación de trabajo del usuario. En una red cualquier equipo puede ser el servidor o el cliente. El cliente es la entidad que solicita la realización de una tarea, el servidor es quien la realiza en nombre del cliente. Este es el caso de aplicaciones de acceso a bases de datos, en las cuales las estaciones ejecutan las tareas del interfaz de usuario (pantallas de entrada de datos o consultas, listados, etc.) y el servidor realiza las actualizaciones y recuperaciones de datos en la base (altas , bajas y modificaciones lo que normalmente se referencia con la palabra CRUD ).

**Protocolos. HTTP**

Un protocolo podríamos definirlo como uan serie de reglas que permite que una o más entidades intercambien información utilizando la red física (cables y placas de comunicación). Es una manera de establecer una comunicación en base a unas reglas.

Dentro de estas reglas existen formas por las cuales se puede identificar y distinguir cada uno de los recursos y lo que ellos pueden ofrecer a todos los demás recursos de la red.

Un sistema operativo puede soportar varios protocolos, pero solamente los dispositivos que utilizan el mismo protocolo pueden comunicarse entre sí.

Cuando se conecta una computadora a la red (utilizando una tarjeta NIC, módem), la computadora asocia automáticamente un protocolo con dicho dispositivo. El protocolo asociado por defecto con el dispositivo dependerá del sistema operativo instalado en la computadora.

Por ejemplo, Windows 10 instala por defecto el protocolo TCP/IP.

TCP/IP es realmente un conjunto de protocolos entre los que destacan Http para la trasferencia de páginas web, SMYP y POP3 para el correo electrónico, FTP para la trasferencia de archivos etc…

En el protocolo HTTP (Hyper Text Transfer Protocol) por defecto a un servidor web le llega un comando GET. Siempre es asi, el protocolo Http envía comandos al servidor web que es donde están alojadas las páginas web y de esta manera comenzamos la comunicación con el servidor en base a este protocolo. Otros ejemplos de comandos http que podemos enviar al servidor son POST en caso de enviar datos desde un formulario usando el método post o HEAD en caso de que el navegador solicite las cabeceras Http( información acerca de algunos parámetros como por ejemplo la lectura de cookies) al servidor.

**MODELO CONCEPTUAL OSI**

El modelo de referencia OSI ofrece varias funciones a la comunidad que participa del **internet working:**

Proporciona una forma de entender cómo opera un **internet working**

de redes.

Sirve como framework de trabajo para crear e implementar

estándares de red, dispositivos y esquemas de **internet working.**

Ventajas :

* Separa la compleja operación de **internetworking** en elementos más

simples.

* Permite a los ingenieros centrarse en el diseño y desarrollo de

funciones modulares.

* Proporciona la posibilidad de definir interfaces estándar para

compatibilidad “plug-and-play” e integración multifabricante.

El modelo de referencia OSI consta de siete capas.

Las cuatro capas de nivel más bajo definen rutas para que los puestos finales puedan conectarse unos con otros y poder intercambiar datos. Las tres capas superiores definen cómo han de comunicarse las aplicaciones de los ordenadores de trabajo entre ellas y con los usuarios.

Descripción de las capas OSI:

**Capa de aplicación**. Es la capa de nivel superior del modelo.

En esta capa, el usuario o la aplicación dialoga con los protocolos para acceder a la red. Por ejemplo, se accede a un procesador de textos por el servicio de transferencia de archivos de esta capa.

**Capa de presentación.** La capa de presentación proporciona diversas funciones de conversión y codificación que se aplican a los datos de la capa de aplicación. Estas funciones aseguran que los datos enviados desde la capa de aplicación de un sistema podrán ser leídos por la capa de aplicación de otro sistema. Un ejemplo de funciones de codificación serían los formatos de imágenes jpeg y gif que se insertan en el código HTML de las páginas web.

**Capa de sesión**. La capa de sesión es la responsable de establecer, administrar y concluir las sesiones de comunicaciones entre entidades de la capa de presentación. La comunicación en esta capa consiste en peticiones de servicios y respuestas entre aplicaciones ubicadas en diferentes dispositivos. Un ejemplo de este tipo de comunicación podría ser el que se produce entre un servidor y un cliente de base de datos.

**Capa de trasporte**

En este nivel se realiza y se garantiza la calidad de la comunicación, ya que asegura la integridad de los datos. Es aquí donde se realizan las retransmisiones cuando la información que se corrompió o porque alguna trama (del nivel 2) detectó errores de formato y se necesita volver a reenviar el paquete o datagrama.

El nivel de transporte notifica a las capas superiores si se está logrando la calidad requerida. Este nivel utiliza reconocimiento, números de secuencia y control de flujo.

**Capa de red**

Es la encargada de preparar la información codificada en forma binaria en formatos previamente definidos por el protocolo que vamos a usar.

Se aplica en el marco de redes WAN y LAN ya que como se estableció previamente la transmisión de datos , consiste en el envió en forma ordenada de bits de información. Podríamos de hecho concebir a ésta como una cadena de bits que se alinean en una fila (para el caso de transmisiones seriales), cadena que carece de significado hasta el momento en que las señales binarias se agrupan bajo reglas, con el objetivo de permitir su interpretación en el lado receptor de una manera constante.

**Capa de enlace de datos**

En el nivel de enlace de datos se lleva a cabo el direccionamiento físico de la información; es decir, se leerán los encabezados que definen las direcciones de los nodos (para el caso WAN) o de los segmentos (para el caso LAN) por donde viajarán las tramas. Decimos que son direcciones físicas ya que las direcciones lógicas o de la aplicación que pretendemos transmitir serán direccionadas o enviadas a través de un router en el nivel de red

En este nivel de enlace sólo se da tratamiento a las direcciones MAC (Media Access Control) para el caso de LAN y a las direcciones de las tramas síncronas como HDLC (High-Level Data Link Control), SDLC (Synchronous Data Link Control, de IBM), LAP B (Link Access Procedure Balance) .

**Capa física**

Es el primer nivel del modelo OSI y en él se definen y reglamentan todas las características físicas, mecánicas y eléctricas que debe cumplir el sistema para poder operar. Como es el nivel más bajo, es el que se va a encargar de las comunicaciones físicas entre dispositivos y de cuidar su correcta operación. Es bien sabido que la información computarizada es procesada y transmitida en forma digital siendo esta de bits: 1 y 0.

El protocolo TCP/IP tiene que estar a un nivel superior del tipo de red empleado y funcionar de forma transparente en cualquier tipo de red. Y a un nivel inferior de los programas de aplicación (páginas WEB, correo electrónico…) particulares de cada sistema operativo.

Esquema del modelo de referencia:

**Capa de aplicación (HTTP, SMTP, FTP, TELNET...)**

**Capa de transporte (UDP, TCP)**

**Capa de red (IP)**

**Capa de acceso a la red (Ethernet, Token Ring...)**

**Capa física (cable coaxial, par trenzado...)**

El nivel más bajo es la capa física. Aquí nos referimos al medio físico por el cual se transmite la información. Generalmente será un cable aunque no se descarta cualquier otro medio de transmisión como ondas o enlaces vía satélite.

La capa de acceso a la red determina la manera en que las estaciones (ordenadores) envían y reciben la información a través del soporte físico proporcionado por la capa anterior. Es decir, una vez que tenemos un cable, esta capa soluciona cómo se transmite la información por ese cable, cuándo puede una estación transmitir, si tiene que esperar algún turno o transmite sin más, cómo sabe una estación que un mensaje es para ella, etc…

Las dos capas anteriores quedan a un nivel inferior del protocolo TCP/IP, es decir, no forman parte de este protocolo. La capa de red define la forma en que un mensaje se transmite a través de distintos tipos de redes hasta llegar a su destino. El principal protocolo de esta capa es el IP .

Esta capa proporciona el direccionamiento IP y determina la ruta óptima a través de los routers que debe seguir un paquete desde el origen al destino.

La capa de transporte (protocolos TCP y UDP) ya no se preocupa de la ruta que siguen los mensajes hasta llegar a su destino.

La capa de aplicación nos proporciona los distintos servicios de Internet: correo electrónico, páginas Web, FTP, TELNET…

### Direcciones IP

Todas las máquinas conectadas dado a Internet tienen una dirección IP asignada, la cual debe ser distinta a todas las demás direcciones que estén vigentes en ese momento en la Red. Una dirección IP identifica de manera unívoca a un ordenador en la Red.

Todos los servidores de Internet con conexión permanente tienen asignada una dirección IP fija, la cual no puede ser cedida a ninguna otra máquina. Sin embargo, los ordenadores que se conecten de forma remota no tienen porqué tener siempre la misma dirección. Cada vez que nos conectamos, nuestro proveedor nos asignará una dirección disponible, se conoce como IP dinámica y cada vez que nos conectamos nos asigna una de forma automática.

Las direcciones IP están formadas por 4 bytes (32 bits). Se suelen representar de la forma a.b.c.d donde cada una de estas letras es un número comprendido entre el 0 y el 255.

Sin embargo nosotros no trabajamos directamente con direcciones IP sino que utilizamos un nombre del estilo www.google.com.

Esto es posible gracias a un servidor DNS (Domain Name System) el cual se encarga de traducir los nombres que nosotros escribimos en direcciones IP

Existen unas tablas que asignan dirección IP con nombre de dominio y a esta traducción se le denomina resolver la dirección.

### Protocolo IP

IP es el principal protocolo de la capa de red. Este protocolo define la unidad básica de transferencia de datos entre el origen y el destino, atravesando toda la red de internet. El software IP es el encargado de elegir la ruta más adecuada por la que los datos serán enviados los paquetes de información a través del router

Se trata de un sistema de entrega de paquetes (llamados datagramas) que tiene las siguientes características:

* No orientado a conexión debido a que cada uno de los paquetes puede seguir rutas distintas entre el origen y el destino. Entonces pueden llegar duplicados o desordenados.
* No fiable porque los paquetes pueden perderse, dañarse o llegar retrasados.

### Encaminamiento

Internet es una red de redes que están interconectadas por distintos routers. Cuando enviamos un mensaje desde un ordenador hasta otro, éste mensaje tiene que ser capaz de encontrar la ruta más adecuada para llegar a su destino. Esto es lo que se conoce como encaminamiento. Normalmente no existe un único camino entre el origen y el destino, y los distintos mensajes pueden seguir rutas alternativas. Por ello decíamos que el protocolo IP es no orientado a conexión.

Los routers son los encargados de elegir la ruta más óptima para que un mensaje llegue a su destino. Los routers suelen ser ordenadores con varias tarjetas de red o bien, aparatos dedicados. Su objetivo es reenviar lo que reciben por una de sus tarjetas de red a otra de ellas.

## Puertos

Un ordenador puede estar conectado con distintos servidores a la vez; por ejemplo, con un servidor de noticias y un servidor de correo. Para distinguir las distintas conexiones dentro de un mismo ordenador se utilizan los puertos.

Un puerto es un número de 16 bits, por lo que existen 65536 puertos en cada ordenador. Las aplicaciones utilizan estos puertos para recibir y transmitir mensajes.

Los números de puerto de las aplicaciones cliente son asignados dinámicamente y generalmente son superiores al 1024. Cuando una aplicación cliente quiere comunicarse con un servidor, busca un número de puerto libre y lo utiliza.

En cambio, las aplicaciones servidoras utilizan unos números de puerto prefijados: son los llamados puertos well-known (bien conocidos). Estos puertos están definidos en la RFC 1700 y se pueden consultar en <http://www.ietf.org/rfc/rfc1700.txt>. A continuación se enumeran los puertos well-known más usuales:

Los puertos tienen una memoria intermedia (buffer) situada entre los programas de aplicación y la red. De tal forma que las aplicaciones transmiten la información a los puertos. Aquí se va almacenando hasta que pueda enviarse por la red. Una vez que pueda transmitirse, la información irá llegando al puerto destino donde se irá guardando hasta que la aplicación esté preparada para recibirla.

Los dos protocolos principales de la capa de transporte son UDP y TCP. El primero ofrece una transferencias de mensajes no fiable y no orientada a conexión y el segundo, una transferencia fiable y orientada a conexión.

PROTOCOLO UDP Y TCP

**UDP**: User Datagram Protocol

**TCP** : Transport Control Protocol

**IP** :¿ A qué aplicación va el paquete?

# Diferencias entre TCP y UDP

**TCP**: Protocolo de flujo de bytes , fiable , orientado a conexión y con control de flujo.

Fiable: Se encarga de comprobar que los datos llegan al destino. Si no , los vuelve a retrasmitir.

Orientado a conexión: Antes de enviar se establece conexión y hasta que no está establecida no se envían datos. ( En UDP e IP no sucede asi).

Flujo de Bytes: TCP maneja los datos como un stream . El stream se parte en trozos (paquetes) los envia ( no necesariamente tienen que llegar en el mismo orden) . El software TCP del receptor los recompone y los coloca en orden.

Posibilidad de recuperar los paquetes perdidos ( IP no detecta paquetes perdidos).

Confirmación y control de flujo: TCP manda las confirmaciones al emisor.

Si no hay confirmación no sigue mandando los datos. Así se consigue el control de flujo , que no nos envíen más datos que los que somos capaces de procesar.

Entrega a la aplicación correcta: Cuando llega un paquete mira las cabeceras para ver a que aplicación va el paquete.

**UDP:** Protocolo de trasporte de datagramas , no fiable sin conexión y sin control de flujo. La unidad de trasporte se llama datagrama y son independientes.

No fiable: Envia un paquete IP y si no llega se pierde.

Sin conexión

No control de flujo

Se utiliza cuando queremos hacer una consulta rápida a una base de datos (no de gestión) por ejemplo un servidor de DNS.

UDP es mucho más rápido .

Se usa también en aplicaciones de audio o video. Si algún datagrama se pierde se produce una interferencia.

# Los Números de puertos

Los números de puerto se representan con 16 bits ( 2 bytes).

2 elevado a 16 =65.535 puertos distintos . Nos van a permitir identificar una aplicación dentro de un Host.

Junto con la dirección IP trasferimos el puerto.

Se dividen en 2 grupos:

Well Known Ports ( 0..1023 ) puertos bajos al Servidor.

Anonymous Ports (1024..65535) puertos altos al cliente.

Los números de puerto están estandarizados ( IANA)

21🡪 FTP

80🡪http

25🡪SMTP

23🡪TELNET

110🡪POP3

53🡪DNS

El cliente coge un puerto cualquiera que esté libre. El sistema nos da uno aleatorio.

Tanto para TCP como UDP están estandarizados los mismos puertos.

Donde aparecen publicados los puertos y servicios?

UNIX🡪/etc/services

WINDOWS🡪 /Windows/services

En Unix los primeros 1024 puertos solo puede usarlos el superusuario

No se permite al usuario instalar servicios en los puertos bajos( caballos de Troya).

Ejemplo:

Enviar una petición a un Servidor DNS.

Cliente IP: 120.181.31.4 Servidor IP:80.41.12.3

Puerto: 53 (DNS)

Dir. IP Origen: 120.181.31.4

Dir. IP Destino: 80.41.12.3

Puerto Origen:1300

Puerto Destino:53

Pregunta:www.Microsoft.com

Dir. IP Origen: 120.181.31.4

Dir. IP Destino: 80.41.12.3

Puerto Origen:1300

Puerto Destino:53

Respuesta: 181.12.131.4

# Tipo de Servidores TCP /UDP

Servidores Iterativos:

Típico de UDP(Atiende peticiones sueltas).

Consiste en hacer un bucle que atiende peticiones de clientes . Coge la petición la resuelve y manda la respuesta al cliente. Después atiende la siguiente petición.

Son fáciles de implementar . El inconveniente es que si tardamos mucho en atender al cliente , se va a hacer una cola de espera . Los utilizamos para resolver preguntas sencillas (resolver DNS´s ) .No van a ser muy útiles para resolver problemas del tipo human in the middle ( nos conectamos , la dejamos abierta y nos vamos a hacer otra cosa mientras la conexión permanece abierta).

Servidores concurrentes:

Normal en TCP.

Cuando se conecta el cliente, creamos un nuevo hilo (thread) que se encarga de atender a ese cliente , cuando el cliente se desconecta el hilo morirá.

