



**TRIMESTRE 24O**

**Materia: Proyecto terminal I de Ingeniería de Software**

**Profesor: Eduardo Filemón Vázquez Santacruz**

**Nombre del alumno: Gabriela Anaid Caballero Flores**

**Número de matrícula: 2193013206**

**Trabajo escrito-Creación de un libro digital sobre Ing.  
Software (HTML, CSS, JS) para nivel secundaria.**

**Grupo: XXXX**

## Índice

### Listas de Tablas

Aquí vemos el índice numerado en el que aparece el número de la página en la que está la tabla. Los textos están vinculados a las tablas para que así puedas pulsar sobre ellos.

(TABLA)

#### 1. Introducción

Este

## 2. Glosario de Términos



A continuación, se van a describir unos conceptos clave que se deben de tener en cuenta antes de empezar con la lectura del libro. Estos se encuentran en la tabla siguiente:

Concepto	Significado
----------	-------------

Dato	Es la unidad mínima de la información, por sí sólo no sirve de nada, se requiere de varios de ellos para poderlos procesar como Información.
Información	Es un conjunto de datos debe de ser específico de un tema.
Informática	Es un conjunto de técnicas y herramientas que nos permiten: búsqueda, selección, análisis, procesar, presentar la información.
Lenguajes de programación	Son un conjunto de reglas y símbolos, sistema de comunicación, que son usados para definir una secuencia de instrucciones para el procesamiento, también, controla el comportamiento físico y lógico de una máquina.
Software	Es conocido como soporte lógico, software de sistema (intangible), se define como el conjunto de programas, aplicaciones y sistemas operativos que permiten que una computadora realice varias tareas específicas, se pueden modificar y actualizar fácilmente.
Hardware	Es conocido como soporte físico, soporte duro (tangible), son los componentes físicos de una computadora e interactúan con el software, se clasifica en los dispositivos de entrada, procesamiento, almacenamiento y salida.
Firmware	Es conocido como software especializado para ejecutar los dispositivos de hardware específicos como lo son: routers, impresoras, cámaras

	digitales,etc. Además, controla y gestiona el hardware lo cual permite que los dispositivos puedan realizar funciones específicas. Se almacenan en la memoria ROM.
Humanware	Es la interacción de los seres humanos y la tecnología (máquina), se define como el conjunto de habilidades, conocimientos, capacidades humanas que facilitan el uso y la integración de una computadora.
Malware	Se refiere al software mal intencionado , software malicioso, su propósito es dañar la información de un sistema informático.
Computadora	Es una máquina de tipo electrónica que sirve para procesar y almacenar información basada en números. También, llamado ordenador.
Sistema Informático	Es un conjunto de partes que nos permite manejar automáticamente la información en una computadora.
Sistema Operativo	Se define como un conjunto de programas en la categoría de software de sistema encargado de administrar e integrar los recursos de tipos físicos y lógicos de un sistema informático.
Programa	Es un conjunto de instrucciones (intangible) escritas en un lenguaje de programación que una computadora puede ejecutar para que haga cierta tarea, estas están para resolver los problemas, optimizar los procesos, realizar funciones específicas dentro de un sistema

	informático. Pueden ser simples o complejos.
Archivo	Es un conjunto de datos o información (software) almacenados en una computadora. Contienen todo tipo de contenido como por ejemplo, documentos, imágenes, videos, música, código fuente de programas, etc. Tienen un nombre y una extensión que indica el tipo de contenido que contiene, por ejemplo: .txt para archivos de texto, .exe para ejecutar programas ejecutables. Pueden ser leídos, escritos, modificados, copiados y eliminados.
Algoritmo	Es un conjunto de instrucciones o pasos definidos (deben ser claros y precisos), ordenados (deben seguir un orden lógico), finitos (deben tener un número limitado de pasos para llegar a la solución en un cierto tiempo) que se usan para resolver un problema o realizar una tarea concreta. Son implementados en distintos lenguajes de programación para ser ejecutados con una computadora.
Código	Es la implementación de un algoritmo en un lenguaje de programación específico. También, es el conjunto de instrucciones escritas para que la computadora pueda ejecutar. Es el lenguaje que permite a los programadores (usuarios) poder comunicarse con la computadora.
Memoria ROM (Memoria de Solo Lectura)	Es un tipo de memoria no volátil que almacena datos de manera permanente, incluso cuando el

	dispositivo se apaga. Guarda el Firmware y las instrucciones para que el dispositivo funcione como en este caso la inicialización del sistema, configuraciones del hardware. Sólo se pueden leer y no modificarse fácilmente.
Memoria RAM (Memoria de Acceso Aleatorio)	Es un tipo de memoria volátil que almacena temporalmente los datos e instrucciones que se están usando activamente (de manera inmediata) en el CPU para que el procesador pueda leer y escribir datos mientras realiza varias operaciones. Es volátil porque pierde los datos al apagar el dispositivo.
CPU (Unidad Central de Procesamiento)	Es el cerebro de la computadora que realiza los cálculos y procesa las instrucciones.
Dispositivo	Es cualquier aparato o herramienta que realiza una función o una tarea específica, también, se refiere a un componente o equipo que interactúa con otros o con los usuarios para cumplir una o más funciones concretas. Pueden ser componentes tangibles (como una computadora o un teclado) o componentes intangibles (como un programa).
Usuario	Es una persona o entidad o un individuo que interactúa con un sistema, aplicación o un servicio, se refiere a quien utiliza el sistema para poder realizar tareas específicas.
Seguridad web	Se refiere a las medidas para proteger los sistemas web y los datos que se transmiten a través de la web. Esta previene los accesos no autorizados, ataques

	cibernéticos y la pérdida de información de las aplicaciones.
Computación	
Cibernética	
https	
Internet	
Interfaz	
API	
Pseudocódigo	
Sitio web	
Página web	
Sistema web	

Tabla 1: Glosario de términos antes de comenzar.

