Programación Web

El desarrollo de sitios web para internet, emplea en su inmensa mayoría, tecnologías del lado del servidor y tecnologías del lado del cliente. Esto involucra una combinación de procesos de base de datos con el uso de un navegador web a fin de realizar determinadas tareas o mostrar información.

Como desarrolladores web, debemos tener conocimiento de qué sucede en las diferentes etapas. Como primer detalle podemos hacer mención a cómo se desarrolla software en la actualidad.

Bases de la arquitectura web

Si bien aún no daremos la profundidad qué la temática lo amerita, haremos mención del desarrollo basado en microservicios. Detalle qué implica “romper” una aplicación en múltiples etapas con tareas específicas, haciendo hincapié en lo qué se conoce como desacoplamiento y modularidad de código.

Previamente los desarrollos se encontraban todos en un mismo bloque trabajando, a medida qué el desarrollo de software se fue extendiendo y detalles como la reutilización de código y la adaptabilidad de los desarrollos empezaron a cobrar mayor importancia, lo qué se conoce como desarrollo monolítico, fue quedando atrás.

Otra base qué encontramos para la arquitectura web, es el modelo de capas OSI. El conjunto de estas capas conforma lo qué se conoce como pila

De esta lógica desacoplada, es qué parte el desarrollo web qué estudiaremos a continuación. El mismo se establece en lo qué se conoce como “capas”, donde en su conjunto conforman una “pila” y se encargan de vincularse entre sí, logrando qué la información viaje y se produzca entre usuarios y servidores, en lo qué se conoce como modelo OSI.

En la capa Física definimos los cables, el medio de transmisión que vamos a utilizar, la radiofrecuencias a manejar, aspectos físicos.

En la capa de Enlace de Datos trabajos los switches, los elementos que van a poder interconectar segmentos de la red.

En Red hablamos de direccionamiento IP, Enrutamiento

En la capa de Transporte, TCP-UDP siendo estos otros protocolos.

En la capa de Sesión, veremos cómo se comunican y cómo administramos las sesiones entre host

En la capa de Presentación, estandarización de la forma en que se presentan los datos.

En la capa de Aplicación, ya es donde nuestras aplicaciones interactúan directamente con los usuarios.

Familia de protocolos de internet

La familia de protocolos de internet es un conjunto de protocolos de red en los que se basa internet y que permiten la transmisión de datos entre computadoras.

En ocasiones se le denomina conjunto de protocolos TCP/IP, en referencia a los dos protocolos más importantes que la componen, que fueron de los primeros en definirse, y que son los dos más utilizados de la familia:
TCP: protocolo de control de transmisión.
IP: protocolo de internet.
Existen tantos protocolos en este conjunto que llegan a ser más de cien diferentes,[cita requerida] entre ellos se encuentran:
ARP: protocolo de resolución de direcciones, para encontrar la dirección física (MAC) correspondiente a una determinada IP.
FTP: protocolo de transferencia de archivos, popular en la transferencia de archivos.
HTTP: protocolo de transferencia de hipertexto, que es popular porque se utiliza para acceder a las páginas web.
POP: protocolo de oficina de correo, para correo electrónico.
SMTP: protocolo para transferencia simple de correo, para el correo electrónico.
Telnet (Telecommunication Network), para acceder a equipos remotos.
IDENTD: (Identification Daemon)
IRC: (Internet Relay Chat)
UDP: (User Datagram Protocol) Protocolo de Datagramas de usuario
ICMP:(Internet Control Message Protocol)
TFTP (Trivial FTP) FTP trivial

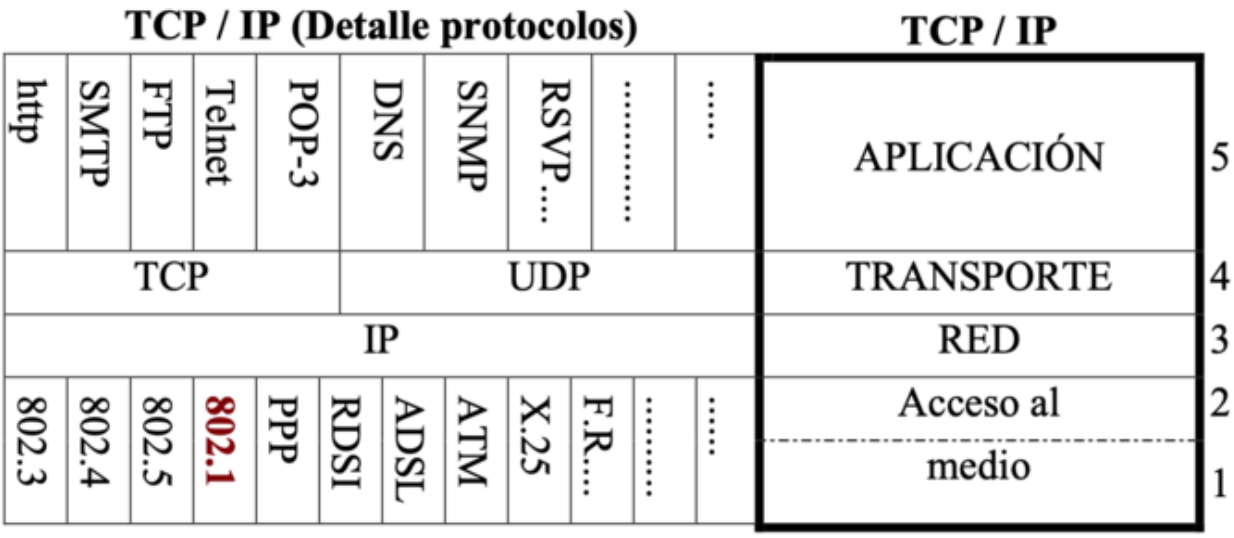
DHCP: (Dynamic Host Configuration Protocol) Protocolo de configuración dinámica del Host
NTP: (Network Time Protocol) Protocolo de tiempo de Red
DNS: (Domain Name Service) Servicio de Nombres de Dominio
SNMP: (Simple Network Management Protocol)
RIP: (Routing Information Protocol)
PPP: (Point to Point Protocol)

Se debe tener en cuenta también que esta familia de protocolos, o pila TCP/IP, no es un estándar estricto, como lo es el modelo OSI de ISO, en el cual cada nivel o capa responde a una normativa muy específica. En el caso de TCP/IP, se basa en la mejores prácticas y/o las diferentes RFC que se van publicando. Es por esta razón por lo que es común que diferentes autores lo presenten como un modelo de cuatro o cinco capas, y ambas posturas son igual de válidas, pues son la interpretación de cada autor.

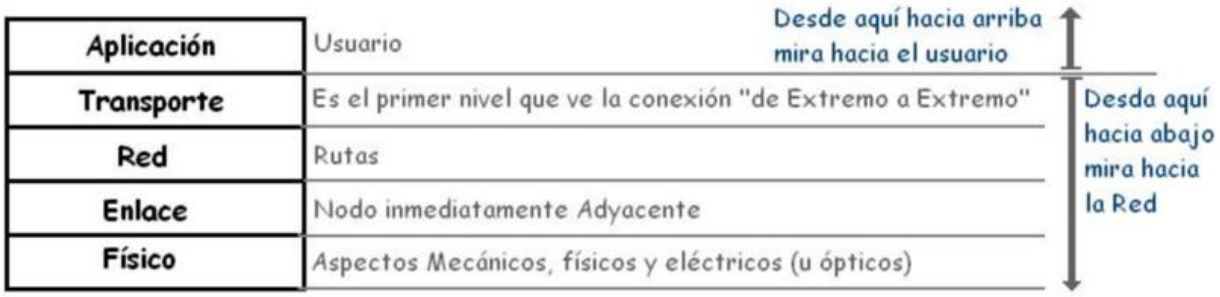
Teniendo en cuenta estos conceptos, se puede plantear una correspondencia entre ambos modelos de la forma que se presenta en la imagen

<table>
<tr><th>OSI</th><th>DARPA o TCP/IP</th></tr>
<tr><td>Aplicación</td><td rowspan="3">Aplication</td></tr>
<tr><td>Presentación</td></tr>
<tr><td>Sesión</td></tr>
<tr><td>Transporte</td><td>Transport</td></tr>
<tr><td>Red</td><td>Internetwork</td></tr>
<tr><td>Enlace</td><td>Medium Access</td></tr>
<tr><td>Físico</td><td>Phisical</td></tr>
</table>

Por esa razón si queremos asociar los protocolos mencionados anteriormente con este modelo de capas TCP/IP, los deberíamos presentar según la imagen que se presenta a continuación.



Por último, es muy importante considerar que estos diferentes niveles o capas, se han diseñado para desempeñar servicios y/o funciones muy específicas, en particular hacia dos grandes conceptos de informática: redes y TI, tal cual se presenta en la imagen que figura a continuación.



Sitio Web

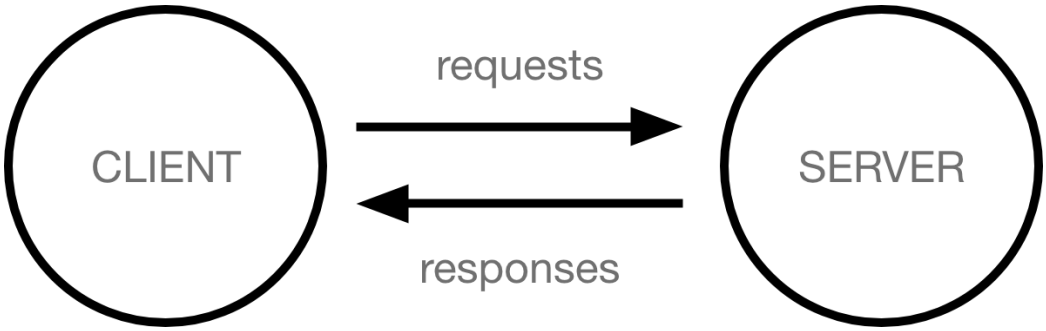
Un sitio web, (website) portal o cibersitio es una colección de páginas web relacionadas y comunes a un dominio de internet o subdominio en la World Wide Web dentro de Internet.
A las páginas de un sitio web se accede frecuentemente a través de un URL raíz común llamado portada, que normalmente reside en el mismo servidor físico. Los URL organizan las páginas en una jerarquía, aunque los hiperenlaces, entre ellas controlan más particularmente cómo el lector percibe la estructura general y cómo el tráfico web fluye entre las diferentes partes de los sitios.
Algunos sitios web requieren una subscripción para acceder a algunos o todos sus contenidos. Ejemplos de sitios con subscripción incluyen algunos sitios de noticias, de juegos, foros, servicios de correo electrónico basados en web, sitios que proporcionan datos en tiempo real tal como datos meteorológicos, información económica, horarios, etc.