De esta forma vamos a poder estar atendiendo a muchos clientes a la vez. Cuando el hilo no hace nada se duerme hasta que recibe una orden nueva de este cliente, así hasta que cierra la conexión y muere el hilo.

**El Protocolo TCP**

En TCP las conexiones siempre son full-duplex 🡪 ambos interlocutores pueden leer o escribir al mismo tiempo. La orden primeramente irá al buffer de TCP.

En TCP se envían y reciben bytes.

IN

IN DIVISIÓN EN **OOOOO**

TCP

FRAGMENTOS APLICACION

TCP

**OOOOOOOO**  **OOO** / **OOO** / **OO**

APLICACION OUT

**OOOOOOOO**  **OOOOO**

OUT

TCP se encarga de la confirmación de que los datos hayan llegado .

Para ello TCP utiliza un sistema de **ACK (Acknowledge).**

Una vez hemos recibido datos envío un ACK ( es un paquete adicional que se envía ).

No existe el concepto de NAK 🡪Si llega mal o no llega aviso al cliente (emisor)

Si no llega confirmación de una trama , volvería a enviarla pasado un tiempo ( t. Máximo).

Sistema de parada y espera:

Es bastante lento , hay que esperar a que llegue el ACK.

P1

ACK 1

P2

ACK 2

# Sistema de ventana deslizante:

P1

P2

P3

# Stall (Parada)

# ...

**...**

**...**

ACK 1

ACK 2

P4

ACK3

ACK 4

Podemos enviar varias tramas seguidas sin la necesidad de tener que parar para cada una de ellas en espera de cada uno de sus ACK.

El problema es que podemos saturar al receptor .El receptor (TCP del receptor) irá cogiendo datos y se irá llenando el Buffer. Puede ser que no seamos capaces de procesar rápidamente los datos del Buffer y se saturaría.

Ponemos un límite máximo de datos que podemos enviar al receptor sin que este nos mande confirmación (Ventana deslizante).

Se utiliza como una técnica de control de flujo. Si no le enviamos un ACK le estamos parando el envío de datos.

APLICACION

BUFFER

TCP

# V=4 bytes

## C- a- r- a –c- t- e- r –e- s

## C- a- r -a / c- t- e- r –e- s

C- a- r -a

**ACK** hasta la r ( confirmo hasta la r , técnica de parada de flujo)

a-c-t-e Realmente me han enviado c-t-e ( no queremos que nos envíen mas de 3 bytes)

**ACK** hasta la e

r-e-s

**ACK** confirmamos hasta la s

# Formato de la cabecera TCP

DATOS **APLICACIÓN**

CABECERA TCP | DATOS **TCP**

CABECERA IP | CABECERA TCP | DATOS **IP**

Dirección IP de origen y Destino

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Puerto Oigen** | **Puerto Destino** | **Nº de Secuencia** | **ACK Number** | **HLEN** | **Reservado** | **Bits de Control** | **Tamaño Ventana** | **CheckSun** | **URG Pointer** | **Opciones** |
| 16 bits | 16 bits | 32 bits | 32 bits | 4 bits | 6 bits | 6 bits | 16 bits | 16 bits | 16 bits | Tamaño Variable |

ISN ( Initial Sequential

Number )

Desplazamiento del

primer byte de datos

respecto del inicio

de flujo de TCP por

cada paquete .

Offset

Número de secuencia del emisor/recptor + 1

Número que debe ser comprobado

**Establecimiento y cierre de la conexión TCP.**

**Establecimiento e la conexión**. Algoritmo Three Steps Handshake

El servidor elige un puerto y se lo pide a TCP , para instalarse. El hilo queda dormido esperando a que alguien se conecte🡪**passive open**

Después cuando el cliente se quiere conectar al servidor envia un segmento TCP formado exclusivamente por una cabecera TCP y dentro activa el flag de SYN y en el campo de número de secuencia el cliente pone un número a partir del cual quiere empezar a sincronizar los bytes de la conexión .A ese número se le llama ISN ( initial séquense number ).

No se empieza a contar por 0 .

Si abrimos una conexión , la cerramos y la volvemos a abrir rápidamente , en la segunda conexión pueden llegar segmentos de la primera conexión , para evitar este efecto lateral , se lleva un contador interno que lleva el número de conexión para poder diferenciar a qué conexión pertenece el paquete Es un contador

Cliente :1030 Flag:**SYN** ISN:**1413821** Tamaño Dat:**0**  Servidor : 80

Flag:**SYN,ACK** (activos) ACK number:**Num. De Sec. Cliente +1**

Num. Secuencia:**84345**🡪ISN del Servidor

Por cada byte enviado se incrementa

Flag: **ACK** Num. De sec.:**1413821** ACK number:**84346** Tamaño Dat.:**0**

En este caso no hay datos y el número de secuencia es el mismo.

La conexión ya está establecida reconocida pero la aplicación puede estar haciendo otra cosa🡪cola de listener.

ISN🡪 Lo genera el SW de TCP cada vez que se establece la conexión.

Cola de listener🡪50 clientes como máximo.

En servidores http el número es mayor.

**Cierre de la conexión.** Algoritmo Two Ways Handshake.

Durante el cierre se ve como una doble conexión. Una de A 🡪B y otra de B🡪A

Al cerrar no se distingue entre cliente y servidor .

Cuando cerramos de A🡪B , A ya no puede enviar datos a B pero B si puede enviar datos a A y ahí estamos en estado *half close .* Sí podemos enviar ACK’s.

Cuando se cierra de B🡪A ya no podemos enviar en ninguna dirección.

A *cierra* B

TCP envía los datos que tiene el Buffer y luego envía el flag de fin en el último segmento TCP o bien envía flag de fin en uno independiente.

El SW de TCP de B recibe este segmento e informa a la aplicación de que se va a cerrar la conexión , pero la forma informar es que la Aplicación pide datos a TCP nó al revés pero la Aplicación puede tardar responder .

A B

Flag: **FIN** , **ACK** Num. De sec: **141837** ACK Num.:**85372** Tamaño Dat.:**0**

**Half Close** Flag:**ACK** Num. Sec: **85371** ACK number:**141838** Tamaño Dat.:**0**

## Le envía un segmento de confirmación a A sin que la Aplicación de B se entere.

**Es el mismo número de secuencia porque no se han enviado datos.**

Más tarde la Aplicación de B verá que no hay más datos que leer y la Aplicación e B pedirá a su SW de TCP cerrar la conexión .

El SW de TCP enviará otro segmento con el flag de FIN y datos que quedasen en el buffer de B.

A B

Flag: **FIN** Num. De sec.: **85371** Tamaño Dat.:**100**

Flag:**ACK**  Num. Sec:**141837** Tamaño Dat.:**0** ACK number:**85472**

Conexión TCP: Timeout

* Tiempo que ha de transcurrir para que un cliente tratando de establecer una conexión indique que dicha conexión no puede ser establecida.No se especifica el motivo en la aplicaciones estándares.
* Este timeout varía de unas implementaciones a otras.
* En el caso del UNIX BSD y derivados, este timeout vale 75 segundos en los que se suelen mandar entre 3 y 5 paquetes deestablecimiento.
* Funciona en base a ticks de 500 mseg.

##### TCP: Segmentos de Reset

• Un segmento es de Reset cuando se activa en la cabecera TCP el flag RST.

• Se activa el bit de Reset en una conexión TCP cuando el paquete que ha llegado no parece, en principio, estar relacionado con la conexión a la que está referido el paquete.

• Las causas de generar un paquete con este bit para

una conexión TCP pueden ser varias:

a)Intento de conexión a un puerto no existente

b)Abortar una conexión

c)Respuesta ante conexiones semi-abiertas

La mayoría de los servidores TCP son concurrentes:

* Un servidor recibe una nueva petición de conexión.
* El servidor acepta la conexión e invoca a un nuevo proceso (fork o threads) para gestionar esa conexión.
* Estado de las conexiones TCP: netstat –n
* Demultiplexación en base a dir. IP de destino, puerto de destino, dir. IP de origen y puertode origen:

Sólo el proceso en estado LISTEN recibirá los segmentos SYN.

Sólo los procesos en ESTABLISHED reciben datos (nunca el del estado LISTEN).

* Cola de entrada de peticiones de conexión:

Todo punto final de conexión (socket) en escucha tiene una cola de longitud fija de conexiones que han sido aceptadas por TCP y no por la aplicación.

La aplicación especifica un límite (backlog) a esta cola (entre 0 y 5).

Cuando llega un segmento SYN pidiendo una nueva conexión, TCP aplica un

algoritmo para decidir en función del backlog y del nº de conexiones encoladas si acepta o no la nueva.

Si se puede aceptar la nueva conexión \_ TCP lo hace.

* La aplicación del servidor no “ve” la nueva conexión hasta que no se recibe el tercer segmento del establecimiento de la conexión.
* Si el cliente comenzara a enviar datos antes de que esta notificación se produjera, estos datos se encolarían en un buffer de entrada.

Si no hay sitio en la cola para la nueva conexión \_ Se rechaza sin enviar nada de

vuelta al cliente. En el cliente se producirá un timeout.

* **Puerto fuente** (16 bits). Puerto de la máquina origen. Al igual que el puerto destino es necesario para identificar la conexión actual.
* **Puerto destino** (16 bits). Puerto de la máquina destino.
* **Número de secuencia** (32 bits). Indica el número de secuencia del primer byte que trasporta el segmento.
* **Número de acuse de recibo** (32 bits). Indica el número de secuencia del siguiente byte que se espera recibir. Con este campo se indica al otro extremo de la conexión que los bytes anteriores se han recibido correctamente.
* **HLEN** (4 bits). Longitud de la cabecera medida en múltiplos de 32 bits (4 bytes). El valor mínimo de este campo es 5, que corresponde a un segmento sin datos (20 bytes).
* **Reservado** (6 bits). Bits reservados para un posible uso futuro.
* **Bits de código** o indicadores (6 bits). Los bits de código determinan el propósito y contenido del segmento. A continuación se explica el significado de cada uno de estos bits (mostrados de izquierda a derecha) si está a 1.
* **URG.** El campo Puntero de urgencia contiene información válida.
* **ACK.** El campo Número de acuse de recibo contiene información válida, es decir, el segmento actual lleva un ACK. Observemos que un mismo segmento puede transportar los datos de un sentido y las confirmaciones del otro sentido de la comunicación.
* **PSH.** La aplicación ha solicitado una operación push (enviar los datos existentes en la memoria temporal sin esperar a completar el segmento).
* **RST.** Interrupción de la conexión actual.
* **SYN.** Sincronización de los números de secuencia. Se utiliza al crear una conexión para indicar al otro extremo cual va a ser el primer número de secuencia con el que va a comenzar a transmitir (veremos que no tiene porqué ser el cero).
* **FIN.** Indica al otro extremo que la aplicación ya no tiene más datos para enviar. Se utiliza para solicitar el cierre de la conexión actual.
* **Ventana** (16 bits). Número de bytes que el emisor del segmento está dispuesto a aceptar por parte del destino.
* **Suma de verificación** (24 bits). Suma de comprobación de errores del segmento actual. Para su cálculo se utiliza una pseudo-cabecera que también incluye las direcciones IP origen y destino.
* **Puntero de urgencia** (8 bits). Se utiliza cuando se están enviando datos urgentes que tienen preferencia sobre todos los demás e indica el siguiente byte del campo Datos que sigue a los datos urgentes. Esto le permite al destino identificar donde terminan los datos urgentes. Nótese que un mismo segmento puede contener tanto datos urgentes (al principio) como normales (después de los urgentes).
* **Opciones** (variable). Si está presente únicamente se define una opción: el tamaño máximo de segmento que será aceptado.
* **Relleno.** Se utiliza para que la longitud de la cabecera sea múltiplo de 32 bits.
* **Datos.** Información que envía la aplicación.

# Nombres de dominio

Generalmente nosotros no trabajamos con direcciones IP sino con nombres de dominio del estilo de www.maizin.com o news.microsoft.com. Para que esto pueda ser posible es necesario un proceso previo de conversión de nombres de dominio a direcciones IP, ya que el protocolo IP requiere direcciones IP al enviar sus datagramas. Este proceso se conoce como resolución de nombres.

## Métodos para resolver nombres de dominio

|  |  |
| --- | --- |
| **Método de resolución** | **Descripción** |
| **1. Local host name** | Nombre de host configurado para la máquina (Entorno de Red, TCP/IP, configuración DNS) |
| **2. Fichero HOSTS** | Fichero de texto situado en el directorio de Windows que contiene una traducción de nombres de dominio en direcciones IP. |
| **3. Servidor DNS** | Servidor que mantiene una base de datos de direcciones IP y nombres de dominio |
| **4. Servidor de nombres NetBIOS** | Servidor que mantiene una base de datos de direcciones IP y nombres NetBIOS. Los nombres NetBIOS son los que vemos desde Entorno de Red y no tienen porqué coincidir con los nombres de dominio |
| **5. Local Broadcast** | Broadcasting a la subred local para la resolución del nombre NetBIOS |
| **6. Fichero LMHOSTS** | Fichero de texto situado en el directorio de Windows que contiene una traducción de nombres NetBIOS en direcciones IP |

Cada vez que escribimos un nombre de dominio en una utilidad TCP/IP, por ejemplo:

C:\>**ping www.google.com**

se van utilizando cada uno de los métodos descritos desde el primero al último hasta que se consiga resolver el nombre. Si después de los 6 métodos no se ha encontrado ninguna coincidencia, se producirá un error.

El fichero HOSTS proporciona un ejemplo muy sencillo de resolución de nombres:

127.0.0.1 localhost  
192.168.0.69 servidor  
129.168.0.1 router

## DNS

En los orígenes de Internet, cuando sólo había unos cientos de ordenadores conectados, la tabla con los nombres de dominio y direcciones IP se encontraba almacenada en un único ordenador con el nombre de HOSTS.TXT. El resto de ordenadores debían consultarle a éste cada vez que tenían que resolver un nombre. Este fichero contenía una estructura plana de nombres, tal como hemos visto en el ejemplo anterior y funcionaba bien ya que la lista sólo se actualizaba una o dos veces por semana.

Sin embargo, a medida que se fueron conectando más ordenadores a la red comenzaron los problemas: el fichero HOSTS.TXT comenzó a ser demasiado extenso, el mantenimiento se hizo difícil ya que requería más de una actualización diaria y el tráfico de la red hacia este ordenador llegó a saturarla.

Es por ello que fue necesario diseñar un nuevo sistema de resolución de nombres que distribuyese el trabajo entre distintos servidores. Se ideó un sistema jerárquico de resolución conocido como DNS (Domain Name System, sistema de resolución de nombres).

* **Clientes DNS** (resolvers). Los clientes DNS envían las peticiones de resolución de nombres a un servidor DNS. Las peticiones de nombres son preguntas de la forma: ¿Qué dirección IP le corresponde al nombre nombre.dominio?
* **Servidores DNS** (name servers). Los servidores DNS contestan a las peticiones de los clientes consultando su base de datos. Si no disponen de la dirección solicitada pueden reenviar la petición a otro servidor.
* **Espacio de nombres de dominio** (domain name space). Se trata de una base de datos distribuida entre distintos servidores.

### Espacio de nombres de dominio

El espacio de nombres de dominio es una estructura jerárquica con forma de árbol que clasifica los distintos dominios en niveles.

Y como ya comentamos el servicio DNS traduce esos nombres de dominio por sus direcciones IP correspondientes.

El punto más alto de la jerarquía es el dominio raíz. Los dominios de primer nivel (es, edu, com...) parten del dominio raíz y los dominios de segundo nivel (upm, ucm, microsoft...), de un dominio de primer nivel; y así sucesivamente. Cada uno de los dominios puede contener tanto hosts como más subdominios.

Un **nombre de dominio** es una secuencia de nombres separados por el carácter delimitador punto. Por ejemplo, www.ce.ucm.es. Esta máquina pertenece al dominio ce (Centro de estudios) que a su vez pertenece al dominio ucm (Universidad Complutense de Madrid) y éste a su vez, al dominio es (España).