### 3. Desarrollo

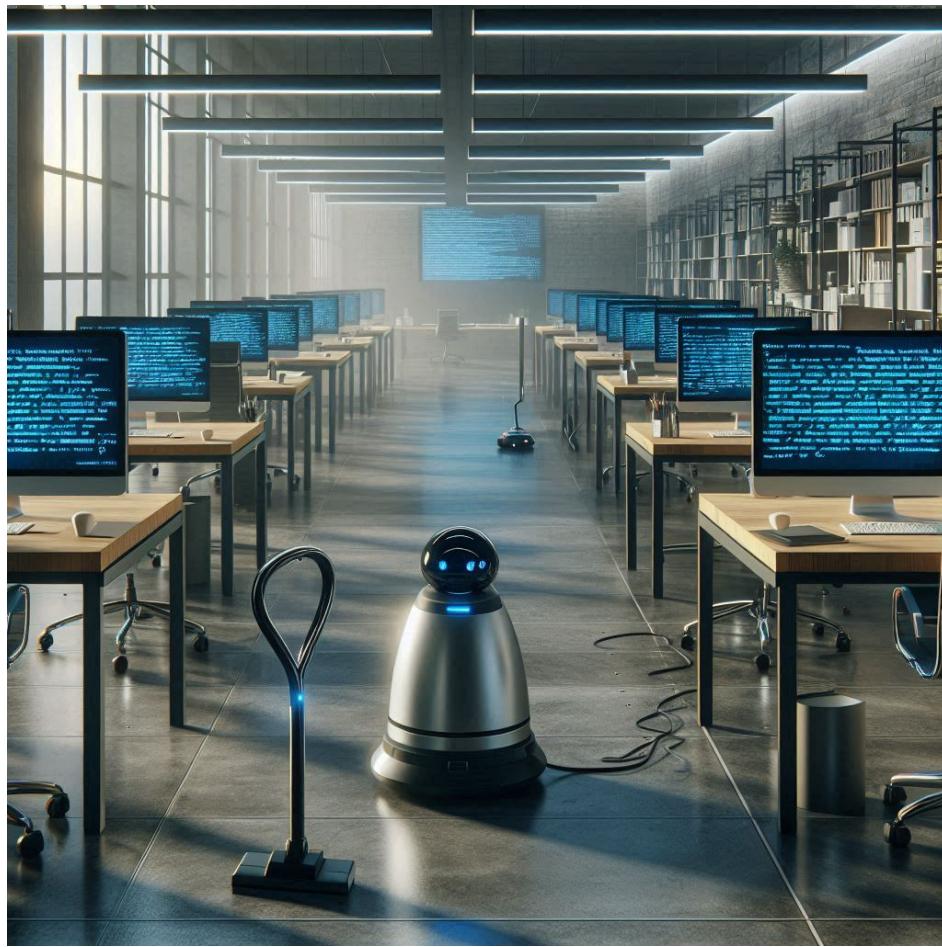


En esta sección, se encuentra el desarrollo del proyecto de investigación I y II que se hace en la carrera de la Licenciatura en Computación en la Universidad Autónoma Metropolitana de la Unidad Iztapalapa (UAM-I).



Figura 1: Logo de la UAM-I.

## CAPÍTULO 1. INICIACIÓN Y CONCEPTUALIZACIÓN



En este apartado se comenzará con los conocimientos básicos de la materia de Ingeniería de Software de la UAM-I.

La Ingeniería de Software es una disciplina en la cual se encarga de diseñar, desarrollar, mantener y gestionar sistemas o programas informáticos (software).

Además, se usan algunos de los conceptos clave para tratar de entender esta disciplina, estos son los siguientes:

- **Datos:** Es la unidad mínima de la información, por sí sólo no sirve de nada, se requiere de varios de ellos para poderlos procesar como información.
- **Información:** Es un conjunto de datos, deben ser específicos de un tema.
- **Computadora u ordenador:** Máquina electrónica que sirve para procesar y almacenar información. Asimismo, esta cuenta con cierta capacidad de cálculo numérico y decisiones lógicas, controlada por un programa almacenado y con posibilidades de comunicación. Está destinada a realizar múltiples y diversas tareas o actividades, según convenga al usuario, recibe y procesa datos para convertirlos en información útil para facilitar el trabajo.

- **Computación:** Es la ciencia que se encarga de estudiar los procesos y sistemas informáticos. También, estudia la administración de los diferentes métodos, técnicas y procesos para poder almacenar, procesar y transmitir la información digital.



Estos son otros conceptos de los cuales debes saber ya que te permite cómo se desarrollan los sistemas de software, estos serían:

- **Requisitos o requerimientos:** Es la traducción de un problema de un negocio, también, es lo que el software debe de hacer, en este caso serían: funcionalidades, rendimiento, seguridad, entre otros, para que después se pueda realizar el diseño de ese.
- **Diseño:** Se define la estructuración del software (componentes) en la cual se deben cumplir las necesidades y los requerimientos para favorecer la siguiente etapa de desarrollo.
- **Desarrollo:** Se empieza a desarrollar el código del software de acuerdo con los requerimientos y el diseño mencionado antes.
- **Pruebas:** Se hacen varios experimentos para verificar y comprobar si el software funciona correctamente o no y esto debe satisfacer con los requerimientos anteriores, existen 3 tipos de pruebas, estos son: unitarias, segmentos y sistema.
- **Implantación:** Es el proceso de instalar y poner en funcionamiento un sistema o software a un nuevo entorno sin tener que cambiar el código original.
- **Mantenimiento:** Actualización, mejoras y ajustes del software para añadir nuevos cambios en su entorno.



## LENGUAJES DE PROGRAMACIÓN

A continuación, se hablarán de los diferentes lenguajes de programación que existen y sus características pertinentes.

Hay que aclarar que existen muchas definiciones o significados para explicar lo que son los lenguajes de programación, esto puede depender del contexto o de los autores de cómo lo dicen, así que primero hay que decir qué son los lenguajes de programación.

## DEFINICIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

### ¿Qué son los lenguajes de programación?

Los lenguajes de programación son herramientas o utensilios en las cuales nos permite desarrollar software o **programas informáticos** [1] para poder comunicarnos con una computadora, tablet, celular o cualquier otro dispositivo electrónico.

También, son como una conexión entre el ser humano y las máquinas porque es muy difícil hablar con el sistema binario (sistema de numeración que los dígitos o números que se utilizan son el 0 y el 1).

Además, son un conjunto de reglas gramaticales, estas incluyen la semántica y la sintaxis para que los programadores puedan dar indicaciones o secuencia de órdenes a

una computadora, tablet, celular o cualquier otro dispositivo electrónico para que hagan unas tareas determinadas.



## CARACTERÍSTICAS DE LOS LENGUAJES DE PROGRAMACIÓN

### ¿Cuáles son las principales características de los lenguajes de programación?

De ese mismo modo, se mencionan las principales características de los lenguajes de programación. Estas se determinan por la funcionalidad, flexibilidad o facilidad de cómo.

A continuación, te digo las características clave, estas son:

- **Sintaxis:** Es el conjunto de reglas y estructuras gramaticales que determinan cómo se escriben las instrucciones en cada uno de los lenguajes de programación.
- **Semántica:** Es la definición de las sentencias y las instrucciones dentro del lenguaje. Esto es lo qué hace realmente el código al ser ejecutado.
- **Portabilidad:** Se refiere a la capacidad del programa en la cual se ejecuta en distintas plataformas sin tener que hacer modificaciones pertinentes.
- **Tipado:** Se refiere a cómo se manipulan los distintos tipos de datos (cadenas, números, objetos, etc) en el lenguaje.
- **Modularidad:** Se refiere a la habilidad de un sistema o programa para ser desglosado en componentes más pequeños o "módulos", donde cada módulo funciona de manera independiente y está diseñado para realizar una tarea específica dentro del conjunto general del sistema.
- **Paradigma:** Un paradigma de programación es una forma o método para abordar la resolución de problemas mediante programación. Cada paradigma propone una manera distinta de estructurar y escribir el código, lo que influye en cómo se organiza y resuelve un problema.



## TIPOS DE LENGUAJES DE PROGRAMACIÓN

### ¿Cuáles son los tipos de lenguajes de programación?

Los lenguajes de programación es un lenguaje formal específico serie de instrucciones para una computadora, puede usarse para crear programas que pongan en práctica algoritmos.



