Sumario

Colección de apuntes de Java EE

No poseo los derechos de autor de todo el material expuesto en esta colección de apuntes. Utilice innumerables fuentes de datos. Algunas partes las escribí o las modifique para adaptarlas a las versiones Java EE 8.

Muchas gracias a todas las fuentes de datos que enriquecieron este material de estudio no comercial.

Internet

Internet es un conjunto descentralizado de redes de interconectadas que utilizan la familia de protocolos TCP/IP para conectarse, lo cual garantiza que las redes físicas heterogéneas que la componen formen una red lógica única de alcance mundial. Sus orígenes se remontan a 1969, cuando se estableció la primera conexión de computadoras, conocida como ARPANET, entre tres universidades en California (Estados Unidos).
Uno de los servicios más exitosos de internet ha sido la World Wide Web (WWW o la Web), hasta tal punto que es habitual la confusión entre ambos términos.
La WWW es un conjunto de protocolos que permite, de forma sencilla, la consulta remota de archivos de hipertexto. Esta fue un desarrollo posterior (1990) y utiliza internet como medio de transmisión.
Existen muchos otros servicios y protocolos en internet, aparte de la Web: el envío de correo electrónico SMTP), la transmisión de archivos (FTP), las conversaciones en línea (IRC), la mensajería instantánea y presencia, la transmisión de contenido y comunicación multimedia —telefonía (VoIP), televisión (IPTV)—, los boletines electrónicos (NNTP), el acceso remoto a otros dispositivos (SSH y Telnet) o los juegos en línea.
En los últimos 20, el uso de internet se ha multiplicado por 100, la mayoría de las industrias de comunicación, incluyendo telefonía, radio, televisión, correo postal y periódicos tradicionales están siendo transformadas o redefinidas por el Internet, permitiendo el nacimiento de nuevos servicios como correo electrónico (email), telefonía por internet, televisión por Internet, música digital, y video digital. Las industrias de publicación de periódicos, libros y otros medios impresos se están adaptando a la tecnología de los sitios web, o están siendo reconvertidos en blogs, web feeds o agregadores de noticias online (p. ej., Google Noticias). Internet también ha permitido o acelerado nuevas formas de interacción personal a través de mensajería instantánea, foros de Internet, y redes sociales como Facebook. El comercio electrónico ha crecido exponencialmente para tanto grandes cadenas como para pequeños y mediana empresa o nuevos emprendedores, ya que permite servir a mercados más grandes y vender productos y servicios completamente en línea.
Relaciones business-to-business y de servicios financieros en línea en internet han afectado las cadenas de suministro de industrias completas.

Arquitectura CLIENTE/SERVIDOR

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras. La interacción cliente-servidor es el soporte de la mayor parte de la comunicación por redes.
El servidor debe negociar con su Sistema Operativo un puerto (casi siempre bien conocido) donde esperar las solicitudes. El servidor espera pasivamente las peticiones en un puerto bien conocido que ha sido reservado para el servicio que ofrece. El cliente también solicita, a su sistema operativo, un puerto no usado desde el cual enviar su solicitud y esperar respuesta. Un cliente ubica un puerto arbitrario, no utilizado y no reservado, para su comunicación.
En una interacción se necesita reservar solo uno de los dos puertos, asignados un identificador único de puerto para cada servicio, se facilita la construcción de clientes y servidores.
Los servidores por lo general son más difíciles de construir que los clientes pues aunque se implantan como programas de aplicación deben manejar peticiones concurrentes, así como reforzar todos los procedimientos de acceso y protección del sistema computacional en el que corren, y protegerse contra todos los errores posibles. El cliente y el servidor pueden interactuar en la misma máquina. En esta arquitectura la computadora de cada uno de los usuarios, llamada cliente, produce una demanda de información a cualquiera de las computadoras que proporcionan información, conocidas como servidores estos últimos responden a la demanda del cliente que la produjo.
Los clientes y los servidores pueden estar conectados a una red local o una red amplia, como la que se puede implementar en una empresa o a una red mundial como lo es la Internet.
Bajo este modelo cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o distantes y de procesarla como según le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura.

Partes de un sistema Cliente / Servidor

Cliente: Programa que participa activamente en el establecimiento de las conexiones. Envía una petición al servidor y se queda esperando por una respuesta. Su tiempo de vida es finito una vez que son servidas sus solicitudes, termina el trabajo.

Servidor: Es un programa que ofrece un servicio que se puede obtener en una red. Acepta la petición desde la red, realiza el servicio y devuelve el resultado al solicitante. Al ser posible implantarlo como aplicaciones de programas, puede ejecutarse en cualquier sistema donde exista TCP/IP y junto con otros programas de aplicación. El servidor comienza su ejecución antes de comenzar la interacción con el cliente. Su tiempo de vida o de interacción es “interminable”.
Los servidores pueden ejecutar tareas sencillas o complejas. Los servidores sencillos procesan una petición a la vez (son secuenciales o interactivos), por lo que no revisan si ha llegado otra petición antes de enviar la respuesta de la anterior. Los más complejos trabajan con peticiones concurrentes aún cuando una sola petición lleve mucho tiempo para ser servida. Son complejos pues tienen altos requerimientos de protección y autorización. Pueden leer archivos del sistema, mantenerse en línea y acceder a datos protegidos y a archivos de usuarios.

Características Arquitectura Cliente-Servidor

● Combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos a compartir. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema.

● Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidades del disco e input-output devices.

● Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.

● Existe una clara distinción de funciones basadas en el concepto de"servicio", que se establece entre clientes y servidores.

● La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a los recursos compartidos.

● No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicios.

● El ambiente es heterogéneo. La plataforma de hardware y el sistema operativo del cliente y del servidor no son siempre los mismos. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.

● El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema Cliente-Servidor. La escalabilidad horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

Ventajas del Esquema Cliente-Servidor

● Facilita la integración entre sistemas diferentes y comparte información, permitiendo por ejemplo que las máquinas ya existentes puedan ser utilizadas pero utilizando interfaces más amigables el usuario. De esta manera, se puede integrar PCs con sistemas medianos y grandes, sin necesidad de que todos tengan que utilizar el mismo sistema operativo.

● Al favorecer el uso de interfaces gráficas interactivas, los sistemas construidos bajo este esquema tienen una mayor y más intuitiva con el usuario. En el uso de interfaces gráficas para el usuario, presenta la ventaja, con respecto a uno centralizado, de que no siempre es necesario transmitir información gráfica por la red pues esta puede residir en el cliente, lo cual permite aprovechar mejor el ancho de banda de la red.

● La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.

Desventajas del Esquema Cliente-Servidor

● El mantenimiento de los sistemas es más difícil pues implica la interacción de diferentes partes de hardware y de software, distribuidas por distintos proveedores, lo cual dificulta el diagnóstico de fallas.

● Cuenta con muy escasas herramientas para la administración y ajuste del desempeño de los sistemas.

● Es importante que los clientes y los servidores utilicen el mismo mecanismo (por ejemplo sockets o RPC), lo cual implica que se deben tener mecanismos generales que existan en diferentes plataformas.

● Hay que tener estrategias para el manejo de errores y para mantener la consistencia de los datos.

● El desempeño (performance), problemas de este estilo pueden presentarse por congestión en la red, dificultad de tráfico de datos, etc.

Conceptos Generales

Protocolo TCP/IP

TCP/IP Son las siglas de Protocolo de Control de Transmisión/Protocolo de Internet (en inglés Transmission Control Protocol/Internet Protocol), un sistema de protocolos que hacen posibles servicios Telnet, FTP, E-mail, y otros entre computadores que no pertenecen a la misma red.
El Protocolo de Control de Transmisión (TCP) permite a dos anfitriones (HOSTs) establecer una conexión e intercambiar datos. El TCP garantiza la entrega de datos, es decir, que los datos no se pierdan durante la transmisión y también garantiza que los paquetes sean entregados en el mismo orden en el cual fueron enviados.
El Protocolo de Internet (IP) utiliza direcciones que son series de cuatro números octetos (byte) con un formato de punto decimal, por ejemplo: 69.5.163.59 .
Los Protocolos de Aplicación como HTTP y FTP se basan y utilizan TCP/IP.

Puertos IP

En TCP/IP, el protocolo que usan los computadores para entenderse en Internet y actualmente casi en cualquier otra red, el puerto es una numeración lógica que se asigna a las conexiones, tanto en el origen como en el destino. No tiene ninguna significación física. El permitir o denegar acceso a los puertos es importante porque las aplicaciones servidoras (que aceptan conexiones originadas en otro computador) deben 'escuchar' en un puerto conocido de antemano para que un cliente (que inicia la conexión) pueda conectarse. Esto quiere decir que cuando el sistema operativo recibe una petición a ese puerto, la pasa a la aplicación que escucha en él, si hay alguna, y a ninguna otra. Los servicios más habituales tienen asignados los llamados puertos bien conocidos, por ejemplo el 80 para web, el 21 para ftp, el 23 para telnet, etc. Así pues, cuando se pide una página web, el navegador realiza una conexión al puerto 80 del servidor web, y si este número de puerto no se supiera de antemano o estuviera bloqueado no podría recibir la página.

Un puerto puede estar:

- Abierto: Acepta conexiones. Hay una aplicación escuchando en este puerto. Esto no quiere decir que se tenga acceso a la aplicación, sólo que hay posibilidad de conectarse.

- Cerrado: Se rechaza la conexión. Probablemente no hay aplicación escuchando en este puerto, o no se permite el acceso por alguna razón. Este es el comportamiento normal del sistema operativo.
- Bloqueado o Sigiloso: No hay respuesta. Este es el estado ideal para un cliente en Internet, de esta forma ni siquiera se sabe si el ordenador está conectado. Normalmente este comportamiento se debe a un firewall.
- En una URL (Universal Resource Locator) los puertos se denotan con ':' a continuación del nombre de la máquina, por ejemplo http://www.facebook.com:80 quiere decir pedir el documento 'index.html' mediante http conectándose al puerto 80 de este servidor. Como 80 es el como es el puerto por defecto para http se puede omitir.

HTTP

Que es un protocolo?

Un protocolo es un reglamento o una serie de instrucciones que se fijan por tradición o por convenio. Partiendo de este significado, es posible emplear la noción en diferentes contextos. Un protocolo puede ser un documento o una normativa que establece cómo se debe actuar en ciertos procedimientos. De este modo, recopila conductas, acciones y técnicas que se consideran adecuadas ante ciertas situaciones.

Que es un protocolo de red ?

Protocolo es el término que se emplea para denominar al conjunto de normas, reglas y pautas que sirven para guiar una conducta o acción. Red, por su parte, es una clase de estructura o sistema que cuenta con un patrón determinado.
El concepto de protocolo de red se utiliza en el contexto de la informática para nombrar a las normativas y los criterios que fijan cómo deben comunicarse los diversos componentes de un cierto sistema de interconexión. Esto quiere decir que, a través de este protocolo, los dispositivos que se conectan en red pueden intercambiar datos.
También conocido como protocolo de comunicación, el protocolo de red establece la semántica y la sintaxis del intercambio de información, algo que constituye un estándar. Las computadoras en red, de este modo, tienen que actuar de acuerdo a los parámetros y los criterios establecidos por el protocolo en cuestión para lograr comunicarse entre sí y para recuperar datos que, por algún motivo, no hayan llegado a destino.