Los dominios Top Level Domains, los del nivel 1 , se han clasificado tanto en función de su estructura organizativa como geográficamente.

Ejemplos:

Por estructura organización:

|  |  |
| --- | --- |
| **Nombre de dominio** | **Significado** |
| com | organizaciones comerciales |
| net | redes |
| org | otras organizaciones |
| edu | instituciones educativas y universidades |
| gov | organizaciones gubernamentales |
| mil | organizaciones militares |

Por PAISES:

|  |  |
| --- | --- |
| **Nombre de dominio** | **Significado** |
| es | España |
| tw | Taiwán |
| fr | Francia |
| tv | Tuvalu |

### Tipos de servidores DNS

Dependiendo de la configuración del servidor, éste puede desempeñar distintos papeles:

* **Servidores primarios** (primary name servers). Estos servidores almacenan la información de su zona en una base de datos local. Son los responsables de mantener la información actualizada y cualquier cambio debe ser notificado a este servidor.
* **Servidores secundarios** (secundary name servers). Son aquellos que obtienen los datos de su zona desde otro servidor que tenga autoridad para esa zona. El proceso de copia de la información se denomina transferencia de zona.
* **Servidores maestros** (master name servers). Los servidores maestros son los que transfieren las zonas a los servidores secundarios. Cuando un servidor secundario arranca busca un servidor maestro y realiza la transferencia de zona. Un servidor maestro para una zona puede ser a la vez un servidor primario o secundario de esa zona. Estos servidores extraen la información desde el servidor primario de la zona. Así se evita que los servidores secundarios sobrecargen al servidor primario con transferencias de zonas.
* **Servidores locales** (caching-only servers). Los servidores locales no tienen autoridad sobre ningún dominio: se limitan a contactar con otros servidores para resolver las peticiones de los clientes DNS. Estos servidores mantienen una memoria caché con las últimas preguntas contestadas. Cada vez que un cliente DNS le formula una pregunta, primero consulta en su memoria caché. Si encuentra la dirección IP solicitada, se la devuelve al cliente; si no, consulta a otros servidores, apunta la respuesta en su memoria caché y le comunica la respuesta al cliente.

**Lenguajes de construcción**

2.1 Introducción al HTML

¿Qué es HTML?

**HTML** es la abreviatura de HyperText Markup Language, y es el lenguaje que todos los programas navegadores usan para presentar información en la World Wide Web (WWW).

Su origen se sitúa en la comunidad científica, era una forma de estar comunicados entre los científicos , después pasó al ámbito estatal , posteriormente Universidades y finalmente empresas (consorcios) privados.

El MIT (Instituto tecnológico de Massachusetts) es una universidad privada localizada en Cambridge, Massachusetts.

Fundado en 1861 en respuesta a la creciente industrialización de los Estados Unidos, utilizó el modelo de universidades politécnicas y participó de forma activa en el desarrollo de este lenguaje.

HTML es un lenguaje muy fácil de entender que se basa en el uso de etiquetas, consistentes en un texto ASCII encerrado dentro de un par de paréntesis angulares(símbolos de mayor y menor <..>). El texto incluido dentro de los paréntesis nos dará una explicación de la utilidad de la etiqueta. Así por ejemplo la etiqueta <**IMG**> nos permitirá definir una imagen dentro de nuestro documento.

Las etiquetas podrán incluir una serie de atributos o propiedades, en su mayoría opcionales, que nos permitirán definir diferentes posibilidades o características de la misma. Y por lo tanto nos debemos acostumbrar a utilizar las comillas dobles a la hora de definir propiedades o atributos dentro de una etiqueta HTML.

<IMG **SRC=”sol.gif”** > en este ejemplo el atributo SRC hace referencia a la imagen que vamos a insertar dentro del documento y lo debemos encerrar entre comillas dobles.

Estos atributos quedarán definidos por su nombre y su valor que toman separados por un signo de igual( atributo=”valor”). En el caso de que el valor que tome el atributo tenga más de una palabra deberá expresarse entre comillas, en caso contrario no será necesario. Aunque sí que es recomendable Así por ejemplo la etiqueta <TABLE border=2> nos permitirá definir una tabla con borde de tamaño 2.

El manejo de estas etiquetas nos permitirá:

* Definir la estructura lógica del documento HTML.
* Aplicar distintos estilos al texto (**negrita**, cursiva, ...).
* La inclusión de hiperenlaces, que nos permitirán acceder a otros documentos relacionados con el actual.
* La inclusión de imágenes y ficheros multimedia (gráficos, vídeo, audio).

¿Qué vamos a necesitar para crear un documento HTML?

* Un procesador de textos para escribir y editar el código HTML. Este podrá ser cualquiera que no formatee el texto, ya que el leguaje HTML está basado en el código ASCCI. Si usamos un procesador como el NotePad de windows, tendremos que guardar el documento como " texto ".
* Un navegador web como el Chrome, Firefox, Explorer (Edge en Windows 10),etc., el cual se encargará del interpretar el código HTML de nuestro documento y mostrárnoslo con todos los elementos que hayamos definido tanto textos como elementos multimedia.

La forma de trabajo para crear nuestro documento es:

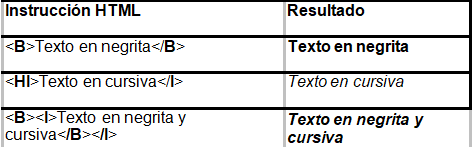
1. Con el procesador de texto crearemos el documento HTML y lo guardaremos con el nombre que deseemos, a excepción de la extensión, que deberá ser necesariamente .htm ó .html.
2. Para visualizar nuestro documento HTML abriremos el documento con el navegador( browser).

Reglas de formato

Todos los navegadores usan unas reglas básicas para poder mostrar una página web con un buen formato:

* **El espacio en blanco es ignorado.** Ya que un documento HTML puede estar en cualquier tipo de fuente y además la ventana del navegador puede ser de cualquier tamaño.
* **Las etiquetas pueden ser escritas en mayúsculas o en minúsculas.** En todo caso se aconseja su escritura en mayúsculas para poder distinguirlas del texto normal.
* **Existe normalmente una etiqueta de inicio y otra de fin.** La etiqueta de fin contendrá el mismo texto que la de inicio añadiéndole al principio una barra inclinada /. La etiqueta afectará por tanto a todo lo que esté incluido entre las etiquetas de inicio y fin. Existen algunas que no necesitan cierre, ya que en estas etiquetas se presupone su final, como por ejemplo; <**P**> párrafo, <**BR**> salto de línea ó <**IMG**> inclusión de una imagen, aunque también podemos optar como veremos más adelante por cerrarla de la siguiente forma: </BR> o </IMG> para pasar de un documento HTML a XHTML que sería para entenderlo ahora al principio un documento HTML pero en formato XML ( Después veremos que es esta tecnología XML y para que se usa)

Algunos formatos de los textos:



M1\_img\_3

2.2 Estructura de un documento HTML

Un documento HTML se divide básicamente en cabecera y cuerpo, siendo el cuerpo donde se va a mostrar absolutamente todo el contenido del documento tanto textos como imágenes, videos, mapas etc…

La estructura básica de un documento HTML es la siguiente:

<HTML> Indica el inicio del documento

<HEAD> Indica el inicio de la cabecera

<TITLE> Inicio del título del documento

</TITLE> Final del título del documento

</HEAD> Final de la cabecera

<BODY> Inicio del cuerpo del documento Instrucciones HTML

</BODY> Final del cuerpo del documento

</HTML> Final del documento

Ninguno de estos elementos es obligatorio, pudiendo crear documentos HTML sin incluir estas etiquetas de identificación. Pero es muy recomendable la construcción de páginas HTML siguiendo esta estructura, para hacer el documento más legible y ordenado a la hora de codificar nosotros u otro programador que tuviese que añadir etiquetas nuevas a nuestra página.

Nuestra primera página HTML

Una vez que conocemos la estructura básica de un documento HTML, vamos a crear una página HTML muy sencilla:

1. Lo primero será abrir el procesador de textos que hayamos elegido y crear el esqueleto del documento:

<HTML>

<HEAD>

<TITLE> Mi primera página HTML </TITLE>

</HEAD>

<BODY>

mi primera página <B>HTML</B>

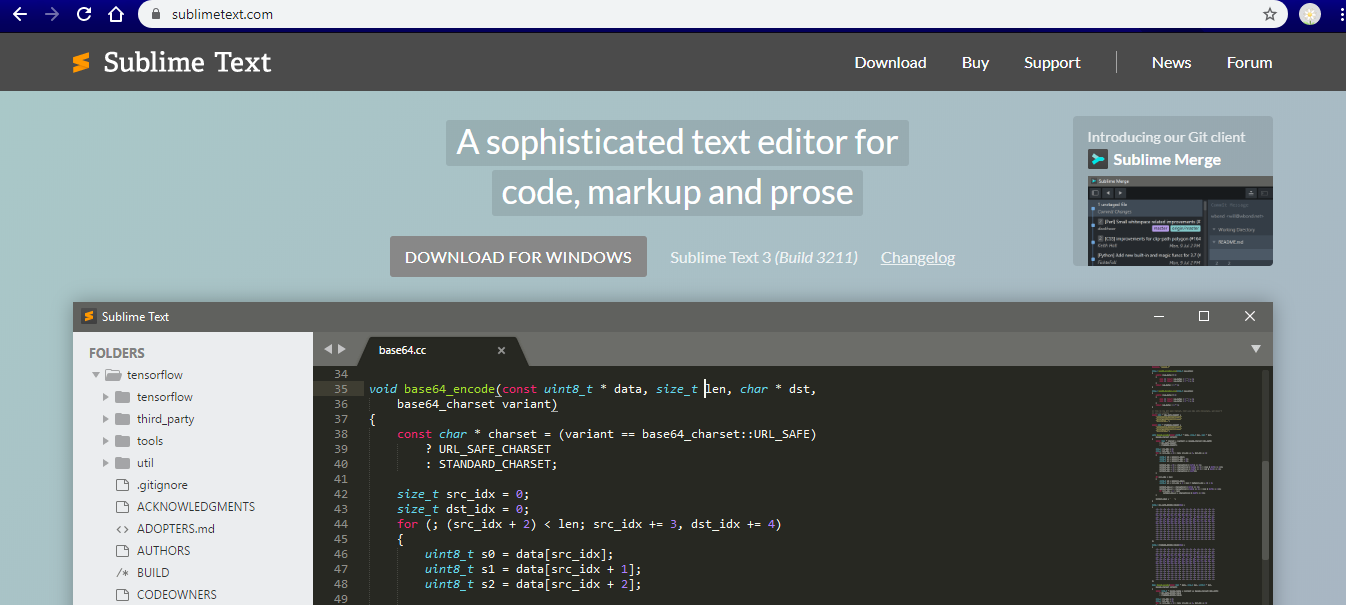
</BODY>

</HTML>

1. Una vez tecleado el código, guardamos el documento como Ejemplo1.htm, y procedemos a abrirlo con el navegador correspondiente. El resultado será:

Como editir de páginas HTML y para no estar utilizando el bloc de notas vamos a usar el editor Sublime Text , que es uno de los más modernos y utilizados en la comunidad HTML.

Lo podemos descargar desde la página oficial sublimetext.com



M1\_img\_4

Le damos al botón de descargar( download for Windows) y procedemos a su descarga:



M1\_img\_5

El proceso de instalación es muy simple, le vamos clickando en los botones de siguiente, hasta finalizar.

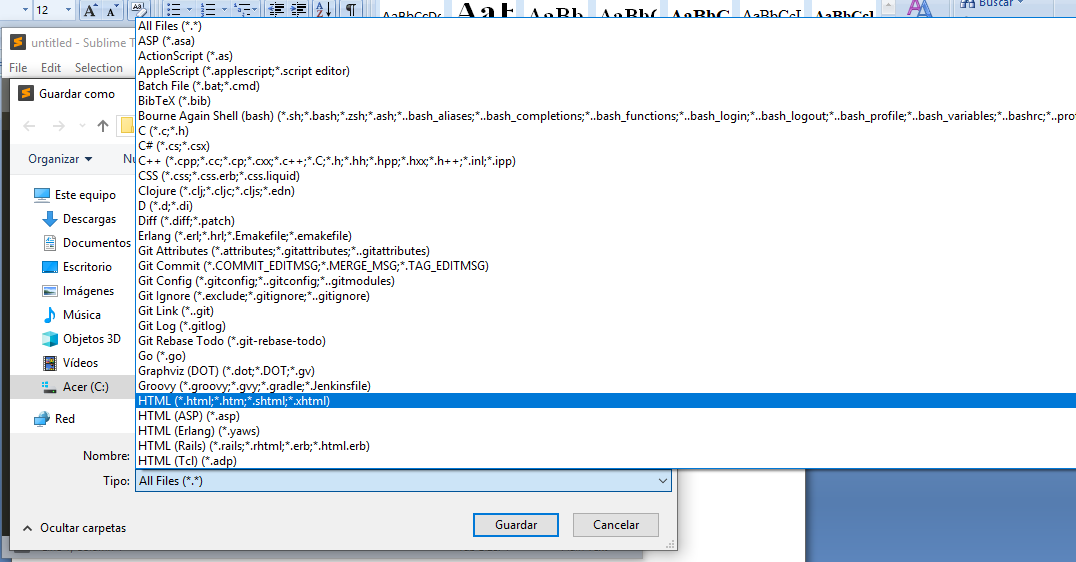
Al arrancarlo tendremos este entorno:



M1\_img\_6

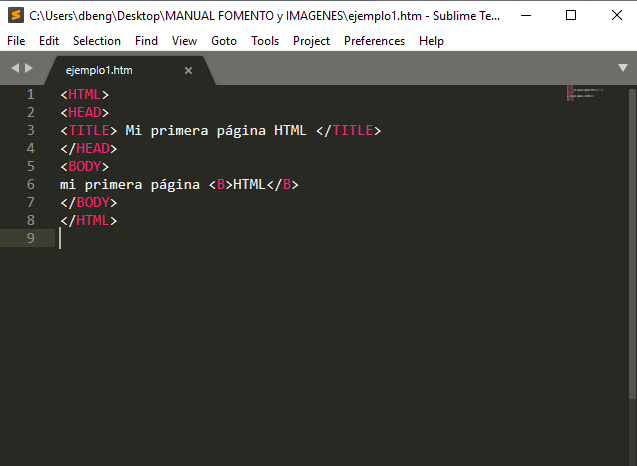
Para crear un nuevo archivo , nos dirigimos al menú File y le damos a New file

Para que tome el formato de HTML lo tendremos que guardar previamente con la extensión htm o html en el menú Save



M1\_img\_7

En nuestro caso crearemos el código anterior(Ejemplo1.htm), y nos quedará de la siguiente forma:



M1\_img\_8

Ya hora abrimos el archivo desde el Google Chrome:



Como podemos observar no nos reconoce los acentos, esto lo solucionaremos un poco más adelante con la etiqueta <META>

2.3 Las URL

La URL(Uniform Resource Locator) nos van a permitir localizar y acceder a cualquier recurso de la red desde nuestro programa navegador. A través de las URL podremos referenciar cualquier fichero que deseemos incluir en nuestro documento HTML. Cada elemento de Internet vendrá definido por su URL correspondiente, independientemente del tipo de servidor en el que se encuentre.

La forma general del URL es:

Tipo de servicio: //máquina.dominio:puerto/camino/archivo

El Tipo de servicio, como su propio nombre indica, hace referencia a alguno de los servicios de Internet. El servicio correspondiente al WWW es http (HiperText Transport Protocol)*,* protocolo para la transmisión de hipertexto. Por lo que cualquier referencia a un documento HTML debería comenzar por http:// .Otros servicios son: ftp, news, telnet o mailto.

La máquina.dominio indica el nombre del servidor en que se encuentra el documento al que estamos haciendo referencia, junto con el dominio al que pertenece dicho servidor.

El puerto es opcional, y lo normal es no ponerlo, indicándose sólo en el caso de que el servidor utilice un puerto distinto al puerto por defecto.

El camino indica la ruta de directorios que hay que seguir para encontrar el documento que estamos referenciando.