Existen varias clasificaciones en las cuales se pueden dividir los lenguajes de programación, esto puede depender los atributos, la finalidad o el nivel de abstracción.

A continuación, se explicará de acuerdo al nivel de abstracción, estos son los siguientes:

### 1. Segundo por el nivel de abstracción (Bajo nivel y Alto nivel)

Se pueden catalogar o dividir en 2 tipos, estos serían los lenguajes de programación de bajo nivel y los lenguajes de programación de alto nivel.

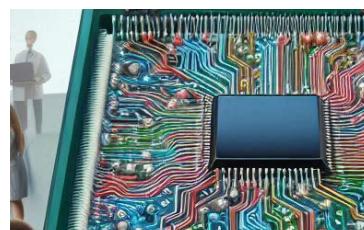
Vamos a comenzar por los lenguajes de bajo nivel, estos son:

#### Lenguajes de Bajo nivel

- **Lenguaje máquina:** Es el código que ejecuta la unidad de procesamiento de un ordenador, en este caso es la Unidad Central de Procesamientos, mejor conocido como la CPU, por sus siglas en inglés “Central Processing Unit” y este usa el sistema binario.



- **Lenguaje ensamblador (Assembly):** Es un lenguaje que se usan en los microprocesadores, este contiene instrucciones que se convierten al lenguaje máquina, este es un nivel superior que se encuentra por encima del lenguaje máquina.



Luego, se encuentran los lenguajes de programación de alto nivel, estos son:

#### Lenguajes de Alto nivel

Estos lenguajes están diseñados para que se puedan ser más fáciles de leer, escribir y entender para los seres humanos, además, permiten que los

programadores vean cómo funciona la lógica de cada programa. Por ejemplo: Java, C, C++, C#, Python, Haskell, Prolog, etc.



Igualmente, se explicará de acuerdo al paradigma de programación, estos son los siguientes:

## 2. Según el paradigma de programación:

- **Lenguajes imperativos:** Los programadores indican paso a paso cómo debe realizarse una tarea mediante instrucciones que cambian el estado del programa.
  - Ejemplos: C, Java, Python.
- **Lenguajes declarativos:** El programador describe qué quiere lograr sin especificar cómo hacerlo. El sistema decide cómo resolverlo.
  - Ejemplos: SQL (para bases de datos), HTML (para estructuras web), Prolog (para programación lógica).
- **Lenguajes funcionales:** El programa se basa en el uso de funciones matemáticas, y los datos son inmutables. Las funciones se aplican a los datos sin alterar su estado.
  - Ejemplos: Haskell, Lisp, F#.
- **Lenguajes lógicos:** Se basan en la lógica formal y el razonamiento. Los programadores definen hechos y reglas, y el sistema resuelve consultas basadas en esos hechos.
  - Ejemplos: Prolog, Mercury.
- **Lenguajes orientados a objetos:** Se organizan en objetos, que son instancias de clases que agrupan datos y métodos para operar sobre esos datos.
  - Ejemplos: Java, C++, Python, Ruby.
- **Lenguajes concurrentes:** Permiten ejecutar múltiples tareas o procesos simultáneamente.

- Ejemplos: Go, Erlang, Java (con hilos).

De igual forma, se explicará de acuerdo al propósito del lenguaje, estos son los siguientes:

### 3. Segundo el propósito del lenguaje

- **Lenguajes de propósito general:** Son lenguajes diseñados para ser versátiles y utilizables en una amplia variedad de aplicaciones o proyectos.
  - Ejemplos: Python, Java, C++, JavaScript.
- **Lenguajes de propósito específico:** Están creados para abordar problemas dentro de un dominio particular o campo, y no están destinados a ser tan generales como los de propósito general.
  - **Lenguajes para desarrollo web:** Se utilizan principalmente en el diseño y desarrollo de aplicaciones y páginas web.
    - Ejemplos: JavaScript, HTML/CSS, PHP.
  - **Lenguajes para bases de datos:** Están especializados en la consulta y manipulación de grandes volúmenes de datos almacenados en bases de datos.
    - Ejemplo: SQL.
  - **Lenguajes científicos y matemáticos:** Se usan principalmente para cálculos complejos, análisis de datos y simulaciones.
    - Ejemplos: MATLAB, R.
  - **Lenguajes para sistemas embebidos:** Son utilizados en dispositivos con recursos limitados (como microcontroladores), donde el control sobre el hardware es esencial.
    - Ejemplos: C, Ada.

También, se explicará de acuerdo al modelo de ejecución, estos son los siguientes:

### 4. Segundo el modelo de ejecución

- **Lenguajes compilados:** El código escrito en el lenguaje fuente se convierte a código máquina o binario antes de ejecutarse. Esto generalmente permite una ejecución más rápida, pero requiere un paso de compilación previo.
  - Ejemplos: C, C++, Rust.
- **Lenguajes interpretados:** En lugar de compilar el código de antemano, un programa llamado intérprete lee y ejecuta el código fuente línea por línea, en tiempo real, durante la ejecución.
  - Ejemplos: Python, Ruby, JavaScript.

- **Lenguajes híbridos:** Combinan características de lenguajes compilados e interpretados, donde el código se convierte a un formato intermedio que luego puede ser interpretado o compilado, dependiendo del entorno de ejecución.
  - Ejemplo: Java (el código se compila a bytecode y luego se ejecuta por la Java Virtual Machine - JVM).

Existe otra forma de clasificar a los tipos de lenguajes, estos son los siguientes:

##### 5. Otros tipos de lenguajes

- **Lenguajes de marcado:** Aunque no son estrictamente lenguajes de programación, se utilizan para describir la estructura de documentos y datos de forma que sean comprensibles tanto para las máquinas como para los seres humanos.
  - Ejemplos: HTML, XML, Markdown.
- **Lenguajes de consulta:** Están diseñados específicamente para interactuar con bases de datos y extraer o manipular información de manera eficiente mediante consultas.
  - Ejemplo: SQL, SPARQL.

## TABLA COMPARATIVA DE LA CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

Asimismo, se mostrará una tabla comparativa de la clasificación de los distintos lenguajes de programación con sus ventajas y desventajas, esta es la siguiente:

Clasificación de los lenguajes de programación	Descripción	Ventajas	Desventajas	Ejemplos
Lenguajes de bajo nivel	Son lenguajes cercanos al hardware. Permiten un control detallado sobre los recursos de la máquina.	<ul style="list-style-type: none"> <li>• Alta eficiencia y velocidad.</li> <li>• Control total sobre el hardware.</li> <li>• Adecuados para sistemas embebidos.</li> </ul>	<ul style="list-style-type: none"> <li>• Dificultad para escribir y mantener.</li> <li>• Portabilidad limitada.</li> <li>• Requieren conocimiento profundo del hardware.</li> </ul>	Ej. Assembly, C