El protocolo HTTP

El protocolo HTTP (HyperText Transfer Protocol) es el protocolo más importante que debemos conocer como desarrolladores Web. Fue diseñado para transmitir HTML (HyperText Markup Language) pero hoy en día se utiliza para transmitir todo tipo de documentos (imágenes, audio, video, PDF, etc.)
Un protocolo es un documento que define las reglas y la estructura de los mensajes que se van a intercambiar entre máquinas. Además de HTTP, ejemplos de protocolos incluyen SMTP (Simple Mail Transfer Protocol) para transmitir mensajes de correo electrónico y FTP (File Transfer Protocolo) para transmitir archivos. Sin embargo, cualquier persona puede crear un protocolo. Skype, por ejemplo, desarrolló su propio protocolo de comunicación peer-to-peer.
HTTP es un protocolo cliente-servidor, lo que significa que el cliente envía una petición al servidor y espera un mensaje de respuesta del servidor. Es un protocolo sin estado, lo que significa que el servidor no guarda información del cliente, cada petición es independiente de las demás.

Mensajes HTTP

Un mensaje HTTP (no importa si es de petición o respuesta) se compone de 3 partes:

- La primera línea (que es diferente para la petición y la respuesta).
- Los encabezados.
- El cuerpo (opcional)

Request

```
GET /index.html HTTP/1.1
Host: wikipedia.org
Accept: text/html
```

En el ejemplo anterior se solicita el recurso /index.html de wikipedia.org. La primera línea del mensaje de petición se compone de:

- El verbo (en este caso GET)
- El recurso (en este caso /index)
- La versión de HTTP (en este caso HTTP/1.1)

Y la respuesta sería:

Response

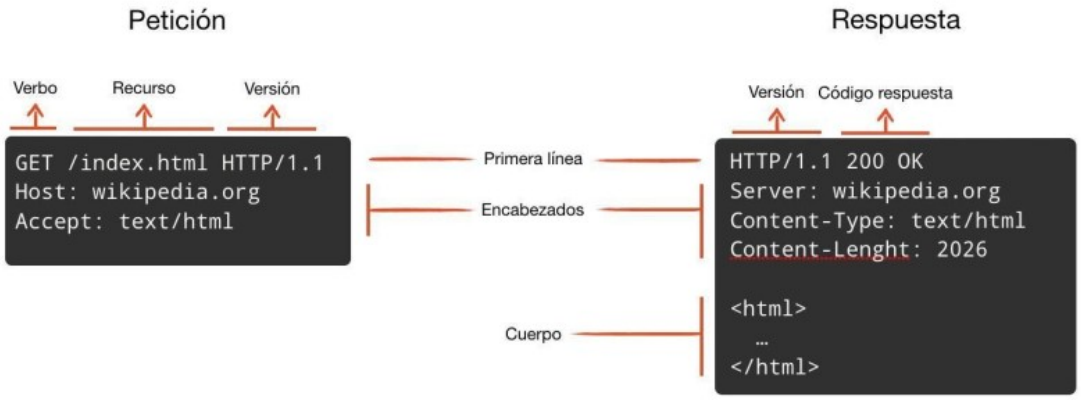
```
HTTP/1.1 200 OK
Server: wikipedia.org
Content-Type: text/html
Content-Lenght: 2026

<html>
  …
</html>
```

La primera línea del mensaje de respuesta se compone de:

- La versión de HTTP (HTTP/1.1)
- El código de respuesta (200 OK)- Cabeceras del mensaje
- Contenido de la respuesta

La siguiente imagen muestra mejor las partes de cada mensaje:



Clasificación de las peticiones HTTP

Estas peticiones las podemos clasificar en Safe e Idempotent:

Peticiones HTTP *Safe*

Un método HTTP es considerado safe o seguro si no altera el estado del servidor. En otras palabras, un método es seguro si conduce a una operación de 'sólo lectura'. Algunos de los métodos HTTP más comunes son seguros: OPTIONS, GET o HEAD. Todos los métodos seguros son también a su vez idempotent (así como también algunos, pero no todos, los métodos inseguros como DELETE o PUT).

Incluso si los métodos seguro tienen una semántica de sólo lectura, los servidores pueden alterar su estado, como por ejemplo: pueden registrar o mantener estadísticas. Lo importante es que de esta forma los usuarios pueden hacer uso de los métodos seguros sin preocuparse de que se realicen cambios en el servidor o de crearle cargas innecesarias a este último. Es por ello también que los navegadores hacen uso de estos métodos al realizar operaciones tales como la de anticiparse a otras operaciones (pre-fetching) sin temor a causar algún problema o riesgo.

Peticiones HTTP *Idempotent*

Así como un objeto cualquiera tiene la propiedad de idempotencia si al realizar una operación muchas veces da el mismo resultado cual si se hubiese realizado la operación una sola vez, un método HTTP es idempotente si una solicitud idéntica puede realizarse una o demasiadas veces consecutivamente obteniendo el mismo resultado dejando al servidor en el mismo estado.

En la vida y en los números reales, podríamos llamar al 1 y al 0 como los únicos idempotentes para la operación de multiplicación, pues estos números al multiplicarse en muchas ocasiones, da como resultado él mismo (1^n = 1). Volviendo a HTTP, los métodos que (implementados correctamente) son idempotentes son el GET, HEAD, PUT y DELETE y como lo explicamos en los métodos Safe, todos éstos últimos son idempotentes.

GET /index.html HTTP/1.2 es idempotente, pues si lo llamamos el cualquier número de ocasiones de forma consecutiva, el cliente recibirá el mismo resultado:

```
1 GET /index.html HTTP/1.1
2 GET /index.html HTTP/1.1
3 GET /index.html HTTP/1.1
4 GET /index.html HTTP/1.1
```

El verbo HTTP

La primera línea de un mensaje de petición empieza con un verbo (también se le conoce como método). Los métodos definen la acción que se quiere realizar sobre el recurso. Los más comunes son:

- Get
- Options
- Head
- Put
- Post
- Delete
- Connect
- Trace

Existen otros pero estos son los más comunes. Cuando ingresas a una página desde un navegador, por debajo el navegador envía un mensaje GET, lo mismo cuando oprimes un vínculo a otra página.

Peticiones HTTP: GET

El método GET significa recuperar cualquier información (en forma de una entidad) identificada por el Request-URI. Si el Request-URI se refiere a un proceso de producción de datos, son los datos producidos los que se devolverán como entidad en la respuesta y no el texto fuente del proceso. Cabe señalar que todas las peticiones que usan el método GET, deberán recuperar únicamente datos.

Ejemplo:

```
GET /index.html HTTP/1.1
User-Agent: Mozilla/4.0
(compatible; MSIE5.01;
Windows NT)
Host:
www.michelletorres.mx
Accept-Language: es-mx
Accept-Encoding: gzip,
deflate
Connection: Keep-Alive
```

A lo que el servidor respondería algo parecido a:

```
Ejemplo respuesta del
servidor:
HTTP/1.1 200 OK
Date: Wed, 08 Nov 2017
12:28:53 GMT
Server: Apache/2.2.14
(Win32)
Last-Modified: Mon, 22 Jul
2014 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Vary: Authorization,Accept
Accept-Ranges: bytes
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

```
<html>
<body>
<h1>Howdy, Michelle!</h1>
</body>
</html>
```

Peticiones HTTP: OPTIONS

El método OPTIONS representa una solicitud de información acerca de las opciones de comunicación disponibles en el canal de solicitud/respuesta identificada por el Request-URI. En otras palabras, éste método es el que utilizamos para describir las opciones de comunicación existentes de un recurso destino. ***Dato:*** *El cliente como tal puede especificar una URL para este método o, en su lugar, utilizar un asterisco para referirse al servidor completo.*

Ejemplo:

Se necesita saber cuáles métodos de solicitud soporta el servidor de nuestra profesora, podemos utilizar curl y una solicitud OPTIONS:

```
curl -X OPTIONS http://michelletorres.mx -i
```

Para lo cual el servidor podría contestar algo como lo siguiente:

```
HTTP/1.1 200 OK
Date: Wed, 8 Nov 2017 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Type: httpd/unix-directory
```

Peticiones HTTP: HEAD

El método HEAD es muy similar al GET (funcionalmente hablando), a excepción de que el servidor responde con líneas y headers, pero no con el body de la respuesta.

Ejemplo:

```
GET /index.html HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows
NT)
Host: www.michelletorres.mx
Accept-Language: es-mx
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

A lo que el servidor nos respondería algo como:

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Vary: Authorization,Accept
Accept-Ranges: bytes
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

Peticiones HTTP: PUT

El método PUT es usado para solicitar que el servidor almacene el cuerpo de la entidad en una ubicación específica dada por el URL.

Ejemplo:

Se solicita al servidor que guarde el cuerpo de la entidad dada en index.htm en la raíz del servidor:

```
PUT /index.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.michelletorres.mx
Accept-Language: es-mx
Connection: Keep-Alive
Content-type: text/html
Content-Length: 182
```

```
<html>
<body>
<h1>Howdy, Michelle!</h1>
</body>
</html>
```

Y el servidor por su parte, responde con lo siguiente al cliente (habiendo ya guardado el cuerpo de la entidad en el archivo index.htm):

```
HTTP/1.1 201 Created
Date: Mon, 27 Nov 2017 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Content-type: text/html
Content-length: 30
Connection: Closed
```

Peticiones HTTP: POST

El método POST es usado cuando se requiere enviar información al servidor como, por ejemplo, un archivo de actualización, información de formulario, etc. En otras palabras, éste método se usa cuando se necesita enviar una entidad para algún recurso determinado. La diferencia entre PUT y POST es que PUT es idempotente, mientras que si realizamos repetidas idénticas peticiones con el método POST, podría haber efectos adicionales como pasar una orden varias ocasiones.

Ejemplo: Se enviará información de formulario al servidor, que será procesada por un process.cgi:

```
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.michelletorres.mx
Content-Type: text/xml; charset=utf-8
Content-Length: 88
Accept-Language: es-mx
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

Peticiones HTTP: DELETE

Este método es utilizado para solicitar al servidor que elimine un archivo en una ubicación específica dada por la URL. En otras palabras más simples, este método elimina un recurso determinado.

Ejemplo: Se le solicitará al servidor eliminar el archivo hello.htm en la ruta raíz del servidor:

```
DELETE /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.michelletorres.mx
Accept-Language: es-mx
Connection: Keep-Alive
```

El servidor por su parte responderá eliminando dicho archivo y respondiendo al cliente lo siguiente:

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Content-type: text/html
Content-length: 30
Connection: Closed

<html>
<body>
<h1>URL deleted.</h1>
</body>
```

Peticiones HTTP: CONNECT

Este método por su parte es usado por el cliente para establecer una conexión de red con un servidor web mediante HTTP misma que se establece en forma de un túnel.

Ejemplo: Se quiere establecer conexión con un servidor corriendo en michelletorres.mx:

```
CONNECT www.michelletorres.mx HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
```

Una vez establecida la conexión, se regresa al cliente una respuesta similar a:

```
HTTP/1.1 200 Connection established
Date: Mon, 27 Nov 2017 02:22:53 GMT
Server: Apache/2.2.14 (Win32)
```

Peticiones HTTP: TRACE

Éste método se utiliza para realizar pruebas de eco (de retornos) de mensajes en el camino que existe hacia un recurso determinado. Es un método muy utilizado para la depuración y también para el desarrollo.