Página web

No debemos confundir sitio web con página web; esta última es solo un archivo HTML, una unidad HTML, que forma parte de algún sitio web. Al ingresar una dirección web, como por ejemplo www.wikipedia.org, siempre se está haciendo referencia a un sitio web, el que tiene una página HTML inicial, que es generalmente la primera que se visualiza. La búsqueda en Internet se realiza asociando el DNS ingresado con la dirección IP del servidor que contiene el sitio web en el cual está la página HTML buscada.

Los clientes y servidores

Las computadoras conectadas a la web se llaman clientes y servidores.



Los clientes son dispositivos de los usuarios conectados a Internet (por ejemplo, tu ordenador conectado a la red Wi-Fi o el teléfono conectado a la red de telefonía móvil) y el software que se encuentra disponible y permite acceder a Internet en dichos dispositivos (normalmente, un navegador web como Firefox o Chrome).

Los servidores son computadores que almacenan páginas web, sitios o aplicaciones. Cuando un dispositivo cliente quiere acceder a una página web, una copia de la página web se descarga desde el servidor en el equipo cliente y se muestra en el navegador web del usuario.

Además del cliente y el servidor:

Tu conexión a Internet: permite enviar y recibir datos en la web. Básicamente es el recorrido entre tu casa y la tienda.

TCP/IP: Protocolo de Control de Transmisión y Protocolo de Internet, son los protocolos de comunicación que definen cómo deben viajar los datos a través de la web. Esto es, los medios de transporte que te permiten hacer un pedido, ir a la tienda y comprar los productos. En nuestro ejemplo, podría ser un coche, una bicicleta o tus propios pies.
DNS: los servidores del Sistema de Nombres de Dominio (DNS, por sus siglas en inglés), son como una libreta de direcciones de sitios web. Cuando escribes una dirección web en el navegador, el navegador busca los DNS antes de recuperar el sitio web. El navegador necesita averiguar en qué servidor vive el sitio web y así enviar los mensajes HTTP al lugar correcto (ver más abajo). Esto es como buscar la dirección de la tienda para que puedas llegar a ella.
HTTP: el Protocolo de Transferencia de Hipertexto es un protocolo de aplicación que define un idioma para que los clientes y servidores se puedan comunicar. Esto es como el idioma que utilizas para ordenar tus compras.
Archivos componentes: un sitio web se compone de muchos archivos diferentes, que son como las diferentes partes de los productos que comprarás en la tienda. Estos archivos se dividen en dos tipos principales:
Archivos de código: los sitios web se construyen principalmente con HTML, CSS y JavaScript, aunque te encontrarás con otras tecnologías más adelante.
Recursos: este es un nombre colectivo para el resto de materiales que conforman un sitio web, como imágenes, música, video, documentos de Word, archivos PDF, etc.

Entonces, ¿qué sucede exactamente?

Cuando escribes una dirección web en el navegador (usando nuestra analogía para ir a la tienda):

1. El navegador va al servidor DNS y encuentra la dirección real del servidor donde el sitio web vive (encontrar la dirección de la tienda).
2. El navegador envía un mensaje de petición HTTP al servidor, pidiéndole que envíe una copia de la página web para el cliente (ir a la tienda y hacer un pedido). Este mensaje y todos los datos enviados entre el cliente y el servidor, se envían a través de tu conexión a Internet usando TCP/IP.
3. Siempre que el servidor apruebe la solicitud del cliente, el servidor enviará al cliente un mensaje «200 OK», que significa, «¡por supuesto que puedes ver ese sitio web! Aquí está.», y comenzará a enviar los archivos de la página web al navegador como una serie de pequeños trozos llamados paquetes de datos (la tienda te entrega tus productos y los llevas de regreso a casa).
4. El navegador reúne los pequeños trozos, forma un sitio web completo y te lo muestra (llegas a casa con tus nuevas compras).

DNS

Las direcciones webs reales no son las agradables y fácilmente recordables secuencias que tecleas en la barra de direcciones para encontrar tus sitios webs favoritos. En realidad, se trata de secuencias de números, algo como 63.245.217.105.

Lo anterior se llama dirección IP y representa un lugar único en la web. Sin embargo, no es muy fácil de recordar, ¿verdad? Por eso se inventaron los servidores de nombres de dominio. Estos son

servidores especiales que hacen coincidir una dirección web tecleada desde tu navegador («mozilla.org», por ejemplo) con la dirección real del sitio web (IP).

Los sitios webs se pueden acceder directamente a través de sus direcciones IP. Intenta acceder a la página web de Mozilla escribiendo 63.245.217.105 en la barra de dirección de una nueva pestaña en tu navegador. Puedes encontrar la dirección IP de un sitio web escribiendo su dominio en una herramienta como DNS lookup.

Paquetes

Anteriormente hemos utilizado el término paquetes para describir el formato en que los datos se envían desde el servidor al cliente. Básicamente, los datos se envían a través de la web como miles de trozos pequeños, permitiendo que muchos usuarios pueden descargar la misma página web al mismo tiempo.

¿Qué es la programación de sitios web de lado servidor?

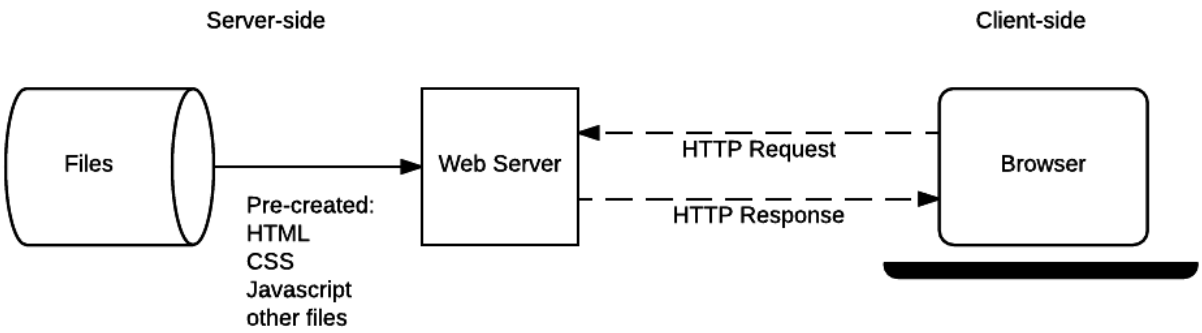
Los exploradores web se comunican con los servidores web usando el Protocolo de Transporte de Hyper Texto (HyperText Transport Protocol (HTTP). Cuando pinchas en un enlace en una página web, envías un formulario o ejecutas una búsqueda, se envía una peticion HTTP desde tu explorador web al servidor web de destino. La petición incluye un URL que identifica el recurso afectado, un método que define la acción requerida (por ejemplo, obtener, borrar o publicar el recurso), y puede incluir información adicional codificada en parámetros en el URL (los pares campo-valor enviados en una cadena de consulta (query string), como datos POST (datos enviados mediate el método POST de HTTP, HTTP POST method), o en associated cookies.

Los servidores web esperan los mensajes de petición de los clientes, los procesan cuando llegan y responden al explorador web con un mensaje de respuesta HTTP. La repuesta contiene una línea de estado indicando si la petición ha tenido éxito o no (ej, "HTTP/1.1 200 OK" en caso de éxito). El cuerpo de una respuesta exitosa a una petición podría contener el resurso solicitado (ej, una nueva página HTML, o una imagen, etc...), que el explorador web podría presentar en pantalla.

Sitios Estáticos

El diagrama de abajo muestra una arquitectura de servidor web básica correspondiente a un sitio estático (un sitio estático es aquél que devuelve desde el servidor el mismo contenido insertado en el código "hard coded" siempre que se solicita un recurso en particular). Cuando un usuario quiere navegar a una página, el explorador envía una petición HTTP "GET" especificando su URL. El servidor recupera de su sistema de ficheros el documento solicitado y devuelve una respuesta HTTP

que contiene el documento y un estado de éxito "success status" (normalmente 200 OK). Si el fichero no puede ser recuperado por alguna razón, se devuelve un estado de error (ver respuestas de error del cliente y respuestas de error del servidor).

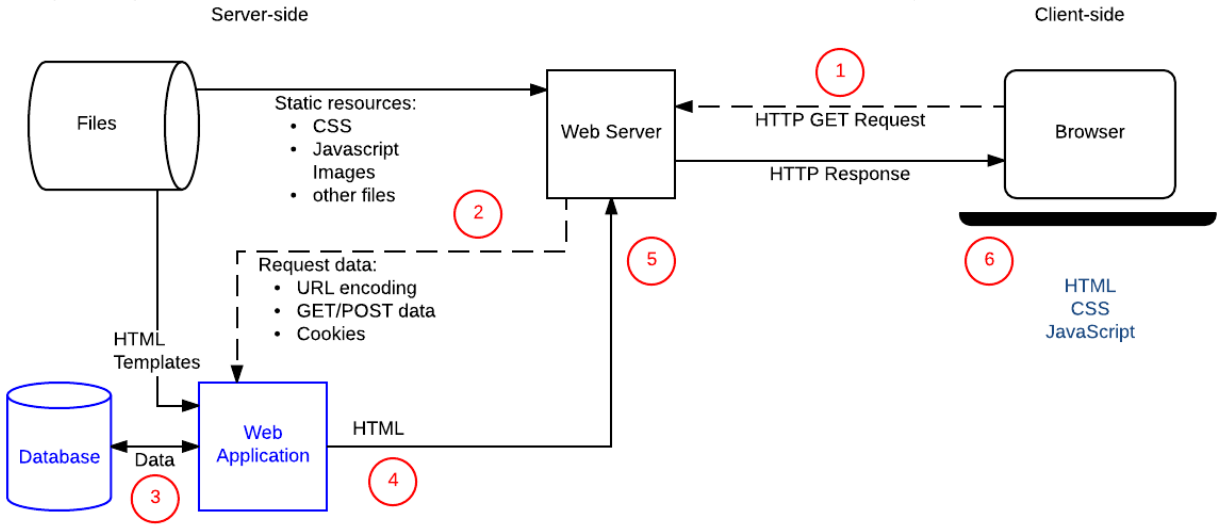


Sitios Dinámicos

Un sitio dinámico es aquél en que algun contenido de la respuesta está generado dinámicamente sólo cuando se necesita. En un sitio web dinámico las páginas HTML se crean normalmente insertando datos desde una base en variables dentro de plantillas HTML (esta es una forma mucho más eficiente de almacenar gran cantidad de contenido que la que usan los sitios web estáticos). Un sitio dinámico puede devolver datos diferentes para un URL basados en la información proporcionada por el usuario o sus preferencias almacenadas y puede realizar otras operaciones como parte de la devolución de respuesta (ej, enviar notificaciones).