<b>Lenguajes de alto nivel</b>	Lenguajes que están más alejados del hardware, diseñados para facilitar la programación, con abstracciones que ocultan detalles complejos del sistema.	<ul style="list-style-type: none"> <li>Facilidad de uso y aprendizaje.</li> <li>Portabilidad entre plataformas.</li> <li>Amplia comunidad de soporte.</li> </ul>	<ul style="list-style-type: none"> <li>Menor control sobre los recursos del sistema.</li> <li>Puede ser más lento que los lenguajes de bajo nivel.</li> </ul>	Ej. Python, Java, C++
<b>Lenguajes orientados a objetos</b>	Enfoque en la programación basada en objetos, que encapsula datos y métodos en "clases".	<ul style="list-style-type: none"> <li>Facilita la reutilización del código.</li> <li>Modularidad y mantenimiento más fácil.</li> <li>Ideal para grandes aplicaciones.</li> </ul>	<ul style="list-style-type: none"> <li>Curva de aprendizaje más alta.</li> <li>Puede ser menos eficiente en términos de rendimiento.</li> </ul>	Ej. Java, C++, Python
<b>Lenguajes funcionionales</b>	Basados en funciones matemáticas puras, sin cambios de estado ni datos mutables.	<ul style="list-style-type: none"> <li>Promueve código limpio y predecible.</li> <li>Mejor manejo de problemas concurrentes.</li> <li>Facilidad para tratar con datos inmutables.</li> </ul>	<ul style="list-style-type: none"> <li>Difícil de aprender para quienes están acostumbrados a la programación imperativa.</li> <li>Menor rendimiento en algunos casos debido a la inmutabilidad.</li> </ul>	Ej. Haskell, Lisp, F#
<b>Lenguajes de ejecuciones comandos</b>	Lenguajes utilizados para escribir scripts que automatizan tareas o controlan aplicaciones.	<ul style="list-style-type: none"> <li>Rápido desarrollo y prototipado.</li> <li>Amplia comunidad y soporte.</li> <li>Portabilidad entre plataformas.</li> </ul>	<ul style="list-style-type: none"> <li>No ideales para tareas de alto rendimiento.</li> <li>Menos control sobre el sistema.</li> </ul>	Ej. Python, JavaScript, Ruby
<b>Lenguajes narrativos</b>	El programador especifica lo que desea lograr, y no	<ul style="list-style-type: none"> <li>Facilitan la descripción de resultados sin</li> </ul>	<ul style="list-style-type: none"> <li>Limitados en cuanto a</li> </ul>	Ej. SQL, HTML, CSS

	cómo hacerlo (en contraste con los lenguajes imperativos).	<ul style="list-style-type: none"> <li>preocuparse por los detalles de implementación.</li> <li>Sintaxis clara y concisa.</li> </ul>	<ul style="list-style-type: none"> <li>flexibilidad y control.</li> <li>Pueden no ser adecuados para tareas complejas.</li> </ul>	
<b>lenguajes de ramación lógica</b>	Basados en la lógica matemática, donde se define un conjunto de reglas y hechos, y el sistema deduce respuestas.	<ul style="list-style-type: none"> <li>Útiles para resolver problemas complejos de lógica.</li> <li>Facilita la solución de problemas con grandes bases de datos de conocimiento.</li> </ul>	<ul style="list-style-type: none"> <li>Menor eficiencia en comparación con otros paradigmas.</li> <li>Dificultad para integrarse con sistemas tradicionales.</li> </ul>	Ej. Prolog
<b>lenguajes de ramación concurrente</b>	Están diseñados para manejar procesos simultáneos de manera eficiente, facilitando la programación de sistemas que requieren concurrencia.	<ul style="list-style-type: none"> <li>Adecuado para sistemas distribuidos y aplicaciones de alta concurrencia.</li> <li>Mayor rendimiento en sistemas multi-core.</li> </ul>	<ul style="list-style-type: none"> <li>Complejidad en la gestión de la concurrencia.</li> <li>Requiere una mentalidad diferente respecto a la programación tradicional.</li> </ul>	Ej. Go, Erlang
<b>lenguajes de ramación visual</b>	Utilizan representaciones gráficas para construir programas mediante la manipulación de elementos visuales.	<ul style="list-style-type: none"> <li>Muy fáciles de aprender y usar, especialmente para principiantes y niños.</li> <li>Promueven la creatividad en el diseño de programas.</li> </ul>	<ul style="list-style-type: none"> <li>Limitados en cuanto a complejidad y funcionalidad.</li> <li>No son adecuados para desarrollar aplicaciones profesionales complejas.</li> </ul>	Ej. Scratch, Blockly

Tabla 2: Tabla comparativa de las diferentes clasificaciones de los tipos de lenguajes de programación.

## EJEMPLOS DE LOS LENGUAJES DE PROGRAMACIÓN MÁS

## COMUNES

Algunos de los lenguajes de programación más comunes son:

- Java
- C
- C++
- C#
- HTML
- CSS
- JavaScript
- Python
- Haskell
- Prolog
- SQL

## TABLA COMPARATIVA DE LOS DIFERENTES TIPOS DE LOS LENGUAJES DE PROGRAMACIÓN MÁS COMUNES

Aquí se muestra una tabla comparativa de los lenguajes de programación más comunes, con sus descripciones, ventajas y desventajas, esta sería:

Lenguaje de Programación	Descripción	Ventajas	Desventajas
Java	Lenguaje de programación orientado a objetos, ampliamente utilizado para aplicaciones de gran escala, aplicaciones web y móviles (Android).	<ul style="list-style-type: none"><li>• Portabilidad (gracias a la JVM).<ul style="list-style-type: none"><li>• Gran comunidad y soporte.</li><li>• Seguridad robusta.</li><li>• Buen rendimiento en aplicaciones grandes.</li></ul></li></ul>	<ul style="list-style-type: none"><li>• Más lento que lenguajes como C o C++.</li><li>• Consumo de memoria alto.</li><li>• Verbosidad en el código.</li></ul>
C	Lenguaje de bajo nivel, ampliamente utilizado en sistemas	<ul style="list-style-type: none"><li>• Alto rendimiento y eficiencia.</li><li>• Control total sobre el hardware.</li></ul>	<ul style="list-style-type: none"><li>• Difícil de aprender para principiantes.</li><li>• Gestión manual de</li></ul>

	operativos, controladores de hardware y aplicaciones embebidas.	<ul style="list-style-type: none"> <li>• Amplia portabilidad.</li> </ul>	<ul style="list-style-type: none"> <li>memoria.</li> <li>• Propenso a errores (ej. desbordamiento de búfer).</li> </ul>
<b>C++</b>	Lenguaje de programación que extiende a C con soporte para la programación orientada a objetos. Usado en aplicaciones de alto rendimiento como videojuegos y sistemas operativos.	<ul style="list-style-type: none"> <li>• Gran control sobre los recursos del sistema.</li> <li>• Soporte para programación orientada a objetos y genérica.</li> <li>• Alto rendimiento.</li> </ul>	<ul style="list-style-type: none"> <li>• Complejo y difícil de aprender.</li> <li>• Administración manual de memoria.</li> <li>• Puede ser propenso a errores difíciles de rastrear.</li> </ul>
<b>C#</b>	Lenguaje orientado a objetos desarrollado por Microsoft, utilizado principalmente en el desarrollo de aplicaciones para Windows y en el ecosistema .NET.	<ul style="list-style-type: none"> <li>• Gran integración con el ecosistema Microsoft.</li> <li>• Más sencillo que C++ para desarrolladores.</li> <li>• Buen soporte para desarrollo de aplicaciones web y móviles.</li> </ul>	<ul style="list-style-type: none"> <li>• Menos eficiente que C o C++.</li> <li>• Limitado a la plataforma Windows (aunque ahora es multiplataforma con .NET Core).</li> </ul>
<b>HTML</b>	Lenguaje de marcado utilizado para estructurar contenido web. No es un lenguaje de programación, sino un lenguaje de marcado.	<ul style="list-style-type: none"> <li>• Fácil de aprender.</li> <li>• Esencial para la creación de sitios web.</li> <li>• Muy utilizado en el desarrollo web.</li> </ul>	<ul style="list-style-type: none"> <li>• No es interactivo ni dinámico.</li> <li>• Depende de otros lenguajes (como CSS y JavaScript) para crear sitios completos.</li> </ul>
<b>CSS</b>	Lenguaje de estilo utilizado para definir la presentación visual de las páginas web	<ul style="list-style-type: none"> <li>• Facilidad para separar el contenido de la presentación.</li> <li>• Gran flexibilidad para diseñar</li> </ul>	<ul style="list-style-type: none"> <li>• Difícil de depurar en proyectos grandes.</li> <li>• Algunas características</li> </ul>