Ejemplo:

```
TRACE / HTTP/1.1
Host: www.michelletorres.mx
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
```

Utilizando este método como se muestra en la parte anterior, obtendríamos una respuesta del servidor como la siguiente:

```
HTTP/1.1 200 OK
Date: Mon, 27 Nov 2017 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Connection: close
Content-Type: message/http
Content-Length: 39

TRACE / HTTP/1.1
Host: www.michelletorres.mx
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
```

El código de respuesta

La primera línea de un mensaje de respuesta tiene un código de 3 dígitos que le indica al cliente cómo interpretar la respuesta. Los códigos de respuesta se dividen en cinco categorías dependiendo del dígito con el que inician:

1XX: Información
2XX: Éxito
3XX: Redirección
4XX: Error en el cliente
5XX: Error en el servidor

Las Cabeceras

Las cabeceras brindan información adicional sobre la petición o la respuesta.
Los encabezados tienen la siguiente sintaxis:

```
[nombre del encabezado]: [valor del encabezado]
```

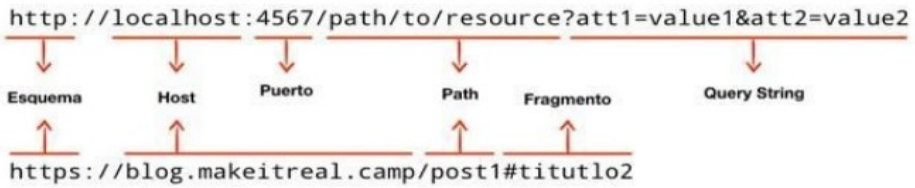
Las cabeceras comunes incluyen:
- Content-Type: Tipo de contenido que se está enviando en el cuerpo de un request, por ejemplo text/html.
- Accept: Tipo de contenido que el cliente está esperando.
- User-Agent: Tipo de navegador que está haciendo la petición

URL

Una URL (Uniform Resource Locator) se utiliza para ubicar un recurso en Internet.

Los URLs no solo se pueden utilizar para el protocolo HTTP, se utilizan en muchos otros protocolos.

El siguiente ejemplo muestra las partes de un URL utilizando dos ejemplos:



- Esquema: El esquema define el protocolo a utilizar, para HTTP puede ser http o https.
- Host: La IP o el nombre del servidor que se quiere acceder (p.e. 127.0.0.1, localhost, google.com, www.google.com.co, etc.)
- Puerto: El puerto en el que está escuchando el servidor HTTP. Si se omite se asume que es el 80.
- Path: La ruta al recurso que se quiere acceder.
- Query String: Contiene información adicional para el servidor en forma de propiedades (atributo=valor). Las propiedades se separan por &.
- Fragmento: La referencia a una ubicación interna del documento.

Sitios Web

Un sitio web (web site) es un sitio en la World Wide Web que contiene documentos (páginas web) organizados jerárquicamente. Cada documento contiene texto y/o gráficos que aparecen como información digital en la pantalla de un dispositivo. Cada sitio web tiene una página de inicio (Home Page / Index), que es el primer documento que ve el usuario cuando entra en el sitio web poniendo el nombre del dominio de ese sitio web en un navegador. A veces se utiliza erróneamente el término página web para referirse a sitio web. Una página web es parte de un sitio web y es un único archivo con un nombre de archivo asignado, mientras que un sitio web es un conjunto de archivos llamados páginas web.

¿Qué es una página web?

Una página web es un documento adaptado para la Web y que normalmente forma parte de un sitio web. Su principal característica son los hiperenlaces a otras páginas, siendo esto el fundamento de

la Web. Dentro de una página web, como documento que es, podemos encontrar diferentes tipos de contenidos: textos, imágenes, enlaces, plugins, etc.

¿Qué es una página web estáticas?

Una Web estática es aquella página enfocada principalmente a mostrar una información permanente, donde el navegante se limita a obtener dicha información, sin poder interactuar con la página web visitada.

¿Qué es una página web dinámica?

Una web dinámica es aquella que contiene aplicaciones dentro de la propia web, otorgando mayor interactividad con el cliente. Ejemplos de aplicaciones dinámicas son encuestas y votaciones, foros de soporte, libros de visita, envío de e-mails inteligentes, reserva de productos, pedidos on-line, atención al cliente personalizada. Es importante no confundir multimedia e interactividad, con páginas dinámicas. Una página web estática puede ser multimedia (contener varios tipos de medios vídeo, sonido, imágenes...) e interactiva a través de enlaces e hipervínculos, sin ser dinámica por ella. En las páginas dinámicas, el contenido suelen generarse en el momento de visualizarse, pudiendo variar por tanto este, mientras que en las estáticas el contenido suele estar predeterminado. Lo importante de esta clasificación entre dinámicas y estáticas, es que una página web estática la vamos a poder almacenar fácilmente, mientras que una dinámica no será así.

Extensiones de páginas web

Como se mencionó las páginas web son documentos y, como tales, suelen tener una extensión que nos da una idea sobre del tipo de contenido que posee ese documento. Las extensiones más habituales de las páginas web con las que nos podemos encontrar a la hora de navegar son:

● html, y htm: Suele ser la extensión estándar para páginas web estáticas, formadas por contenido html.

● aspx (Active server pages): Son páginas dinámicas que se generan en el momento de ser visitadas. Están escritas framework de programación de Microsoft.

● jsp (Java server pages): Son páginas dinámicas al igual que las anteriores pero que están escritas con el lenguaje de programación Java de Oracle.

● php (Personal Home Page): Son páginas dinámicas escritas con un lenguaje de propósito general para ser incrustado junto al código HTML.

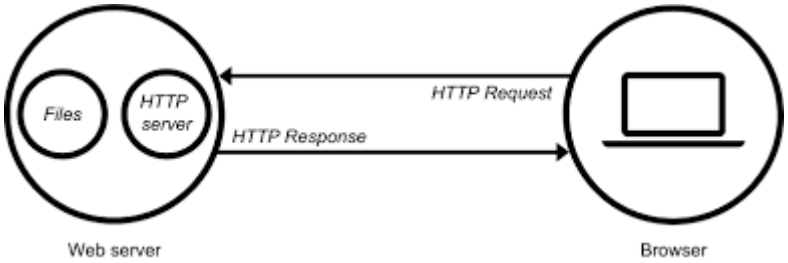
Si bien todos los documentos con las extensiones anteriores son páginas web, únicamente los .html o .html son páginas web estáticas, y por tanto las únicas que podremos ver directamente con el navegador. Las páginas .aspx, .jsp o .php, para poder visualizarse deberán estar contenidas en un servidor web.

Web Server

Cuando hablamos de un servidor web, podemos referirnos a un hardware o un software, o a ambos trabajando juntos. En cuanto a hardware, un servidor web es una computadora que almacena los archivos que componen un sitio web (ej. documentosHTML , imágenes, hojas de estilos CSS y archivo JavaScript) y los entrega al dispositivo del usuario final. Está conectado a internet y es accesible a través de un nombre de dominio.

En cuanto a software, un servidor web tiene muchas partes encargadas del control sobre cómo tienen acceso los usuarios a los archivos, por lo menos un servidor HTTP. Un servidor HTTP es una pieza de software que comprende URLs y HTTP.
Al nivel más básico, siempre que un navegador necesite un archivo almacenado en un servidor web, el navegador lo solicita vía HTTP. Cuando la petición llega al servidor web correcto (hardware), el servidor HTTP (software) envía el archivo antes solicitado, también a través de HTTP.

Funcionamiento de un Web Server



El Servidor web se ejecuta en un ordenador manteniéndose a la espera de peticiones por parte de un cliente (un navegador web) y responde a estas peticiones adecuadamente, mediante una *página web* que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error. A modo de ejemplo, al teclear *www.wikipedia.org* en nuestro navegador, éste realiza una petición HTTP al servidor de dicha dirección. El servidor responde al cliente enviando el código HTML de la página; el cliente, una vez recibido el código, lo interpreta y lo exhibe en pantalla. Como vemos con este ejemplo, el cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página; el servidor tan sólo se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma.

Además de la transferencia de código HTML, los Servidores web pueden entregar aplicaciones web. Estas son porciones de código que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP. Hay que distinguir entre:

- Aplicaciones en el lado del cliente: el cliente web es el encargado de ejecutarlas en la máquina del usuario. Son las aplicaciones tipo Java "applets" o Javascript: el servidor proporciona el código de las aplicaciones al cliente y éste, mediante el navegador, las ejecuta. Es necesario, por tanto, que el cliente disponga de un navegador con capacidad para ejecutar aplicaciones (también llamadas *scripts*). Comúnmente, los navegadores permiten ejecutar aplicaciones escritas en lenguaje *javascript* y *java*, aunque pueden añadirse más lenguajes mediante el uso de *plugin*s.

- Aplicaciones en el lado del servidor: el servidor web ejecuta la aplicación; ésta, una vez ejecutada, genera cierto código HTML; el servidor toma este código recién creado y lo envía al cliente por medio del protocolo HTTP.

Las aplicaciones de servidor muchas veces suelen ser la mejor opción para realizar aplicaciones web. La razón es que, al ejecutarse ésta en el servidor y no en la máquina del cliente, éste no necesita ninguna capacidad añadida, como sí ocurre en el caso de querer ejecutar aplicaciones

javascript o java. Así pues, cualquier cliente dotado de un navegador web básico puede utilizar este tipo de aplicaciones.

El hecho de que HTTP y HTML estén íntimamente ligados no debe dar lugar a confundir ambos términos. HTML es un lenguaje de marcas y HTTP es un "protocolo".

Apache HTTP Server

El servidor Apache HTTP, también llamado Apache, es un servidor web HTTP de código abierto para la creación de páginas y servicios web. Es un servidor multiplataforma, gratuito, muy robusto y que destaca por su seguridad y rendimiento.
Para entender mejor lo que es Apache, primeramente definiremos lo que es un servidor web. La definición más sencilla de servidor web, que es un programa especialmente diseñado para transferir datos de hipertexto, es decir, páginas web con todos sus elementos (textos, widgets, baners, etc). Estos servidores web utilizan el protocolo http. Los servidores web están alojados en un ordenador que cuenta con conexión a Internet. El web server, se encuentra a la espera de que algún navegador le haga alguna petición, como por ejemplo, acceder a una página web y responde a la petición, enviando código HTML mediante una transferencia de datos en red.

Ventajas
● Instalación/Configuración: Software de código abierto.
● Coste: El servidor web Apache es completamente gratuito.
● Funcional y Soporte: Alta aceptación en la red y muy popular, esto hace que muchos programadores de todo el mundo contribuyen constantemente con mejoras, que están disponibles para cualquier persona que use el servidor web y que Apache se actualice constantemente.
● Multi-plataforma: Se puede instalar en muchos sistemas operativos, es compatible con Windows, Linux y MacOS.
● Rendimiento: Capacidad de manejar más de un millón de visitas/día.
● Soporte de seguridad SSL y TLS.

Inconvenientes
● Falta de integración
● Posee formatos de configuración NO estándar.
● No posee un buen panel de configuración

Apache Tomcat

Apache Tomcat es un contenedor de servlets (Servlet Container) desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Oracle Corporation.
Tomcat es un contenedor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.
Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.
Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo

que disponga de la máquina virtual Java.

