UNIDAD 1

Fundamentos de la Web

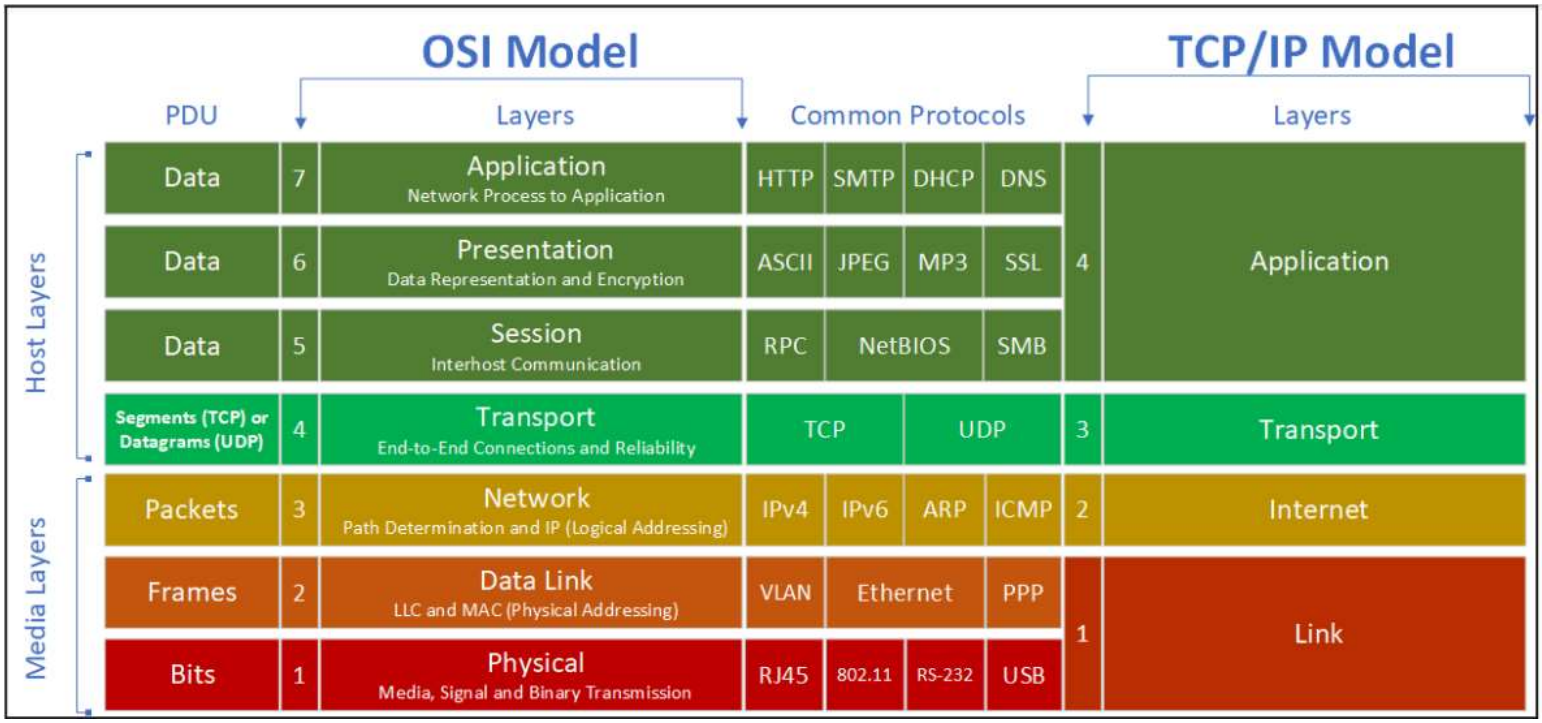
Protocolos de comunicación

Los protocolos más comunes están: TCP, IP, ARP o HTTP. A un conjunto de protocolos que trabajan juntos se le llama pila de protocolos.

Los protocolos tienen que lidiar con los siguientes problemas:

- **Inicio de la conexión:** Inicia el servidor o el cliente, que se necesita
- **Negociación de las características de la conexión**: Cifrado y encriptado
- **Formato de los datos:** Formato del paquete y orden de lectura
- **Detección y corrección de errores:** Demasiado tiempo de respuesta o desconexión
- **Fin de la conexión:** Indicar fin de conexión

El modelo OSI o también llamado la pila de capas o niveles del modelo OSI, es un modelo de referencia ideal para los protocolos de la red, los cuales están separados de acuerdo a sus funciones. Está formado por siete capas o layers:



- Las Host layers son las responsables de proporcionar una entrega precisa de los datos entre los equipos.
- Las Media layers son las responsables de controlar la entrega física de los datos a través de la red.

- La capa 7 o de **aplicación** es el nivel más alto del modelo OSI, es la capa vista por los usuarios finales. (HTTP, SMPT, FTP, etc.)
- La capa 6 o de **presentación** se encarga de transformar los datos que recibe en un formato que pueda ser leído por la capa de aplicación o por las capas inferiores. (ASCII, JPEG, MP3)
- La capa 5 o de **sesión** controla el diálogo entre los dispositivos. Establece, gestiona, mantiene y termina las conexiones entre los equipos involucrados en la comunicación.
- La capa 4 o de **transporte** tiene como principal objetivo garantizar la correcta transferencia de los datos entre equipos (protocolos TCP y UDP)
- La capa 3 o de **red** es responsable de enrutar los datos entre las distintas redes físicas. Utiliza las direcciones lógicas de los dispositivos de red. También a veces se encarga de dividir los datos en fragmentos más pequeños y de la detección de errores. (IPv4, IPv6)
- La capa 2 o de **enlace** proporciona la transferencia de datos de nodo a nodo, un enlace o medio de transporte entre dos o más dispositivos directamente conectados. Generalmente se divide en dos subcapas: LLC que define la forma de transferencia de los datos y MAC que se encarga del direccionamiento. (Ethernet, VLAN, etc.)
- La capa 1 o **física** es el medio físico por el cual los datos de red son transferidos, la información se transfiere en bits. (cables como el RJ45 o el 802.11, USB, etc.)