Finalmente especificaremos el nombre del *archivo* en que está guardado el documento en cuestión. Si no indicamos un archivo, accederemos al archivo por defecto del directorio al que estemos referenciando.(En el caso del servicio http este archivo por defecto suele ser default.htm ó default.html).



M1\_img\_10

Localizando un documento hipertexto (http):

Sintaxis: http://maquina.dominio:puerto/camino/archivo Ejemplo1: [http://www.google.es](http://www.eidos.es/)

En este ejemplo estamos indicando el servicio al que estamos accediendo (http), la máquina y el dominio, tomándose el resto de parámetros por defecto, de forma que accederemos al directorio raíz del servidor y al documento por defecto de ese directorio, en este caso estamos accediendo a la página de inicio de Google.

Ejemplo2: <http://www.deportes.es/futbol/resultados.htm>

Con esta URL estamos accediendo al fichero resultados.htm que se encuentra en el directorio futbol del servidor www.deportes.es.

En los nuevos navegadores no necesitamos incluir el protocolo (http) ya que con poner la dirección sin él también funciona, incluso muchas páginas sin necesidad tampoco de poner las tres w (www).

Sin embargo como veremos en el lenguaje HTML cuando codificamos una etiqueta , al hacer referencia a una dirección url deberemos incluir el protocolo obligatoriamente.

Servicio FTP (ftp:File Transfer Protocol)

Es un servicio para la trasferencia de archivos de una máquina a otra.

Es usado normalmente para trasferir archivos a un servidor web (proveedor de internet) desde nuestro sistema local ( c:/ ).

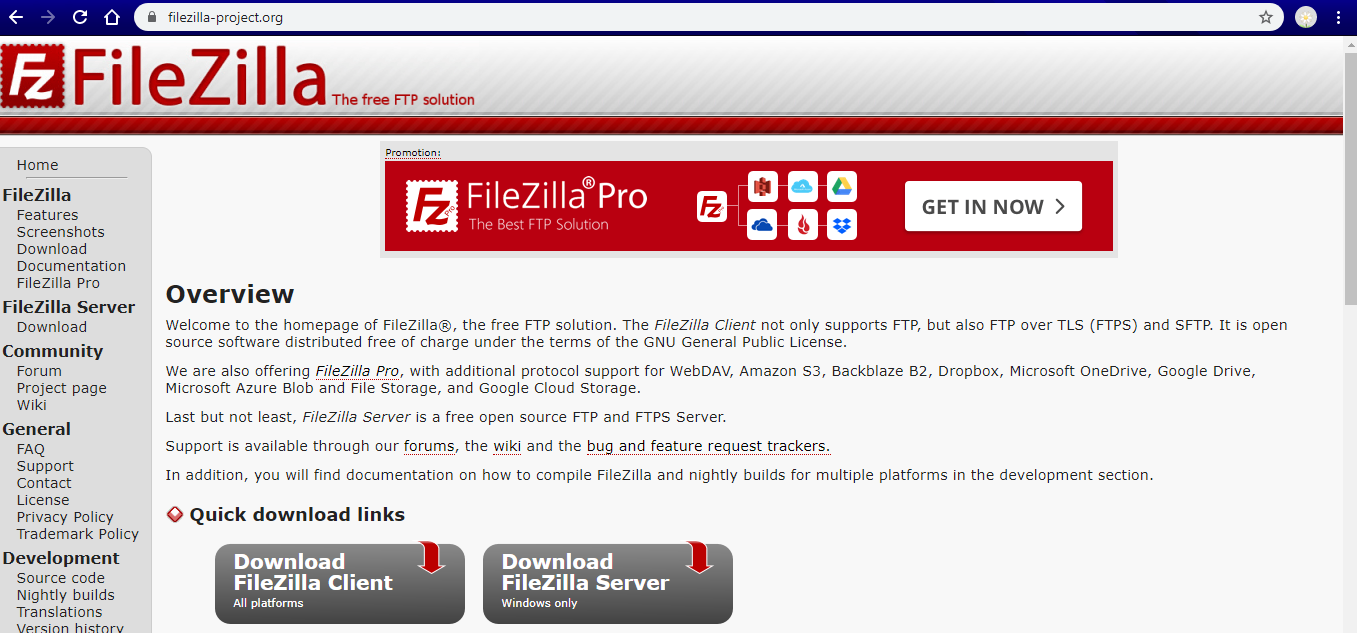
También usado para realizar descarga de archivos . Con lo que se puede realizar la trasferencia en ambos sentidos.

Sintaxis: ftp://maquina.dominio:puerto/camino/archivo Ejemplo: ftp:// ftp.maizin.es/pub/mi\_ftp/

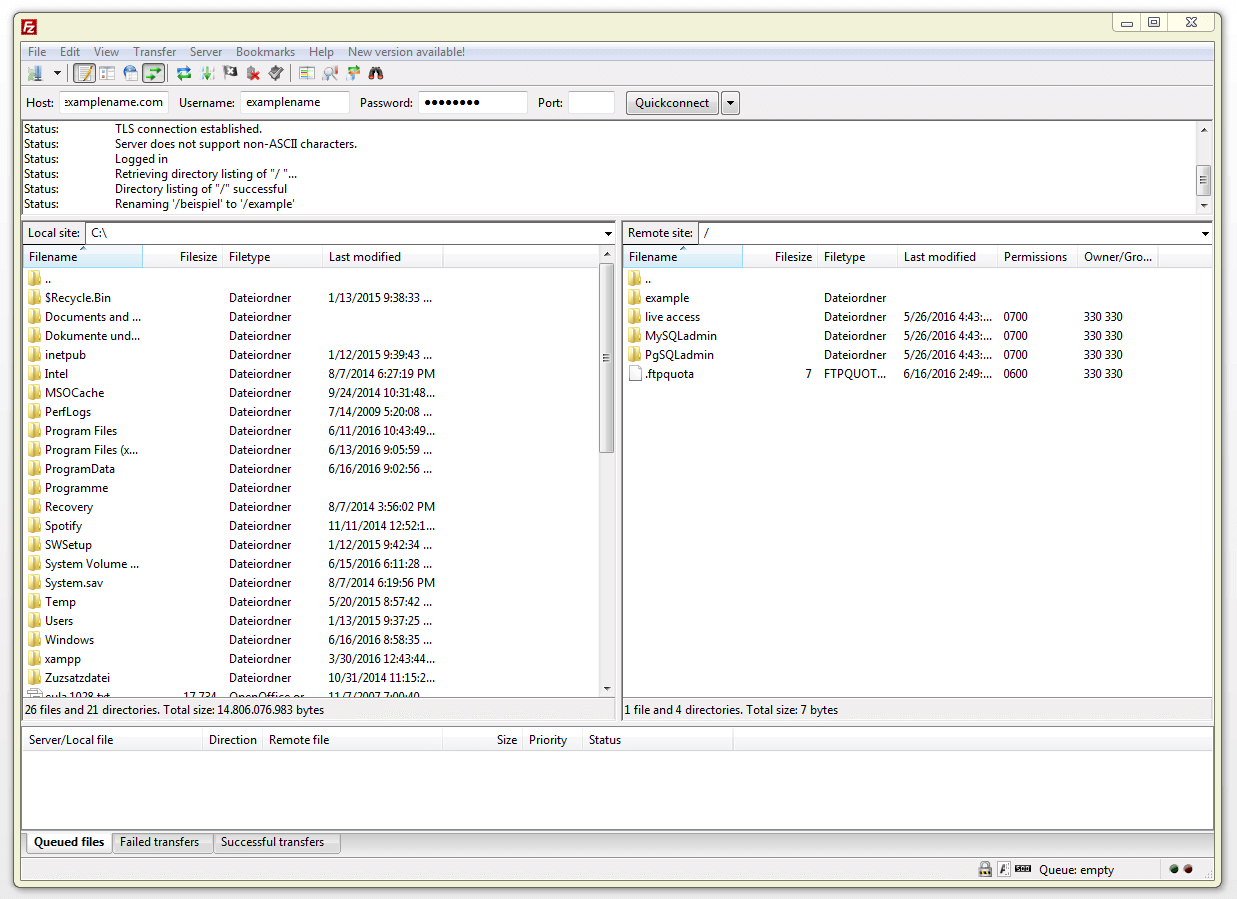
En este caso, estamos accediendo al servicio ftp (protocolo de transferencia de archivos), ftp.maizin.es. En este caso no estamos indicando ningún archivo, con lo que se referenciará un índice con los contenidos del directorio: /pub/mi\_ftp.

El programa Filezilla es uno de los más populares para el uso del servicio FTP

Es gratuito y te lo puedes descargar desde la web oficial de Filezilla



M1\_img\_11



M1\_img\_12

La pantalla aparece dividida en dos partes , una representa nuestro sistema local y la otra el sistema remoto.

Basta con arrastrar con el ratón archivos de un lado a otro para comenzar la trasferencia de archivos.

Si alguna vez desarrollas tu propia web y contratas un servicio de proveedor probablemente tendrás que usar este programa para subir tus páginas html a la world wide web.

Grupo de noticias (news):

Noticias publicadas en directorios en internet

Sintaxis:

news:GRUPO

Ejemplo: news:maizin.es

En este ejemplo estamos accediendo al grupo de noticias del servidor de news maizin.es.

Accediendo a una dirección e-mail (mailto):

Normalmente usado desde formulario de datos , en cuyo caso los daros se enviarán a la dirección o direcciones indicadas en el mailto.

Sintaxis:

mailto: Destinatario\_1, Destinatario\_2,...,Destinatario\_n

Ejemplo:

<mailto:info@maizin.es>

Enviará un mail a la dirección de correo electrónico indicada.

Localización de un archivo cualquiera (file):

File es un protocolo que hace referencia a archivos de nuestra computadora.

Sintaxis: file://maquina.dominio:puerto/camino/archivo Ejemplo: file://c:/windows/

En este caso estamos accediendo a la carpeta Windows de la unidad C: de nuestro ordenador local.

Como podemos observar este protocolo utiliza las 3 barras en vez de 2 como en http.

Es un protocolo muy utilizado en lenguajes de programación sobretodo cuando accedemos a imágenes por ejemplo que queremos cargar desde nuestro disco duro

**2.4 Cabecera y cuerpo del documento.**

Los documentos HTML están basados en etiquetas (apertura y cierre).

Todos los documentos HTML comienzan con la etiqueta de apertura y cierre del

Documento <HTML> :

<HTML>

………..

</HTML>

El símbolo / se utiliza para indicar que la etiqueta es de cierre.

Un documento HTML se divide en cabecera y cuerpo

Todo el contenido que nosotros vamos a visualizar en la página web (documento web o html) está dentro de la sección <BODY> es decir dentro del cuerpo.

Cabecera

Cabecera del documento HTML:

<HEAD>

</HEAD>

En la cabecera de un documento HTML incluiremos las definiciones generales que afecten a todo el documento, como por ejemplo el título de la página que aparecerá en la parte superior de la ventana del navegador.

Las etiquetas <HEAD> y </HEAD> son las que van a delimitar la cabecera de nuestro documento. Todas la etiquetas de la cabecera son opcionales, pero se recomienda incluir la etiqueta <TITLE> </TITLE> que representa el título de la página que se visualizará en la barra superior del navegador.

A continuación vamos a tratar cada uno de los posibles componentes de la cabecera por separado.

Título del documento

**<**TITLE**>**

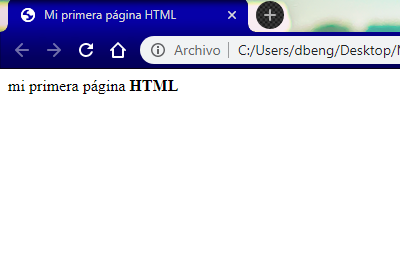
Título del documento

**<**/TITLE**>**

El título de nuestro documento viene especificado por las etiquetas: <TITLE> y </TITLE> y como ya hemos dicho, este título es el que aparecerá en la ventana de nuestro programa

Lo recomendable es que el título guarde relación con el contenido de nuestro documento, ya que por ejemplo es utilizado por algunos servidores de búsqueda para poder intuir el contenido de nuestro documento, sobretodo actualmente en el sistema de posicionamiento del buscador Google.

<TITLE> Mi primera página HTML </TITLE>



M1\_img\_13

Etiquetas META

Las etiquetas <META> contienen lo que llamamos la meta información de la página.

Es información que no se ve en el documento pero que realiza alguna función dentro de la propia página tal como por ejemplo poder escribir con acentos y con caracteres especiales.

Han de ir dentro de la sección <HEAD> de nuestro documento , aunque no es estrictamente obligatorio ya que en algunos casos nos puede ser útil ponerlas fuera.

<META charset="UTF-8">

Nos permite poder escribir en base al sistema de codificación UTF-8 que usa 3 bytes para representar un carácter con lo que lograremos poder escribir con acentos y ñ dentro de nuestro documento HTML.

<META http-equiv="refresh" content="número\_segundos;url=URL de refresco">

Esta etiqueta sirve para indicar un documento que deberá sustituir al actual transcurrido un número determinado de segundos. Una posible utilidad de esta etiqueta podría ser para visualizar documentos de forma secuencial, de forma que se empieza por un documento y se pasa al siguiente transcurrido un período de tiempo sin necesidad de que la persona que lee el documento realice ninguna acción.

<META http-equiv="refresh" [content="5;url=http://www.google.es">](http://www.eidos.es/)

Si insertamos esta instrucción en la cabecera de nuestro página HTML, transcurridos 5 segundos ésta se refrescará con la página web de inicio del buscador Google.

**Cuerpo**

Cuerpo del documento HTML:

<BODY>

</BODY>

Todo lo que va dentro de esta sección son etiquetas que al contrario de la sección <HEAD> sí que muestran contenido dentro de la página ya sea texto,imágenes, mapas, etc…

En el cuerpo de un documento HTML es donde incluiremos las distintas instrucciones del lenguaje junto con el contenido en sí de nuestra página web.

Las etiquetas <BODY> y </BODY> son las que van a delimitar el cuerpo de nuestro documento. Esta posee una serie de propiedades que nos van a permitir variar las características del documento en su conjunto, como por ejemplo el color del texto o del color de fondo de nuestra página web.

Vamos a ver como se especifican los colores en un documento HTML.

Estos se pueden especificar por su nombre (name) o por su código de color, que es un número compuesto de tres pares de cifras hexadecimales que nos indican la proporción de los colores básicos (rojo, verde y azul) que forman el color deseado (#rrggbb), este sistema también es conocido como RGB de red, green, blue, que son los tres colores con los que combinándolos entre ellos se pueden obtener toda la gama o espectro de los mismos.

En Hexadecimal por cada componente de color tenemos valores que van desde el 0 al 9 y dese la A hasta la F.