HTML

Introducción

HTML5 no es una nueva versión del antiguo lenguaje de etiquetas, ni siquiera una mejora de esta ya antigua tecnología, sino un nuevo concepto para la construcción de sitios web y aplicaciones en unos tiempos donde se combinan dispositivos móviles, computación en la nube y trabajos en red.
Todo comenzó mucho tiempo atrás con una simple versión de HTML propuesta para crear la estructura básica de páginas web, organizar su contenido y compartir información. El lenguaje y la web misma nacieron principalmente con la intención de comunicar información por medio de texto.
El limitado objetivo de HTML motivó a varias compañías a desarrollar nuevos lenguajes y programas para agregar características a la web nunca antes implementadas. Estos desarrollos iniciales crecieron hasta convertirse en populares y poderosos accesorios.
Simples juegos y bromas animadas pronto se transformaron en sofisticadas aplicaciones, ofreciendo nuevas experiencias que cambiaron el concepto de la web para siempre. De las opciones propuestas, Java y Flash fueron las más exitosas; ambas fueron masivamente adoptadas y ampliamente consideradas como el futuro de Internet. Sin embargo, tan pronto como el número de usuarios se incrementó e Internet pasó de ser una forma de conectar amantes de los ordenadores a un campo estratégico para los negocios y la interacción social, limitaciones presentes en estas dos tecnologías probaron ser una sentencia de muerte. El mayor inconveniente de Java y Flash puede describirse como una falta de integración. Ambos fueron concebidos desde el principio como complementos (plug-ins), algo que se inserta dentro de una estructura pero que comparte con la misma solo espacio en la pantalla.
No existía comunicación e integración alguna entre aplicaciones y documentos. La falta de integración resultó ser crítica y preparó el camino para la evolución de un lenguaje que comparte espacio en el documento con HTML y no está afectado por las limitaciones de los plug-ins.
Javascript, un lenguaje interpretado incluido en navegadores, claramente era la manera de mejorar la experiencia de los usuarios y proveer funcionalidad para la web. Sin embargo, después de algunos años de intentos fallidos para promoverlo y algunos malos usos, el mercado nunca lo adoptó plenamente y pronto su popularidad declinó. Los detractores tenían buenas razones para oponerse a su adopción. En ese momento, Javascript no era capaz de reemplazar la funcionalidad de Flash o Java. A pesar de ser evidente que ambos limitaban el alcance de las aplicaciones y aislaban el contenido web, populares funciones como la reproducción de video se estaban convirtiendo en una parte esencial de la web y solo eran efectivamente ofrecidas a través de estas tecnologías.
A pesar del suceso inicial, el uso de Java comenzó a declinar. La naturaleza compleja del lenguaje, su evolución lenta y la falta de integración disminuyeron su importancia hasta el punto en el que hoy día no es más usado en aplicaciones web de importancia. Sin Java, el mercado volcó su atención a Flash. Pero el hecho de que Flash comparte las mismas características básicas que su competidor en la web lo hace también susceptible de correr el mismo destino. Mientras esta competencia silenciosa se llevaba a cabo, el software para acceder a la web continuaba evolucionando. Junto con nuevas funciones y técnicas rápidas de acceso a la red, los navegadores también mejoraron gradualmente sus intérpretes Javascript. Más potencia trajo más oportunidades y este lenguaje estaba listo para aprovecharlas. En cierto punto durante este proceso, se hizo evidente para algunos desarrolladores que ni Java o Flash podrían proveer las herramientas que ellos necesitaban para crear las aplicaciones demandadas por un número creciente de usuarios. Estos desarrolladores, impulsados por las mejoras otorgadas por los navegadores, comenzaron a aplicar Javascript en sus aplicaciones de un modo nunca visto. La innovación y los increíbles resultados obtenidos llamaron la atención de más programadores. Pronto lo que fue llamado la “Web 2.0”

nació y la percepción de Javascript en la comunidad de programadores cambió radicalmente. Javascript era claramente el lenguaje que permitía a los desarrolladores innovar y hacer cosas que nadie había podido hacer antes en la web. En los últimos años, programadores y diseñadores web alrededor del mundo surgieron con los más increíbles trucos para superar las limitaciones de esta tecnología y sus iniciales deficiencias en portabilidad. Gracias a estas nuevas implementaciones, Javascript, HTML y CSS se convirtieron pronto en la más perfecta combinación para la necesaria evolución de la web. HTML5 es, de hecho, una mejora de esta combinación, el pegamento que une todo. HTML5 propone estándares para cada aspecto de la web y también un propósito claro para cada una de las tecnologías involucradas. A partir de ahora, HTML provee los elementos estructurales, CSS se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista, y Javascript tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales. Las barreras entre sitios webs y aplicaciones finalmente han desaparecido. Las tecnologías requeridas para el proceso de integración están listas. El futuro de la web es prometedor y la evolución y combinación de estas tres tecnologías (HTML, CSS y Javascript) en una poderosa especificación está volviendo a Internet la plataforma líder de desarrollo. HTML5 indica claramente el camino.

World Wide Web Consortium (W3C)

El **Consorcio WWW**, en inglés: ***World Wide Web Consortium*** (**W3C**), es un consorcio internacional que genera recomendaciones y estándares que aseguran el crecimiento de la *World Wide Web* a largo plazo.
Este consorcio fue creado en octubre de 1994, y está dirigido por Tim Berners-Lee, el creador original del URL (*Uniform Resource Locator*, Localizador Uniforme de Recursos), del HTTP (*HyperText Transfer Protocol*, Protocolo de Transferencia de HiperTexto) y del HTML (*Hyper Text Markup Language*, Lenguaje de Marcado de HiperTexto), que son las principales tecnologías sobre las que se basa la Web.
El objetivo del W3C es guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web. Debajo tratamos importantes aspectos de este objetivo, los cuales promueven la visión del W3C de **Web Única**.

Documentos HTML5

En la actualidad la mayoría de los navegadores de internet tienen soporte completo para HTML5

Componentes básicos

HTML5 provee básicamente tres características:
estructura, estilo y funcionalidad. Nunca fue declarado oficialmente pero, incluso cuando algunas APIs y la especificación de CSS3 por completo no son parte del mismo, HTML5 es considerado el producto de la combinación de HTML, CSS y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML5. HTML está a cargo de la estructura, CSS presenta esa estructura y su contenido en la pantalla y Javascript hace el resto que es extremadamente significativo. Más allá de esta integración, la estructura sigue siendo parte esencial de un documento. La misma provee los elementos necesarios para ubicar contenido estático o dinámico, y es también una plataforma básica para aplicaciones. Con la variedad de dispositivos para acceder a Internet y la diversidad de interfaces disponibles para interactuar con la web, un aspecto básico como la estructura se vuelve parte vital del documento. Ahora la estructura debe proveer forma, organización y flexibilidad, y debe ser tan fuerte como los fundamentos de un edificio.

Para trabajar y crear sitios webs y aplicaciones con HTML5, necesitamos saber primero cómo esa estructura es construida. Crear fundamentos fuertes nos ayudará más adelante a aplicar el resto de los componentes para aprovechar completamenteestas nuevas tecnologías. Por lo tanto, empecemos por lo básico, paso a paso. A partir de ahora comenzaremos a construir una plantilla para futuros proyectos usando los nuevos elementos HTML introducidos en HTML5.

Estructura global

El DOCTYPE, En primer lugar necesitamos indicar el tipo de documento que estamos creando. Esto en HTML5 se hace de la siguiente manera:

El elemento <!DOCTYPE html>

```
<!DOCTYPE html>
```

Esta línea debe ser la primera línea del archivo, sin espacios o líneas que la preceden. De esta forma, el modo estándar del navegador es activado y las incorporaciones de HTML5 son interpretadas siempre que sea posible, o ignoradas en caso contrario.

El elemento <html>

Luego de declarar el tipo de documento, debemos comenzar a construir la estructura HTML. Como siempre, la estructura tipo árbol de este lenguaje tiene su raíz en el elemento mismo mencionado:

```
<!DOCTYPE html>
<html lang=”en”>
</html>
```

El atributo lang en la etiqueta de apertura es el único atributo que necesitamos especificar en HTML5. Este atributo define el idioma humano del contenido del documento que estamos creando, en este caso es por ingles. HTML usa un lenguaje de etiquetas para construir páginas web. Estas etiquetas HTML son palabras clave y atributos rodeados de los signos mayor y menor. En este caso, html es la palabra clave y lang es el atributo con el valor (puede ser en o es o otro lenguaje). La mayoría de las etiquetas HTML se utilizan en pares, una etiqueta de apertura y una de cierre, y el contenido se declara entre ellas. En el ejemplo anterior, indica el comienzo del código HTML y indica el final. Las etiquetas de apertura y cierre se distinguen por una barra invertida antes de la palabra clave. El resto de nuestro código será insertado entre estas dos etiquetas.
HTML5 es extremadamente flexible en cuanto a la estructura y a los elementos utilizados para construirla. El elemento puede ser incluido sin ningún atributo o incluso ignorado completamente. Con el propósito de preservar compatibilidad se recomienda seguir las reglas básicas.

El elemento <head>

El código HTML insertado entre las etiquetas tiene que ser dividido entre dos secciones principales. Al igual que en versiones previas de HTML, la primera sección es la cabecera y la segunda el cuerpo. El siguiente paso, por lo tanto, será crear estas dos secciones en el código usando los elementos ya conocidos. El elemento va primero, por supuesto, y al igual que el resto de los elementos estructurales tiene una etiqueta de apertura y una de cierre:

```
<!DOCTYPE html>
<html lang="es">
      <head>
      </head>
</html>
```

La etiqueta no cambió desde versiones anteriores y su propósito sigue siendo exactamente el mismo. Dentro de las etiquetas definiremos el título de nuestra página web, declararemos el set de caracteres correspondiente, proveeremos información general acerca del documento e incorporaremos los archivos externos con estilos, códigos Javascript o incluso imágenes necesarias para generar la página en la pantalla. Excepto por el título y algunos íconos, el resto de la información incorporada en el documento entre estas etiquetas es invisible para el usuario. <body>La siguiente gran sección que es parte principal de la organización de un documento HTML es el cuerpo. El cuerpo representa la parte visible de todo documento y es especificado entre etiquetas . Estas etiquetas tampoco han cambiado en relación con versiones previas de HTML:

```
<!DOCTYPE html>
<html lang="es">
      <head>
      </head>
      <body>
      </body>
</html>
```

Hasta el momento tenemos un código simple pero con una estructura compleja. Esto es porque el código HTML no está formado por un conjunto de instrucciones secuenciales. HTML es un lenguaje de etiquetas, un listado de elementos que usualmente se utilizan en pares y que pueden ser anidados.