	creadas con HTML.	<p>interfaces.</p> <ul style="list-style-type: none"> <li>• Amplio soporte en todos los navegadores.</li> </ul>	no son completamente compatibles entre navegadores.
<b>JavaScript</b>	Lenguaje de programación interpretado que permite crear páginas web interactivas y dinámicas, y también se utiliza en el desarrollo del lado del servidor (Node.js).	<ul style="list-style-type: none"> <li>• Esencial para el desarrollo web moderno.</li> <li>• Compatible con todos los navegadores web.</li> <li>• Gran comunidad y bibliotecas como React y Angular.</li> </ul>	<ul style="list-style-type: none"> <li>• Puede ser difícil de depurar.</li> <li>• Falta de tipado fuerte (aunque TypeScript lo soluciona).</li> <li>• No es tan eficiente como otros lenguajes como C++.</li> </ul>
<b>Python</b>	Lenguaje de alto nivel, interpretado, conocido por su sintaxis simple y su enfoque en la legibilidad del código. Usado en desarrollo web, ciencia de datos, inteligencia artificial, y más.	<ul style="list-style-type: none"> <li>• Sintaxis clara y fácil de aprender.</li> <li>• Amplia comunidad y soporte.</li> <li>• Gran versatilidad y numerosas bibliotecas.</li> </ul>	<ul style="list-style-type: none"> <li>• Menor rendimiento que lenguajes compilados como C++.</li> <li>• Consumo de memoria alto.</li> </ul>
<b>Haskell</b>	Lenguaje funcional puro, utilizado para tareas que requieren una programación altamente matemática o lógica, como en investigación y sistemas complejos.	<ul style="list-style-type: none"> <li>• Excelente para resolver problemas matemáticos y lógicos complejos.</li> <li>• Programación pura e inmutabilidad.</li> <li>• Concisión en el código.</li> </ul>	<ul style="list-style-type: none"> <li>• Difícil de aprender y usar.</li> <li>• Comunidad y recursos limitados en comparación con otros lenguajes.</li> </ul>
<b>Prolog</b>	Lenguaje de programación lógica, basado en hechos y reglas,	<ul style="list-style-type: none"> <li>• Ideal para aplicaciones de inteligencia artificial y</li> </ul>	<ul style="list-style-type: none"> <li>• No es adecuado para aplicaciones de</li> </ul>

	utilizado principalmente en inteligencia artificial y procesamiento de lenguaje natural.	razonamiento lógico. • Permite resolver problemas complejos de forma declarativa.	alto rendimiento. • Difícil de integrar con otros lenguajes o sistemas.
<b>SQL</b>	Lenguaje utilizado para la gestión y manipulación de bases de datos relacionales. Permite realizar consultas, actualizaciones y administración de datos.	• Esencial para trabajar con bases de datos. • Sintaxis simple y fácil de aprender. • Potente y eficiente para gestionar grandes volúmenes de datos.	• No es adecuado para tareas complejas fuera de la gestión de bases de datos. • La optimización de consultas puede ser compleja en bases de datos grandes.

Tabla 3: Tabla comparativa de los diferentes tipos de lenguajes de programación más comunes.

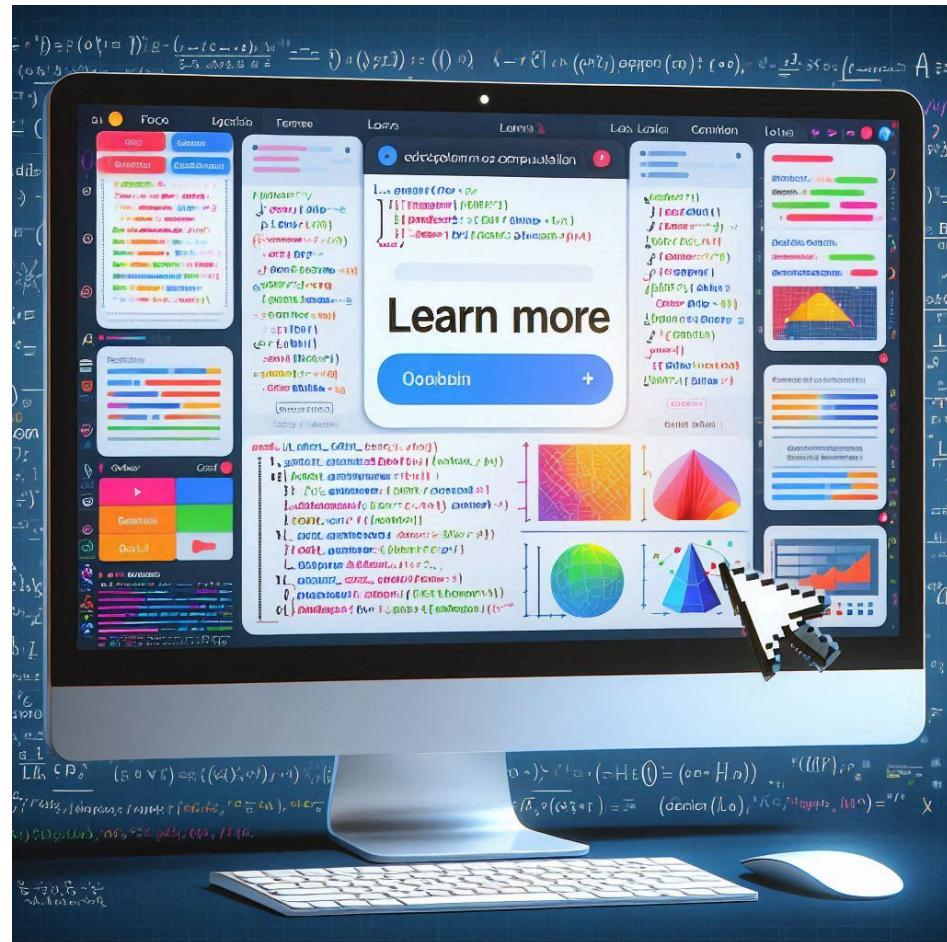
**Notas adicionales:**

- Java: Ideal para aplicaciones grandes y multiplataforma.
- C: Perfecto para sistemas de bajo nivel y alto rendimiento.
- C++: Extiende a C, adecuado para desarrollo de videojuegos y software de alto rendimiento.
- C# : Lenguaje moderno, fácil de aprender, ideal para aplicaciones Windows.
- HTML/CSS: Imprescindibles para la creación de sitios web, aunque no son lenguajes de programación.
- JavaScript: Esencial para desarrollo web interactivo, usado tanto en el frontend como en el backend.
- Python: Lenguaje versátil y fácil de aprender, usado en ciencia de datos, IA y desarrollo web.
- Haskell: Lenguaje funcional ideal para problemas lógicos y matemáticos complejos.
- Prolog: Orientado a la lógica y la inteligencia artificial, basado en hechos y reglas.
- SQL: El estándar para la manipulación y gestión de bases de datos.