La mayor parte del código para soportar un sitio web dinámico debe correr en el servidor. La creación de este código se conoce como "programación de lado-servidor" (o algunas veces "back-end scripting").

El diagrama de abajo muestra una arquitectura simple para un sitio web dinámico. Como en el diagrama previo, los exploradores web envían peticiones HTTP al servidor, el servidor procesa a continuación las peticiones y devuelve las respuestas HTTP apropiadas. Las peticiones de recursos estáticos son gestionadas de la misma manera que para los sitios estáticos (los recursos estáticos son cualquier fichero que no cambia - generalmente: CSS, JavaScript, Imágenes, ficheros PDF creados previamente, etc...)



Las peticiones de recursos dinámicos, por el contrario, son reenviadas (2) al código del lado-servidor (mostrado en el diagrama como Web Application). Para las "peticiones dinámicas" el servidor interpreta la petición, lee de la base de datos la información requerida (3), combina los datos recuperados con las plantillas HTML (4), y envía de vuelta una respuesta que contiene el HTML generado (5,6).

¿Son iguales la programación del lado-servidor y lado-cliente?

Prestemos ahora nuestra atención al código involucrado en la programación de lado-servidor y lado-cliente. En cada caso, el código es significativamente diferente:

- Tienen diferentes propósitos y preocupaciones.
- Por lo general no usan los mismos lenguajes de programación (siendo la excepción el JavaScript, que puede usarse tanto en lado servidor como en lado cliente).
- Se ejecutan entornos de diferentes sistemas operativos.

El código que se ejecuta en el explorador se conoce como código de lado-cliente, y su principal preocupación es la mejora de la apariencia y el comportamiento de una página web entregada. Esto incluye la selección y estilo de los componentes UI, la creación de layouts, navegación, validación de formularios, etc. Por otro lado, la programación de sitios web de lado servidor en su mayor parte implica la elección de qué contenido se ha de devolver al explorador como respuesta a sus peticiones. El código de lado-servidor gestiona tareas como la validación de los datos enviados y las peticiones, usando bases de datos para almacenar y recuperar datos, y enviando los datos correctos al cliente según se requiera.

El código del lado cliente está escrito usando HTML, CSS, y JavaScript — es ejecutado dentro del explorador web y tiene poco o ningún acceso al sistema operativo subyacente (incluyendo un acceso limitado al sistema de ficheros).

Los desarrolladores web no pueden controlar qué explorador web usará cada usuario para visualizar un sitio web — los exploradores web proporcionan niveles de compatibilidad inconsistentes con las características de codificación lado cliente, y parte del reto de la programación de lado cliente es gestionar con dignidad las diferencias de soporte entre exploradores.

El código del lado servidor puede escribirse en cualquier número de lenguajes de programación — ejemplos de lenguajes de programación populares incluyen Java, PHP, Python, Ruby, C# y NodeJS(JavaScript). El código del lado servidor tiene acceso completo al sistema operativo del servidor y el desarrollador puede elegir qué lenguaje de programación (y qué versión específica) desea usar.

Los desarrolladores generalmente escriben su código usando web frameworks. Los web framworks son colecciones de funciones, objetos, reglas y otras construcciones de código diseñadas para resolver problemas comunes, acelerar el desarrollo y simplificar los diferentes tipos de tareas que se han de abordar en un dominio en particular.

De nuevo, mientras que, tanto el código lado cliente y el lado servidor usan frameworks, los dominios son muy diferentes, y por lo tanto también lo son los frameworks. Los frameworks del lado cliente simplifican los diseños y las tareas de presentación mientras que los del lado servidor proporcionan un montón de funcionalidades "comunes" que tendría que haber implementado uno mismo (ej, soporte para las sesiones, soporte para los usuarios y autenticación, acceso fácil a la base de datos, librerías de plantillas, etc...).

¿Qué se puede hacer en el lado-servidor?

La programación del lado-servidor es muy útil porque nos permite distribuir eficientemente información a medida para usuarios individuales y por lo tanto crear una experiencia de usuario mucho mejor.

Compañías como Amazon utilizan la programación del lado-servidor para construir resultados de búsquedas de productos, hacer sugerencias sobre productos escogidos basados en las preferencias del cliente y sus hábitos de compra previos, simplificar las adquisiciones, etc. Los bancos usan la programación del lado-servidor para almacenar la información sobre las cuentas y permitir ver y realizar transacciones sólo a los usuarios autorizados. Otros servicios como Facebook, Twitter, Instagram y Wikipedia usan la programación de lado-servidor para destacar, compartir y controlar el acceso al contenido interesante.

Almacenaje y distribución eficiente de información

Imagina cuántos productos están disponibles en Amazon, e imagina cuántas entradas se han escrito en Facebook. Crear una página estática separada para cada producto o entrada sería completamente ineficiente.

La programación de lado-servidor nos permite por el contrario almacenar la información en una base de datos y construir dinámicamente y devolver ficheros HTML y de otros tipos (ej, PDFs, imágenes, etc.). También es posible devolver simplemente datos (JSON, XML, etc.) para presentar

mediante los web frameworks adecuados del lado-cliente (esto reduce la carga de procesamiento del servidor y la cantidad de datos que se necesitan enviar).

El servidor no se limita a enviar información de las bases de datos, y podría además devolver el resultado de herramientas de software o datos de servicios de comunicación. El contenido puede incluso ser dirigido por el tipo de dispositivo cliente que lo está recibiendo.

Debido a que la información está en una base de datos, puede también ser compartida y actualizada con otros sistemas de negocio (por ejemplo, cuando se venden los productos online o en una tienda, la tienda debería actualizar su base de datos de inventario.

Acceso controlado al contenido

La programación de lado-servidor permite a los sitios restringir el acceso a usuarios autorizados y servir sólo la información que se le permite ver al usuario.

Ejemplos del mundo real incluyen:

Redes sociales como Facebook permiten a los usuarios controlar totalmente sus propios datos pero permitiendo sólo a sus amigos ver o comentar sobre ellos. El usuario determina quien puede ver sus datos, y por extensión, los datos de quienes aparecen en sus notificaciones — autorización es una parte central de la experiencia de usuario!

Almacenar información de sesión/estado

La programación de lado-servidor permite a los desarrolladores hacer uso de las sesiones — es básicamente un mecanismo que permite al servidor almacenar información sobre el usuario actual del sitio u enviar diferentes respuestas basadas en esa información. Esto permite, por ejemplo, que un sitio sepa que un usuario ha iniciado sesión previamente y presente enlaces a sus correos, o a su historial de órdenes, o quizá guardar el estado de un simple juego de forma que el usuario pueda volver al sitio de nuevo y retomar el juego donde lo dejó.
Notificaciones y comunicación
Los servidores pueden enviar notificaciones de tipo general o específicas de usuario a través del propio sitio web o vía correo electrónico, SMS, mensajería instanténea, conversaciones de video u otros servicios de comunicación.

Unos pocos ejemplos incluyen:

- Facebook y Twitter envían mensajes de correo y SMS para notificarte de nuevas comunicaciones.
- Amazon envía con regularidad emails que sugieren productos similares a aquellos comprados o vistos anteriormente y en los que podrías estar interesado.
- Un servidor web podría enviar mensajes de aviso a los administradores del sistema alertandoles de memoria baja en el servidor o de actividades de usuario sospechosas.

Análisis de datos

Un sitio web puede recolectar un montón de datos acerca de los usuarios: qué es lo que buscan, qué compran, qué recomiendan, cuánto tiempo permanecen en cada página. La programación de lado-servidor puede utilizarse para refinar las respuestas basándose en el análisis de estos datos.
Por ejemplo, Amazon y Google anuncian ambos productos basados en búsquedas previas (y adquisiciones).

Servidores Web y HTTP

Los exploradores web se comunican con los servidores web usando el Protocolo de Transferencia de HyperTexto (HyperTextTransfer Protocol HTTP). Cuando haces click en un enlace sobre una página web, envías un formulario o ejecutas una búsqueda, el explorador envía una petición (Request) HTTP al servidor.

Esta petición incluye:

- Una URL que identifica el servidor de destino y un recurso (ej. un fichero HTML, un punto de datos particular en el servidor, o una herramienta a ejecutar).
- Un método que define la acción requerida (por ejemplo, obtener un fichero o salvar o actualizar algunos datos). Los diferentes métodos/verbos y sus acciones asociadas se listan debajo:
  - GET: Obtener un recurso específico (ej. un fichero HTML que contiene información acerca de un producto o una lista de productos).
  - POST: Crear un nuevo recurso (ej. añadir un nuevo artículo a una wiki, añadir un nuevo contacto a una base de datos).
  - HEAD: Obtener la información de los metadatos sobre un recurso específico sin obtener el cuerpo entero tal como haría GET. Podrías, por ejemplo, usar una petición HEAD para encontrar la última vez que un recurso fue actualizado, y a continuación usar la petición GET (más "cara") para descargar el recurso sólo si éste ha cambiado.
  - PUT: Actualizar un recurso existente (o crear uno nuevo si no existe).
  - DELETE: Borrar el recurso específico.
  - TRACE, OPTIONS, CONNECT, PATCH: Estos verbos son para tareas menos comunes/avanzadas, por lo que no los trataremos aquí.
- Se puede codificar información adicional con la petición (por ejemplo, datos de un formulario HTML). La información puede ser codificada como:
  - Parámetros URL: Las peticiones GET codifican los datos en la URL enviada al servidor añadiendo al final pares nombre/valor — por ejemplo, http://mysite.com?name=Fred&age=11. Siempre encontrarás un signo de interrogación(?) separando los parámetros URL del resto de la misma, un signo igual (=) separando cada nombre de su valor asociado y un ampersand (&) separando cada par. Los parámetros URL son inherentemente "inseguros" ya que pueden ser modificados por los usuarios y ser enviados de nuevo. Como consecuencia los parámetros URL/peticiones GET no se usan para peticiones de actualización de datos en el servidor.

- Datos POST. Las peticiones POST añaden nuevos recursos, cuyos datos están codificados dentro del cuerpo de la petición.
- Cookies de lado cliente. Los Cookies contienen datos de sesión acerca del cliente, incluyendo las claves que el servidor puede usar para determinar su estado de incio de sesión y los permisos/accesos a los recursos.