Encapsulación de los datos

Los protocolos de las capas del modelo OSI intercambian los datos entre ellas con la ayuda de la encapsulación. El proceso de la encapsulación da como resultado una unidad de protocolo de datos o **PDU**, el cual incluye los datos que se envían y además todas las headers o footers añadidos.

Arquitectura Cliente-Servidor y Protocolo HTTP

La arquitectura cliente-servidor es un modelo de diseño de software que distribuye las funciones de una aplicación entre dos entidades principales: el cliente y el servidor.

- **Cliente**: Es la interfaz de usuario o la aplicación que solicita servicios y consume recursos proporcionados por el servidor. Su función principal es enviar solicitudes al servidor y mostrar los resultados al usuario.
- **Servidor**: Es el componente que almacena, procesa y gestiona los datos y recursos de la aplicación. Recibe las solicitudes del cliente, las procesa y envía las respuestas adecuadas.

Esta comunicación puede ser síncrona, donde el cliente espera una respuesta inmediata del servidor, o asíncrona, donde el cliente continúa su ejecución mientras espera la respuesta del servidor en segundo plano.

De forma básica, cuando un usuario se conecta a Internet con un dispositivo cualquiera, se le asigna un identificador único mediante los protocolos **TCP/IP**. El protocolo TCP proporciona el medio para crear las conexiones y el protocolo IP proporciona el mejor “camino” para alcanzar su destino.

Este identificador único, más conocido como **dirección IP**, suele estar compuesto por cuatro códigos de 8 bits y vinculado a un nombre único (dominio) como https://google.es.

Un **DNS** sirve cuando un usuario ingresa un nombre de dominio en su navegador, el DNS busca la dirección IP correspondiente a ese nombre.

PROTOCOLO HTTP

Una vez que se tiene el objetivo al que dirigirse, el cliente en términos de comunicaciones, abre una instancia de comunicación con el Servidor mediante el **protocolo HTTP**, que es quién dicta las normas la comunicación y es, además, quién define la sintaxis y semántica que se debe utilizar en cada conexión.

HTTPS es la versión segura de HTTP. Incorpora una capa adicional de seguridad mediante el uso de certificados SSL o TLS para cifrar las comunicaciones entre el navegador del usuario y el servidor web

El protocolo HTTP define una serie de **métodos** que especifican la acción que el cliente desea realizar sobre un recurso determinado en el servidor.

- **GET**: Solicita la recuperación de un recurso específico del servidor.
- **POST**: Envía datos al servidor para que sean procesados. Es utilizado comúnmente en formularios web
- **PUT**: Envía datos al servidor para ser almacenados en un recurso específico.
- **DELETE**: Solicita la eliminación de un recurso específico en el servidor.

Además de los métodos, el protocolo HTTP define **códigos de estado** que indican el resultado de la solicitud realizada por el cliente. Algunos de los códigos de estado más comunes son:

- **200 OK**: Indica que la solicitud se ha completado con éxito y que el servidor ha devuelto los datos solicitados al cliente.
- **404 Not Found**: Indica que el recurso solicitado no se ha encontrado en el servidor.
- **500 Internal Server Error**: Indica que se ha producido un error interno en el servidor

Las **cabeceras HTTP** son componentes clave de las solicitudes y respuestas HTTP que se intercambian entre clientes y servidores. Estas cabeceras proporcionan información adicional sobre la solicitud o la respuesta, lo que permite una comunicación más completa y eficiente entre los componentes de la aplicación web.

Tipos de Cabeceras HTTP:

- **Cabeceras de Solicitud:** Enviadas por el cliente al servidor para proporcionar información sobre la solicitud
- **Cabeceras de Respuesta**: Enviadas por el servidor al cliente para proporcionar información sobre la respuesta

Protocolo FTP y SSL

El Protocolo de Transferencia de Archivos (FTP) es un protocolo estándar utilizado para la transferencia de archivos entre un cliente y un servidor en una red de computadoras.

Algunas características importantes del Protocolo FTP incluyen:

- **Modos de Transferencia**: En el **modo activo**, el servidor inicia la conexión de datos con el cliente, mientras que, en el **modo pasivo**, el cliente inicia la conexión de datos con el servidor.
- **Autenticación**: FTP proporciona mecanismos de autenticación para garantizar la seguridad durante la transferencia de archivos
- **Seguridad:** Por defecto no viene cifrado, pero existen las versiones FTPS y SFTP (SSH)
- **Flexibilidad y Versatilidad:** Sirve para transferencia usuario-usuario como para grandes volúmenes de datos.

Protocolo SSH

El **Protocolo SSH** es un protocolo de red que proporciona una forma segura de acceder y administrar servidores remotos a través de una conexión cifrada.

- **Cifrado de Datos:** Solo el cliente y servidor autorizados pueden acceder a la información.
- **Autenticación del Servidor y del Cliente:** La autenticación del servidor se realiza mediante el intercambio de claves públicas y privadas, mientras que la autenticación del cliente puede basarse en contraseñas, claves públicas, tokens, etc.
- **Túneles Seguros:** Esto es especialmente útil para acceder a servicios y recursos internos de una red privada desde ubicaciones externas.
- **Gestión de Sesiones Interactivas y No Interactivas:** SSH permite ejecutar comandos remotos para administrar servidores y sistemas.

Técnicas de cifrado

Hay tres tecnologías de cifrado diferentes utilizadas por SSH:

1. Cifrado simétrico: Se utiliza una clave secreta tanto para el cifrado como para el descifrado de un mensaje, tanto por el cliente como por el host.
   Es muy seguro porque la clave nunca se transmite entre el cliente y el host. Los dos equipos comparten datos públicos y luego los manipulan para calcular de forma independiente la clave secreta. Incluso si otra máquina captura los datos públicamente compartidos, no será capaz de calcular la clave porque el algoritmo de intercambio de clave no se conoce.

2. Cifrado asimétrico: Se utiliza dos claves separadas para el cifrado y el descifrado. Estas dos claves se conocen como la clave pública y la clave privada. La clave pública se distribuye abiertamente. Un mensaje cifrado por la clave pública de una máquina, sólo puede ser descifrado por la misma clave privada de la máquina. Sólo se utiliza durante el algoritmo de intercambio de claves de cifrado simétrico
3. Hashing: Se usa como forma de verificar si una entrada es correcta, si un cliente tiene la entrada correcta, pueden generar el hash criptográfico y comparar su valor para verificar si poseen la entrada correcta.
   Mientras se selecciona el algoritmo de cifrado simétrico, también se selecciona un algoritmo de autenticación de mensajes adecuado.