El elemento <meta>

Es momento de construir la cabecera del documento. Algunos cambios e innovaciones fueron incorporados dentro de la cabecera, y uno de ellos es la etiqueta que define el juego de caracteres a utilizar para mostrar el documento. Ésta es una etiqueta que especifica cómo el texto será presentado en pantalla:

```
<!DOCTYPE html>
<html lang="es">
      <head>
            <meta charset=”iso-8859-1”>
      </head>
      <body>
      </body>
</html>
```

La innovación de este elemento en HTML5, como en la mayoría de los casos, es solo simplificación. La nueva etiqueta para la definición del tipo de caracteres es más corta y simple. Por supuesto, podemos cambiar el tipo ISO-8859-1 por el necesario para nuestros documentos y agregar otras etiquetas como description o keywords para definir otros aspectos de la página web, como es mostrado en el siguiente ejemplo:

```
<!DOCTYPE html>
      <html lang="es">
      <head>
            <meta charset="iso-8859-1">
            <meta name=”description” content=”Ejemplo de HTML5”>
            <meta name=”keywords” content=”HTML5, CSS3, Javascript”>
      </head>
      <body>
      </body>
</html>
```

Hay varios tipos de etiqueta <meta> que pueden ser incluidas para declarar información general sobre el documento, pero esta información no es mostrada en la ventana del navegador, es solo importante para motores de búsqueda y dispositivos que necesitan hacer una vista previa del documento u obtener un sumario de la información que contiene. Como se comentó anteriormente, aparte del título y algunos íconos, la mayoría de la información insertada entre las etiquetas <head> no es visible para los usuarios.
En HTML5 no es necesario cerrar etiquetas simples con una barra al final. El anterior código se podría escribir de la siguiente manera:

```
<meta charset="iso-8859-1" />
<meta name="description" content="Ejemplo de HTML5" />
<meta name="keywords" content="HTML5, CSS3, JavaScript" /><title>
```

La etiqueta <title> simplemente especifica el título del documento:

```
<!DOCTYPE html>
<html lang="es">
      <head>
            <meta charset="iso-8859-1">
            <meta name="description" content="Ejemplo de HTML5">
            <meta name="keywords" content="HTML5, CSS3, JavaScript">
            <title>Este texto es el título del documento</title>
      </head>
      <body>
      </body>
</html>
```

El texto entre las etiquetas <title> es el título del documento que estamos creando. Normalmente este texto es mostrado en la barra superior de la ventana del navegador.

El elemento <link>

Otro importante elemento que va dentro de la cabecera del documento es <link>. Este elemento es usado para incorporar estilos, códigos Javascript, imágenes o iconos desde archivos externos. Uno de los usos más comunes para <link> es la incorporación de archivos con estilos CSS:

```
<!DOCTYPE html>
      <html lang="es">
      <head>
            <meta charset="iso-8859-1">
            <meta name="description" content="Ejemplo de HTML5">
            <meta name="keywords" content="HTML5, CSS3, JavaScript">
            <title>Este texto es el título del documento</title>
            <link rel=”stylesheet” href=”misestilos.css”>
      </head>
      <body>
      </body>
</html>
```

En HTML5 ya no se necesita especificar qué tipo de estilos estamos insertando, por lo que el atributo type fue eliminado. Solo necesitamos dos atributos para incorporar nuestro archivo de estilos: rel y href. El atributo rel significa “relación” y es acerca de la relación entre el documento y el archivo que estamos incorporando por medio de href.
En este caso, el atributo rel tiene el valor stylesheet que le dice al navegador que el archivo misestilos.css es un archivo CSS con estilos requeridos para presentar la página en pantalla.
Un archivo de estilos es un grupo de reglas de formato que ayudarán a cambiar la apariencia de nuestra página web (por ejemplo, el tamaño y color del texto). Sin estas reglas, el texto y cualquier

otro elemento HTML sería mostrado en pantalla utilizando los estilos estándar provistos por el navegador.
Los estilos son reglas simples que normalmente requieren sólo unas pocas líneas de código y pueden ser declarados en el mismo documento. Como veremos más adelante, no es estrictamente necesario obtener esta información de archivos externos pero es una práctica recomendada. Cargar las reglas CSS desde un documento externo (otro archivo) nos permitirá organizar el documento principal, incrementar la velocidad de carga y aprovechar las nuevas características de HTML5.

Estructura del cuerpo

La estructura del cuerpo generará la parte visible del documento. Este es el código que producirá nuestra página web. HTML siempre ofreció diferentes formas de construir y organizar la información dentro del cuerpo de un documento. Uno de los primeros elementos provistos para este propósito fue <table>. Las tablas permitían a acomodar datos, texto, imágenes y herramientas dentro de filas y columnas de celdas, incluso sin que hayan sido concebidas para este propósito.
En los primeros días de la web, las tablas fueron una revolución, un gran paso hacia adelante con respecto a la visualización de los documentos y la experiencia ofrecida a los usuarios. Más adelante, gradualmente, otros elementos reemplazaron su función, permitiendo lograr lo mismo con menos código, facilitando de este modo la creación, permitiendo portabilidad y ayudando al mantenimiento de los sitios web.
El elemento comenzó a dominar la escena. Con el surgimiento de webs más interactivas y la integración de HTML, CSS y Javascript, el uso de se volvió una práctica común. Pero este elemento, así como <table>, no provee demasiada información acerca de las parte del cuerpo que está representando. Desde imágenes a menús, textos, enlaces, códigos, formularios, cualquier cosa puede ir entre las etiquetas de apertura y cierre de un elemento . En otras palabras, la palabra clave div solo especifica una división en el cuerpo, como la celda de una tabla, pero no ofrece indicio alguno sobre qué clase de división es, cuál es su propósito o qué contiene.
Para los usuarios estas claves o indicios no son importantes, pero para los navegadores la correcta interpretación de qué hay dentro del documento que se está procesando puede ser crucial en muchos casos. Luego de la revolución de los dispositivos móviles y el surgimiento de diferentes formas en que la gente accede a la web, la identificación de cada parte del documento es una tarea que se ha vuelto más relevante que nunca.
En HTML5 las secciones más importantes son diferenciadas y la estructura principal ya no depende más de los elementos o <table>. Cómo usamos estos nuevos elementos depende de nosotros, pero las palabras clave otorgadas a cada uno de ellos nos dan ayudan a entender sus funciones.
Normalmente una página o aplicación web está dividida entre varias áreas visuales para mejorar la experiencia del usuario y facilitar la interactividad. Las palabras claves que representan cada nuevo elemento de HTML5 están íntimamente relacionadas con estas áreas.

Formularios Web

La Web 2.0 está completamente enfocada en el usuario. Y cuando el usuario es el centro de atención, todo está relacionado con interfaces, en cómo hacerlas más intuitivas, más naturales, más prácticas, y por supuesto más atractivas. Los formularios web son la interface más importante de todas, permiten a los usuarios insertar datos, tomar decisiones, comunicar información y cambiar el comportamiento de una aplicación. Durante los últimos años, códigos personalizados y librerías fueron creados para procesar formularios en el ordenador del usuario. HTML5 vuelve a estas funciones estándar agregando nuevos atributos, elementos y una API completa. Ahora la capacidad de procesamiento de información insertada en formularios en tiempo real ha sido incorporada en los navegadores y completamente estandarizada.

El elemento <form>

Los formularios en HTML no han cambiado mucho. La estructura sigue siendo la misma, pero HTML5 ha agregado nuevos elementos, tipos de campo y atributos para expandirlos tanto como sea necesario y proveer así las funciones actualmente implementadas en aplicaciones web.

```
<!DOCTYPE html>
	<html lang="es">
	<head>
		<title>Formularios</title>
	</head>
	<body>
		<section>
			<form name="miformulario" id="miformulario" method="get">
				<input type="text" name="nombre" id="nombre">
				<input type="submit" value="Enviar">
			</form>
		</section>
	</body>
</html>
```

En el ejemplo anterior creamos una plantilla básica para formularios. Como se puede ver, la estructura del formulario y sus atributos siguen siendo igual que en especificaciones previas. Sin embargo, existen nuevos atributos para el elemento <form>:
● autocomplete: Este es un viejo atributo que se ha vuelto estándar en esta especificación. Puede tomar dos valores: on y off. El valor por defecto es on. Cuando es configurado como off los elementos <input> pertenecientes a ese formulario tendrán la función de autocompletar desactivada, sin mostrar entradas previas como posibles valores. Puede ser implementado en el elemento <form> o en cualquier elemento <input> independientemente.
● novalidate: Una de las características de formularios en HTML5 es la capacidad propia de validación. Los formularios son automáticamente validados. Para evitar este comportamiento, podemos usar el atributo novalidate. Para lograr lo mismo para elementos <input> específicos, existe otro atributo llamado formnovalidate. Ambos atributos son booleanos, ningún valor tiene que ser especificado (su presencia es suficiente para activar su función).

El elemento <input>
El elemento más importante en un formulario es <input>. Este elemento puede cambiar sus características gracias al atributo type (tipo). Este atributo determina qué clase de entrada es esperada desde el usuario. Los tipos disponibles hasta el momento eran el multipropósitos text (para textos en general) y solo unos pocos más específicos como password o submit. HTML5 ha expandido las opciones incrementando de este modo las posibilidades para este elemento.
En HTML5 estos nuevos tipos no solo están especificando qué clase de entrada es esperada sino también diciéndole al navegador qué debe hacer con la información recibida. El navegador procesa los datos ingresados de acuerdo al valor del atributotype y validará la entrada o no. El atributo type trabaja junto con otros atributos adicionales para ayudar al navegador a limitar y controlar en tiempo real lo ingresado por el usuario.

Tipo email

Casi todo formulario en la web ofrece un campo para ingresar una dirección de email, pero hasta ahora el único tipo de campo disponible para esta clase de datos era text. El tipo text representa un texto general, no un dato específico, por lo que teníamos que controlar la entrada con código Javascript para estar seguros de que el texto ingresado correspondía a un email válido. Ahora el navegador se hace cargo de esto con el nuevo tipo email:

```
<input type="email" name="miemail" id="miemail">
```

El texto insertado en el campo generado por el código anterior será controlado por el navegador y validado como un email. Si la validación falla, el formulario no será enviado. Cómo cada navegador responderá a una entrada inválida no está determinado en la especificación de HTML5. Por ejemplo, algunos navegadores mostrarán un borde rojo alrededor del elemento <input> que produjo el error y otros lo mostrarán en azul. Existen formas de personalizar esta respuesta.

Tipo search

El tipo search (búsqueda) no controla la entrada, es solo una indicación para los navegadores. Al detectar este tipo de campo algunos navegadores cambiarán el diseño del elemento para ofrecer al usuario un indicio de su propósito.

```
<input type="search" name="busqueda" id="busqueda">
```

Tipo url

Este tipo de campo trabaja exactamente igual que el tipo email pero es específico para direcciones web. Está destinado a recibir solo URLs absolutas y retornará un error si el valor es inválido.

```
<input type="url" name="miurl" id="miurl">
```

Tipo tel

Este tipo de campo es para números telefónicos. A diferencia de los tipos email y url, el tipo tel no requiere ninguna sintaxis en particular. Es solo una indicación para el navegador en caso de que necesite hacer ajustes de acuerdo al dispositivo en el que la aplicación es ejecutada.

```
<input type="tel" name="telefono" id="telefono">
```

Tipo number

Como su nombre lo indica, el tipo number es sólo válido cuando recibe una entrada numérica. Existen algunos atributos nuevos que pueden ser útiles para este campo:

- min: El valor de este atributo determina el mínimo valor aceptado para el campo.
- max: El valor de este atributo determina el máximo valor aceptado para el campo.
- step: El valor de este atributo determina el tamaño en el que el valor será incrementado o disminuido en cada paso. Por ejemplo, si declara un valor de 5 para step en un campo que tiene un valor mínimo de 0 y máximo de 10, el navegador no le permitirá especificar valores entre 0 y 5 o entre 5 y 10.

```
<input type="number" name="numero" id="numero" min=”0” max=”10” step=”5”>
```

Tipo range

Este tipo de campo hace que el navegador construya una nueva clase de control que no existía previamente. Este nuevo control le permite al usuario seleccionar un valor a partir de una serie de valores o rango. Normalmente es mostrado en pantalla como una puntero deslizable o un campo con flechas para seleccionar un valor entre los predeterminados, pero no existe un diseño estándar hasta el momento.
El tipo range usa los atributos min y max estudiados previamente para configurar los límites del rango. También puede utilizar el atributo step para establecer el tamaño en el cual el valor del campo será incrementado o disminuido en cada paso.

```
<input type="range" name="numero" id="numero" min=”0” max=”10” step=”5”>
```

Tipo date

Este es otro tipo de campo que genera una nueva clase de control. En este caso fue incluido para ofrecer una mejor forma de ingresar una fecha. Algunos navegadores muestran en pantalla un calendario que aparece cada vez que el usuario hace clic sobre el campo. El calendario le permite al usuario seleccionar un día que será ingresado en el campo junto con el resto de la fecha. Un ejemplo de uso es cuando necesitamos proporcionar un método para seleccionar una fecha para un vuelo o la entrada a un espectáculo. Gracias al tipo date ahora es el navegador el que se encarga de construir un almanaque o las herramientas necesarias para facilitar el ingreso de este tipo de datos.

```
<input type="date" name="fecha" id="fecha">
```

La interface no fue declarada en la especificación. Cada navegador provee su propia interface y a veces adaptan el diseño al dispositivo en el cual la aplicación está siendo ejecutada. Normalmente el valor generado y esperado tiene la sintaxis año-mes-día.