Los servidores web esperan los mensajes de petición de los clientes, los procesan cuando llegan y responden al explorador web con un mensaje de respuesta HTTP. La respuesta contiene un código de estado de respuesta HTTP que indica si la petición ha tenido éxito o no (ej. "200 OK" para indicar éxito, "404 Not Found" si el resurso no ha podido ser encontrado, "403 Forbidden" si el usuario no está autorizado a acceder al recurso, etc). El cuerpo de la respuesta de éxito a una petición GET contendría el recurso solicitado.
Cuando se devuelve una página HTML es renderizada por el explorador web. Como parte del procesamiento el explorador puede descubrir enlaces a otros recursos (ej. una página HTML normalmente referencia las páginas JavaScript y CSS), y enviará peticiones HTTP separadas para descargar estos ficheros.
Los sitios web tanto estáticos como dinámicos (abordados en las secciones siguientes) usan exactamente los mismos protocolos/patrones de comunicación

Ejemplo de petición/respuesta GET

Puedes realizar una petición GET simplemente pinchando sobre un enlace o buscando en un sitio (como la página inicial de un motor de búsquedas). Por ejemplo, la petición HTTP que se envía cuando realizas una búsqueda en MDN del término "visión general cliente servidor" se parecerá mucho al texto que se muestra más abajo (no será idéntica porque algunas partes del mensaje dependen de tu explorador/configuración).

La petición

Cada línea de la petición contiene información sobre ella. La primera parte se llama cabecera o header, y contiene información útil sobre la petición, de la misma manera que un HTML head contiene información útil sobre un documento (pero no el contenido mismo, que está en el cuerpo):

```
GET https://developer.mozilla.org/en-US/search?q=client+server+overview&topic=apps&topic=html&topic=css&topic=js&topic=api&topic=webdev HTTP/1.1
Host: developer.mozilla.org
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: https://developer.mozilla.org/en-US/
Accept-Encoding: gzip, deflate, sdch, br
Accept-Charset: ISO-8859-1,UTF-8;q=0.7,*;q=0.7
Accept-Language: en-US,en;q=0.8,es;q=0.6
```

Cookie: sessionid=6ynxs23n521lu21b1t136rhbv7ezngie; csrftoken=zIPUJsAZv6pcgCBJSCj1zU6pQZbfMUAT; dwf_section_edit=False; dwf_sg_task_completion=False; _gat=1; _ga=GA1.2.1688886003.1471911953; ffo=true

La primera y segunda líneas contienen la mayoría de la información de la que hemos hablado arriba:
El tipo de petición (GET).
La URL del recurso de destino (/en-US/search).
Los parámetros URL (q=client%2Bserver%2Boverview&topic=apps&topic=html&topic=css&topic=js&topic=api&topic=webdev).
El destino/host del sitio web (developer.mozilla.org).
El final de la primera linea también incluye una cadena corta que identifica la versión del protocolo utilizado (HTTP/1.1).
La última linea contiene información sobre los cookies del lado cliente — puedes ver que en este caso el cookie incluye un id para gestionar las sesiones (Cookie: sessionid=6ynxs23n521lu21b1t136rhbv7ezngie; ...).
Las líneas restantes contienen información sobre el explorador web usado y la clase de respuestas que puede manejar. Por ejemplo, puedes ver aquí que:
Mi explorador (User-Agent) es Mozilla Firefox (Mozilla/5.0).
Puede aceptar información comprimida gzip (Accept-Encoding: gzip).
Puede aceptar el conjunto de caracteres especificado (Accept-Charset: ISO-8859-1,UTF-8;q=0.7,*;q=0.7) y los idiomas (Accept-Language: de,en;q=0.7,en-us;q=0.3).
La linea Referer indica la dirección de la página web que contenía el enlace a este recurso (es decir, el origen de la petición, https://developer.mozilla.org/en-US/).
Las peticiones HTTP también pueden tener un cuerpo, pero está vacío en este caso.

La respuesta

La primera parte de la respuesta a esta petición se muestra abajo. La cabecera o header contiene información como la siguiente:
La primera linea incluye el código de respuesta 200 OK, que nos dice que la petición ha tenido éxito.
Podemos ver que la respuesta está formateada (Content-Type) en modo text/html.
Podemos ver que usa también el conjunto de caracteres UTF-8 (Content-Type: text/html; charset=utf-8).
La cabecera también nos dice lo grande que es (Content-Length: 41823).
Al final del mensaje vemos el contenido del cuerpo (body) — que contiene el HTML real devuelto por la respuesta.

```
HTTP/1.1 200 OK
Server: Apache
X-Backend-Server: developer1.webapp.scl3.mozilla.com
Vary: Accept,Cookie, Accept-Encoding
Content-Type: text/html; charset=utf-8
Date: Wed, 07 Sep 2016 00:11:31 GMT
Keep-Alive: timeout=5, max=999
Connection: Keep-Alive
```

```
X-Frame-Options: DENY
Allow: GET
X-Cache-Info: caching
Content-Length: 41823

<!DOCTYPE html>
<html lang="en-US" dir="ltr" class="redesign no-js"  data-ffo-opensanslight=false data-ffo-opensans=false >
<head prefix="og: http://ogp.me/ns#">
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=Edge">
  <script>(function(d) { d.className = d.className.replace(/\bno-js/, '');
})(document.documentElement);</script>
  ...
```

El resto de la cabecera de la respuesta incluye información sobre la respuesta (ej. cuándo se generó), el servidor, y cómo espera el explorador manejar la página (ej. la linea X-Frame-Options: DENY le dice que al explorador que no está permitido incrustar esta página en un <iframe> en otro sitio).

Ejemplo de petición/respuesta POST

Un HTTP POST se realiza cuando se envía un formulario que contiene información para ser guardada en el servidor.

La petición

El texto de abajo muestra la petición HTTP realizada cuando un usuario envía al sitio los detalles de un nuevo perfil. El formato de la petición es casi el mismo que en la petición GET del ejemplo mostrado previamente, aunque la primera linea identifica esta petición como POST.

```
POST https://developer.mozilla.org/en-US/profiles/hamishwillee/edit HTTP/1.1
Host: developer.mozilla.org
Connection: keep-alive
Content-Length: 432
Pragma: no-cache
Cache-Control: no-cache
Origin: https://developer.mozilla.org
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: https://developer.mozilla.org/en-US/profiles/hamishwillee/edit
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.8,es;q=0.6
```

Cookie: sessionid=6ynxs23n521lu21b1t136rhbv7ezngie; _gat=1; csrftoken=zIPUJsAZv6pcgCBJSCj1zU6pQZbfMUAT; dwf_section_edit=False; dwf_sg_task_completion=False; _ga=GA1.2.1688886003.1471911953; ffo=true

csrfmiddlewaretoken=zIPUJsAZv6pcgCBJSCj1zU6pQZbfMUAT&user-username=hamishwillee&user-fullname=Hamish+Willee&user-title=&user-organization=&user-location=Australia&user-locale=en-US&user-timezone=Australia%2FMelbourne&user-irc_nickname=&user-interests=&user-expertise=&user-twitter_url=&user-stackoverflow_url=&user-linkedin_url=&user-mozillians_url=&user-facebook_url=

La principal diferencia es que la URL no tiene parámetros. Como puedes ver, la información del formulario se codifica en el cuerpo de la petición (por ejemplo, el nombre completo del nuevo usuario se establece usando: &user-fullname=Hamish+Willee).

La respuesta

La respuesta a la petición se muestra abajo. El código de estado "302 Found" le dice al explorador que el POST ha tenido éxito, y que debe realizar una segunda petición HTTP para cargar la página especificada en el campo Location. La información es de lo contrario similar a la respuesta a una petición GET.

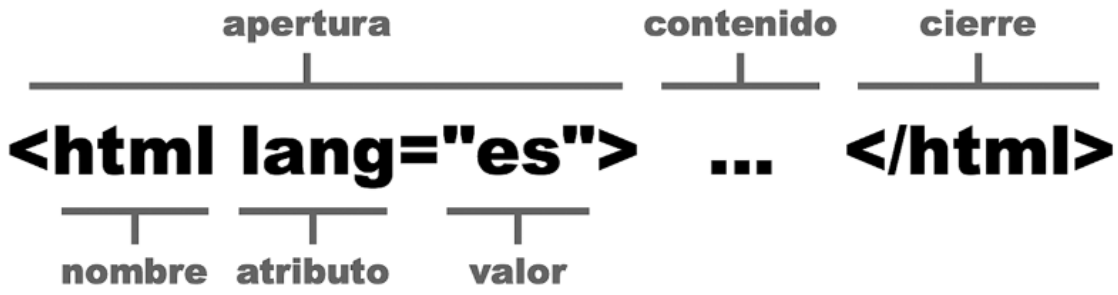
```
HTTP/1.1 302 FOUND
Server: Apache
X-Backend-Server: developer3.webapp.scl3.mozilla.com
Vary: Cookie
Vary: Accept-Encoding
Content-Type: text/html; charset=utf-8
Date: Wed, 07 Sep 2016 00:38:13 GMT
Location: https://developer.mozilla.org/en-US/profiles/hamishwillee
Keep-Alive: timeout=5, max=1000
Connection: Keep-Alive
X-Frame-Options: DENY
X-Cache-Info: not cacheable; request wasn't a GET or HEAD
Content-Length: 0
```

HTML

HTML (HyperText Markup Language) es un lenguaje compuesto por un grupo de etiquetas definidas con un nombre rodeado de paréntesis angulares. Los paréntesis angulares delimitan la etiqueta y el nombre define el tipo de contenido que representa.

Por ejemplo, la etiqueta <html> indica que el contenido es código HTML. Algunas de estas etiquetas son declaradas individualmente (por ejemplo,
) y otras son declaradas en pares, que incluyen una de apertura y otra de cierre, como <html></html> (en la etiqueta de cierre el nombre va precedido por una barra invertida). Las etiquetas individuales y las de apertura pueden incluir atributos para ofrecer información adicional acerca de sus contenidos (por ejemplo, <html lang="es">). Las etiquetas individuales y la combinación de etiquetas de apertura y cierre se llaman elementos. Los elementos compuestos por una sola etiqueta se usan para modificar el contenido que

los rodea o incluir recursos externos, mientras que los elementos que incluyen etiquetas de apertura y cierre se utilizan para delimitar el contenido del documento, tal como ilustra la Figura



Se deben combinar múltiples elementos para definir un documento. Los elementos son listados en secuencia de arriba abajo y pueden contener otros elementos en su interior. Por ejemplo, el elemento <html> que se muestra en la Figura 1-4 declara que su contenido debe ser interpretado como código HTML. Por lo tanto, el resto de los elementos que describen el contenido de ese documento se deben declarar entre las etiquetas <html> y </html>. A su vez, los elementos dentro del elemento <html> pueden incluir otros elementos.