Negociación de cifrado de sesión

Cuando un cliente intenta conectarse al servidor a través de TCP, el servidor presenta los protocolos de cifrado y las versiones respectivas que soporta. Si el cliente tiene un par similar de protocolo y versión, se alcanza un acuerdo y se inicia la conexión con el protocolo aceptado.
Una vez que esto se establece se ejecuta el Algoritmo de Intercambio de Claves:

1. Tanto el cliente como el servidor coinciden en un número primo muy grande (semilla), que por supuesto no tiene ningún factor en común

2. Luego, las dos partes acuerdan un mecanismo de cifrado común para generar otro conjunto de valores manipulando los valores semilla de una manera algorítmica específica.

3. Ambas partes generan independientemente otro número primo. Esto se utiliza como una clave **privada secreta para la interacción**

4. Esta clave privada recién generada, con el número compartido y el algoritmo de cifrado, se utiliza para calcular una **clave pública** que se distribuye a la otra computadora.

5. Las partes utilizan su clave privada personal, la clave pública compartida de la otra máquina y el número primo original para crear una clave compartida final, calculada de forma independiente por ambos equipos.

6. Ahora que ambas partes tienen una clave compartida, pueden cifrar simétricamente toda la sesión SSH.

Uniform Resource Locators - URL

Una **URL** es una dirección que se utiliza para identificar de manera única un recurso en internet.

- **Protocolo**: El protocolo es la parte inicial de una URL y define cómo se debe acceder al recurso. HTTP, HTTPS, FTP y mailto.
- **Dominio:** También conocido como nombre de host, identifica la ubicación específica en la web donde se encuentra el recurso.
- **Ruta**: La ruta especifica la ubicación exacta del recurso dentro del servidor.
- **Parámetros**: Los parámetros son datos adicionales que se pueden incluir en la URL para proporcionar información adicional al servidor.
- **Fragmento**: El fragmento, también conocido como ancla, identifica una sección específica dentro de un recurso más grande, como una página web

Las URL solo se pueden enviar a través de Internet utilizando el juego de caracteres ASCII. **Si una URL contiene caracteres fuera del conjunto ASCII, la URL debe convertirse**.

UNIDAD 2

HTML (HyperText Markup Language)

Es un lenguaje de marcado que permite a los desarrolladores web organizar y presentar información en forma de documentos hipertextuales, los cuales pueden contener texto, imágenes, enlaces, formularios, videos y mucho más. HTML proporciona una forma estructurada de organizar la información, utilizando etiquetas que definen el significado y la función de cada elemento en una página web.

Estructura básica de un documento HTML

- **Etiqueta <!DOCTYPE html>**: Esta etiqueta define el tipo de documento y la versión de HTML que se está utilizando. En HTML5, la declaración <!DOCTYPE> se simplificó a <!DOCTYPE html>, y no requiere un Documento de Tipo de Documento (DTD) externo.
- **Etiqueta <html>**: La etiqueta <html> envuelve todo el contenido de la página web.
- **Etiqueta <head>**: La etiqueta <head> contiene información meta sobre el documento. Esta sección no se muestra en la ventana del navegador, pero proporciona información importante sobre la página.
- **Etiqueta <title>**: La etiqueta <title> se encuentra dentro del elemento <head> y define el título de la página que se muestra en la barra de título del navegador
- **Etiqueta <body>**: La etiqueta <body> contiene todo el contenido visible de la página web

Elemento HTML

Un elemento HTML se define mediante una etiqueta de inicio, algo de contenido y una etiqueta de cierre: **<nombredeetiqueta>**El contenido va aquí...**</nombredeetiqueta>**

- **HTML Headings** – etiquetas <h1> a <h6>: Son etiquetas que se utilizan para definir la estructura jerárquica y el contenido de una página web.
- **HTML Párrafos** - etiqueta <p>: Esta etiqueta se utiliza para indicar que un bloque de texto forma un párrafo dentro de un documento HTML
- **HTML Imágenes -** etiqueta <img>: <img src="imagen.jpg" alt="Descripción de la imagen">
- La etiqueta **<hr>** define una ruptura temática en una página HTML y suele mostrarse como una regla horizontal.
- El elemento HTML **
** define un salto de línea.
- El elemento HTML **<pre>** define texto preformateado.

HTML Elementos de formato

- **<b>**: Se utiliza para aplicar negrita al texto. Obsoleta
- ****: Se utiliza para aplicar negrita visualmente, también enfatiza el significado semántico del texto.
- **<i>**: Se utiliza para aplicar cursiva al texto. Obsoleta

- ****: Se utiliza para aplicar cursiva visualmente, también enfatiza el significado semántico del texto.
- **<mark>**: Se utiliza para resaltar o marcar un fragmento de texto.
- **<small>**: Se utiliza para hacer que el texto sea más pequeño.
- **<del>**: Se utiliza para indicar que el texto ha sido eliminado o marcado para su eliminación. Por lo general, se muestra con una línea a través del texto.
- **<ins>**: Se utiliza para indicar que el texto ha sido insertado o añadido recientemente. Por lo general, se muestra subrayado.
- **<sub>**: Se utiliza para representar texto en formato de subíndice.
- **<sup>**: Se utiliza para representar texto en formato de superíndice.

Otras etiquetas

- **<blockquote>**: Se utiliza para representar citas o bloques de texto que se han tomado directamente de otra fuente
- **<q>:** Define una cita breve
- **<abbr>:** Define una abreviatura o un acrónimo, es más útil con: title=" Nombre completo"
- **<address>:** Define la información de contacto del autor de un documento o artículo.
- **<cite>**: Define el título de una obra creativa
- **<!-- Este es un comentario -->**

HTML Enlaces - etiqueta <a>

Los enlaces permiten a los usuarios navegar de una página a otra, o dentro de la misma página. Para apuntar a ubicaciones dentro de la misma página se usa el atributo href con un identificador.

El atributo target especifica dónde abrir el documento vinculado.