Input Type Radio

<input type="radio">	Define un radio button.Input Type Checkbox
<input type="checkbox">	Define un checkbox.

Input Type Button

<input type="button">	Define un button.

CSS

introducción

CSS (*Hojas de Estilo en Cascada*) es el código que usas para dar estilo a tu página web. *CSS Básico* te lleva a través de lo que tú necesitas para empezar. Contestaremos preguntas tal: ¿Como hago mi texto rojo o negro?, ¿Como hago que mi contenido se muestre en tal y tal lugar de la pantalla? Como decoro mi página web con imágenes de fondo y colores?

Ejemplo de uso

Como HTML, CSS no es realmente un lenguaje de programación. Es un *lenguaje de hojas de estilo*, es decir, te permite aplicar estilos de manera selectiva a elementos en documentos HTML. Por ejemplo, para seleccionar **todos** los elementos de párrafo en una página HTML y volver el texto dentro de ellos de color rojo, has de escribir este CSS:

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

Vamos a probarlo: pega estas tres líneas de CSS en un nuevo archivo en tu editor de texto, y guarda este archivo como `style.css` en tu directorio `estilos`.

Archivo index.html de ejemplo para probar el código css

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
    <link href='http://fonts.googleapis.com/css?family=Open+Sans'
rel='stylesheet' type='text/css'>
    <link href="styles/style.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <h1>Mozilla is cool</h1>
    <img src="images/firefox-icon.png" alt="The Firefox logo: a flaming fox
surrounding the Earth.">

    <p>At Mozilla, we’re a global community of</p>

    <ul> <!-- changed to list in the tutorial -->
      <li>technologists</li>
      <li>thinkers</li>
      <li>builders</li>
    </ul>

    <p>working together to keep the Internet alive and accessible, so people
worldwide can be informed contributors and creators of the Web. We believe this
act of human collaboration across an open platform is essential to individual
growth and our collective future.</p>

    <p>Read the <a href="https://www.mozilla.org/en-US/about/manifesto/">Mozilla
Manifesto</a> to learn even more about the values and principles that guide the
pursuit of our mission.</p>
  </body>
</html>
```

Pero aun debemos aplicar el CSS a tu documento HTML, de otra manera el estilo CSS no cambiará cómo tu navegador muestra el documento HTML.

1- Abre tu archivo `index.html` y pegue la siguiente línea en algún lugar dentro del head, es decir, entre las etiquetas `<head>` y `</head>`:

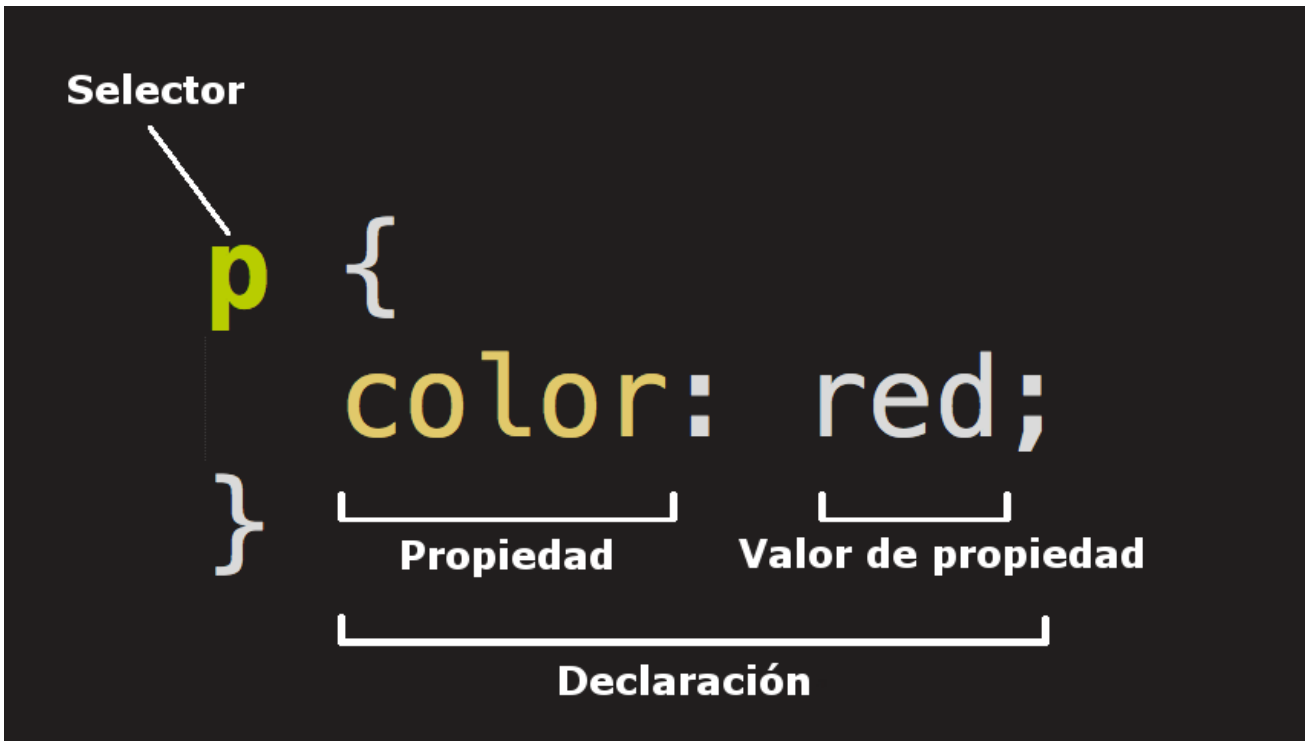
```
<link href="estilos/style.css" rel="stylesheet" type="text/css">
```

2- Guarda el archivo `index.html` y cárgalo en tu navegador. Debes ver algo como esto:

Si tu texto del parrafo ahora es rojo, ¡Felicitaciones, ya has escrito tu primer CSS de forma exitosa!

Anatomía de una regla CSS

Observemos al CSS de arriba, un poco más a detalle:



La estructura completa es llamada **regla predeterminada** (pero a menudo "regla" para abreviar). Nota también los nombres de las partes individuales:

Selector
: El elemento HTML en el que comienza la regla. esta selecciona el(los) elemento(s) a dar estilo (en este caso, los elementos `p`). Para dar estilo a un elemento diferente, solo cambia el selector.

Declaración
: Una sola regla como `color: red;` especifica a cuál de las **propiedades** del elemento quieres dar estilo.

Propiedades
: Maneras en las cuales puedes dar estilo a un elemento HTML. (En este caso, `color` es una propiedad del elemento `p`.) En CSS, seleccionas que propiedad quieres afectar en tu regla.

Valor de la propiedad
: A la derecha de la propiedad, después de los dos puntos (:), tenemos el **valor de la propiedad**, para elegir una de las muchas posibles apariencias para una propiedad determinada (hay muchos valores para `color` además de `red`).

Nota las otras partes importantes de la sintaxis:

- Cada una de las reglas (aparte del selector) deben estar encapsulada entre corchetes (`{}`).
- Dentro de cada declaración, debes usar los dos puntos (`:`) para separar la propiedad de su valor.
- Dentro de cada regla, debes usar el punto y coma (`;`) para separar una declaración de la siguiente.

Asi que para modificar varios valores de propiedad a la vez, solo necesitas escribirlos separados por punto y coma (;), así:

```
p {
```

```
  color: red;
  width: 500px;
  border: 1px solid black;
}
```

Seleccionando varios elementos

También puedes seleccionar varios elementos y aplicar una sola regla a todos ellos. Incluye varios selectores separados por comas(,). Por ejemplo:

```
p,li,h1 {
  color: red;
}
```

Diferentes tipos de selector

Existen muchos tipos diferentes de selectores. Antes, solo vimos los **selectores de elementos**, los cuales seleccionan todos los elementos de un tipo dado en los documentos HTML. Sin embargo podemos hacer selecciones más específicas que esas. Enseguida están algunos de los tipos de selectores más comunes:

Nombre del selector	Que selecciona?	Ejemplo
Selector de elemento (llamado algunas veces selector de etiqueta o tipo)	Todos los elementos HTML del tipo especificado.	`p` Selecciona `<p>`
Selector de identificación (ID)	El elemento en la página con el ID especificado (en una página HTML dada, solo se permite un unico elemento por ID).	`#my-id` Selecciona `<p id="my-id">` y `<a id="my-id">`
Selector de Clase	Los elementos en la página con la clase especificada (una clase puede aparecer varias veces en una página).	`.my-class` Selecciona `<p class="my-class">` y `<a class="my-class">`
Selector de atributo	Los elementos en una página con el atributo especificado.	`img[src]` Selecciona `<img src="myimage.png">` pero no `<img>`
Selector de Pseudo-clase	Los elementos especificados, pero solo cuando esté en el estado especificado, por ejemplo cuando el puntero esté sobre él.	`a:hover` Selecciona `<a>`, pero solo cuando el puntero esté sobre el enlace.

Existen muchos más selectores para explorar, estos son los mas usados.

Fuentes y texto

Ahora que hemos explorado lo básico de CSS, empecemos por añadir información y algunas reglas más a nuestro archivo `style.css` para que nuestro ejemplo se vea bonito. Primero, haremos que nuestras fuentes y texto luzcan un poco mejor.

1- Antes que nada, regresa y busca las fuentes de Google Fonts que guardaste en un lugar seguro. Agrega el elemento `<link ... >` en algún lugar del head de tu archivo

`index.html` (de nuevo, en cualquier lugar entre las etiquetas `<head>` y `</head>`). Se debe verse algo así:

```
<link href='http://fonts.googleapis.com/css?family=Open+Sans'
rel='stylesheet' type='text/css'>
```

2- Luego, borra la regla existente en tu archivo `style.css`. Fue una buena prueba, pero el texto en rojo en realidad no se ve muy bien.

3- Agrega en su lugar las siguientes líneas, reemplazando la línea **placeholder** con la actual línea `font-family` que obtuvo de Google Fonts (solamente quiere decir que `font-family` es la fuente que deseas usar en tu texto.) Esta regla selecciona primero una fuente base global para usar en toda la página (ya que `<html>` es el elemento primario de toda la página, y todos los elemento adentro heredan el `font-size` y `font-family`):