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Mi primer documento HTML</title>
</head>
<body>
<p>HOLA MUNDO!</p>
</body>
</html>
```

En la primera línea, se encuentra una etiqueta individual que declara el tipo de documento (<!DOCTYPE html>) seguida por una etiqueta de apertura <html lang="es">.
Entre las etiquetas <html> y </html> se incluyen otros elementos que representan la cabecera y el cuerpo del documento (<head> y <body>), los cuales a su vez encierran más elementos con sus respectivos contenidos (<title> y <p>), demostrando cómo se compone un documento HTML. Los elementos se listan uno a continuación de otro y también dentro de otros elementos, de modo que se construye una estructura de tipo árbol con el elemento <html> como raíz.

Como ya mencionamos, las etiquetas individuales y de apertura pueden incluir atributos. Por ejemplo, la etiqueta de apertura <html> no está solo compuesta por el nombre html y los paréntesis angulares, sino también por el texto lang="es". Este es un atributo con un valor. El nombre del atributo es lang y el valor es se asigna al atributo usando el carácter =. Los atributos ofrecen información adicional acerca del elemento y su contenido. En este caso, el atributo lang declara el idioma del contenido del documento (es por Español).

Debido a que los navegadores son capaces de procesar diferentes tipos de archivos, lo primero que debemos hacer en la construcción de un documento HTML es indicar su tipo. Para asegurarnos de que el contenido de nuestros documentos sea interpretado correctamente como código HTML, debemos agregar la declaración <!DOCTYPE> al comienzo del archivo. Esta declaración, similar en formato a las etiquetas HTML, se requiere al comienzo de cada documento para ayudar al navegador a decidir cómo debe generar la página web. Para documentos programados con HTML5, la declaración debe incluir el atributo html, según la definimos en el siguiente ejemplo.
<!DOCTYPE html>

Elementos estructurales

Los elementos HTML conforman una estructura de tipo árbol con el elemento <html> como su raíz. Esta estructura presenta múltiples niveles de organización, con algunos elementos a cargo de definir secciones generales del documento y
otros encargados de representar secciones menores o contenido. Los siguientes son los elementos disponibles para definir la columna vertebral de la estructura y facilitar la información que el navegador necesita para mostrar la página en la pantalla.

<html>—Este elemento delimita el código HTML. Puede incluir el atributo lang para definir el idioma del contenido del documento.
<head>—Este elemento se usa para definir la información necesaria para configurar la página web, como el título, el tipo de codificación de caracteres y los archivos externos requeridos por el documento.
<body>—Este elemento delimita el contenido del documento (la parte visible de la página).

Después de declarar el tipo de documento, tenemos que construir la estructura de tipo árbol con los elementos HTML, comenzando por el elemento <html>. Este elemento puede incluir el atributo lang para declarar el idioma en el que vamos a escribir el contenido de la página, tal como muestra el siguiente ejemplo

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

El código HTML insertado entre las etiquetas <html> se tiene que dividir en dos secciones principales: la cabecera y el cuerpo. Por supuesto, la cabecera va primero y, al igual que el resto de los elementos estructurales, está compuesta por etiquetas de apertura y cierre

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

Entre las etiquetas <head> debemos definir el título de la página web, declarar el tipo de codificación de caracteres, facilitar información general acerca del documento, e incorporar los archivos externos con estilos y códigos necesarios para generar la página. Excepto por el título e iconos, el resto de la información insertada en medio de estas etiquetas no es visible para el usuario. La otra sección que forma parte de la organización principal de un documento HTML es el cuerpo, la parte visible del documento que se especifica con el elemento <body>.

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

La estructura básica ya está lista. Ahora tenemos que construir la página, comenzando por la definición de la cabecera. La cabecera incluye toda la información y los recursos necesarios para generar la página.

Los siguientes son los elementos disponibles para este propósito.

<title>—Este elemento define el título de la página.
<base>—Este elemento define la URL usada por el navegador para establecer la ubicación real de las URL relativas. El elemento debe incluir el atributo href para declarar la URL base. Cuando se declara este elemento, en lugar de la URL actual, el navegador usa la URL asignada al atributo href para completar las URL relativas.
<meta>—Este elemento representa metadatos asociados con el documento, como la descripción del documento, palabras claves, el tipo de codificación de caracteres, etc. El elemento puede incluir los atributos name para describir el tipo de metadata, content para especificar el valor, y charset para declarar el tipo de codificación de caracteres a utilizar para procesar el contenido.
<link>—Este elemento especifica la relación entre el documento y un recurso externo (generalmente usado para cargar archivos CSS). El elemento puede incluir los atributos href para declarar la ubicación del recurso, rel para definir el tipo de relación, media para especificar el medio al que el recurso está asociado (pantalla, impresora, etc.), y type y sizes para declarar el tipo de recurso y su tamaño (usado a menudo para cargar iconos).
<style>—Este elemento se usa para declarar estilos CSS dentro del documento
<script>—Este elemento se usa para cargar o declarar código JavaScript

Lo primero que tenemos que hacer cuando declaramos la cabecera del documento es especificar el título de la página con el elemento <title>. Este texto es el que muestran los navegadores en la parte superior de la ventana, y es lo que los usuarios ven cuando buscan información en nuestro sitio web por medio de motores de búsqueda como Google o Yahoo.

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

```
<title>Este texto es el título del documento</title>
</head>
<body>
</body>
</html>
```

Además del título, también tenemos que declarar los metadatos del documento. Los metadatos incluyen información acerca de la página que los navegadores, y también los motores de búsqueda, utilizan para generar y clasificar la página web. Los valores se declaran con el elemento <meta>. Este elemento incluye varios atributos, pero cuáles usemos dependerá del tipo de información que queremos declarar. Por ejemplo, el valor más importante es el que define la tabla de caracteres a utilizar para presentar el texto en pantalla, el cual se declara con el atributo charset.

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Este texto es el título del documento</title>
<meta charset="utf-8">
</head>
<body>
</body>
</html>
```

Se pueden incluir múltiples elementos <meta> para declarar información adicional. Por ejemplo, dos datos que los navegadores pueden considerar a la hora de procesar nuestros documentos son la descripción de la página y las palabras claves que identifican su contenido.
Estos elementos <meta> requieren el atributo name con los valores "description" y "keywords", y el atributo content con el texto que queremos asignar como descripción y palabras clave (las palabras clave se deben separar por comas).

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Este texto es el título del documento</title>
<meta charset="utf-8">
<meta name="description" content="Este es un documento HTML5">
<meta name="keywords" content="HTML, CSS, JavaScript">
</head>
<body>
</body>
</html>
```

Otro elemento importante de la cabecera del documento es <link>. Este elemento se usa para incorporar al documento estilos, códigos, imágenes o iconos desde archivos externos.

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

<head>
<title>Este texto es el título del documento</title>
<meta charset="utf-8">
<meta name="description" content="Este es un documento HTML5">
<meta name="keywords" content="HTML, CSS, JavaScript">
<link rel="icon" href="imagenes/favicon.png" type="image/png"
sizes="16x16">
</head>
<body>
</body>
</html>

El elemento <link> se usa comúnmente para cargar archivos CSS con los estilos necesarios para generar la página web. Por ejemplo, el siguiente documento carga el archivo misestilos.css. Después de cargar el archivo, todos los estilos declarados en su interior se aplican a los elementos del documento. En este caso, solo necesitamos incluir el atributo rel para declarar el tipo de recurso (para hojas de estilo CSS debemos asignar el valor "stylesheet") y el atributo href con la URL que determina la ubicación del archivo

<!DOCTYPE html>
<html lang="es">
<head>
<title>Este texto es el título del documento</title>
<meta charset="utf-8">
<meta name="description" content="Este es un documento HTML5">
<meta name="keywords" content="HTML, CSS, JavaScript">
<link rel="stylesheet" href="misestilos.css">
</head>
<body>
</body>
</html>

Con la cabecera lista, es hora de construir el cuerpo. Esta estructura (el código entre las etiquetas <body>) es la encargada de generar la parte visible de nuestro documento (la página web). HTML siempre ha ofrecido diferentes maneras de construir y organizar la información en el cuerpo del documento. Uno de los primeros elementos utilizados con este propósito fue <table> (tabla).

Este elemento permitía a los desarrolladores organizar datos, textos, imágenes, así como herramientas en filas y columnas de celdas. Con la introducción de CSS, la estructura generada por estas tablas ya no resultaba práctica, por lo que los desarrolladores comenzaron a implementar un elemento más flexible llamado (división). Pero ,
así como <table>, no facilita demasiada información acerca de las partes del cuerpo que representa.

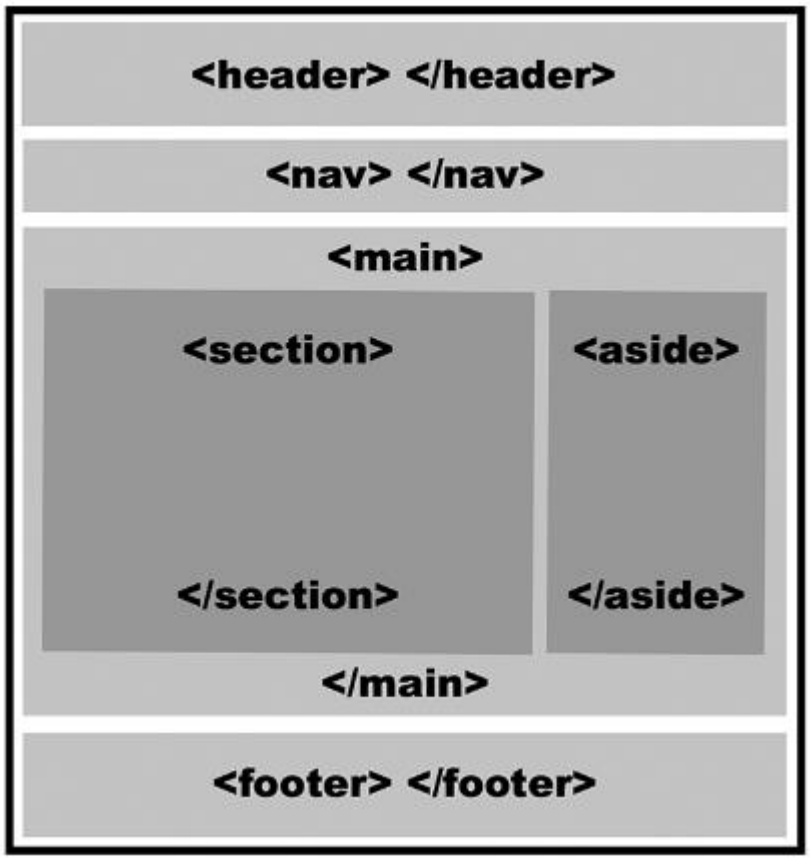
Cualquier cosa, desde imágenes hasta menús, texto, enlaces, códigos o formularios, se puede insertar entre las etiquetas de apertura y cierre de un elemento .

En otras palabras, el nombre div solo especifica una división en el cuerpo, como una celda en una tabla, pero no ofrece ninguna pista acerca del tipo de división que está creando, cuál es su propósito o qué contiene. Esta es la razón por la que HTML5 introdujo nuevos elementos con nombres más descriptivos que permiten a los desarrolladores identificar cada parte del documento. Estos elementos no solo ayudan a los desarrolladores a crear el documento, sino que además informan al navegador sobre el propósito de cada sección. La siguiente lista incluye todos los elementos disponibles para definir la estructura del cuerpo.

—Este elemento define una división genérica. Se usa cuando no se puede aplicar ningún otro elemento.
<main>—Este elemento define una división que contiene el contenido principal del documento (el contenido que representa el tema central de la página)
<nav>—Este elemento define una división que contiene ayuda para la navegación, como el menú principal de la página o bloques de enlaces necesarios para navegar en el sitio web.
<section>—Este elemento define una sección genérica. Se usa frecuentemente para separar contenido temático, o para generar columnas o bloques que ayudan a organizar el contenido principal.
<aside>—Este elemento define una división que contiene información relacionada con el contenido principal pero que no es parte del mismo, como referencias a artículos o enlaces que apuntan a publicaciones anteriores.
<article>—Este elemento representa un artículo independiente, como un mensaje de foro, el artículo de una revista, una entrada de un blog, un comentario, etc.
<header>—Este elemento define la cabecera del cuerpo o de secciones dentro del cuerpo.
<footer>—Este elemento define el pie del cuerpo o de secciones dentro del cuerpo.

A pesar de que cada desarrollador crea sus propios diseños, en general podremos describir todo sitio web considerando estas secciones. En la barra superior, descrita como cabecera, ubicamos el logo, el nombre del sitio, los subtítulos y una descripción breve de nuestro sitio o página web. En la barra de navegación situada debajo es donde la mayoría de los desarrolladores ofrecen un menú o una lista de enlaces para navegar en el sitio. El contenido relevante de la página se ubica en el medio del diseño, donde generalmente encontramos artículos
o noticias, y también enlaces a documentos relacionados o recursos.En la parte inferior de un diseño tradicional, nos encontramos con otra barra llamada barra institucional. La llamamos de este modo porque en este área es donde mostramos información general acerca del sitio web, el autor, la compañía, los enlaces
relacionados con reglas de uso, términos y condiciones, el mapa del sitio, etc.

Como mencionamos anteriormente, los elementos de HTML5 se han diseñado siguiendo este patrón. En la Figura siguiente aplicamos los elementos introducidos anteriormente para definir el diseño