- _self - Por defecto. Abre el documento en la misma pestaña en la que se hizo clic
- _blank - Abre el documento en una nueva ventana o pestaña
- _parent - Abre el documento en el marco principal
- _top - Abre el documento en el cuerpo completo de la ventana

HTML Tablas

Para crear una tabla en HTML, utilizamos las etiquetas **<table>, <tr>, <td> y <th>**. Aquí tienes una explicación de cada una de estas etiquetas:

- **<table>:** Define una tabla en HTML y delimita su contenido e
- **<tr>:** Define una fila en una tabla. Contiene a <td> y/o <th>
- **<td>:** Define una celda de datos en una fila de la tabla.
- **<th>:** Define una celda de encabezado en una fila de la tabla. Estas celdas se utilizan típicamente para encabezados de columna o fila, y su contenido se muestra en negrita y centrado por defecto.

Atributos:

- **colspan**: Este atributo se utiliza para fusionar celdas horizontalmente en una tabla.
- **rowspan**: Este atributo se utiliza para fusionar celdas verticalmente en una tabla.

HTML Listas

Las listas en HTML son elementos que permiten organizar y presentar información de manera estructurada y fácilmente comprensible. Hay tres tipos principales de listas en HTML: **listas desordenadas (<ul>)**, **listas ordenadas (<ol>)** y **listas de definición (<dl>, <dt> y <dd>)**.

Elemento DIV

El elemento ** ** en HTML es un contenedor genérico que se utiliza para agrupar y organizar otros elementos en bloques separados en una página web. No tiene un significado semántico específico por sí sola.

Atributo de clase

El atributo de clase en HTML se utiliza para asignar una o más clases a un elemento HTML. Las clases son nombres o identificadores que se utilizan para aplicar estilos CSS y para seleccionar elementos específicos en un documento HTML utilizando JavaScript u otras técnicas de manipulación del DOM.

Atributo de identificación

El atributo de identificación en HTML se utiliza para proporcionar un identificador único a un elemento HTML dentro de un documento. Este identificador, conocido como ID, permite seleccionar y manipular el elemento de forma única mediante CSS y JavaScript.

Anclas y Enlaces Internos: Los ID se utilizan frecuentemente en combinación con anclas (<a>) para crear enlaces internos dentro de una página web.

HTML Iframe

El elemento **<iframe>** en HTML se utiliza para incrustar otro documento HTML dentro de una página web. Este elemento permite mostrar contenido de otra fuente, como una página web externa, un documento PDF, un video de YouTube, etc., dentro de un marco dentro de la página actual.

- **Navegación Segura:** El contenido incrustado no tiene acceso directo al DOM de la página principal y viceversa.

HTML Rutas de archivos

Ejemplos de ruta de archivo

Camino	Descripción
<img src="picture.jpg">	El archivo "picture.jpg" se encuentra en la misma carpeta que la página actual.
<img src="images/picture.jpg">	El archivo "picture.jpg" se encuentra en la carpeta de imágenes de la carpeta actual
<img src="/images/picture.jpg">	El archivo "picture.jpg" se encuentra en la carpeta de imágenes en la raíz de la página web actual
<img src="../picture.jpg">	El archivo "picture.jpg" se encuentra en la carpeta un nivel arriba de la carpeta actual

HTML Multimedia

Imágenes: En HTML, las imágenes se insertan utilizando el elemento **<img>,** que incluye el atributo src para especificar la URL de la imagen.

Audio: El elemento <audio> se utiliza para incrustar archivos de audio en una página web.

Video: El elemento <video> se utiliza para incrustar archivos de video en una página web.

Formularios y controles de entrada

Los formularios HTML son elementos para la interacción del usuario en la web, ya que permiten recopilar datos y enviarlos al servidor para su procesamiento.
Elemento **<form>**: El elemento <form> es la envoltura principal de un formulario HTML.
El atributo action en los formularios HTML especifica la URL a la cual se enviarán los datos del formulario cuando este sea enviado.
El **atributo method** especifica el método HTTP que se utilizará al enviar los datos del formulario.

- **POST**: Envía los datos del formulario mediante una solicitud POST HTTP.
- **GET**: Envía los datos del formulario como una cadena de consulta agregada a la URL. La longitud de una URL está limitada (2048 caracteres) y los datos son visibles en la URL.

El **atributo autocomplete** controla si el navegador debe completar automáticamente los campos.

El **atributo novalidate** se puede agregar a un elemento <form> en HTML para indicar al navegador que no valide los campos del formulario antes de enviar los datos al servidor.

HTML Input Attributes

Los atributos de entrada en HTML son propiedades adicionales que se pueden agregar a los elementos <input> para modificar su comportamiento o apariencia.

- **type**: Especifica el tipo de entrada. Los valores posibles incluyen text, password, checkbox, radio, submit, button, file, email, date, number, color, image, hidden, entre otros.
- **value**: Define el valor predeterminado del campo de entrada.
- **placeholder**: Muestra una pista del tipo de información que se espera.
- **name**: Especifica el nombre del campo de entrada, que se utiliza para identificar el valor del campo cuando se envía el formulario.
- **id**: Proporciona un identificador único para el campo de entrada, que se puede utilizar para vincularlo con etiquetas de <label> o para manipularlo mediante JavaScript.
- **required**: Indica que el campo de entrada es obligatorio y no se puede enviar el formulario sin completarlo.
- **disabled**: Deshabilita el campo de entrada, impidiendo que el usuario interactúe con él
- **readonly**: Hace que el campo de entrada sea de solo lectura
- **autocomplete**: Controla si el navegador debe completar automáticamente el campo.
- **autofocus**: Hace que el campo de entrada reciba automáticamente el foco cuando la página se carga, lo que permite al usuario comenzar a escribir inmediatamente.

HTML Pattern Attribute

El atributo pattern es utilizado en elementos de entrada HTML para especificar un patrón que debe cumplir el valor introducido por el usuario. Este patrón se define utilizando una expresión regular, y el navegador verificará que el valor ingresado coincida con el patrón antes de permitir que se envíe el formulario.

CSS (Cascading Style Sheets)

CSS (Cascading Style Sheets) es un lenguaje de estilo utilizado para describir cómo se ve y se presenta un documento HTML. Permite controlar el diseño, el formato y la apariencia de múltiples páginas web a la vez.