En resumen, cada tipo de lenguaje tiene su propósito y contexto donde brilla más, y su elección depende de las necesidades específicas del proyecto o la tarea a realizar.



## CAPÍTULO 2. PÁGINAS WEB



En este caso, nos enfocaremos con los lenguajes que nos ayuden a la creación de páginas web, como son: HTML, CSS y JavaScript (JS).

## LENGUAJE HTML (HYPERTEXT MARKUP LANGUAGE)

### CONCEPTO

El lenguaje de programación de HTML es un **acrónimo<sup>1</sup>** del idioma en inglés de “HyperText Markup Language”, en español se traduce como el lenguaje de marcado de hipertexto.

El HTML es el lenguaje de programación que a su vez es un lenguaje de marcado que sirve para crear, gestionar las páginas web. Usa una serie de elementos o etiquetas lo cual indica al navegador cómo se debe de presentar el contenido, pueden ser textos, imágenes, enlaces o links, encuestas, etc.



Figura: Logo del lenguaje de HTML.

### CARACTERÍSTICAS DEL HTML

- Estructuración del contenido: HTML organiza el contenido de una página web en una estructura jerárquica mediante etiquetas. Por ejemplo, `<h1>` se usa para los títulos, `<p>` para los párrafos y `<img>` para las imágenes.
- Etiquetas semánticas: Con la llegada de HTML5, se añadieron etiquetas semánticas como `<header>`, `<footer>`, `<article>`, y `<section>`, que hacen más fácil entender la organización del contenido tanto para los usuarios como para los motores de búsqueda.
- Soporte multimedia: HTML5 permite la inclusión de contenido multimedia de forma nativa, como archivos de audio y video, utilizando etiquetas como `<audio>` y `<video>`, sin necesidad de usar complementos adicionales.

- Creación de formularios: HTML ofrece herramientas para desarrollar formularios interactivos, permitiendo recolectar información de los usuarios mediante etiquetas como `<input>`, `<textarea>`, y `<button>`.

## ESTRUCTURA BÁSICA DE UN DOCUMENTO EN HTML

En adición, se muestra la estructura básica de un documento que utiliza HTML. El archivo HTML (como ejemplo: nombre1.html) define la estructura de la página. Este es el esqueleto y contiene todos los elementos que se mostrarán al usuario, como por ejemplo: títulos, párrafos, imágenes, enlaces o links, encuestas, etc.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mi Página Web</title>
</head>
<body>
    <h1>Bienvenido a mi página web</h1>
    <p>Este es un párrafo para ilustrar la estructura de un documento HTML.</p>
</body>
</html>
```

## EXPLICACIÓN DE LA ESTRUCTURA BÁSICA DE UN DOCUMENTO EN HTML

### **<!DOCTYPE html>:**

- Especifica que el documento es un archivo HTML5.
- Es importante porque le dice al navegador que la página debe ser procesada como HTML5.

### **<html lang="es">:**

- Es el elemento raíz del documento HTML.
- El atributo `lang="es"` indica que el contenido está en español, lo que ayuda a los motores de búsqueda y lectores de pantalla.

### **<head>:**

- Contiene metadatos sobre el documento, como el título de la página, enlaces a hojas de estilo, y configuraciones de codificación.
- No es visible en la página, pero es esencial para la configuración de la página web.

### **<meta charset="UTF-8">:**

- Define la codificación de caracteres del documento. UTF-8 es el estándar más común, que permite usar caracteres especiales y acentuados correctamente.

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**:

- Hace que la página sea responsive, adaptándose a diferentes tamaños de pantalla, como los de dispositivos móviles.

**<title>:**

- Define el título de la página que aparece en la pestaña del navegador.

**<body>:**

- Contiene todo el contenido visible de la página web, como texto, imágenes, enlaces, y otros elementos HTML.
- Es la parte que se muestra al usuario en el navegador.

**<h1>:**

- Define un encabezado de nivel 1, que se utiliza para títulos principales.
- Es importante para la organización del contenido y la accesibilidad.

**<p>:**

- Define un párrafo de texto.
- Es uno de los elementos más comunes en un documento HTML.

## COMENTARIOS EN HTML

En HTML, los comentarios son notas que se incluyen dentro del código para proporcionar explicaciones o aclaraciones, pero no afectan la visualización de la página en el navegador. Son útiles para que otros programadores, o incluso uno mismo en el futuro, comprendan el propósito de ciertas partes del código. Los comentarios no son visibles cuando se carga la página web, ya que solo se encuentran en el código fuente.

Para escribir un comentario en HTML, se utiliza la siguiente estructura:

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

Es decir, los comentarios comienzan con <!- - y terminan con - ->. Todo lo que se coloque entre estas marcas no se mostrará en el navegador, pero estará presente en el código fuente.

Un ejemplo de cómo se usan los comentarios en un código HTML sería:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Mi Página Web</title>
</head>
```

```
<body>
<!-- Este es el encabezado principal -->
<h1>Bienvenido a mi página</h1>

<!-- Este es un párrafo de texto -->
<p>Este es un párrafo de ejemplo para ilustrar los comentarios en HTML.</p>

<!-- El siguiente enlace lleva a la página principal -->
<a href="https://www.example.com">Visita mi página favorita</a>
</body>
</html>
```

Es importante tener en cuenta que en HTML no se pueden anidar comentarios. Es decir, no se puede poner un comentario dentro de otro, ya que esto causaría un error en la interpretación del código.

Un ejemplo incorrecto sería:

```
<!-- Este es un comentario <!-- Este es un comentario anidado --> -->
```

Y el correcto sería:

```
<!-- Este es un comentario -->
<!-- Otro comentario -->
```

Las ventajas de los comentarios incluyen:

1. Documentación del código: Permiten explicar el propósito de una sección del código, facilitando su comprensión.
2. Desactivación temporal de código: Se puede comentar una parte del código para desactivarla sin eliminarla, útil durante el desarrollo.
3. Colaboración en equipo: Ayudan a que otros programadores comprendan el funcionamiento del código.

## FALTAN LAS DESVENTAJAS DE LOS COMENTARIOS EN HTML

Los comentarios en HTML son herramientas fundamentales para organizar, explicar y documentar el código, pero, no afectan ni dañan la estructura ni el rendimiento de la página web.