Por ejemplo este sería un color en hexadecimal

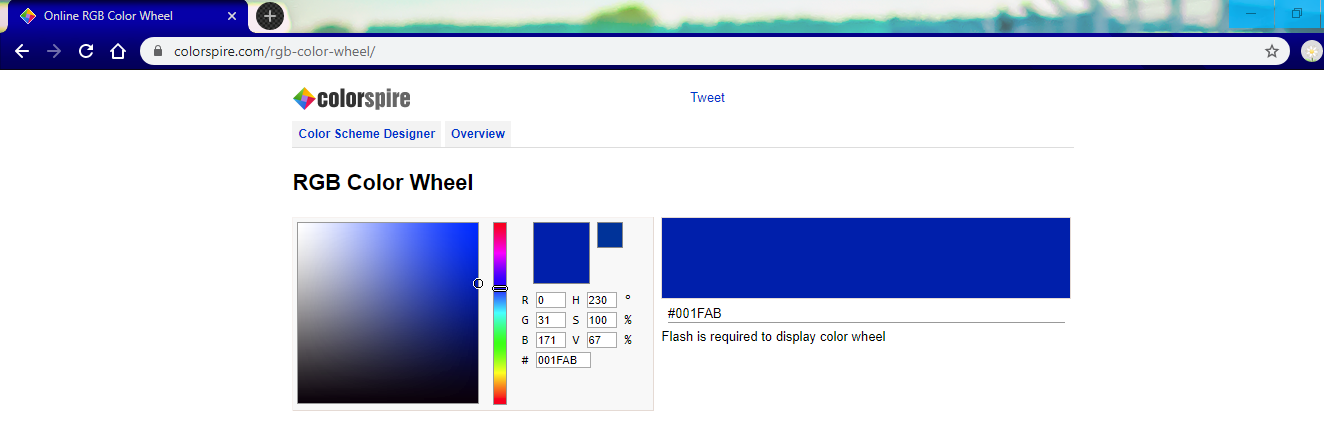
001FAB,componente Rojo 00, componente Green 1F y componente Blue AB

Con lo que obtendríamos este color:



M1\_img\_14

En esta página puedes encontrar un denerador de colores para tu página HTML

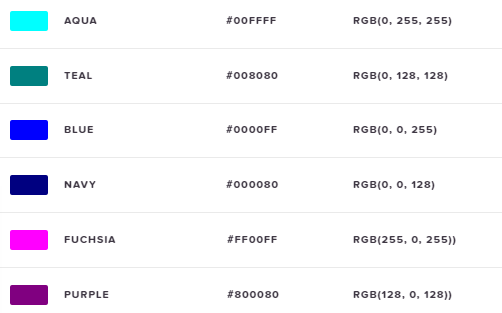


M1\_img\_15

Tabla de colores HTML



M1\_img\_16



M1\_img\_17

La etiqueta <BODY> contiene una serie de atributos que van a afectar a todo el documento en su conjunto. Estos atributos nos van a permitir definir los colores del texto, del fondo, y de los hiperenlaces del documento. Incluso nos permitirán insertar una imagen de fondo en nuestra página.

<BODY background="URL" bgcolor="#rrggbb ó name" text="name" link="name" vlink="name" >

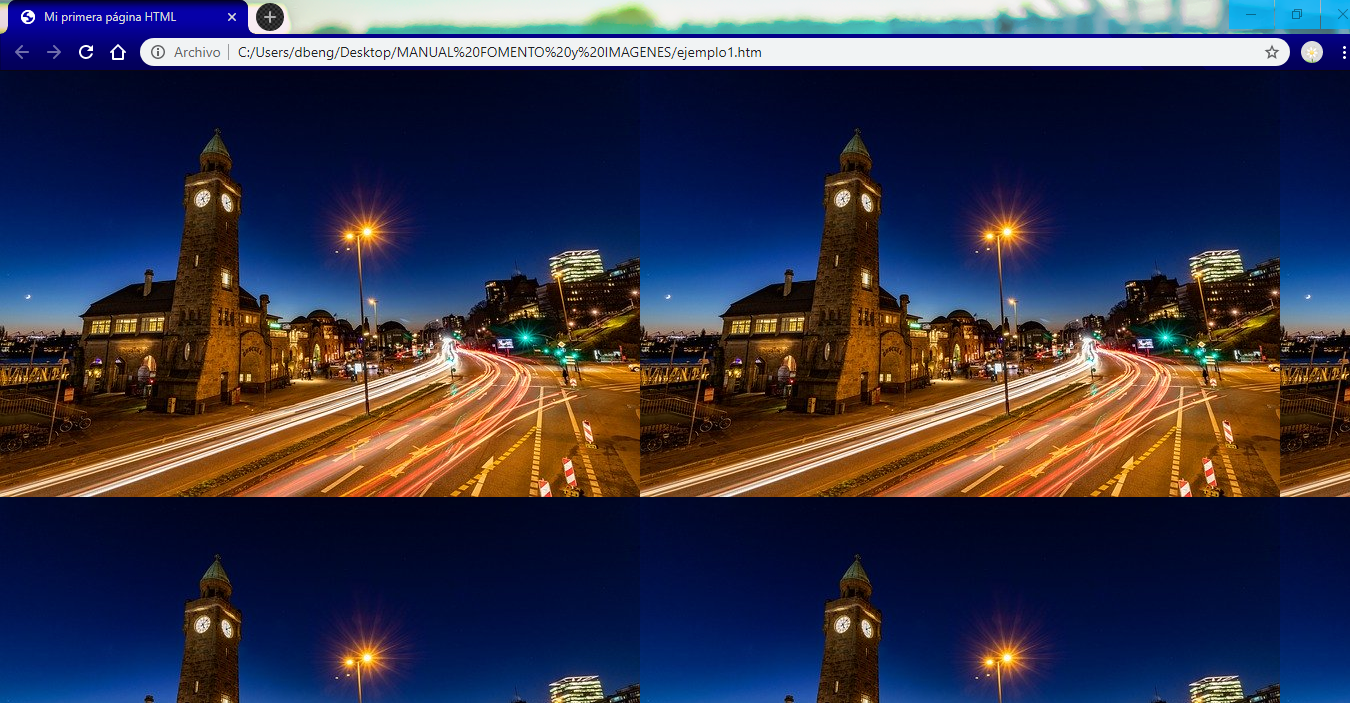
* background="URL". Nos va a permitir mostrar una imagen como fondo de nuestro documento HTML. El camino a esta imagen vendrá especificado por la URL que definamos. Si la imagen no rellena todo el fondo del documento, ésta será reproducida tantas veces como sea necesario hasta completar todo el fondo.Podemos guardar la imagen en el mismo directorio de la página HTML con lo que bastaría con poner solamente el nombre de la imagen.En este caso vamos a insertar la imagen Hamburgo.jpg como fondo de la página

<BODY background="hamburgo.jpg">



M1\_img\_18

El resultado será:

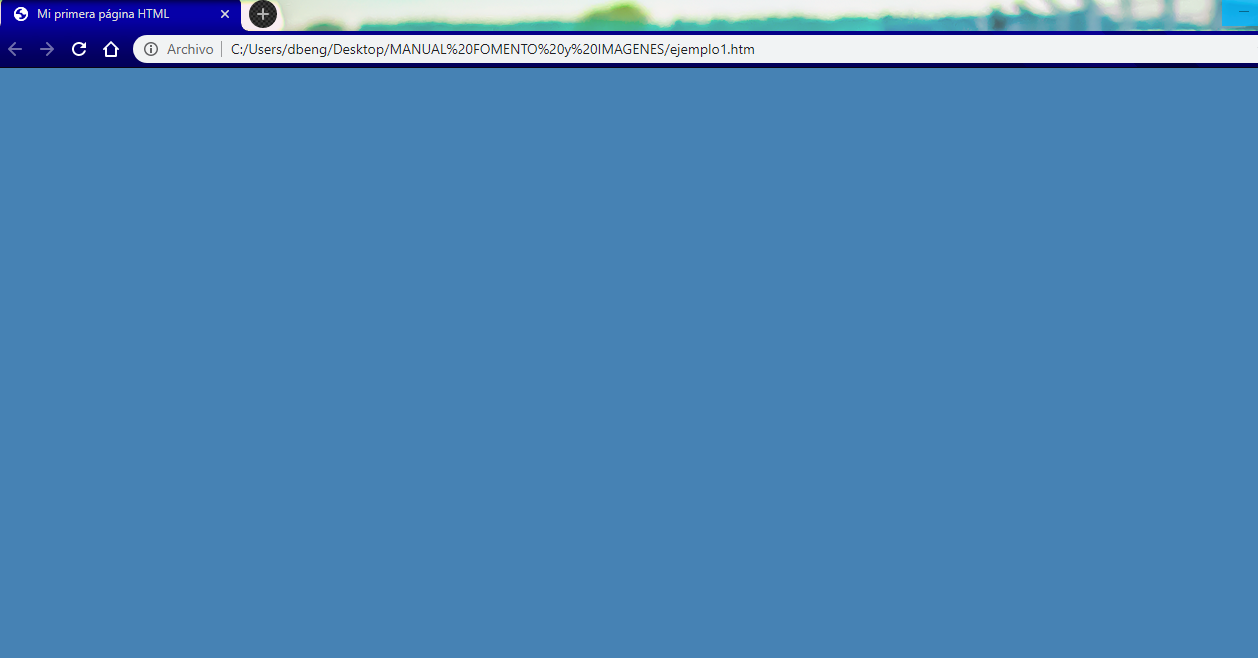


M1\_img1\_19

Como podemos ver la imagen se repite debido a que su tamaño es menor que la resolución de la pantalla. Para que no se repitiese podríamos haber elegido una resolución bastante mayor o al menos igual que la de la pantalla donde se va a visualizar.

* bgcolor=”#rrggbb” ó name *.*Nos va a permitir definir un color para el fondo de nuestro documento. Este atributo será ignorado si previamente hemos utilizado el atributo *background*. En el caso de *name* utiliza una de las constantes de color vistas anteriormente

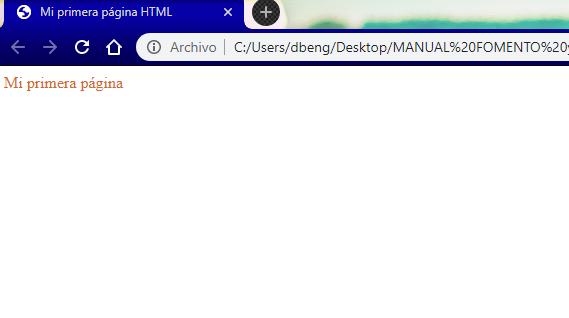
<BODY bgcolor="steelblue">



M1\_img\_20

* text=#rrggbb ó name. Nos permitirá definir un color para el texto de nuestro documento. Por defecto es negro.

<BODY text="chocolate">



M1\_img\_21

<BODY bgcolor="yellow" text="navy">



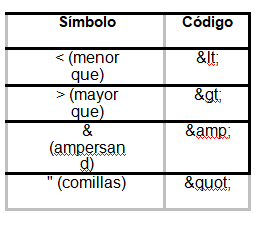
M1\_img\_22

* link=”#rrggbb” ó name. Indica el color que tendrán los hiperenlaces que no han sido accedidos. Por defecto es azul.
* vlink=·”#rrggbb” ó name (*visited links)* Indica el color de los hiperenlaces que ya han sido accedidos. Por defecto es púrpura.
* alink=”#rrggbb” ó name (*actived links)* Indica el color de los hiperenlaces que están siendo pinchados en ese momento.

2.5 Caracteres especiales. Entidades

Hasta ahora hemos escrito algunos ejemplos de código HTML, pero hemos pasado por alto algunas limitaciones de HTML respecto al uso de caracteres especiales. Una de estas limitaciones es, por ejemplo, el uso de los caracteres **<** y **>**, que como ya sabemos indican el inicio y fin de una etiqueta HTML. Pues bien, si quisiéramos representar estos caracteres como parte normal de un texto, el navegador no sabría si se trata de texto normal o del comienzo y final de una etiqueta.

Para ellos existen lo que llamamos entidades, que es la manera de representar algunos caracteres especiales dentro de nuestro documento web.



M1\_img\_23

Ejemplo:

Mañana es Viernes

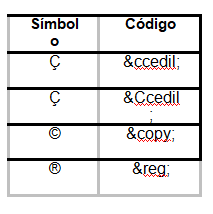
El símbolo < sirve para abrir una etiqueta

La etiqueta &lt;p&gt; abre un p&aacute;rrafo

Si no usásemos las entidades interpretaría <p> como una etiqueta HTML y ni siquiera aparecería en el documento.

Para representar la a acentuada hemos usado también una entidad, aunque en los navegadores modernos no es necesario esto ya que si incluimos la etiqueta <META>correspondiente, los pondrá automáticamente.

Para las letras específicas del idioma castellano: las vocales acentuadas, la ñ, la ü y los signos ¿ y ¡, existen los códigos siguientes:



M1\_img\_24

Para los navegadores actuales, podemos escribir nuestro código HTML de manera directa sin tener en cuenta estos códigos, pero nunca podemos asegurar que las personas que accedan a nuestras páginas con otros navegadores puedan leerlas correctamente.

|  |  |
| --- | --- |
| **Símbolo** | **Código** |
| Ç | &ccedil; |
| Ç | &Ccedil; |
| © | &copy; |
| ® | &reg; |

M1\_img\_25

Ejemplo:

CopyRight &copy;

m1\_img\_26

Para la inclusión de comentarios en nuestra página HTML, podemos usar la etiqueta:

<!-- Esto es un comentario en HTML -->

Estos comentarios nos podrán servir para efectuar anotaciones en nuestro documento HTML que ayuden a una mayor comprensión del código.

Espaciado y saltos de línea

En HTML sólo se admite un único espacio en blanco separando cualquier elemento o texto, el resto de espacios serán ignorados por el navegador, al igual que las tabulaciones, retornos de carro, etc

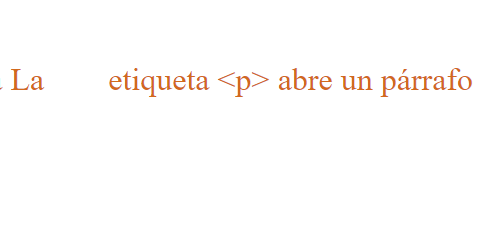
Inclusión de espacios en blanco: &nbsp;

Nos permitirá la inclusión de más de un espacio en blanco entre dos textos consecutivos, de forma que estos espacios se muestren de forma efectiva en el navegador. Tendremos que incluir tantas expresiones &nbsp; como espacios en blanco se deseen incluir.

Ejemplo:

<p>La &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; etiqueta </p

Deja 5 espacios en blanco entre la palabra ‘La’ y ‘etiqueta’



M1\_img\_27

**2.6 Conjunto fundamental de etiquetas**

**Salto de línea <BR>**

Nos permite dar un salto a la línea siguiente en el punto donde la etiqueta sea insertada.

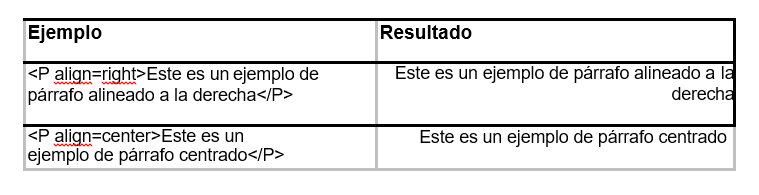
**Párrafo <P>**

Permite definir un párrafo, introduciendo normalmente un espacio de separación de dos líneas con el texto siguiente al punto donde hayamos insertado la etiqueta <P>.

La etiqueta de fin párrafo </P> es opcional, no siendo necesario incluirla. Aunque siempre es recomendable delimitar claramente el inicio y final de un párrafo. Además, cuando usemos esta etiqueta como cerrada <P>. </P>, tenemos la posibilidad de incluirle el atributo align el cual

indica al navegador la forma de justificar el texto incluido en el párrafo. El formato sería el siguiente:

**<P align**= left / right / center / justify >Texto contenido en el párrafo </P>



M1\_img\_28

**Línea Horizontal <HR>**

Nos permite insertar una línea horizontal, cuyo tamaño podremos determinar a través de sus atributos. Si no especificamos ningún atributo dibujará una línea que ocupe el ancho de la pantalla del navegador. Su utilidad es la de permitirnos dividir nuestra página en distintas secciones. No será necesaria la etiqueta de fin </HR>.

El formato de la etiqueta con sus posibles atributos es:

**<HR align**= left / right / center noshade size=n width=n >

* align= left / right / center. Permite establecer la alineación de la línea a la izquierda, a la derecha o centrarla.
* noshade. No muestra la sombra de la línea, evitando el efecto de tres dimensiones.
* size=n. Indica el grosor de la línea en pixels.
* width=n ó n%. Especificará el ancho de la línea, este se podrá especificar en pixels (n) o en tanto por ciento del ancho de la ventana (n%).
* Color=name o #rrggbb. Especifica el color de la línea horizontal

Ejemplo:

<hr>

<br>

<hr noshade>

<br>

<hr width="30%" align="right">

<br>

<hr size="30" width="70%" noshade color="orange" >

<br>



M1\_img\_29

**Texto preformateado: <PRE>**

Nos permitirá visualizar el texto tal y como se ha escrito, respetando los saltos de línea, las tabulaciones, y espacios en blanco de todo el texto incluido entre las etiquetas <PRE> y </PRE>.