Al utilizar reglas CSS, los desarrolladores pueden especificar propiedades como el color, el tamaño y la ubicación de los elementos HTML, lo que proporciona flexibilidad y personalización. Además, CSS permite la creación de diseños responsivos que se adaptan a diferentes dispositivos y tamaños de pantalla.

CSS Syntax

La sintaxis de CSS (Hojas de Estilo en Cascada) consta de un selector, seguido por un conjunto de declaraciones encerradas entre llaves. Cada declaración incluye una propiedad y su valor correspondiente, separados por dos puntos. Múltiples declaraciones se separan por puntos y comas.

```
p {//p es un selector que apunta a las etiquetas <p>
  color: red; //color es una propiedad y red es su valor}
```

CSS Selectors

Los selectores CSS son uno de los conceptos fundamentales en la estilización de páginas web. Permiten a los desarrolladores dirigirse a elementos específicos en un documento HTML y aplicar estilos a esos elementos de manera selectiva.

- **Selector de Tipo**: Apunta a elementos HTML específicos por su nombre de etiqueta.
- **Selector de Clase**: Apunta a elementos que tienen un atributo class específico.
- **Selector de ID**: Se dirige a elementos con un atributo id específico.
- **Selector Universal**: Este selector, representado por un asterisco (*), selecciona todos los elementos en el documento.
- **Selector de Atributo**: Selecciona elementos que tienen un atributo específico, independientemente de su valor.
- **Selector de Atributo y Valor**: Este selector apunta a elementos que tienen un atributo específico con un valor específico.
- **Selector de Descendencia**: Selecciona un elemento que es descendiente directo de otro elemento. Se utiliza con un espacio entre los selectores. Por ejemplo, div p seleccionaría todos los elementos de párrafo que son descendientes directos de elementos .
- **Selector de Clase Anidado**: Selecciona elementos que son descendientes de un elemento con una clase específica.

Formas comunes de insertar CSS en un documento HTML

- **Estilo en línea (Inline Style)**: Esta técnica implica agregar estilos directamente a un elemento HTML utilizando el atributo style.
- **Estilo interno (Internal Style):** Con esta técnica, los estilos se definen dentro del elemento <style> en el encabezado <head> del documento HTML
- **Estilo externo (External Style):** En esta técnica, los estilos se almacenan en un archivo de hoja de estilo separado (con extensión .css) y se vinculan al documento HTML utilizando la etiqueta <link>.

Orden de Cascada

Todos los estilos en una página se "cascadearán" en una nueva hoja de estilo "virtual" según las siguientes reglas, donde el número uno tiene la mayor prioridad:

- Estilo en línea (dentro de un elemento HTML)
- Hojas de estilo externas e internas (en la sección de encabezado)
- Predeterminado del navegador

.

CSS Colors

Los colores en CSS son fundamentales para definir la apariencia visual de los elementos de una página web. Hay varias formas de especificar colores en CSS:

- **Nombres de colores**: CSS proporciona un conjunto de nombres de colores predefinidos
- **Valores hexadecimales**: Los colores se pueden especificar utilizando valores hexadecimales RGB
- **Valores RGB**: Los colores también se pueden especificar utilizando valores RGB (0-255)
- **Valores RGBA:** Similar a RGB, pero con un cuarto parámetro que representa la opacidad
- **Valores HSL**: HSL representa el tono (H), la saturación (S) y el nivel de luminosidad (L) del color.
- **Valores HSLA**: Es lo mismo que HSL pero con un cuarto parámetro para la opacidad.

CSS Backgrounds

La propiedad background en CSS es versátil y permite una amplia gama de posibilidades estilísticas, desde colores simples hasta imágenes complejas.

- **background-color**: Define el color de fondo de un elemento.
- **background-image**: Permite establecer una imagen de fondo para un elemento.
- **background-repeat**: Controla la repetición de la imagen de fondo
- **background-position**: Establece la posición inicial de la imagen de fondo
- **background-size**: Define el tamaño de la imagen de fondo
- **background-attachment**: Controla el comportamiento de desplazamiento de la imagen de fondo. Los valores incluyen scroll (la imagen de fondo se desplaza con el contenido), fixed (la imagen de fondo se fija en su posición y no se desplaza con el contenido), y local (la imagen se desplaza con el contenido del elemento de desplazamiento).
- **Opacity / Transparency:** Puede tomar un valor de 0.0 a 1.0.

CSS Borders

- **border**: Es una propiedad compuesta que permite definir el ancho, el estilo y el color del borde en una sola línea
- **border-width**: Especifica el grosor del borde.
- **border-style**: Define el estilo del borde.
- **border-radius:** Permite redondear las esquinas del borde.

Modelo de caja

El Modelo de Caja es un concepto fundamental que describe cómo se componen y se disponen los elementos en una página web. Cada elemento en CSS se representa como una caja rectangular que incluye su contenido, relleno (padding), borde (border) y margen (margin).

- **margin**: Es una propiedad abreviada que permite definir los márgenes en los cuatro lados de un elemento
- **padding**: Es una propiedad abreviada que permite definir el relleno en los cuatro lados de un elemento
- Las propiedades height y width se utilizan para establecer la altura y el ancho de un elemento.
- **Outline:** Un contorno es una línea dibujada fuera del borde del elemento.

Diseño de página y posicionamiento.

Alinear elementos uno al lado del otro es una práctica común al diseñar páginas web. Hay diferentes métodos para lograrlo, todos requieren un poco de estilizado CSS. Vamos a explorar los métodos más comunes.

- **Float**
- **Inline-block**
- **Flex**
- **Grid (rejillas)**

Elementos Flex (Flex Items): Los elementos contenidos dentro de un contenedor flex se denominan elementos flex. Estos elementos se distribuyen automáticamente dentro del contenedor flex de acuerdo con las reglas de Flexbox.