```
html {
  font-size: 10px; /* px quiere decir 'pixels': la base del tamaño de
fuente es ahora de 1o pixeles*/
  font-family: placeholder: Este debe ser el resto del resultado que
obtuviste de Google fonts
}
```

Nota: He añadido un comentario para explicar que significa "px". Todo lo que está en un documento de CSS entre `/*` y `*/` es un **comentario en CSS**, el cual el navegador descarta cuando carga el código. Este es un espacio donde puedes escribir notas útiles sobre lo que estas haciendo.

4- Ahora escojamos el tamaño de fuente para los elementos que contienen texto dentro del cuerpo del HTML (`<h1>`, `<li>`, y `<p>`). También centramos el texto de nuestro título, escogemos un ancho de línea y espaciado entre letras en el contenido del texto para hacerlo un poco mas legible:

```
h1 {
      font-size: 60px;
      text-align: center;
}

p, li {
      font-size: 16px;
      line-height: 2;
      letter-spacing: 1px;
}
```

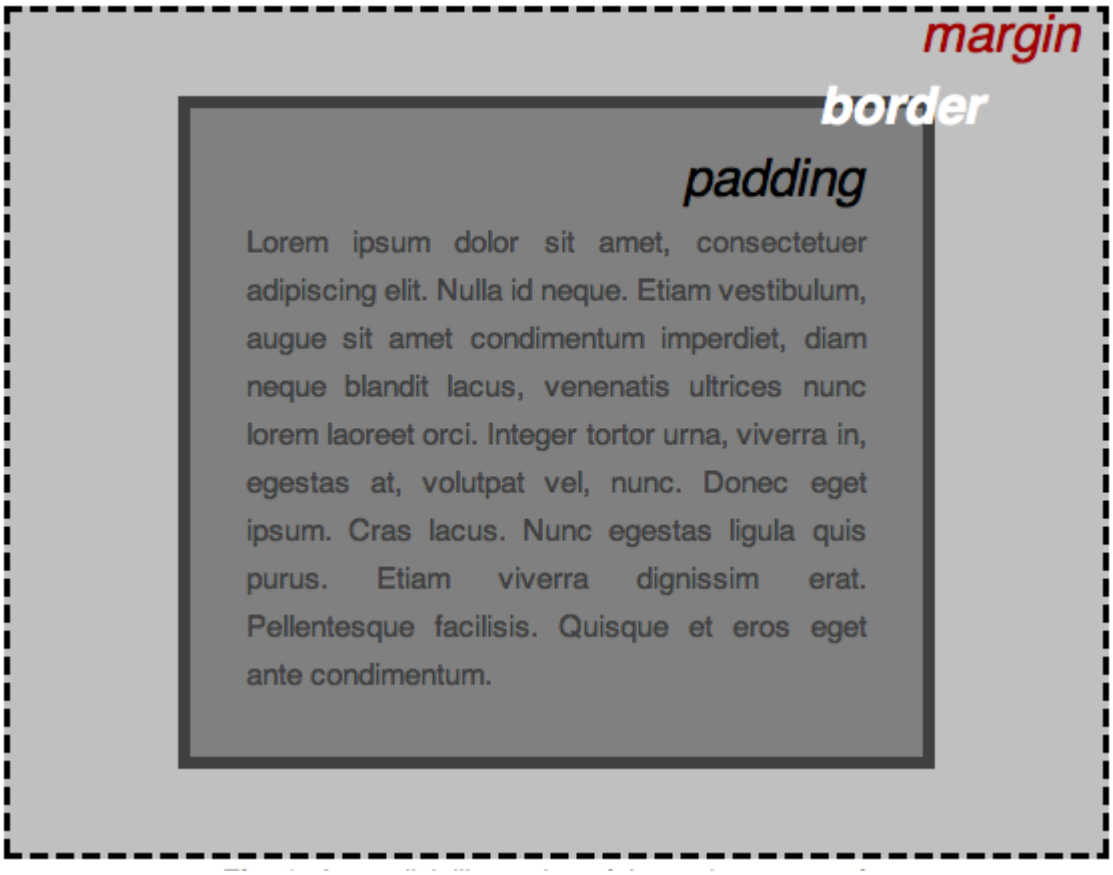
Puedes ajustar estos valores en px para lograr que tu diseño luzca como desees, pero por lo general tu diseño debe verse así:

Cajas, cajas, todo se trata de cajas

Una cosa que notarás sobre la escritura de CSS es que trata mucho sobre cajas — ajustando su tamaño, color, posición, etc. Puedes pensar en la mayoría de los elementos HTML de tu página como cajas apiladas una sobre la otra.

No es de extrañar que, el diseño de CSS este basado principalmente en el *modelo de caja.* Cada uno de los bloques que instala espacio en tu página tiene propiedades como estas:

- `relleno(padding)`, el espacio alrededor del contenido (ej: alrededor del texto de un párrafo)
- `marco(border)`, la línea que se encuentra fuera del relleno
- `margen(margin)`, el espacio fuera del elemento que lo separa de los demás.



En esta sección también usamos:

- `width` (ancho del elemento)
- `background-color`, el color de fondo del contenido y del relleno
- `color`, el color del contenido del elemento (generalmente texto)
- `text-shadow`: coloca una sombra difuminada en el texto dentro del elemento
- `display`: selecciona el modo de visualización para el elemento (no te preocupes de esto por ahora)

Bien, ¡iniciemos y agregamos más CSS a nuestra página! Sigue añadiendo estas reglas nuevas al final de la página, y no temas experimentar cambiando los valores para ver como resulta.

Cambiando el color de la página

```
html {
  background-color: #00539F;
}
```

Esta regla asigna un color de fondo a la página entera. Puedes cambiar el código de color por cualquiera como el que elegiste usar en tu proyecto.

Ordenando el Cuerpo

```
body {
```

```
  width: 600px;
  margin: 0 auto;
  background-color: #FF9500;
  padding: 0 20px 20px 20px;
  border: 5px solid black;
}
```

Ahora el elemento body. Éste tiene varias declaraciones, así que revisemos una por una:

- `width: 600px;` — esto hará que el cuerpo siempre tenga 600 pixeles de ancho.
- `margin: 0 auto;` — Cuando seleccionas dos valores dentro de propiedades como `margin` o `padding`, el primer valor afectara los lados superior (top) **e** inferior (bottom) (en este caso haciéndolo en 0), y el segundo valor los lados izquierda (left) **y** derecha (right) (aquí, `auto` es un valor especial que divide el espacio disponible entre derecha e izquierda). Puedes usar esta propiedad con uno, dos, tres o cuatro valores como se explica aqui.
- `background-color: #FF9500;` — como antes, éste selecciona el color de fondo de un elemento. Hemos usado un naranja rojizo para el `body` en contraste con el azul oscuro del elemento `html`. Sigue y experimenta. Siéntete libre de usar `white` o cualquiera que sea de tu agrado.
- `padding: 0 20px 20px 20px;` — tenemos 4 valores puestos en el relleno, para dar un poco de espacio alrededor del contenido. Esta vez no pondremos relleno en la parte de arriba del `body`, 20 pixeles a la izquierda, abajo y derecha. Los valores se ponen: arribar, derecha, abajo e izquierda, en ese orden.
- `border: 5px solid black;` — éste simplemente coloca un borde de 5 pixeles de ancho, continuo y de color negro alrededor del `body`.

Posicionando y dando estilo a nuestro título de página principal

```
h1 {
  margin: 0;
  padding: 20px 0;
  color: #00539F;
  text-shadow: 3px 3px 1px black;
}
```

Puedes haber notado que hay un hueco horrible en la parte superior del body. Esto sucede porque los navegadores vienen con estilos por defecto, ¡incluso cuando aún no se ha aplicado ningún archivo CSS! Esto podría parecer una mala idea, pero queremos que aun una página sin estilizar sea legible. Para deshacernos de este espacio eliminamos el estilo por defecto, `agregando margin: 0;`.

Enseguida, hemos puesto un relleno arriba y abajo del título de 20 pixeles, e hicimos que el color del texto sea el mismo que el color de fondo del `html`.

Una propiedad muy interesante que hemos usado aquí es `text-shadow`, que aplica una sombra al texto del elemento. Sus cuatro valores son como sigue:

- El primer valor en pixeles asigna el **desplazamiento horizontal** de la sombra desde el texto — que tan lejos la mueve a la derecha: un valor negativo la moverá a la izquierda.

- El segundo valor en pixeles asigna el **desplazamiento vertical** de la sombra desde el texto — que tan lejos la mueve hacia abajo, en este ejemplo, un valor negativo la desplazaría hacia arriba.
- El tercer valor en pixeles asigna **radio de desenfoque** de la sombra — un valor grande es igual a una sombra borrosa.
- El cuarto valor asigna el color base de la sombra.

Una vez más, trata de experimentar con diferentes valores para ver como resulta.

Centrando la imagen

```
img {
  display: block;
  margin: 0 auto;
}
```

Finalmente, centraremos la imagen para hacer que luzca mejor. Podemos usar nuevamente el truco del `margin: 0 auto` que usamos antes para el `body`, pero tenemos que hacer algo más. El elemento body es un elemento en nivel de bloque (**block-level**), lo que significa que tomará espacio en la pagina y que puede tener otros valores de espacio aplicables como margen. Las imágenes, por otra parte, son elementos **inline**, lo que quiere decir que no puedes aplicarles márgenes, debemos dar a la imagen un comportamiento de block-level usando `display: block;`.

Nota: No te preocupes si aun no entiendes `display: block;` y la diferencia entre block-level e inline. Lo entenderás en tanto estudies CSS a profundidad. Puedes encontrar mas en cuanto a los diferentes valores disponibles para display en display reference page.

Conclusión

Si has seguido las instrucciones, deberías terminar con una página que luce algo así:

Ejemplo CSS terminado

```
html {
  font-size: 10px;
  font-family: 'Open Sans', sans-serif;
}


h1 {
  font-size: 60px;
  text-align: center;
```

```
}

p, li {
  font-size: 16px;
  line-height: 2;
  letter-spacing: 1px;
}


html {
  background-color: #00539F;
}

body {
  width: 600px;
  margin: 0 auto;
  background-color: #FF9500;
  padding: 0 20px 20px 20px;
  border: 5px solid black;
}

h1 {
  margin: 0;
  padding: 20px 0;
  color: #00539F;
  text-shadow: 3px 3px 1px black;
}

img {
  display: block;
  margin: 0 auto;
}
```

Tipos MIME

El **tipo Extensiones multipropósito de Correo de Internet (MIME)** es una forma estandarizada de indicar la naturaleza y el formato de un documento. Está definido y estandarizado en IETF RFC 6838. La Autoridad de Números Asignados de Internet (IANA) es el organismo oficial responsable de realizar un seguimiento de todos los tipos MIME oficiales, y puede encontrar la lista más actualizada y completa en la página de tipos de medios (Media Types).

Los navegadores a menudo usan el tipo MIME (y no la extensión de archivo) para determinar cómo procesará un documento; por lo tanto, es importante que los servidores estén configurados correctamente para adjuntar el tipo MIME correcto al encabezado del objeto de respuesta.

Sintaxis

Estructura general