```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Este texto es el título del documento</title>
<meta charset="utf-8">
<meta name="description" content="Este es un documento HTML5">

<meta name="keywords" content="HTML, CSS, JavaScript">
<link rel="stylesheet" href="misestilos.css">
</head>
<body>
<header>
Este es el título
</header>
<nav>
Principal | Fotos | Videos | Contacto
</nav>
<main>
<section>
<article>
Este es el texto de mi primer artículo
</article>
```

<article>
Este es el texto de mi segundo artículo
</article>
</section>
<aside>
Cita del artículo uno
Cita del artículo dos
</aside>
</main>
<footer>
© Derechos Reservados 2023
</footer>

</body>
</html>

Atributos globales

id—Este atributo nos permite asignar un identificador único a un elemento.
class—Este atributo asigna el mismo identificador a un grupo de elementos.

<main>
<section id="noticias">
Artículos largos
</section>
<section id="noticiaslocales">
Artículos cortos
</section>
<aside>
Quote from article one
Quote from article two
</aside>
</main>

Contenido

<h1>—Este elemento representa un título. El título se declara entre las etiquetas de apertura y cierre. HTML también incluye elementos adicionales para representar subtítulos, hasta seis niveles (<h2>, <h3>, <h4>, <h5>, y <h6>)
<p>—Este elemento representa un párrafo. Por defecto, los navegadores le asignan un margen en la parte superior para separar un párrafo de otro.
<pre>—Este elemento representa un texto con formato predefinido, como código de programación o un poema que requiere que los espacios asignados a cada carácter y los saltos de línea se muestren como se han declarado originalmente.

—Este elemento puede contener un párrafo, una frase o una palabra. No aplica ningún estilo al texto pero se usa para asignar estilos personalizados

—Este elemento se usa para insertar saltos de línea.
<wbr>—Este elemento sugiere la posibilidad de un salto de línea para ayudar al navegador a decidir dónde cortar el texto cuando no hay suficiente espacio para mostrarlo entero
—Este elemento se usa para indicar énfasis. El texto se muestra por defecto con letra cursiva.
—Este elemento se utiliza para indicar importancia. El texto se muestra por defecto en negrita.
<i>—Este elemento representa una voz alternativa o un estado de humor, como un pensamiento, un término técnico, etc. El texto se muestra por defecto con letra cursiva.
<u>—Este elemento representa texto no articulado. Por defecto se muestra subrayado.
<b>—Este elemento se usa para indicar importancia. Debería ser implementado solo cuando ningún otro elemento es apropiado para la situación. El texto se muestra por defecto en negrita
<mark>—Este elemento resalta texto que es relevante en las circunstancias actuales (por ejemplo, términos que busca el usuario).
<small>—Este elemento representa letra pequeña, como declaraciones legales, descargos, etc.
<cite>—Este elemento representa el autor o título de una obra, como un libro, una película, etc.
<address>—Este elemento representa información de contacto. Se implementa con frecuencia dentro de los pies de página para definir la dirección de la empresa o el sitio web
<time>—Este elemento representa una fecha en formato legible para el usuario. Incluye el atributo datetime para especificar un valor en formato de ordenador y el atributo pubdate, el cual indica que el valor asignado al atributo datetime representa la fecha de publicación.
<code>—Este elemento representa código de programación. Se usa en conjunto con el elemento <pre> para presentar código de programación en el formato original.
<data>—Este elemento representa datos genéricos. Puede incluir el atributo value para especificar el valor en formato de ordenador.

Enlaces

Conectar documentos con otros documentos mediante enlaces es lo que hace posible la Web. Como mencionamos anteriormente, un enlace es contenido asociado a una URL que indica la ubicación de un recurso. Cuando el usuario hace clic en el contenido (texto o imagen), el navegador descarga el recurso. HTML incluye el siguiente elemento para crear enlaces.
<a>—Este elemento crea un enlace. El texto o la imagen que representa el enlace se incluye entre las etiquetas de apertura y cierre. El elemento incluye el atributo href para especificar la URL del enlace.

Imágenes

Las imágenes pueden se pueden considerar el segundo medio más importante en la Web. HTML incluye los siguientes elementos para introducir imágenes en nuestros documentos.
<img>—Este elemento inserta una imagen en el documento. El elemento requiere del atributo src para especificar la URL del archivo con la imagen que queremos incorporar.

<picture>—Este elemento inserta una imagen en el documento. Trabaja junto con el elemento <source> para ofrecer múltiples imágenes en diferentes resoluciones. Es útil para crear sitios web adaptables.
<figure>—Este elemento representa contenido asociado con el contenido principal, pero que se puede eliminar sin que se vea afectado, como fotos, vídeos, etc.
<figcaption>—Este elemento introduce un título para el elemento <figure>.

Listados

A menudo, la información se debe representar como una lista de ítems. Por ejemplo, muchos sitios web incluyen listados de libros, películas, o términos y descripciones. Para crear estos listados, HTML ofrece los siguientes elementos.

—Este elemento crea una lista de ítems sin orden. Está compuesto por etiquetas de apertura y cierre para agrupar los ítems (<ul> y) y trabaja junto con el elemento <li> para definir cada uno de los ítems de la lista.
—Este elemento crea una lista ordenada de ítems. Está compuesto por etiquetas de apertura y cierre para agrupar los ítems (<ol> y) y trabaja junto con el elemento <li> para definir los ítems de la lista. Este elemento puede incluir los atributos reversed para invertir el orden de los indicadores, start para determinar el valor desde el cual los indicadores tienen que comenzar a contar y type para determinar el tipo de indicador que queremos usar. Los valores disponibles para el atributo type son 1 (números), a (letras minúsculas), A (letras mayúsculas), i (números romanos en minúsculas) e I (números romanos en mayúsculas).
<dl>—Este elemento crea una lista de términos y descripciones. El elemento trabaja junto con los elementos <dt> y <dd> para definir los ítems de la lista. El elemento <dl> define la lista, el elemento <dt> define los términos y el elemento <dd> define las descripciones.

Tablas

Las tablas organizan información en filas y columnas. Debido a sus características, se usaron durante mucho tiempo para estructurar documentos HTML, pero con la introducción de CSS, los desarrolladores pudieron lograr el mismo efecto implementando otros elementos. Aunque ya no se recomienda usar tablas para definir la estructura de un documento, todavía se utilizan para presentar información tabular, como estadísticas o especificaciones técnicas, por ejemplo. HTML incluye varios elementos para crear una tabla. Los siguientes son los más utilizados.
<table>—Este elemento define una tabla. Incluye etiquetas de apertura y cierre para agrupar el resto de los elementos que definen la tabla.
<tr>—Este elemento define una fila de celdas. Incluye etiquetas de apertura y cierre para agrupar las celdas.
<td>—Este elemento define una celda. Incluye etiquetas de apertura y cierre para delimitar el contenido de la celda y puede incluir los atributos colspan y rowspan para indicar cuántas columnas y filas ocupa la celda.
<th>—Este elemento define una celda para la cabecera de la tabla. Incluye etiquetas de apertura y cierre para delimitar el contenido de la celda y puede incluir los atributos colspan y rowspan para indicar cuántas columnas y filas ocupa la celda.

Formularios

Los formularios son herramientas que podemos incluir en un documento para permitir a los usuarios insertar información, tomar decisiones, comunicar datos y cambiar el comportamiento de una aplicación. El propósito principal de los formularios es permitir al usuario seleccionar o insertar información y enviarla al servidor para ser procesada.

Los formularios pueden presentar varias herramientas que
permiten al usuario interactuar con el documento, incluidos campos de texto, casillas de control, menús desplegables y botones. Cada una de estas herramientas se representa por un elemento y el formulario queda definido por el elemento <form>, que incluye etiquetas de apertura y cierre para agrupar al resto de los elementos y requiere de algunos atributos para determinar cómo se envía la información al servidor.

name—Este atributo especifica el nombre del formulario. También se encuentra disponible para otros elementos, pero es particularmente útil para elementos de formulario, como veremos más adelante.
method—Este atributo determina el método a utilizar para enviar la información al servidor. Existen dos valores disponibles: GET y POST. El método GET se usa para enviar una cantidad limitada de información de forma pública (los datos son incluidos en la URL, la cual no puede contener más de 255 caracteres). Por otro lado, el método POST se utiliza para enviar una cantidad ilimitada de información de forma privada (los datos no son visibles al usuario y pueden tener la longitud que necesitemos).
action—Este atributo declara la URL del archivo en el servidor que va a procesar la información enviada por el formulario.
target—Este atributo determina dónde se mostrará la respuesta recibida desde el servidor. Los valores disponibles son _blank (nueva ventana), _self (mismo recuadro), _parent (recuadro padre), y _top (la ventana que contiene el recuadro). El valor _self se declara por defecto, lo que significa que la respuesta recibida desde el servidor se mostrará en la misma ventana enctype—Este atributo declara la codificación aplicada a los datos que envía el formulario. Puede tomar tres valores: application/x-www-form-urlencoded (los
caracteres son codificados), multipart/form-data (los caracteres no son codificados), text/plain (solo los espacios son codificados). El primer valor se asigna por defecto.
accept-charset—Este atributo declara el tipo de codificación aplicada al texto del
formulario. Los valores más comunes son UTF-8 e ISO-8859-1. El valor por defecto se asigna al documento con el elemento <meta>.

El siguiente ejemplo define un formulario básico. El atributo name identifica el formulario con el nombre "formulario", el atributo method determina que los datos se incluirán en la URL (GET), y el atributo action declara que procesar.php es el archivo que se ejecutará en el servidor para procesar la información y devolver el resultado.

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Formularios</title>
```

```
</head>
<body>
<section>
<form name="formulario" method="get" action="procesar.php">
</form>
</section>
</body>
</html>
```