Propiedades Importantes de Flexbox

- **flex-direction**: Controla la dirección en la que se colocan los elementos flex
- **justify-content**: Alinea los elementos flex a lo largo del eje principal del contenedor flex
- **align-items**: Alinea los elementos flex a lo largo del eje transversal del contenedor flex
- **flex-grow, flex-shrink, flex-basis**: Estas propiedades controlan cómo se distribuye el espacio entre los elementos flexibles cuando no tienen un ancho o alto fijo

Grid Layout

El Módulo de Diseño de Rejilla en CSS ofrece un sistema de diseño basado en rejillas, con filas y columnas, lo que facilita el diseño de páginas web sin tener que utilizar flotantes y posicionamiento

Media Queries y Diseño Adaptable (responsive).

El diseño web responsivo hace que tu página web se vea bien en todos los dispositivos. El diseño web responsivo utiliza únicamente HTML y CSS.

Viewport

Es el área de la pantalla en la que se muestra el contenido de una página web. El tamaño del viewport puede cambiar dependiendo del dispositivo y de cómo el usuario esté interactuando con la página.

Media Queries

Las consultas de medios son reglas CSS que permiten aplicar estilos basados en las características del dispositivo en el que se visualiza la página web. Se definen utilizando la sintaxis @media, seguida de una condición que especifica las características del dispositivo, como el ancho, la orientación o la resolución de la pantalla. Ejemplo: `@media only screen and (max-width: 600px)`

Responsive Web Design – Frameworks

Los frameworks proporcionan conjuntos predefinidos de estilos y componentes que facilitan la creación de sitios web receptivos y adaptables a una variedad de dispositivos y tamaños de pantalla

UNIDAD 3
JavaScript

JavaScript es un lenguaje de programación dinámico y versátil que se utiliza principalmente para crear y controlar contenido dinámico en sitios web.

Características Principales

- **Interactividad en el Cliente:** JavaScript permite a los desarrolladores agregar interactividad directamente en el navegador del usuario.
- **Programación Asíncrona**: Promises y async/await.
- **Extensibilidad**: JavaScript puede integrarse con bibliotecas y frameworks
- **Compatibilidad y Accesibilidad:** JavaScript es compatible con todos los navegadores modernos.

JavaScript Puede Cambiar el Contenido HTML
Uno de los muchos métodos de JavaScript para trabajar con HTML es **getElementById()**.

JavaScript Puede Ocultar y Mostrar Elementos HTML
```
document.getElementById("demo").style.display ="none" / "block";
```

¿Cómo insertar un js?

Hay tres formas comunes de llamar a JavaScript en una página web:

- Incrustado en HTML: Puedes escribir JavaScript directamente dentro de las etiquetas <script> en tu archivo HTML.

- Archivo Externo: Puedes escribir tu código JavaScript en un archivo separado con extensión .js y luego enlazarlo en tu archivo HTML usando la etiqueta <script> con el atributo src. **<script src="mi_script.js"></script>**

- Manejadores de Eventos: Puedes llamar a funciones JavaScript cuando ocurren eventos específicos en la página. Esto se hace usando atributos de eventos HTML o añadiendo oyentes de eventos a elementos HTML a través de JavaScript.

<button **onclick="saludar()"**>Haz clic aquí</button>

Posibilidades de Visualización en JavaScript

JavaScript puede "mostrar" datos de diferentes maneras:

- Escribiendo en un elemento HTML, usando innerHTML.
- Escribiendo en la salida HTML usando document.write().
- Escribiendo en un cuadro de alerta, usando window.alert().
- Escribiendo en la consola del navegador, usando console.log().
- Por impresora.

Variables, tipos de datos y operadores.

JavaScript Variables
La declaración de variables en JavaScript se realiza mediante las palabras clave var, let y const, cada una con sus propias características y usos específicos.

Declaración de Variables
- **Var:** Las variables declaradas con var tienen un ámbito de función, lo que significa que están disponibles dentro de la función donde se declaran.
- **let**: Permite declarar variables con un ámbito de bloque. Esto significa que la variable sólo está disponible dentro del bloque {} donde se declara.
- **const**: Se utiliza para declarar variables cuyo valor no debe cambiar a lo largo del programa. Similar a let, tiene un ámbito de bloque.

Asignación de Valores

Las variables pueden ser asignadas a diferentes tipos de datos, como números, cadenas de texto, booleanos, objetos, arreglos, funciones, entre otros.

Reglas de Nomenclatura

Las variables en JavaScript deben seguir ciertas reglas de nomenclatura:

- Deben comenzar con una letra, un signo de dólar $, o un guion bajo _.
- No pueden comenzar con un número.
- No pueden usar palabras reservadas de JavaScript (como class, return, if, etc.).

Ámbito de Variables
El ámbito de una variable determina dónde puede ser utilizada dentro del código:

- **Ámbito global**: Las variables declaradas fuera de cualquier función tienen un ámbito global y están disponibles en cualquier parte del código.
- **Ámbito de función**: Las variables declaradas dentro de una función con var están disponibles solo dentro de esa función.
- **Ámbito de bloque**: Las variables declaradas con let o const dentro de un bloque {} están disponibles solo dentro de ese bloque.

Operadores en JavaScript
Los operadores en JavaScript son símbolos utilizados para realizar operaciones sobre valores y variables. Estos operadores se pueden categorizar en diferentes tipos según su funcionalidad

- **Operadores Aritméticos:** Se utilizan para realizar operaciones matemáticas entre números.
- **Operadores de Asignación:** Se utilizan para asignar valores a variables.
- **Operadores de Comparación:** Se utilizan para comparar dos valores.
- **Operadores Lógicos:** Se utilizan para realizar operaciones lógicas y devuelven un valor booleano (true o false).

Funciones y objetos en JavaScript.

Las funciones en JavaScript son bloques de código reutilizables diseñados para realizar unatarea específica. Una función se define una vez y luego puede invocarse varias veces.

Funciones Anónimas
Las funciones anónimas son funciones sin nombre. A menudo se utilizan como valores de variables o como argumentos a otras funciones.
let multiplicar = function(a, b) {return a * b;}

Funciones Flecha (Arrow Functions)
Las funciones flecha son una forma más concisa de escribir funciones anónimas. Se utilizan el operador de flecha (=>) y son útiles para funciones de una sola línea.

Funciones de Orden Superior
Las funciones de orden superior son funciones que pueden aceptar otras funciones como argumentos o devolver funciones como resultado.

Funciones como Métodos
En JavaScript, las funciones también pueden ser propiedades de objetos. Cuando una función es una propiedad de un objeto, se llama método.