```
tipo/subtipo
```

La estructura de un tipo MIME es muy simple; consiste en un tipo y un subtipo, dos cadenas, separadas por un `'/'`. No se permite espacio. El *tipo* representa la categoría y puede ser de tipo *discreto* o *multiparte*. El *subtipo* es específico para cada tipo.

Un tipo MIME no distingue entre mayúsculas y minúsculas, pero tradicionalmente se escribe todo en minúsculas.

Tipos discretos

```
text/plain
text/html
image/jpeg
image/png
audio/mpeg
audio/ogg
audio/*
video/mp4
application/octet-stream
…
```

Los tipos *discretos* indican la categoría del documento, puede ser uno de los siguientes:

Tipo	Descripción	Ejemplo de subtipos típicos
`text`	Representa cualquier documento que contenga texto y es teóricamente legible por humanos	`text/plain`, `text/html`, `text/css`, `text/javascript`
`image`	Representa cualquier tipo de imagen. Los videos no están incluidos, aunque las imágenes animadas (como el gif animado) se describen con un tipo de imagen.	`image/gif`, `image/png`, `image/jpeg`, `image/bmp`, `image/webp`
`audio`	Representa cualquier tipo de archivos de audio	`audio/midi`, `audio/mpeg`, `audio/webm`, `audio/ogg`, `audio/wav`
`video`	Representa cualquier tipo de archivos de video	`video/webm`, `video/ogg`
`application`	Representa cualquier tipo de datos binarios.	`application/octet-stream`, `application/pkcs12`, `application/vnd.mspowerpoint`, `application/xhtml+xml`, `application/xml`, `application/pdf`

Para documentos de texto sin subtipo específico, se debe usar `text/plain`. De forma similar, para los documentos binarios sin subtipo específico o conocido, se debe usar `application/octet-stream`.

Tipos multiparte

```
multipart/form-data
multipart/byteranges
```

Los tipos de *partes múltiples* indican una categoría de documento que está rota en distintas partes, a menudo con diferentes tipos de MIME. Es una forma de representar un documento compuesto. Con

la excepción de `multipart/form-data`, que se utilizan en relación con formularios HTML y el método `POST`, y `multipart/byteranges`, que se utilizan junto con el mensaje de estado de `Contenido Parcial 206` para enviar solo un subconjunto de un documento completo, HTTP no maneja documentos multiparte de una manera específica: el mensaje simplemente se transmite al navegador (que probablemente propondrá una ventana Guardar como, sin saber cómo mostrar el documento en línea.)

Tipos MIME importantes para desarrolladores Web

`application/octet-stream`

Este es el valor predeterminado para un archivo binario. Como realmente significa un *archivo binario desconocido*, los navegadores generalmente no lo ejecutan automáticamente, o incluso preguntan si debería ejecutarse. Lo tratan como si el encabezado `Content-Disposition` se configurara con el valor `attachment` y proponen un archivo 'Guardar como'.

`text/plain`

Este es el valor predeterminado para los archivos de texto. Incluso si realmente significa un archivo textual desconocido, los navegadores asumen que pueden mostrarlo.

Tenga en cuenta que `text/plain` no significa *cualquier tipo de datos textuales*. Si esperan un tipo específico de datos textuales, probablemente no lo considerarán una coincidencia. Específicamente, si descargan un archivo de texto sin formato `text/plain` de un elemento `<link>` que declara archivos CSS, no lo reconocerán como un archivo CSS válido si se presenta con `text/plain`. Se debe usar el tipo MIME CSS `text/css`.

`text/css`

Todos los archivos CSS que deban interpretarse como tales en una página web **deben** ser de los archivos de `text/css`. A menudo, los servidores no reconocen los archivos con el sufijo `.css` como archivos CSS y los envían con tipo MIME `text/plain` o `application/octet-stream`: en estos casos, la mayoría de los navegadores no los reconocerán como archivos CSS y serán ignorado silenciosamente. Se debe prestar especial atención en servir los archivos CSS con el tipo correcto.

`text/html`

Todo el contenido HTML debe ser servido con este tipo. Los tipos MIME alternativos para XHTML, como `application/xml+html`, son en su mayoría inútiles hoy en día (HTML5 unificó estos formatos).

Tipos de imágenes

Solo un puñado de tipos de imágenes son ampliamente reconocidos y se consideran seguros para la Web, listos para usar en una página Web:

Tipo MIME	Tipo de imagen
`image/gif`	Imágenes GIF (compresión sin pérdida, reemplazada por PNG)

Tipo MIME	Tipo de imagen
`image/jpeg`	Imágenes JPEG
`image/png`	Imágenes PNG
`image/svg+xml`	Imágenes SVG (imágenes vectoriales)

Existe una discusión para agregar WebP (`image/webp`) a esta lista, pero como cada tipo de imagen nuevo aumentará el tamaño de una base de código, esto puede presentar nuevos problemas de seguridad, por lo que los proveedores de navegadores son cautelosos al aceptarlo.

Se pueden encontrar otros tipos de imágenes en documentos Web. Por ejemplo, muchos navegadores admiten tipos de imágenes de iconos para favicones o similares. En particular, las imágenes ICO son compatibles en este contexto con el tipo MIME `image/x-icon`.

Tipos de audio y video

Al igual que las imágenes, HTML no define un conjunto de tipos soportados para usar con los elementos `<audio>` y `<video>`, por lo que solo un grupo relativamente pequeño de ellos puede ser utilizado en la web. Los formatos de medios compatibles con los elementos de audio y video HTML explican tanto los códecs como los formatos de contenedor que se pueden usar.

El tipo MIME de dichos archivos representa principalmente los formatos de contenedor y los más comunes en un contexto web son:

Tipo MIME	Tipo de audio o video

`audio/wave`
`audio/wav`
`audio/x-wav`
`audio/x-pn-wav`
	Un archivo de audio en formato de contenedor WAVE. El códec de audio PCM (códec WAVE "1") a menudo es compatible, pero otros códecs tienen soporte más limitado (si lo hay).

`audio/webm`	Un archivo de audio en formato de contenedor WebM. Vorbis y Opus son los códecs de audio más comunes.
`video/webm`	Un archivo de video, posiblemente con audio, en el formato de contenedor de WebM. VP8 y VP9 son los códecs de video más comunes utilizados en él; Vorbis y Opus los códecs de audio más comunes.
`audio/ogg`	Un archivo de audio en el formato de contenedor OGG. Vorbis es el códec de audio más común utilizado en dicho contenedor.
`video/ogg`	Un archivo de video, posiblemente con audio, en el formato de contenedor OGG. Theora es el códec de video habitual utilizado en él; Vorbis es el códec de audio habitual.
`application/ogg`	Un archivo de audio o video usando el formato de contenedor OGG. Theora es el códec de video habitual utilizado en él; Vorbis es el códec de audio más común.

`multipart/form-data`

El tipo de datos `multipart/form-data` se puede usar al enviar el contenido de un formulario HTML completo desde el navegador al servidor. Como formato de documento multiparte, consta de diferentes partes, delimitadas por un límite (una cadena que comienza con un doble guión `'--'`). Cada parte es una entidad en sí misma, con sus propios encabezados HTTP, `Content-`

`Disposition` y `Content-Type` para los campos de carga de archivos, y los más comunes (`Content-Length` es ignorado ya que la línea límite se usa como delimitador).

```
Content-Type: multipart/form-data; boundary=unaCadenaDelimitadora
(otros encabezados asociados con el documento multiparte como un todo)

--unaCadenaDelimitadora
Content-Disposition: form-data; name="miArchivo"; filename="img.jpg"
Content-Type: image/jpeg

(data)
--unaCadenaDelimitadora
Content-Disposition: form-data; name="miCampo"

(data)
--unaCadenaDelimitadora
(más subpartes)
--unaCadenaDelimitadora--
```

El siguiente formulario:

```
<form action="http://localhost:8000/" method="post" enctype="multipart/form-
data">
  <input type="text" name="miCampoDeTexto">
  <input type="checkbox" name="miCheckBox">Checado</input>
  <input type="file" name="miArchivo">
  <button>Enviar el archivo</button>
</form>
```

enviará este mensaje:

```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20100101
Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=--------------------------
872165604191141565395500449 8
Content-Length: 465

-----------------------------872165604191141565395500449 8
Content-Disposition: form-data; name="miCampoDeTexto"

Test
-----------------------------872165604191141565395500449 8
Content-Disposition: form-data; name="miCheckBox"

on
-----------------------------872165604191141565395500449 8
Content-Disposition: form-data; name="miArchivo"; filename="prueba.txt"
Content-Type: text/plain

Simple file.
-----------------------------872165604191141565395500449 8--
```

`multipart/byteranges`

El tipo MIME `multipart/byteranges` se usa en el contexto del envío de respuestas parciales al navegador. Cuando se envía el código de estado de `Contenido Parcial 206`, este tipo MIME se usa para indicar que el documento está compuesto de varias partes, una para cada rango solicitado. Al igual que otros tipos de varias partes, `Content-Type` usa la directiva `boundary` para definir la cadena delimitadora. Cada una de las diferentes partes tiene un encabezado `Content-Type` con el tipo real del documento y un `Content-Range` con el rango que representan.

```
HTTP/1.1 206 Partial Content
Accept-Ranges: bytes
Content-Type: multipart/byteranges; boundary=3d6b6a416f9b5
Content-Length: 385

--3d6b6a416f9b5
Content-Type: text/html
Content-Range: bytes 100-200/1270

eta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="vieport" content
--3d6b6a416f9b5
Content-Type: text/html
Content-Range: bytes 300-400/1270

-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: "Open Sans", "Helvetica
--3d6b6a416f9b5--
```

Importancia de establecer el tipo MIME correcto

La mayoría de los servidores web envían recursos de tipo desconocido utilizando el tipo MIME predeterminado `application/octet-stream`. Por razones de seguridad, la mayoría de los navegadores no permiten establecer una acción predeterminada personalizada para dichos recursos, lo que obliga al usuario a almacenarlo en el disco para usarlo. Algunas configuraciones de servidor incorrectas que se ven con frecuencia ocurren con los siguientes tipos de archivos:

- Archivos con codificación RAR. En este caso, lo ideal sería establecer el tipo verdadero de los archivos codificados; esto a menudo no es posible (ya que puede no ser conocido por el servidor y estos archivos pueden contener varios recursos de diferentes tipos). En este caso, al configurar el servidor para que envíe el tipo MIME `application/x-rar-compressed`, los usuarios no habrán definido una acción predeterminada útil para ellos.
- Archivos de audio y video. Solo los recursos con el tipo MIME correcto serán reconocidos y reproducidos en elementos `<video>` o `<audio>`. Asegúrese de usar el tipo correcto para audio y video.
- Tipos de archivos propietarios. Preste especial atención al servir un tipo de archivo propietario. Evite el uso de `application/octet-stream` ya que no será posible un manejo especial: la mayoría de los navegadores no permiten definir un comportamiento predeterminado (como "Abrir en Word") para este tipo genérico MIME.

Olfateo MIME (sniffing)

En ausencia de un tipo MIME, o en algunos otros casos en los que un cliente cree que están configurados incorrectamente, los navegadores pueden realizar el rastreo MIME, que es adivinar el tipo MIME correcto mirando el recurso. Cada navegador realiza esto de manera diferente y bajo diferentes circunstancias. Hay algunas preocupaciones de seguridad con esta práctica, ya que algunos tipos MIME representan el contenido ejecutable y otros no. Los servidores pueden bloquear el rastreo de MIME enviando el `X-Content-Type-Options` a lo largo de `Content-Type`.