El texto se mostrará con una fuente de tipo <TT> teletype.

Ejemplo:

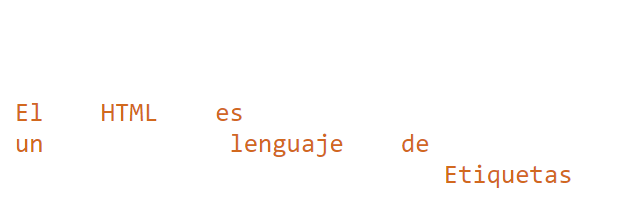
<PRE>

El HTML es

un lenguaje de

Etiquetas

</PRE>



M1\_img\_30

2.6.1Cabeceras y atributos de texto

Cabeceras: <H1>...<H6>

En un documento HTML podemos incluir seis tipos distintos de cabeceras, que van a constituir normalmente el título y los distintos apartados que forman el documento, aunque podremos utilizarlas en el punto que queramos del documento para resaltar cualquier texto. El texto que componga cada cabecera deberá estar incluido entre las etiquetas de inicio (<**Hn**>) y fin (**/Hn**) correspondiente. La cabecera <H1>será la que muestre el texto de mayor tamaño, este tamaño irá disminuyendo hasta llegar a la cabecera </H6>.

<H1> es la de mayor tamaño y podemos llegar hasta <H6> que sería la más pequeña.

Ejemplo:

<H1>HTML es un lenguaje de marcas </H1>

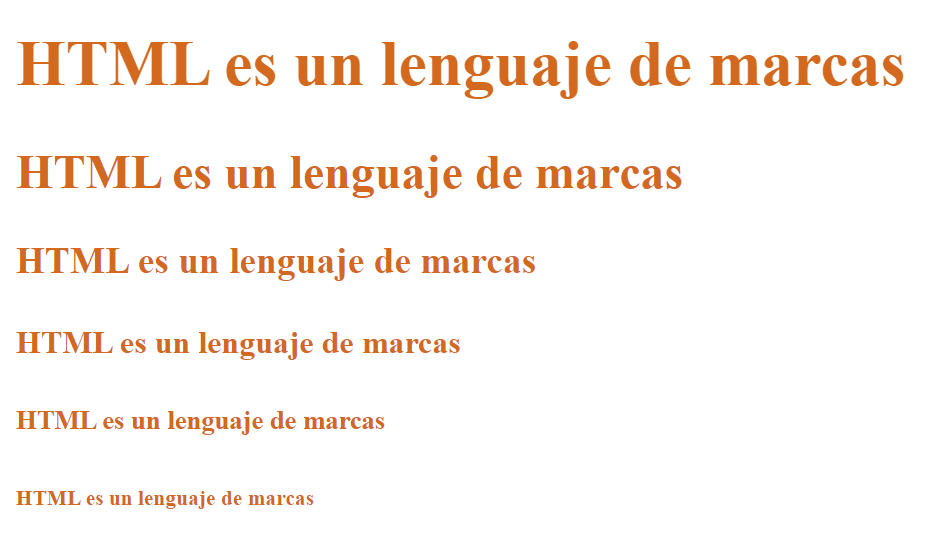
<H2>HTML es un lenguaje de marcas </H2>

<H3>HTML es un lenguaje de marcas </H3>

<H4>HTML es un lenguaje de marcas </H4>

<H5>HTML es un lenguaje de marcas </H5>

<H6>HTML es un lenguaje de marcas </H6>

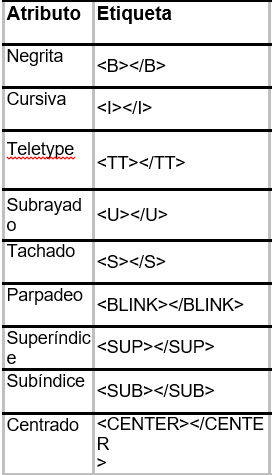


M1\_img\_31

Las etiquetas <H..> llevan implícito ya el salto de línea.

Atributos del texto

Al texto de nuestro documento HTML le podemos aplicar distintos atributos (**negrita**, *cursiva*, subrayado, etc..), al igual que hacemos cuando trabajamos con el procesador de textos en la edición de nuestros documentos. Para aplicar estos atributos disponemos de distintas etiquetas, que aplicarán su efecto al texto incluido entre sus indicadores de inicio y fin.



M1\_img\_32

Ejemplo:

<p><b> Hoy es Jueves</b> </p>

<p><u> Hoy es Jueves</u> </p>

<p><s> Hoy es Jueves</s> </p>

<p><tt> Hoy es Jueves</tt> </p>

<p>2 <sup>3</sup></p>

<p>H <sub>2 </sub> O </p>

<center> Hoy es Jueves </center>



M1\_img\_33

<BLINK> (Parpadeo) no es implementada por el navegador internet Explorer.

Si queremos aplicar efectos más espectaculares a nuestro documento HTML, debemos variar el tamaño, el color y el tipo de letra del texto. La etiqueta que nos permite todo esto es

**<**FONT**>...<**/FONT**>,** por medio de sus atributos size**,** color y face:

**<**FONT size**=**"n" ó "+/- n**"** color**="**#rrggbb ó name**"** face**="**nombre de la fuent**" >**

* size="n" ó "+/- n". El atributo size nos permite especificar un tamaño determinado para la fuente del texto incluido entre las etiquetas de inicio y fin, el cual puede estar entre 1 y

7. El texto de tamaño normal equivale a la fuente de tamaño 3 (fuente base). Por tanto, si especificamos size=+2, el tamaño de la fuente será 5. Y si especificamos size= -1, el tamaño será 2.

* color="#rrggbb ó name"**.** Nos va a permitir definir el color que tendrá el texto incluido entre las etiquetas.

* face="nombre de font". Nos va a permitir escribir texto con el tipo de letra que le especifiquemos. En el caso de que el tipo de letra que le hayamos especificado no esté cargada en el ordenador que lee la página, se usará el font por defecto del navegador.

Ejemplo:

<font color="red" size="+4" face="Verdana">

<p> Hoy es Viernes</p>

</font>



M1\_img\_34

2.6.2 Listas



M1\_img\_35

Podemos representar elementos en forma de lista dentro de nuestros documentos de una forma muy sencilla y con una gran versatilidad. Estas listas podrán incluir cualquiera de los elementos HTML e incluso podemos definir listas anidadas, es decir, listas dentro de listas. HTML nos permite crear tres tipos distintos de listas:

* Listas no numeradas
* **Listas numeradas**
* **Listas de definición**

**Listas no numeradas: <UL>**

Con este tipo de listas podemos especificar una serie de elementos sin un orden predeterminado, precedidos de una marca o viñeta que nosotros mismos podemos definir. Para la definición de los límites de la lista utilizaremos la etiqueta <UL>.... </UL>, y para determinar cada uno de los elementos que la componen usaremos la etiqueta <LI>. El formato es el siguiente:

<UL type="disk" ó "circle" ó "square">

<LH>Título de la lista</LH>

<LI>Elementos </LI>

</UL>

Con el atributo type vamos a especificar el tipo de marca o viñeta que antecederá a cada uno de los elementos de la lista. Estas marcas podrán ser: un disco (disk), un círculo (circle) o un cuadrado (square). La etiqueta <LH> nos va a servir para especificar un posible título de la lista, es opcional.

Ejemplo:

<UL>

<li> HTML </li>

<li> CSS </li>

<li> Base de Datos </li>

<li> Java </li>

</UL>



M1\_img\_36

Ahora vamos a cambiar el punto de la lista por un cuadrado con el atributo type:

<UL type="square">

<li> HTML </li>

<li> CSS </li>

<li> Base de Datos </li>

<li> Java </li>

</UL>



M1\_img\_37

**Listas numeradas: <OL>**

Con este tipo de listas podemos especificar una serie de elementos numerados según el lugar que ocupan en la lista. Para la definición de los límites de la lista utilizaremos la etiqueta

<OL>

</OL>,

y para determinar cada uno de los elementos que la componen usaremos la etiqueta <LI>. El formato es el siguiente:

<OL start="n" type="Tipo de lista">

<LH>Título de la lista</LH>

<LI>Elemento 1

<LI>Elemento 2

....

<LI>Elemento n

</OL>

Ejemplo:

<OL>

<li> Madrid </li>

<li>Segovia</li>

<li>Toledo</li>

<li>Guadalajara</li>

</OL>



M1\_img\_38

Con el atributo start vamos a especificar el número por el que va a empezar la lista.

<OL start="3">

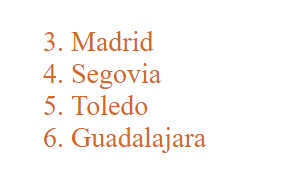
<li> Madrid </li>

<li>Segovia</li>

<li>Toledo</li>

<li>Guadalajara</li>

</OL>



M1\_img\_39

Si no indicamos este argumento la lista empezará a numerarse a partir del 1. Con el atributo type vamos a especificar el tipo de lista numerada.

A: Letras mayúsculas (A, B, C, ...)

<OL type="A">

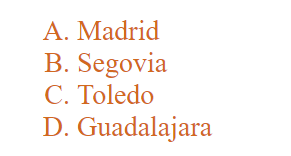
<li> Madrid </li>

<li>Segovia</li>

<li>Toledo</li>

<li>Guadalajara</li>

</OL>



M1\_img\_40

a: Letras minúsculas (a, b, c, ...)

I: Números romanos en mayúsculas (I, II, III, IV, ...)

<OL type="I">

<li> Madrid </li>

<li>Segovia</li>

<li>Toledo</li>

<li>Guadalajara</li>

</OL>



M1\_img\_41

i: Números romanos en minúsculas (i, ii, iii, iv, ...)

Listas de definiciones: <DL>

Estas listas nos van a servir para especificar una serie de términos y sus definiciones correspondientes. Para la definición de la lista usaremos la etiqueta <DL>. </DL>, para especificar los términos usaremos la etiqueta <DT> y para especificar la definición correspondiente a cada término usaremos la

etiqueta <DD>.

El formato es el siguiente:

<DL>

<LH>Título de la lista</LH>

<DT>Término 1

<DD>Definición 1

<DT>Término 2

<DD>Definición 2

....

<DT>Término n

<DD>Definición n

</DL>

Ejemplo:

<DL>

<DT> 1. HTML

<DD>Es un lenguaje de marcas</DD>

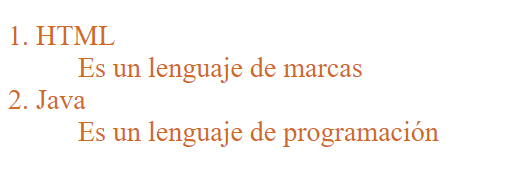
</DT>

<DT> 2. Java

<DD>Es un lenguaje de programación</DD>

</DT>

</DL>



M1\_img\_42

2.6.3 Hiperenlaces

Los hiperenlaces son *enlaces de hipertexto* que nos van a permitir acceder de manera directa a otros documentos HTML independientemente de su ubicación, o aotras zonas dentro de nuestro propio documento.

Enlaces a otras páginas: <A href=...>

Con este tipo de hiperenlaces vamos a poder acceder tanto a otras páginas que estén ubicadas dentro de nuestro propio sistema como a páginas ubicadas en puntos muy distantes del globo. El formato de este tipo de hiperenlaces es:

<A href="URL a la que se accede">Texto Visible</A>

Ejemplo:

<A Href="http://www.google.com"> Ir a Google </A>



M1\_img\_43

Con el atributo href vamos a especificar la URL del documento al que se pretende acceder. El texto contenido entre las etiquetas de comienzo y fin nos va a servir para definir el hiperenlace, por lo que debería ser clarificador del contenido del documento con el que vamos a enlazar.

Esta definición aparecerá resaltada normalmente en azul y subrayada. En la mayoría de los navegadores esta definición del hiperenlace es sensible, por lo que cuando el cursor pasa por encima del mismo este cambia de aspecto indicándolo.

También podemos indicar el hiperenlace por medio de una imagen. Si esta imagen tiene definido un borde, este aparecerá resaltado en color azul. El formato correspondiente sería:

<A href="URL a la que se accede"><IMG src="nombre.extensión"> y/o texto</A>

Ejemplo:

<A Href="http://www.google.com"><img src="google.jpg" width="100" height="50" ></A>



M1\_img\_44

Para cambiar el color del hiperenlace modificamos el atributo link, vlink o alink de la etiqueta <BODY>:

<BODY link=”orange”>

Por defecto aparecerían en color naranja.

También podemos cambiar el color de los links activos ( en el momento que estamos pinchando el enlace) y los ya visitados con propiedades en el <BODY>

<body bgcolor="#FFFFFF" text="chocolate" alink="yellow" link="orange" vlink="green">

Anclas.Enlaces locales dentro de la misma página: <A name=...>

Este tipo de hiperenlaces nos va a permitir marcar distintas zonas del documento activo con el objeto de tener un acceso directo a las mismas. Una buena utilidad de este tipo de enlaces radica en la creación de índices para documentos largos, de forma que si pinchamos en el hiperenlace correspondiente al título de un capítulo determinado, el navegador saltará automáticamente hasta el comienzo de dicho capítulo.

Para la creación de estos hiperenlaces, debemos seguir dos pasos:

1. Marcar las distintas zonas o secciones del documento. Esto lo haremos con el parámetro name:

<A name="Identificador de sección">Texto de la sección</A>

A cada sección le asignaremos un identificador distinto, para poder referenciarlas posteriormente de manera inequívoca.

1. Especificar un enlace a cada una de las secciones que hayamos definido.

<A href="#Identificador de sección">Texto del enlace a la sección</A>

Si pinchamos en este enlace, el navegador saltará automáticamente a la sección correspondiente.

Si nos creamos una página HTML con distintos capítulos, podríamos crear una sección para cada uno de ellos, de forma que si pinchamos en el hiperenlace correspondiente al Tema1 1, el navegador saltaría directamente a la sección correspondiente al **Tema 1**.

Ejemplo:

<html>

<head>

<title>Ejemplo 2</title>

</head>

<body>

<h1> Programación Estructurada</h1>

<hr>

<h4> <a href="#c1">Capítulo 1</a> </h4>

<h4> <a href="#c2">Capítulo 2 </a></h4>

<h4><a href="#c3">Capítulo 3</a></h4>

<br><br><br><br><br><br><br>

<br><br><br><br><br><br><br>

<br><br><br><br><br><br><br>

<br><br><br><br><br><br><br>

<br><br><br><br><br><br><br>

<br><br><br><br><br><br><br>

<br><br><br><br><br><br><br>

<br><br><br><br><br><br><br>

<br><br><br><br><br><br><br>

<br><br><br><br><br><br><br>

<br><br><br><br><br><br><br>

<br><br><br><br><br><br><br>

<font color="lime" face="Verdana">

<A name="c3">

<h2>Capítulo 3</h2>

</font>

What is Lorem Ipsum?

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of.

</body>

</html>

2.6.4 Imágenes



M1\_img\_45

Podremos dar un aspecto mucho más vistoso a nuestros documentos HTML con la inclusión de imágenes. Actualmente los navegadores reconocen casi todos los formatos de imagen como gif, jpg, bmp, png, etc…

El formato gif es más recomendado para la inclusión de iconos, gráficas, y el formato jpg para el caso de insertar fotografías en nuestras páginas.

La etiqueta que nos va a permitir la inserción de imágenes es <IMG>. La imagen se insertará justo en el punto del documento donde coloquemos dicha etiqueta, pudiendo insertar imágenes dentro de listas, tablas o formularios.

El formato de la etiqueta, con sus posibles argumentos, es el siguiente:

**<**IMG src**=**"URL de la imagen**"** alt**="**Texto alternativo**"** align**=**"top/middle/botton**/**left/rigth**"** border**=**"Tamaño" height**=**"Tamaño" width**=**"Tamaño" hspace**=**"margen" vspace**=**"margen"**>**

* src="URL de la imagen". El atributo src nos va servir para indicar la URL de la imagen que queremos insertar, es decir, el servidor y camino hasta llegar al fichero de la imagen.