Ámbito de las Funciones
Las variables definidas dentro de una función solo están disponibles dentro de esa función.
Este se llama ámbito local. Las variables definidas fuera de cualquier función tienen ámbito global y están disponibles en cualquier parte del código

Funciones Recursivas
Una función recursiva es una función que se llama a sí misma. La recursión es útil para resolver problemas que pueden dividirse en sub problemas similares.

Objetos en JavaScript
En JavaScript, un objeto es una colección de propiedades, y una propiedad es una asociación entre un nombre (o clave) y un valor

Creación de Objetos
Literal de Objeto: Esta es la forma más común y directa de crear un objeto.

```
let persona = {
   nombre: "Juan",
   edad: 30,
   }
};
```

Constructor de Objeto: Utiliza la función Object.

```
let persona = new Object();
persona.nombre = "Juan";
persona.edad = 30;
```

Funciones Constructoras: Permiten crear múltiples instancias de un objeto.

```
function Persona(nombre, edad) {
   this.nombre = nombre;
   this.edad = edad; }
let juan = new Persona("Juan", 30);
```

Acceso a Propiedades
Las propiedades de los objetos pueden accederse usando la notación de punto (persona.dni) o la notación de corchetes(persona['nombre']).

Propiedades y Métodos Dinámicos
En JavaScript, se pueden agregar y eliminar propiedades y métodos de un objeto en cualquier momento

Iteración sobre Propiedades
Se puede iterar sobre las propiedades de un objeto utilizando un bucle **for...in.**

```
for (let clave in persona) {
   console.log(clave + ": " + persona[clave]);
}
```

Objetos Anidados
Las propiedades de un objeto pueden ser otros objetos, permitiendo estructuras de datos anidadas.

Prototipos
Cada objeto en JavaScript tiene un prototipo, un objeto del cual hereda propiedades y métodos. Esto es fundamental para la herencia en JavaScript.

Clases
Las clases proporcionan una forma más clara y simple de crear objetos y manejar herencia en JavaScript.

```
class Persona {
   constructor(nombre, edad) {
      this.nombre = nombre;
      this.edad = edad;
   }
```

Estructuras de Control en JavaScript
Las estructuras de control en JavaScript permiten dirigir el flujo de ejecución de un programa. Las principales estructuras de control incluyen condicionales y bucles.

- **Condicionales: if, else if, else:** Los condicionales se utilizan para ejecutar diferentes bloques de código en función de una condición.
- **Bucles: for, while, do...while:** Los bucles permiten ejecutar un bloque de código varias veces
- **Bucles Anidados:** Los bucles anidados son bucles dentro de otros bucles. Son útiles para trabajar con estructuras de datos multidimensionales como matrices bidimensionales.
- **Control de Bucles: break y continue**
    - **break**: Termina el bucle actual inmediatamente.
    - **continue**: Salta a la siguiente iteración del bucle.

Tipos de datos:
- **BigInt en JavaScript**: Permite representar enteros con precisión arbitraria, útil números muy grandes que exceden el límite de los números enteros tradicionales en JavaScript.
- **Number en JavaScript:** El tipo Number representa tanto números enteros como de punto flotante, almacenado como un valor de punto flotante de doble precisión según el estándar IEEE 754.

Manipulación del DOM (Document Object Model)
El DOM (Document Object Model) es una representación estructurada de un documento HTML o XML que permite a los lenguajes de programación acceder y modificar el contenido, la estructura y el estilo del documento de manera dinámica.
Estructura del DOM
El DOM representa un documento como una jerarquía de nodos. Estos nodos pueden ser elementos, atributos, texto, comentarios, entre otros. La estructura del DOM se asemeja a un árbol
Selección de Elementos del DOM
Para manipular el DOM, primero debemos seleccionar los elementos que queremos modificar. getElementById o getElementByClassName
Creación y Eliminación de Elementos del DOM

- **Crear Elementos**: Usando createElement y appendChild.
- **Eliminar Elementos**: Usando removeChild.

Eventos y Manejo de Eventos en JavaScript
Los eventos en JavaScript son acciones o sucesos que ocurren en el navegador, los cuales pueden ser desencadenados por el usuario o por el sistema.

Manejo de Eventos

- **Atributos HTML**: Definir manejadores de eventos directamente en los atributos HTML.
- **Propiedades del DOM**: Asignar funciones a las propiedades de eventos del DOM.
- **Método addEventListener**: Añadir eventos de manera más flexible y moderna, permitiendo agregar múltiples manejadores para el mismo evento.

Eventos y Funciones
Podemos pasar el objeto event como parametro a las funciones manejadoras para obtener información adicional sobre el evento:

Eliminación de Manejadores de Eventos
Si necesitamos eliminar un manejador de eventos, podemos usar el método removeEventListener.

Prevención del Comportamiento Predeterminado
Algunos eventos tienen comportamientos predeterminados que pueden ser prevenidos usando preventDefault.

UNIDAD 4

Arquitectura Cliente-Servidor para Aplicaciones Web
La arquitectura cliente-servidor es un modelo de diseño utilizado en el desarrollo de aplicaciones web, donde las tareas y responsabilidades están divididas entre dos entidades principales: el cliente y el servidor

Cliente: El cliente es el lado del usuario, típicamente un navegador web o una aplicación que solicita recursos o servicios desde el servidor

- **Interfaz de Usuario (UI)**: La parte de la aplicación que interactúa directamente con el usuario, como formularios, botones, y gráficos.
- **Lógica de Presentación**: Gestiona cómo se muestra la información al usuario y cómo se manejan las interacciones del usuario.

Servidor: El servidor es el lado que proporciona recursos o servicios solicitados por el cliente. Puede ser un servidor web, un servidor de aplicaciones, o una base de datos.

- **Servidor Web**: Gestiona las solicitudes HTTP/HTTPS y responde con el contenido adecuado, como páginas HTML, archivos, o datos JSON.
- **Servidor de Aplicaciones**: Procesa la lógica de la aplicación, como la autenticación de usuarios, la gestión de sesiones y la ejecución de operaciones comerciales.
- **Base de Datos**: Almacena y gestiona los datos utilizados por la aplicación, respondiendo a consultas y actualizaciones de datos.