Elementos

Un formulario puede incluir diferentes herramientas para permitir al usuario seleccionar o insertar información. HTML incluye múltiples elementos para crear estas herramientas. Los siguientes son los más utilizados.

<input>—Este elemento crea un campo de entrada. Puede recibir diferentes tipos de entradas, dependiendo del valor del atributo type.
<textarea>—Este elemento crea un campo de entrada para insertar múltiples líneas de texto. El tamaño se puede declarar en números enteros usando los atributos rows y cols, o en píxeles con estilos CSS
<select>—Este elemento crea una lista de opciones que el usuario puede elegir. Trabaja junto con el elemento <option> para definir cada opción y el elemento <optgroup> para organizar las opciones en grupos.
<button>—Este elemento crea un botón. Incluye el atributo type para definir el propósito del botón. Los valores disponibles son submit para enviar el formulario (por defecto), reset para reiniciar el formulario, y button para realizar tareas personalizadas.
<output>—Este elemento representa un resultado producido por el formulario. Se implementa por medio de código JavaScript para mostrar el resultado de una operación al usuario.
<meter>—Este elemento representa una medida o el valor actual de un rango.
<progress>—Este elemento representa el progreso de una operación.
<datalist>—Este elemento crea un listado de valores disponibles para otros controles. Trabaja junto con el elemento <option> para definir cada valor.
<label>—Este elemento crea una etiqueta para identificar un elemento de formulario.
<fieldset>—Este elemento agrupa otros elementos de formulario. Se usa para crear secciones dentro de formularios extensos. El elemento puede contener un elemento <legend> para definir el título de la sección.

El elemento <input> es el más versátil de todos. Este elemento genera un campo de entrada en el que el usuario puede seleccionar o insertar información, pero puede adoptar diferentes características y aceptar varios tipos de valores dependiendo del valor de su atributo type. Los siguientes son los valores disponibles para este atributo.

text—Este valor genera un campo de entrada para insertar texto genérico.
email—Este valor genera un campo de entrada para insertar cuentas de correo.
search—Este valor genera un campo de entrada para insertar términos de búsqueda.

url—Este valor genera un campo de entrada para insertar URL.
tel—Este valor genera un campo de entrada para insertar números de teléfono.
number—Este valor genera un campo de entrada para insertar números.
range—Este valor genera un campo de entrada para insertar un rango de números.
date—Este valor genera un campo de entrada para insertar una fecha.
datetime-local—Este valor genera un campo de entrada para insertar fecha y hora.
week—Este valor genera un campo de entrada para insertar el número de la semana (dentro del año).
month—Este valor genera un campo de entrada para insertar el número del mes.
time—Este valor genera un campo de entrada para insertar una hora (horas y minutos).
hidden—Este valor oculta el campo de entrada. Se usa para enviar información complementaria al servidor.
password—Este valor genera un campo de entrada para insertar una clave. Reemplaza los caracteres insertados con estrellas o puntos para ocultar información sensible.
color—Este valor genera un campo de entrada para insertar un color.
checkbox—Este valor genera una casilla de control que permite al usuario activar o desactivar una opción.
radio—Este valor genera un botón de opción para seleccionar una opción de varias posibles.
file—Este valor genera un campo de entrada para seleccionar un archivo en el ordenador del usuario.
button—Este valor genera un botón. El botón trabaja como el elemento <button> de tipo button. No realiza ninguna acción por defecto; la acción debe ser definida desde JavaScript, como veremos en próximos capítulos.
submit—Este valor genera un botón para enviar el formulario.
reset—Este valor genera un botón para reiniciar el formulario.
image—Este valor carga una imagen que se usa como botón para enviar el formulario. Un elemento <input> de este tipo debe incluir el atributo src para especificar la URL de la imagen.

CSS

Hemos aprendido a crear un documento HTML, a organizar su estructura, y a determinar qué elementos son más apropiados para representar su contenido. Esta información nos permite definir el documento, pero no determina cómo se mostrará en pantalla.
Desde la introducción de HTML5, esa tarea es responsabilidad de CSS. CSS es un lenguaje que facilita instrucciones que podemos usar para asignar estilos a los elementos HTML, como colores, tipos de letra, tamaños, etc. Los estilos se deben definir con CSS y luego asignar a los elementos hasta que logramos el diseño visual que queremos para nuestra página.
Por razones de compatibilidad, los navegadores asignan estilos por defecto a algunos elementos HTML. Algunos de estos estilos son útiles, pero la mayoría deben ser reemplazados o complementados con estilos personalizados. En CSS, los estilos personalizados se declaran con propiedades. Un estilo se define declarando el nombre de la propiedad y su valor separados por dos puntos.

La mayoría de las propiedades CSS pueden modificar un solo aspecto del elemento (el tamaño de la letra en este caso). Si queremos cambiar varios estilos al mismo tiempo, tenemos que declarar múltiples propiedades. CSS define una sintaxis que simplifica el proceso de asignar múltiples

propiedades a un elemento. La construcción se llama regla. Una regla es una lista de propiedades declaradas entre llaves e identificadas por un selector. El selector indica qué elementos se verán afectados por las propiedades.

Aplicando estilos

Las propiedades y las reglas definen los estilos que queremos asignar a uno o más elementos, pero estos estilos no se aplican hasta que los incluimos en el documento. Existen tres técnicas disponibles para este propósito. Podemos usar estilos en línea, estilos incrustados u hojas de estilo. El primero, estilos en línea, utiliza un atributo global llamado style para insertar los estilos directamente en el elemento. Este atributo está disponible en cada uno de los elementos HTML y puede recibir una propiedad o una lista de propiedades que se aplicarán al elemento al que pertenece. Si queremos asignar estilos usando esta técnica, todo lo que tenemos que hacer es declarar el atributo style en el elemento que queremos modificar y asignarle las propiedades CSS.

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Este texto es el título del documento</title>
<meta charset="utf-8">
</head>
<body>
<main>
<section>
<p style="font-size: 20px;">Mi Texto</p>
</section>
</main>
</body>
</html>
```

Los estilos en línea son una manera práctica de probar estilos y ver cómo modifican los elementos, pero no se recomiendan para proyectos extensos. La razón es que el atributo style solo afecta al elemento en el que se ha declarado. Si queremos asignar el mismo estilo a otros elementos, tenemos que repetir el código en cada uno de ellos, lo cual incrementa innecesariamente el tamaño del documento, complicando su actualización y mantenimiento.

Por ejemplo, si más adelante decidimos que en lugar de 20 píxeles el tamaño del texto en cada elemento <p> deber ser de 24 píxeles, tendríamos que cambiar cada estilo en cada uno de los elementos <p> del documento completo y en todos los documentos de nuestro sitio web.
Una alternativa es la de insertar las reglas CSS en la cabecera del documento usando selectores que determinan los elementos que se verán afectados. Para este propósito, HTML incluye el elemento <style>.

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Este texto es el título del documento</title>
```

```
<meta charset="utf-8">
<style>
p {
font-size: 20px;
}
</style>
</head>
<body>
<main>
<section>
<p>Mi texto</p>
</section>
</main>
</body>
</html>
```

Declarar estilos en la cabecera del documento ahorra espacio y hace que el código sea más coherente y fácil de mantener, pero requiere que copiemos las mismas reglas en cada uno de los documentos de nuestro sitio web. Debido a que la mayoría de las páginas compartirán el mismo diseño, varias de estas reglas se duplicarán. La solución es mover los estilos a un archivo CSS y luego usar el elemento <link> para cargarlo desde cada uno de los documentos que lo requieran.

Este tipo de archivos se denomina hojas de estilo, pero no son más que archivos de texto con la lista de reglas CSS que queremos asignar a los elementos del documento.

El elemento <link> se usa para incorporar recursos externos al documento. Dependiendo del tipo de recurso que queremos cargar, tenemos que declarar diferentes atributos y valores. Para cargar hojas de estilo CSS, solo necesitamos los
atributos rel y href. El atributo rel significa relación y especifica la relación entre el documento y el archivo que estamos incorporando, por lo que debemos declararlo con el valor stylesheet para comunicarle al navegador que el recurso es un archivo CSS con los estilos requeridos para presentar la página. Por otro lado, el atributo href declara la URL del archivo a cargar.

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Este texto es el título del documento</title>
<meta charset="utf-8">
<link rel="stylesheet" href="misestilos.css">
</head>
<body>
<main>
<section>
<p>Mi texto</p>
</section>
</main>
```

</body>
</html>

El documento carga los estilos CSS desde el archivo misestilos.css. En este archivo tenemos que declarar las reglas CSS que queremos aplicar al documento, tal como lo hemos hecho anteriormente dentro del elemento <style>. El siguiente es el código que debemosinsertar en el archivo misestilos.css para producir el mismo efecto logrado en ejemplos anteriores.