No será necesario que la imagen que queremos insertar en nuestro documento esté en el ordenador local, aunque sí es recomendable ya que el acceso a las mismas puede ser más lento.

Por tanto, es una buena práctica copiar todas las imágenes, que vayamos a incluir en nuestro documento HTML, al ordenador local.

* Alt/title="Texto alternativo" .Nos va a permitir mostrar un texto alternativo para el caso en que el navegador no sea capaz de mostrar la imagen. Su uso es muy recomendable ya que cuando publiquemos nuestras páginas en la WWW, no sabremos desde qué tipo de navegadores accederán los internautas a nuestras páginas. Actualmente usamos el atributo title en vez de alt porque es más moderno y es aconsejable para el posicionamiento SEO
* align=top/middle/botton left/rigth. Nos permite indicar cómo se alineará el texto que siga a la imagen.

top: alinea el texto con la parte superior de la imagen.

middle: con la parte central.

bottom: con la parte inferior.

left: imagen alineada a la izquierda de la página.

rigth: imagen alineada a la derecha de la página.

En el caso de que el texto que acompaña a la imagen tenga más de una línea, tan sólo la primera será la que se alineará según alguna de las anteriores alineaciones, el resto de líneas se incluirán debajo de la imagen quedando un efecto bastante feo.

Para solventar esto, están las alineaciones **LEFT** (imagen alineada a la izquierda de la página) y **RIGHT** (imagen alineada a la derecha de la página). Las cuales permiten que la imagen se inserte dentro del párrafo que la rodea, produciendo un resultado mucho más profesional en la integración de texto e imágenes.

Estas dos alineaciones no se podrán usar en combinación con las anteriores.

* border="Tamaño" Permite añadirle un borde a la imagen del tamaño que le especifiquemos.

Si la imagen forma parte de un hiperenlace esta aparecerá con un borde del color del hiperenlace. Si no deseáramos que aparezca este borde tendríamos que especificar border=0.

* height="Tamaño". Permite especificar el alto de la imagen, en puntos de pantalla (pixels) o en % del tamaño del documento
* width="Tamaño". Con este argumento especificaremos el ancho al que mostraremos la imagen: en puntos de pantalla (pixels) o en % del tamaño del documento

En el segundo de los ejemplos de la [Tabla 31,](#bookmark46) el ancho de la imagen se ha extendido al 50% del total del ancho de la celda donde está contenida la imagen. Si pruebas este ejemplo en tu documento HTML, la imagen se extenderá horizontalmente hasta ocupar el 50% del ancho de la página.

* hspace="Margen" .Permite especificar el número de espacios horizontales (en puntos) que separarán a la imagen de cualquier elemento que le siga o le anteceda.
* vspace="Margen". Con este argumento especificaremos el margen vertical (en puntos) que habrá entre la imagen y cualquier otro elemento de nuestro documento.

Ejemplo:

<IMG src="oso.jpg" width="160" height="90" border="1" title="oso polar" alt="oso polar">

M1\_img\_46

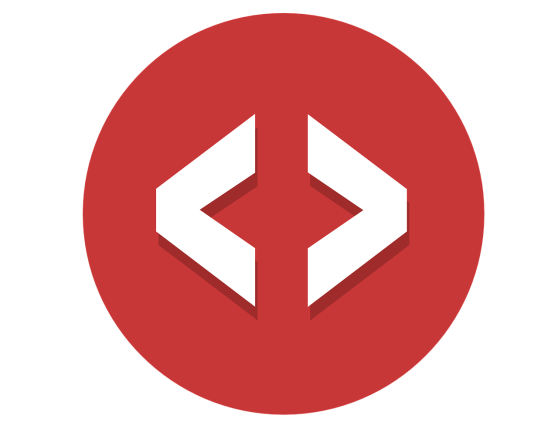
Una imagen puede servir como hipervínculo a otra página web:

<A HREF="http://www.faunia.es">

<IMG src="oso.jpg" width="160" height="90" border="1" title="oso polar" alt="oso polar">

</A>

2.6.5 Tablas



M1\_img\_47

HTML nos da la posibilidad de la insertar de cualquier etiqueta HTML (texto, imágenes, hiperenlaces, listas, etc.), dentro de una tabla. Gracias a lo cual conseguiremos dar una mayor y mejor estructuración a los contenidos de nuestros documentos. Además, la definición de las tablas en HTML es muy abierta, pudiendo en cualquier momento redimensionar la tabla, es decir, cambiar su número de filas o de columnas, cambiar el tamaño de alguna de sus celdas, etc.

La etiqueta que nos va a permitir la definición de tablas es:

<TABLE>

</TABLE>

El formato general de la etiqueta sin ningún argumento, es el siguiente:

**<**TABLE**>**

**<**TR **>**

<TD> ……………</TD>

**<**/TR**>**

**<**/TABLE**>**

**<**TH**>**Cabecera de la celda, se escribe en negrita **<**/TH**>**

<TD>Contenido de la celda </TD>

Ejemplo:

<TABLE border="1">

<TR>

<TD> 1 </TD>

<TD> 2 </TD>

</TR>

<TR>

<TD> 3 </TD>

<TD> 4 </TD>

</TR>

</TABLE>



M1\_img\_48

Dentro de una celda podemos guardar cualquier elemento HTML como por ejemplo una imagen:

<TABLE border="1">

<TR>

<TD**> <img src="oso.jpg”>** </TD>

<TD> 2 </TD>

</TR>

<TR>

<TD> 3 </TD>

<TD> 4 </TD>

</TR>

</TABLE>



M1\_img\_49

Elementos de la tabla:

1. <TABLE></TABLE>: <TR> </TR>: Definición de las filas de la tabla. Por cada etiqueta <TR> que incluyamos se creará una fila en la tabla. No será necesario indicar la etiqueta de cierre.
2. <TH></TH> ó <TD></TD>: Definición de cada una de las celdas de la tabla. Vemos que estas etiquetas están contenidas dentro de otra etiqueta de definición de fila, de forma que por cada etiqueta <TH> o <TD> que incluyamos se creará una celda dentro de la fila correspondiente. La etiqueta <TH> la usaremos para crear celdas de tipo cabecera, es decir, celdas cuyo contenido está resaltado en **negrita**, dejando la etiqueta <TD> para definir celdas de datos. En este caso tampoco es necesario indicar las etiquetas de cierre.

Todas estas etiquetas tienen sus respectivas propiedades o atributos

Vamos ahora a analizar lo distintos argumentos que componen las etiquetas de la tabla, analizándolos desde lo general (la definición de la tabla), hasta lo específico (la definición de las celdas):

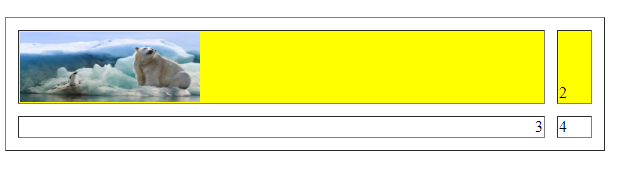
1. Definición de la tabla: <TABLE>.

<TABLE border="n" cellpadding="n" cellspacing="n" width="n ó %" height="n ó %" bgcolor="#rrggbb ó nombre" >

</TABLE>

* + border="n". Si especificamos este argumento, se dibujará un borde alrededor de la tabla del ancho que le indiquemos. Si ponemos cero, o no ponemos este atributo el borde se ve trasparente.
  + cellpadding="n". Distanci en puntos que separa el contenido de la celda con el borde de la misma, siendo 1 por defecto.
  + cellspacing="n". Indica el espacio en puntos que separa a las celdas contenidas dentro de la tabla, siendo 2 por defecto.

<TABLE border="1" width="60%" cellspacing="12">



M1\_img\_50

* + width="n ó %". Indica la anchura de la tabla en puntos o en % en función del ancho de la ventana del navegador. Si no indicamos este argumento, el ancho de la tabla se ajustará al tamaño del contenido de las celdas.

<TABLE border="1" width="80%">

* + height="n ó %". Nos permite definir la altura de la tabla en puntos o en % de la altura de la ventana del visualizador. Si indicamos este argumento, lo recomendable es darlo en puntos ya que es más complicado adecuarnos a la altura de la ventana del visualizador. Al igual que en el caso anterior, si no especificamos este argumento la altura se adecuará al contenido de las celdas.
  + bgcolor= "#rrggbb" o "nombre del color". Nos permite definir un color de fondo para todas las celdas de la tabla o por cada celda de forma independiente.

1. Definición de las filas de la tabla: <TR>

**<**TR align**=**"left/right/center" valign**=**"top/middle/bottom bgcolor="#rrggbb ó nombre"**>**

* + align="left/right/center". Con este argumento vamos a indicar la alineación horizontal (*left*:izquierda, *right*:derecha, *center*:centrado) del contenido de todas las celdas de la fila correspondiente. También se puede hacer con cada celda por separado.
  + valign="top/middle/bottom". Con este argumento vamos a indicar la alineación vertical (*top*:arriba, *middle*:centrado, *bottom*:abajo) del contenido de todas las celdas de la fila correspondiente.
  + bgcolor= "#rrggbb" o "nombre del color". En este caso, este argumento nos permitirá definir un color de fondo para todas las celdas de una fila.

Ejemplo:

<TABLE border="1" **width="60%"**>

<TR bgcolor="yellow">

<TD> <img src="m1\_img\_46.jpg" width="180" height="70">

</TD>

<TD> 2 </TD>

</TR>

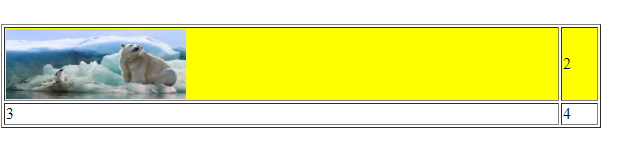
<TR>

<TD> 3 </TD>

<TD> 4 </TD>

</TR>

</TABLE>



M1\_img\_51

1. Definición de las celdas de la tabla: <TH> ó <TD>

**<**TH ó TD align**=**"left/right/center/justify" valign**=**"top/middle/bottom bgcolor="#rrggbb ó nombre" width**=**"n" **>**

<TABLE border="1" width="60%" >

<TR bgcolor="yellow">

**<TH>Columna 1</TH>**

**<TH >Columna 2 </TH**>

</TR>

<TR >

<TD><img src="m1\_img\_46.jpg" width="180" height="70"></TD>

<TD valign="bottom"> 2 </TD>

</TR>

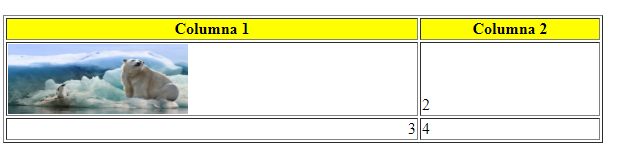
<TR>

<TD align="right"> 3 </TD>

<TD> 4 </TD>

</TR>

</TABLE>



M1\_img\_52

* + align="left/right/center/justify". Con este argumento vamos a indicar la alineación horizontal (*left*: izquierda, *right*: derecha, *center*: centrado, *justify*:justificado) del contenido de cada celda por separado. Su significado es igual que en la etiqueta

<TR>.

* + valign="top/middle/bottom". Con este argumento vamos a indicar la alineación vertical (*top*:arriba, *middle*:centrado, *bottom*:abajo) del contenido de cada celda

Ejemplo:

<TABLE border="1" **width="60%"**>

<TR bgcolor="yellow">

<TD> <img src="m1\_img\_46.jpg" width="180" height="70">

</TD>

<TD **valign=”bottom”**> 2 </TD>

</TR>

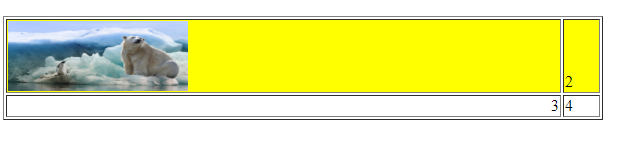
<TR>

<TD **align=”right”**> 3 </TD>

<TD> 4 </TD>

</TR>

</TABLE>



M1\_img\_53

* + bgcolor= "#rrggbb" o "nombre del color". En este caso, podremos definir un color de fondo para cada una de las celdas de manera independiente.
  + width="n ó %". Indica el ancho de la celda en puntos o en % en función del tamaño de la tabla

2.6.6 Formularios

Un formulario como su nombre indica es un conjunto de elementos como los formularios que solemos rellenar a mano, con todos sus elementos de entrada de datos.

El formulario es el elemento de HTML que nos va a permitir interactuar con los usuarios de nuestras páginas web, de forma que podemos solicitarle información al usuario y procesarla. De esta forma, podremos crear en nuestras páginas: *encuestas* para solicitar la opinión del visitante sobre algún tema e incluso sobre el contenido de nuestra propia página web, *cuestionarios* para evaluar la asimilación de contenidos sobre un tema concreto que se trate en la página, etc.

El contenido de la información introducida por medio del formulario será enviado a la dirección URL donde resida el programa que se encargará de procesar los datos.

La etiqueta HTML que nos va a permitir la creación de formularios es <FORM>. Su formato es el siguiente:

**<**FORM

action**=**"programa que procesa los datos que se reciben **"** method**=**"post/get"**>**

Elementos del formulario

**<**/FORM**>**

* **action**="Programa que procesa los datos que llegan". A través de este argumento indicaremos, como ya hemos mencionado, la dirección del fichero de programa que va a tratar la información suministrada por el formulario.

Tipos de programas:

* Servlets
* PHP
* Páginas JSP
* Páginas ASP o ASPx

También podríamos incluir como URL que trata el formulario una dirección de correo electrónico. Por tanto, el valor de este argumento podría ser: action ="mailto: dirección de correo".

* **method**="post/get". A través de este argumento indicaremos el método empleado para transferir la información generada por el formulario.

Get: Los datos se ven en la URL. Podemos enviar un máximo de 240 caracteres.

Post: Los datos no se ven en la URL. Ho hay limitación en caracteres.

Por defecto a un servidor web siempre le llega un Get

Entrada básica de datos

Para definir los distintos tipos de campos básicos de entrada de datos usaremos la etiqueta <INPUT>. El formato es el siguiente:

**<**INPUT type**=** text / password / checkbox / radio / hidden / submit / image / reset name**=**"Variable" value="Valor inicial"**>**

* El argumento type determinará el tipo de campo de entrada que estamos insertando.
* El argumento name especifica el nombre de la variable que tomará el valor introducido en el campo.
* El argumento value especifica el valor por defecto que tendrá el campo.

Campo de texto

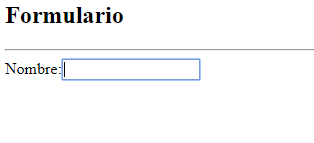
Con este argumento vamos a indicar que el campo a introducir será un texto. El formato sería:

**<**INPUT type= text name="nombre" value="Valor inicialización" size="Tamaño" maxlength="Longitud máxima">

El significado de los nuevos atributos es:

* size="Tamaño". Indicaremos el tamaño en caracteres de la ventana de introducción de texto.
* maxlength="Longitud máxima". Indicaremos el número máximo de caracteres a introducir en el campo.

Nombre:<input **type="text"** size="15" name="nombre"><br>



M1\_img\_54

Campos Password

Con este argumento indicamos que el campo a introducir será una palabra clave, por lo que los caracteres que se introduzcan serán sustituidos por asteriscos en la visualización por pantalla. El formato es:

**<**INPUT type= password name="nombre" value="Valor inicialización" size="Tamaño" maxlength="Longitud máxima">

Clave:<input **type="password"** size="5" name="clave" maxlength="3"><br>



M1\_img\_55

CheckBoxes

El checkbox es un botón que presenta dos estados: marcado (1) y desmarcado (0). Podremos variar su estado simplemente pinchando con el ratón. El formato es:

**<**INPUT type= checkbox name="nombre" value="Valor" checked>

Si especificamos el argumento checked, el botón aparecerá marcado por defecto.

Si incluimos el argumento value, cuando el botón esté marcado su variable asociada adquirirá el valor dado por value.