Frontend

El frontend es la parte de la aplicación web con la que los usuarios interactúan directamente. Está compuesto por la interfaz de usuario y la experiencia de usuario

- **HTML, CSS, JavaScript**, **Frameworks y Bibliotecas**

Backend

El backend es la parte de la aplicación web que se ejecuta en el servidor y es responsable de gestionar la lógica de negocio, la interacción con bases de datos, la autenticación de usuarios y otras tareas críticas.

- **Lenguajes de Programación**: Cada lenguaje tiene sus propias características. La elección del lenguaje depende de los requisitos específicos del proyecto.

- **Frameworks**: Proporcionan estructuras y herramientas predefinidas para gestionar rutas, manejar solicitudes HTTP, y acceder a bases de datos.

- **Bases de Datos**: Las aplicaciones web necesitan almacenar y recuperar datos. Los desarrolladores backend son responsables de diseñar esquemas de bases de datos, escribir consultas SQL o trabajar con ORM para interactuar con las bases de datos.

- **APIs (Application Programming Interfaces)**: El backend expone servicios a través de APIs que el frontend puede consumir. Las APIs pueden ser **RESTful** o basadas en GraphQL, y permiten que los diferentes componentes de la aplicación se comuniquen entre sí.

Interacción entre Frontend y Backend

El frontend y el backend se comunican a través de peticiones **HTTP**. El frontend envía solicitudes al backend para obtener datos o ejecutar acciones, y el backend responde con la información necesaria.

¿Qué son los Servicios Web?
Los servicios web son aplicaciones que utilizan protocolos estándar como **HTTP/HTTPS** para intercambiar datos entre sistemas diferentes a través de una red, generalmente Internet. Estos servicios permiten que las aplicaciones interactúen entre sí

Tipos de Servicios Web:

- **SOAP (Simple Object Access Protocol):** Es un protocolo basado en XML que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos **XML**.
- **REST (Representational State Transfer)**: Es un estilo arquitectónico que utiliza HTTP para realizar operaciones CRUD (Create, Read, Update, Delete) en recursos web representados en formatos como **JSON** o **XML**.

¿Qué es una API?
Una API (Application Programming Interface) es un conjunto de reglas y definiciones que permiten que las aplicaciones se comuniquen entre sí.
Tipos de APIs:

- **APIs Públicas**: Están disponibles para cualquier desarrollador y son utilizadas para integrar servicios externos en una aplicación.
- **APIs Privadas**: Se utilizan dentro de una organización para mejorar la interoperabilidad entre sistemas internos.
- **APIs Asociadas**: Se utilizan para compartir datos y servicios entre socios comerciales o desarrolladores externos específicos.

¿Qué es una API RESTful?

Una API RESTful es una implementación de una API que sigue los principios de REST. REST es un estilo arquitectónico para diseñar servicios de red escalables.

Beneficios de las APIs RESTful:
- **Simplicidad**: Utilizan los verbos HTTP estándar, son fáciles de entender y utilizar.
- **Escalabilidad**: Diseñadas para trabajar en una arquitectura distribuida y escalar horizontalmente.
- **Flexibilidad**: Pueden ser utilizadas por diferentes lenguajes y plataformas.
- **Ligereza**: Generalmente utilizan JSON.

JSON (JavaScript Object Notation) es un formato de intercambio de datos ligero y fácil de leer y escribir tanto para humanos como para las máquinas.

JSON está compuesto por dos estructuras principales:

- **Objetos**: Se representan como una colección de pares clave-valor delimitados por llaves {}. Los pares clave-valor están separados por comas
- **Arreglos**: Son una colección de valores ordenados delimitados por corchetes [].

Tipos de Datos en JSON
JSON admite los siguientes tipos de datos: **Cadenas, Números, Booleanos, Nulos, Objetos y Arreglos**

Desventajas de JSON
- Sin soporte para comentarios: JSON no admite comentarios, lo que puede dificultar la inclusión de anotaciones o explicaciones en los datos.
- Menos robusto que XML: JSON es menos expresivo que XML y puede no ser adecuado para ciertos tipos de datos complejos o jerárquicos.

Consumo de servicios web desde JavaScript.
JavaScript, con su capacidad de ejecutar tanto en el navegador como en el servidor, ofrece varias formas de interactuar con servicios web y APIs.
Métodos para Consumir Servicios Web en JavaScript
- **XMLHttpRequest (XHR)**: es una API integrada en los navegadores web que permite realizar solicitudes HTTP.
- **Fetch API:** Es una interfaz moderna que proporciona una forma más sencilla y flexible de realizar solicitudes HTTP.
- **Axios**: Es una biblioteca basada en promesas para realizar solicitudes HTTP.

Pasos Generales para Consumir Servicios Web

- **Hacer una solicitud HTTP**: Utilizando uno de los métodos mencionados (XHR, Fetch API, Axios), se envía una solicitud al servicio web.

- **Manejar la Respuesta**: Las respuestas del servicio web se manejan típicamente en formato JSON. Estas respuestas se procesan para usarlas en la aplicación.

- **Manejo de Errores**: Es importante manejar errores de red y respuestas incorrectas para asegurar la robustez de la aplicación.

Introducción a AJAX

AJAX, que significa "Asynchronous JavaScript and XML", es una técnica utilizada en el desarrollo web para crear aplicaciones web interactivas. Con AJAX, las aplicaciones web pueden enviar y recibir datos del servidor de manera asíncrona sin recargar la página completa.

El funcionamiento de AJAX se basa en la combinación de varias tecnologías
- **HTML/CSS**: Para la estructura y estilo de la página web.
- **JavaScript**: Para manipular y controlar el contenido dinámico.
- **XMLHttpRequest**: Para realizar solicitudes HTTP asíncronas al servidor.
- **JSON/XML**: Formatos de datos utilizados para intercambiar información entre el servidor y el cliente.

Ventajas de Usar AJAX

- **Interactividad:** Permite actualizar partes de la página web sin recargar por completo.
- **Velocidad:** Mejora la velocidad de la aplicación web, ya que solo se actualizan los datos necesarios.
- **Mejor Experiencia de Usuario**: Al reducir los tiempos de carga y proporcionar respuestas rápidas, mejora significativamente la experiencia del usuario