## ETIQUETAS EN HTML

Las etiquetas son los componentes esenciales que determinan la estructura y el contenido de una página web. Cada etiqueta se compone de dos partes: una etiqueta de apertura y una de cierre, y el contenido que se desea mostrar o modificar se coloca entre estas dos etiquetas.

Las etiquetas HTML son instrucciones que le indican al navegador cómo debe interpretar y mostrar el contenido de una página web. Estas etiquetas están rodeadas por corchetes angulares (< >) y sirven para señalar el comienzo y el final de un bloque o elemento de contenido.

## ESTRUCTURA BÁSICA DE UNA ETIQUETA

- Etiqueta de apertura: Esta etiqueta, que se coloca entre corchetes angulares, marca el inicio de un elemento. Ejemplo: <h1>.
- Contenido: Es el texto o los elementos que se encuentran dentro de la etiqueta. Ejemplo: "Bienvenido a mi página".
- Etiqueta de cierre: Es similar a la etiqueta de apertura, pero con una barra diagonal (/) antes del nombre de la etiqueta. Ejemplo: </h1>.

## EJEMPLO

```
<h1>Bienvenido a mi página</h1>
```

En este ejemplo, se explica que:

- <h1> es la etiqueta de apertura que indica que el contenido debe mostrarse como un encabezado de nivel 1.
- Contenido:"Bienvenido a mi página" es el texto que aparecerá en la página web.
- </h1> es la etiqueta de cierre, que marca el final del encabezado.

## TIPOS DE ETIQUETAS EN HTML

1. **Etiquetas de contenedor:** Son etiquetas que tienen tanto una de apertura como una de cierre. Ejemplo: <p></p>, <div></div>.
  - <p>: Se utiliza para definir un párrafo de texto.
  - <div>: Sirve para agrupar otros elementos dentro de un contenedor genérico.
2. **Etiquetas auto-cerradas:** Estas etiquetas no necesitan una etiqueta de cierre porque no contienen contenido dentro de ellas. Ejemplo: <img />, <br />.
  - <img />: Se usa para insertar una imagen y solo necesita el atributo de la fuente de la imagen.
  - <br />: Se utiliza para hacer un salto de línea.
3. **Etiquetas de formato de texto:** Estas etiquetas permiten aplicar estilos específicos al texto, como negritas, cursivas o subrayados.
  - <b>: Hace que el texto sea negrita.
  - <i>: Aplica cursiva al texto.
  - <u>: Subraya el texto.
  - <strong>: Indica que el texto es importante, aplicando un énfasis adicional.
4. **Etiquetas de enlaces:** Permiten crear vínculos a otras páginas web o recursos.

- <a>: Se utiliza para definir un enlace, y su atributo href especifica la dirección del enlace. Ejemplo: <p>Bienvenido a mi sitio web. <a href="https://www.ejemplo.com">Haz clic aquí</a> para obtener más información.</p> (En este ejemplo: <p> inicia un párrafo y </p> lo cierra.)
- <a href="https://www.ejemplo.com"> crea un enlace hacia la página web indicada.)

## EJEMPLOS DE LAS ETIQUETAS MÁS COMUNES EN HTML

Las etiquetas más comunes en HTML son:

### 1. Etiquetas de estructura básica:

- <!DOCTYPE html>: Declara el tipo de documento y especifica que se está utilizando HTML5.
- <html>: La etiqueta raíz de un documento HTML, que contiene todo el contenido de la página.
- <head>: Contiene metadatos sobre el documento, como el título de la página, vínculos a hojas de estilo, y scripts.
- <body>: Contiene el contenido visible de la página web, como textos, imágenes, y enlaces.

### 2. Etiquetas de texto y formato:

- <h1>, <h2>, <h3>, <h4>, <h5>, <h6>: Son las etiquetas de encabezado, donde <h1> es el más importante (generalmente para el título principal) y <h6> es el menos importante.
- <p>: Define un párrafo de texto.
- <b>: Aplica negrita al texto.
- <i>: Aplica cursiva al texto.
- <u>: Subraya el texto.
- <strong>: Indica que el texto es importante, usualmente se muestra en negrita.
- <em>: Hace énfasis en el texto, comúnmente en cursiva.
- <br />: Inserta un salto de línea.
- <hr />: Inserta una línea horizontal para dividir secciones del contenido.

### 3. Etiquetas de enlaces y navegación:

- <a>: Define un enlace. Se usa con el atributo href para especificar la dirección de destino del enlace.
- <nav>: Define una sección de navegación en la página, como un menú de enlaces.

### 4. Etiquetas de listas:

- <ul>: Define una lista desordenada (con viñetas).
- <ol>: Define una lista ordenada (numerada).
- <li>: Define un elemento dentro de una lista (se usa dentro de <ul> o <ol>).

### 5. Etiquetas de imágenes y multimedia:

- <img />: Inserta una imagen en la página. Usa el atributo src para especificar la ruta de la imagen.
- <audio>: Permite insertar contenido de audio.
- <video>: Permite insertar contenido de video.

#### 6. Etiquetas de formularios:

- <form>: Define un formulario donde el usuario puede ingresar datos.
- <input>: Define un campo de entrada, como una caja de texto o un botón de envío.
- <label>: Define una etiqueta para un control de formulario.
- <textarea>: Define un área de texto multilínea.
- <button>: Define un botón que puede ser clicado por el usuario.

#### 7. Etiquetas de contenedores y estructuras:

- <div>: Define un contenedor genérico para agrupar otros elementos. Es muy útil para la organización del contenido.
- <span>: Similar a <div>, pero se utiliza para aplicar estilos o manipular pequeñas porciones de contenido en línea (sin afectar el flujo del texto).
- <section>: Define una sección del documento, como una división temática de contenido.
- <article>: Define un artículo independiente dentro de la página.
- <header>: Define el encabezado de la página o de una sección.
- <footer>: Define el pie de página de la página o de una sección.

#### 8. Etiquetas de tablas:

- <table>: Define una tabla.
- <tr>: Define una fila dentro de una tabla.
- <td>: Define una celda dentro de una fila.
- <th>: Define una celda de encabezado (normalmente en negrita y centrada).
- <thead>, <tbody>, <tfoot>: Definen las secciones de una tabla para encabezado, cuerpo y pie de tabla.

#### 9. Etiquetas de metadatos:

- <meta>: Proporciona información sobre la página, como el conjunto de caracteres o la descripción de la página.
- <title>: Define el título de la página, que aparece en la barra del navegador.

#### 10. Otras etiquetas comunes:

- <link> : Se usa para vincular un documento HTML con recursos externos, como hojas de estilo CSS.
- <script>: Se utiliza para incluir o vincular código JavaScript en la página web.

Las etiquetas HTML son los bloques fundamentales que estructuran el contenido y determinan cómo se presenta en una página web. Pueden ser de tipo contenedor (con apertura y cierre) o auto-cerradas (sin necesidad de cierre). Son esenciales para desarrollar páginas web organizadas, comprensibles y accesibles.