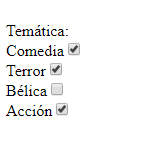
Temática:

Comedia<inpu**t type="checkbox"** value="comedia"><br>

Terror<input type="checkbox" value="terror"><br>

Bélica<input type="checkbox" value="belica"><br>

Acción<input type="checkbox" value="accion"><br>



M1\_img\_56

Se pueden marcar uno o varios o ninguno.

RadioButtons

Este argumento se usa cuando hay que hacer una selección entre varias alternativas excluyentes, pudiéndose seleccionar únicamente una de las alternativas. Debemos incluir una etiqueta radio por cada una de las posibles alternativas. El formato es:

<INPUT type=radio name="nombre" value="Valor" checked>

Si especificamos el argumento checked, el botón aparecerá marcado por defecto. En este caso únicamente uno de los botones de radio podrá aparecer marcado, ya que se usa para el caso de opciones excluyentes.

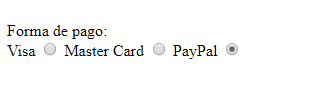
Cuando un botón esté seleccionado la variable asociada a la lista de botones adquirirá el valor dado por value.

Forma de pago:

Visa <input **type="radio"** value="visa" name="pagos">

Master Card <input type="radio" value="master" name="pagos">

PayPal <input type="radio" value="paypal" name="pagos">



M1\_img\_57

Los radio son excluyentes entre ellos por eso les ponemos a todos el mismo name.

Campos ocultos

Este tipo de campos no son visibles para el usuario. Su uso tiene sentido en el caso de enviar algún tipo de información que no deba ser visualizada o variada por el lector de nuestra página Web. El formato es:

<INPUT type= hidden name="nombre" value="Valor" >

<input **type="hidden"** size="5" name="camisas" value="345" >

Entrada de texto en múltiples líneas

En un formulario también podremos introducir un campo de texto que abarque varias líneas. Para ello usaremos la etiqueta <TEXTAREA> </TEXTAREA>. Su formato es el siguiente:

<TEXTAREA name="nombre" rows=Filas cols=Columnas> Contenido por defecto.

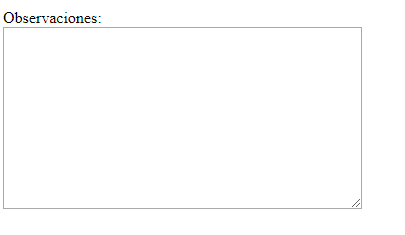
</TEXTAREA>

Los argumentos rows y cols nos van a permitir especificar, respectivamente, el número de filas de texto visibles y el número de columnas de texto visibles en el campo.

Observaciones:<br>

<**TEXTAREA** cols="30" rows="10">

</TEXTAREA>



M1\_img\_58

Listas desplegables

Con los campos de selección podremos desplegar una lista de opciones, entre las que podremos seleccionar una. Para ello usaremos la etiqueta <SELECT> </SELECT>. Su formato es el siguiente:

<SELECT name="nombre" multiple size=n>

<OPTION selected value="Valor\_1">Primera Opción

<OPTION value="Valor\_2">Segunda Opción

..........

</SELECT>

* La variable referenciada por name tomará el valor(value) de la opción seleccionada.
* Si especificamos el argumento múltiple, se mostrarán todas la opciones en forma de tabla. El número de opciones visibles en la tabla vendrá dado por el argumento size.
* Con la etiqueta <OPTION> iremos definiendo cada una de las opciones de la lista.
* El argumento selected indica cuál es la opción por defecto.

<SELECT name="colores" >

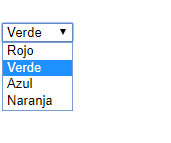
<OPTION value="red">Rojo</OPTION>

<OPTION value="green" selected>Verde</OPTION>

<OPTION value="blue">Azul</OPTION>

<OPTION value="orange">Naranja</OPTION>

</SELECT>



M1\_img\_59

Por defecto solo podemos seleccionar un valor.

Si ponemos el atributo **multiple** se convierte en una lista múltiple donde podremos seleccionar más de un valor pulsando a la vez que seleccionamos la tecla Control.

<SELECT name="colores" **multiple**>

<OPTION value="red">Rojo</OPTION>

<OPTION value="green" selected>Verde</OPTION>

<OPTION value="blue">Azul</OPTION>

<OPTION value="orange">Naranja</OPTION>

</SELECT>



M1\_img\_60

Botón de envío de datos

Con este argumento especificamos un botón en el que al pulsar, los datos serán enviados al programa o dirección de correo encargada de procesar la información recogida por el formulario. El formato es:

<INPUT type= submit value="Mensaje a mostrar">

Botón de borrado de datos

Con este argumento especificamos un botón que al ser pulsado borrará el contenido actual de todos los campos, dejándolos con sus valores por defecto. Su formato es:

<INPUT type= reset value="Texto del botón">

Con el parámetro value especificamos el texto que etiquetará al botón.

2.6.7 Marcos

Los marcos o frames son una utilidad del HTML que nos permite dividir la ven tana del navegador en diferentes ventanas, que podrán ser manipuladas de manera independiente. Esto nos permitirá mostrar una página web diferente en cada una de la ventanas o frames.

Si una página lleva frames no debe de llevar <BODY> ya que realmente los frames son contenedores de otras páginas que son las que sí llevan el código HTML que se muestra.

Estructura de un documento con frames

Un documento con frames se estructura de manera diferente a un documento HTML normal. Así, en este tipo de documentos no aparecerá la etiqueta <BODY>. En su lugar se incluirá la etiqueta

<FRAMESET> </FRAMESET> dentro de la cual se definirán las distintas frames que se incluyan en el documento**.**

La estructura general de un documento de este tipo sería, por tanto:

<HTML>

<HEAD>Cabecera</HEAD>

<FRAMESET>

……………

……………

</FRAMESET>

<NOFRAMES>

Instrucciones HTML en caso de que la página no admota la etiqueta <FRAME>

</NOFRAMES>

</HTML>

Puede darse el caso de que programemos nuestro documento con frames siguiendo la estructura anterior y que el navegador que se use para leer el documento no soporte la característica de frames, con lo cual no se mostrará nada de lo indicado entre las etiquetas <FRAMESET> y </FRAMESET>.

Para solucionar este problema, todas las instrucciones HTML incluidas entre las etiquetas

<NOFRAMES> y </NOFRAMES> serán ejecutadas por los navegadores que no soportan las frames.

La sintaxis de la etiqueta <FRAMESET> es:

<FRAMESET rows="porcentaje de filas" cols="porcentaje de columnas">

<FRAME src="URL de la frame 1" name="Nombre del frame 1”>

<FRAME src="URL de la frame 2" name="Nombre del frame 2”>

......

</FRAMESET>

Los argumentos de <FRAMESET> son rows y cols, definiéndose únicamente uno de los dos atributos en función de si la división de la pantalla se realizará por filas (rows) o por columnas (cols).

Para cada uno de los marcos que creemos debemos especificar su tamaño en puntos o en porcentaje del tamaño de cadamarco. Así por ejemplo, si especificamos rows= "50%, 25%, 25%" crearemos tres marcos horizontales ocupando la primera el 50% de la ventana principal y las otras dos el resto. Si especificamos cols="100,150,\* " crearemos tres marcos verticales ocupando la primera 100 puntos de la ventana principal, la segunda 150 puntos y tercera el espacio restante.

Definición de cada frame

Como ya sabemos, cada una de las frames que incluyamos en nuestro documento vendrá definida por la etiqueta <FRAME> correspondiente. Su formato es el siguiente:

<FRAME src="URL" name="Nombre" marginwidth="n" marginheight="n" scrolling="yes/no/auto noresize>

* Src = "URL". El marco mostrará el contenido del documento HTML que se indique con dicha URL.
* Name = "Nombre". Indica el nombre por el que nos referiremos a ese marco.
* marginwidth = "n". Establecemos los márgenes izquierdo y derecho del contenido de la frame en puntos por pantalla.
* marginheight = "n". Establecemos los márgenes superior e inferior del contenido de la frame en puntos por pantalla.
* scrolling = "yes / no / auto". Indicaremos si se aplica una barra de desplazamiento al marco . El valor "**yes**" muestra siempre la barra de desplazamiento, "**no**" no la muestra nunca y "**auto**" la muestra solo en caso de que sea necesario para poder ver la página.
* Noresize. Cuando el usuario visualiza nuestra página con frames en su navegador podrá redimensionar los marcos seleccionando un borde con el cursor del ratón y desplazándolo. Si se indica este argumento, el usuario no podrá variar el tamaño de los marcos al visualizarlas con el navegador.

Ejemplo de un frameset:

pagina\_derecha.html

<html>

<body>

<H1> Página Derecha</H1>

</body>

</html>

pagina\_izquierda.htm

<html>

<body bgcolor="steelblue">

</body>

</html>

<html>

<**FRAMESET**  cols="25%,75%" noresize border="0">

<FRAME src="pagina\_izquierda.html" >

<FRAME src="pagina\_derecha.htm" >

**</FRAMESET**>

</html>



M1\_img\_61

2.6.8 Iframes

Un iframe es una manera de incrustar un fichero normalmente de video dentro de nuestra página HTML

Por ejemplo para insertar un video de youtube hacemos lo siguiente:

Abrimos youtube y buscamos un video cualquiera



M1\_img\_62

Ahora con botón derecho sobre el video seleccionamos la opción copiar código de inserción



M1\_img\_63

Y ahora lo insertamos(copiamos) dentro de nuestro código HTML:

<**iframe** width="697" height="392"

src="https://www.youtube.com/embed/QtYV3N3dTJ4"

frameborder="0"

allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen>

<**/iframe**>

Y se mostrará de este modo:



M1\_img\_64

Inserción de cualquier tipo de fichero

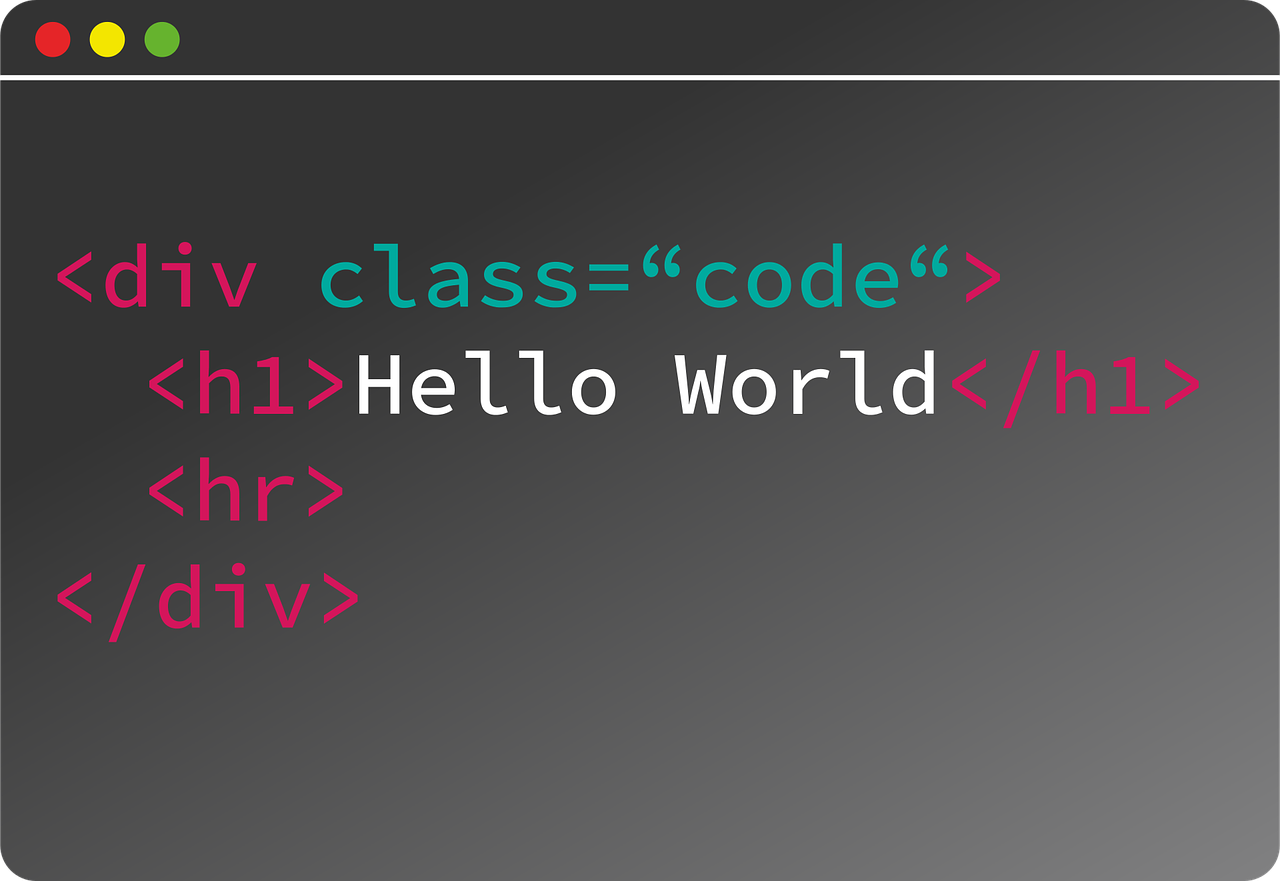
<EMBED src="URL del fichero" width= n ó n% height= n ó n%>

Con esta etiqueta podremos insertar cualquier tipo de fichero en nuestro documento: fichero de sonido, de video, gráfico,etc.

El browser tendrá que tener la capacidad de leer el tipo de fichero especificado o ayudarse de algún otro programa externo que le permita visualizar el fichero. En caso contrario dará un mensaje de error informándonos de que no puede leer el fichero especificado.

Con el argumento src indicamos la URL del fichero a mostrar. Con los argumentos width y height indicaremos el tamaño del objeto insertado, estos argumentos son opcionales.

**2.6.9 Capas DIV**

****

M1\_img\_65

Representa un área( capa) donde vamos a poner cualquier elemento HTML con el fin de posteriormente poder posicionarlo y darle tamaño dentro de la página con CSS.

El formato de esta etiqueta es:

<DIV>

Elementos HTML

</DIV>

Como atributos de <DIV> podemos indicar el ancho y alto con width y height y también el color de fondo de la capa, con atributos CSS de la propiedad style.

Ejemplo:

<DIV style="background-color:orange;width:200px;height:200px">

Hoy es Lunes

</DIV>



M1\_img\_66

**2.6.10 Mapas sensibles**

Los mapas sensibles consisten en utilizar una imagen y dividirla en zonas sensibles de tal modo que al hacer click sobre una determinada zona de la imagen podemos saltar a otra página web que tenga que ver con la zona donde clickamos.

La etiqueta correspondiente es <MAP>

<MAP name=”#nombre\_mapa”>

Definición de las áreas sensibles.

<AREA SHAPE=”opciones” HREF=”URL donde iremos al hacer click” >

</MAP>

Tenemos las siguientes zonas sensibles como opciones del atributo SHAPE:

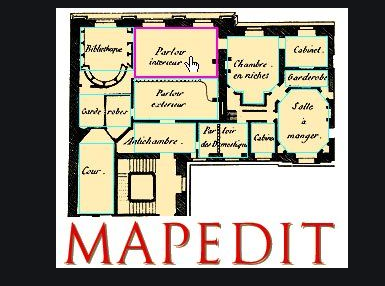
* shape=“rect” : Esta opción crea un mapa de imagen rectangular. Para definir la zona que incluirá ese mapa deberemos definir la esquina superior izquierda del área y la esquina inferior derecha.
* shape=“circle” : Esta opción de área crea un mapa de imagen circular. Nosotros sólo debemos indicarle el centro de la circunferencia y el radio del mismo.
* shape=“poly” : Definimos los diferentes puntos del polígono que hagamos y de una forma ordenada, siguiendo el camino que nosotros hemos trazado para hacerlo.

La imagen la definimos con una etiqueta <IMG SRC=”nombre de la imagen del mapa”

USEMAP=”nombre\_mapa”>

Nombre\_mapa está relacionado con el atributo name de la etiqueta <MAP>.

Existen utilidades en internet para crear las zonas de manera automática como por ejemplo MAPEDIT, que se puede descargar desde internet.



M1\_img\_67