```
p {
font-size: 20px;
}
```

La práctica de incluir estilos CSS en un archivo aparte es ampliamente utilizada por diseñadores y se recomienda para sitos web diseñados con HTML5, no solo porque podemos definir una sola hoja de estilo y luego incluirla en todos los documentos con el elemento <link>, sino porque podemos reemplazar todos los estilos a la vez simplemente cargando un archivo diferente, lo cual nos permite probar diferentes diseños y adaptar el sitio web a las pantallas de todos los dispositivos disponibles

Hojas de estilo en cascada

Una característica importante del CSS es que los estilos se asignan en cascada (de ahí el nombre hojas de estilo en cascada o Cascading Style Sheets en inglés). Los elementos en los niveles bajos de la jerarquía heredan los estilos asignados a los elementos en los niveles más altos. Por ejemplo, si asignamos la regla a los elementos <section> en lugar de los elementos <p>, como se muestra a continuación, el texto en el elemento <p> de nuestro documento se mostrará con un tamaño de 20 píxeles debido a que este elemento es hijo del elemento <section> y, por lo tanto, hereda sus estilos.

```
section {
font-size: 20px;
}
```

Los estilos heredados de elementos en niveles superiores se pueden reemplazar por nuevos estilos definidos para los elementos en niveles inferiores de la jerarquía. Por ejemplo, podemos declarar una regla adicional para los elementos <p> que sobrescriba la propiedad font-size definida para el elemento <section> con un valor diferente.

```
section {
font-size: 20px;
}
p {
font-size: 36px;
}
```

Introducción a JavaScript

HTML y CSS incluyen instrucciones para indicar al navegador cómo debe organizar y visualizar un documento y su contenido, pero la interacción de estos lenguajes con el usuario y el sistema se limita solo a un grupo pequeño de respuestas predefinidas. Podemos crear un formulario con campos de entrada, controles y botones, pero HTML solo provee la funcionalidad necesaria para enviar la información introducida por el usuario al servidor o para limpiar el formulario. Algo similar pasa con CSS; podemos construir instrucciones (reglas) con seudoclases como :hover para aplicar un grupo diferente de propiedades cuando el usuario mueve el ratón sobre un elemento, pero si queremos realizar tareas personalizadas, como modificar los estilos de varios elementos al mismo tiempo, debemos cargar una nueva hoja de estilo que ya presente estos cambios. Con el propósito de alterar elementos de forma dinámica, realizar operaciones personalizadas, o responder al usuario y a cambios que ocurren en el documento, los navegadores incluyen un tercer lenguaje llamado JavaScript.
JavaScript es un lenguaje de programación que se usa para procesar información y manipular documentos. Al igual que cualquier otro lenguaje de programación, JavaScript provee instrucciones que se ejecutan de forma secuencial para indicarle al sistema lo que queremos que haga (realizar una operación aritmética, asignar un nuevo valor a un elemento, etc.). Cuando el navegador encuentra este tipo de código en nuestro documento, ejecuta las instrucciones al momento y cualquier cambio realizado en el documento se muestra en pantalla.

Implementando JavaScript

Siguiendo el mismo enfoque que CSS, el código JavaScript se puede incorporar al documento mediante tres técnicas diferentes: el código se puede insertar en un elemento por medio de atributos (En línea), incorporar al documento como contenido del elemento <script> o cargar desde un archivo externo. La técnica En línea aprovecha atributos especiales que describen un evento, como un clic del ratón. Para lograr que un elemento responda a un evento usando esta técnica, todo lo que tenemos que hacer es agregar el atributo correspondiente con el código que queremos que se ejecute.

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>JavaScript</title>
</head>
<body>
<section>
<p onclick="alert('Hizo clic!')">Clic aquí</p>
<p>No puede hacer clic aquí</p>
</section>

</body>
</html>
```

La lista de atributos disponibles es extensa, pero se pueden organizar en grupos

dependiendo de sus propósitos. Por ejemplo, los siguientes son los atributos más usados asociados con el ratón.
onclick—Este atributo responde al evento click. El evento se ejecuta cuando el usuario hace clic con el botón izquierdo del ratón. HTML ofrece otros dos atributos similares llamados ondblclick (el usuario hace doble clic con el botón izquierdo del ratón) y oncontextmenu (el usuario hace clic con el botón derecho del ratón).
onmousedown—Este atributo responde al evento mousedown. Este evento se desencadena cuando el usuario pulsa el botón izquierdo o el botón derecho del ratón.
onmouseup—Este atributo responde al evento mouseup. El evento se desencadena cuando el usuario libera el botón izquierdo del ratón.
onmouseenter—Este atributo responde al evento mouseenter. Este evento se desencadena cuando el ratón se introduce en el área ocupada por el elemento.
onmouseleave—Este atributo responde al evento mouseleave. Este evento se desencadena cuando el ratón abandona el área ocupada por el elemento.
onmouseover—Este atributo responde al evento mouseover. Este evento se desencadena cuando el ratón se mueve sobre el elemento o cualquiera de sus elementos hijos.
onmouseout—Este atributo responde al evento mouseout. El evento se desencadena cuando el ratón abandona el área ocupada por el elemento o cualquiera de sus elementos hijos.
onmousemove—Este atributo responde al evento mousemove. Este evento se desencadena cada vez que el ratón se encuentra sobre el elemento y se mueve.
onwheel—Este atributo responde al evento wheel. Este evento se desencadena cada vez que se hace girar la rueda del ratón.

Los siguientes son los atributos disponibles para responder a eventos generados por el teclado.

Estos tipos de atributos se aplican a elementos que aceptan una entrada del usuario, como los elementos <input> y <textarea>.

onkeypress—Este atributo responde al evento keypress. Este evento se desencadena cuando se activa el elemento y se pulsa una tecla.
onkeydown—Este atributo responde al evento keydown. Este evento se desencadena cuando se activa el elemento y se pulsa una tecla.
onkeyup—Este atributo responde al evento keyup. Este evento se desencadena cuando se activa el elemento y se libera una tecla.

También contamos con otros dos atributos importantes asociados al documento:

onload—Este atributo responde al evento load. El evento se desencadena cuando un recurso termina de cargarse.
onunload—Este atributo responde al evento unload. Este evento se desencadena cuando un recurso termina de cargarse.

Servidor de aplicaciones web

“Básicamente, un servidor de aplicaciones consiste en un contenedor que abarca la lógica de negocio de un sistema, y que provee respuestas a las peticiones de distintos dispositivos que tienen acceso a ella. Son un claro ejemplo del modelo cliente-servidor, cuyo lado cliente ejecuta requerimientos de procesamiento al otro lado, donde el servidor se encarga de procesar y responder.”

En la actualidad existe una gama muy amplia de tipos de servidores de aplicación, basados en distintas tecnologías. Los más usados son los que funcionan con arquitectura J2EE 7 hechos con tecnología Java, ya que garantizan multiplataformidad. Entre ellos se puede encontrar:

- JBoss.
- GlassFish.
- Oracle application server.
- Jetty.
- IBM WebSphere.
- JOnAS
- Geronimo

Características

Los servidores de aplicaciones incluyen middleware (o software de conectividad) que les permite intercomunicarse con otros servicios, para efectos de confiabilidad, seguridad y no-repudio. Estos servidores también brindan a los desarrolladores una Interfaz para Programación de Aplicaciones (API), de tal manera que no tengan que preocuparse por el sistema operativo o por la gran cantidad de interfaces requeridas en una aplicación web moderna.
Brindan soporte a una gran variedad de estándares, tales como HTML, XML, IIOP, JDBC, SSL, etc., que les permiten su funcionamiento en ambientes web (como Internet) y la conexión a una gran variedad de fuentes de datos, sistemas y dispositivos.

Un servidor de aplicaciones es un programa de servidor en un equipo en una red distribuida que proporciona la lógica de negocio para un programa de aplicación. El servidor de aplicaciones se ve frecuentemente como parte de una aplicación de tres niveles, que consta de un servidor gráfico de interfaz de usuario (GUI), un servidor de aplicaciones (lógica empresarial) y un servidor de bases de datos y transacciones. De manera más descriptiva, se puede visualizar como la división de una aplicación en:

Una interfaz gráfica de usuario de primer nivel, de front-end, basada en el navegador web, normalmente en un equipo de cómputo personal o una estación de trabajo.

Una aplicación de lógica de negocio de nivel medio o conjunto de aplicaciones, posiblemente en una red de área local o un servidor de intranet.
Un servidor de back-end, base de datos y transacciones de tercer nivel, a veces en un mainframe o servidor grande.
Las bases de datos de aplicaciones más antiguas y las aplicaciones de administración de transacciones forman parte del backend o tercer nivel. El servidor de aplicaciones es el intermediario entre bases de datos basadas en navegador y bases de datos de back-end y sistemas heredados.
En muchos usos, el servidor de aplicaciones combina o funciona con un servidor Web (Hypertext Transfer Protocol) y se denomina servidor de aplicaciones web. El navegador web admite un front-end fácil de crear basado en HTML para el usuario. El servidor Web proporciona varias formas diferentes de reenviar una solicitud a un servidor de aplicaciones y de reenviar una página Web nueva o modificada al usuario. Estos enfoques incluyen la interfaz de gateway común (CGI), FastCGI, la página Active Server de Microsoft y la página de servidor de Java. En algunos casos, los servidores de aplicaciones Web también admiten interfaces de "intermediación" de solicitud, como el Protocolo de internet Inter-ORB (IIOP) de CORBA.

Lectura requerida:
Gauchat, J. D. (2017). El gran libro de HTML5, CSS3 y JavaScript. Barcelona: Marcombo.

Lectura ampliatoria:
https://es.wikipedia.org/wiki/Familia_de_protocolos_de_internet
https://es.wikipedia.org/wiki/Sitio_web

https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Introduction

https://developer.mozilla.org/es/docs/Learn/Common_questions/How_does_the_Internet_work

https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Client-Server_overview

Tonina, Marcos. Disponible en "http://es.over-blog.com/Que_es_un_servidor_de_aplicaciones-1228321779-art127891.html". [citado 2015 septiembre,9].

Tesis Pompa Rodríguez, Leandro. Repositorio Institucional de la Universidad de las Ciencias Informáticas. Disponible en "http://repositorio_institucional.uci.cu". [citado 2015 septiembre,9].

https://www.computerweekly.com/es/definicion/Servidor-de-aplicaciones-y-proveedor-de-servicios-de-aplicaciones

https://es.wikipedia.org/wiki/Servidor_de_aplicaciones