## **IMPORTANCIA DEL HTML**

HTML es fundamental para el desarrollo de la web, ya que es la base sobre la cual se construyen todas las páginas. Sin él, no sería posible la existencia de las páginas web tal como las conocemos. Su función principal es organizar y estructurar el contenido de manera que los navegadores puedan interpretarlo y presentarlo correctamente al usuario.

## **VENTAJAS DEL HTML**

Las ventajas del HTML son las siguientes:

Fácil de aprender y utilizar:

- HTML es uno de los lenguajes más accesibles, ya que no requiere conocimientos avanzados de programación. Su sintaxis es simple y clara, lo que facilita que los principiantes aprendan rápidamente a crear páginas web.

Estandarización:

- HTML está estandarizado por el W3C (World Wide Web Consortium), lo que garantiza su compatibilidad entre diferentes navegadores. Cualquier navegador moderno puede interpretar el código HTML de manera consistente.

Amplio soporte:

- HTML es el lenguaje base de la web y es compatible con todos los navegadores web, lo que permite que las páginas HTML sean visibles en cualquier dispositivo (computadoras, tabletas, teléfonos móviles).

Compatibilidad con otras tecnologías:

- HTML se complementa con otras tecnologías web como CSS (para el diseño) y JavaScript (para interactividad). Esto permite crear páginas web dinámicas y visualmente atractivas.

Estructuración de contenido:

- HTML permite organizar y estructurar el contenido de manera lógica utilizando etiquetas, como encabezados, párrafos, listas, tablas, etc., lo que mejora la accesibilidad y el SEO (optimización para motores de búsqueda).

Gratis y de código abierto:

- HTML es gratuito y no requiere licencias de uso. Además, es de código abierto, lo que significa que cualquier persona puede utilizarlo y contribuir a su desarrollo.



## DESVENTAJAS DEL HTML

Las desventajas del HTML son las siguientes:

Limitado en términos de diseño y funcionalidad:

- HTML solo se utiliza para estructurar el contenido, pero no tiene capacidades avanzadas para crear diseños complejos o agregar interactividad (para eso, se necesita CSS y JavaScript).

No soporta interactividad avanzada:

- A pesar de que HTML puede incluir elementos interactivos como formularios, botones, y enlaces, la interactividad avanzada, como animaciones o validaciones, requiere el uso de JavaScript. Sin JavaScript, HTML por sí solo no puede hacer mucho en términos de interacción dinámica.

No es adecuado para aplicaciones web complejas:

- Para desarrollar aplicaciones web complejas, HTML solo no es suficiente. Se necesita usar frameworks y lenguajes adicionales, como JavaScript, para crear funcionalidades dinámicas y gestionar la lógica del programa.

No es adecuado para programación de lógica:

- HTML no tiene capacidades de programación lógica o control de flujo (como condicionales o bucles), por lo que no puede realizar tareas que impliquen decisiones o cálculos complejos, para lo cual se deben utilizar otros lenguajes como JavaScript.

Dependencia de otros lenguajes:

- HTML, por sí solo, no puede proporcionar una experiencia de usuario avanzada ni el diseño visual completo. Necesita ser combinado con CSS (para el estilo y diseño) y JavaScript (para la interactividad).

No es flexible para diseño responsivo:

- Aunque HTML5 ha mejorado el diseño responsivo (que se adapta a diferentes tamaños de pantalla), HTML por sí solo no tiene un mecanismo completo para

crear interfaces de usuario que respondan bien a diferentes dispositivos. Para esto, es necesario utilizar CSS y técnicas como el diseño adaptable (responsive design).



## CREACIÓN DE UN PROGRAMA EN HTML

En este apartado, vamos a comenzar a usar el lenguaje de HTML, pero, primero vamos a crear un programa en HTML usando Visual Studio Code.

### REALIZACIÓN DE UN PROGRAMA EN HTML

En esta sección, se van a exponer los pasos a seguir en la realización de un programa, por medio de un ejemplo. Para poder escribir programas se necesita un editor de código fuente, esto es: Visual Studio Code (Este se encuentra en el apéndice X)

#### Creación de tu primer programa

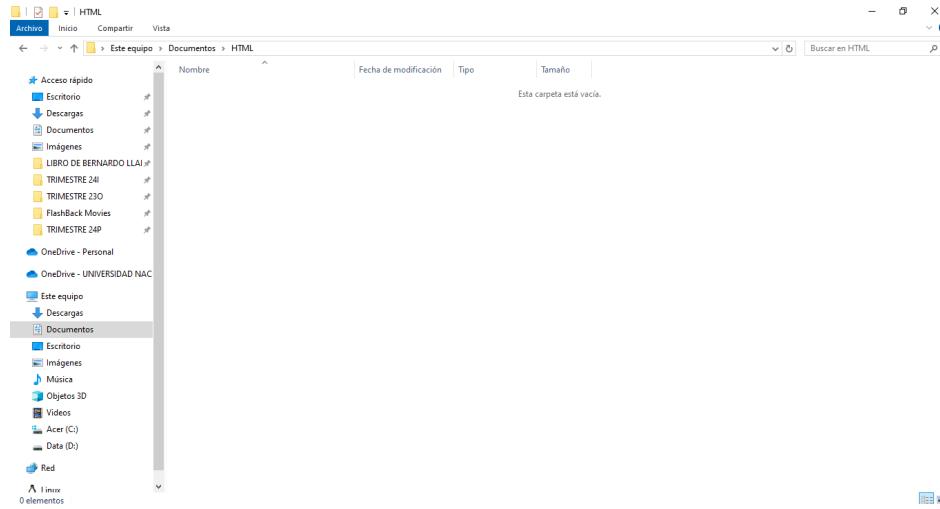
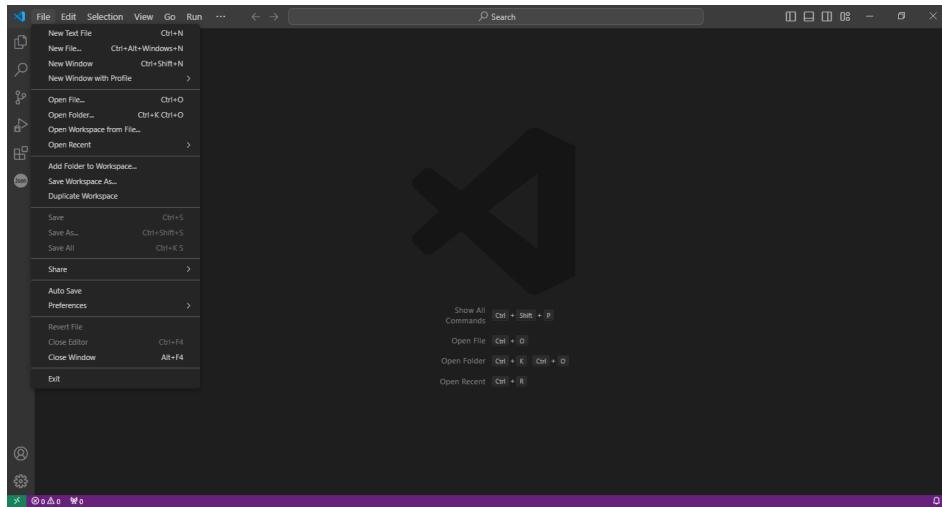
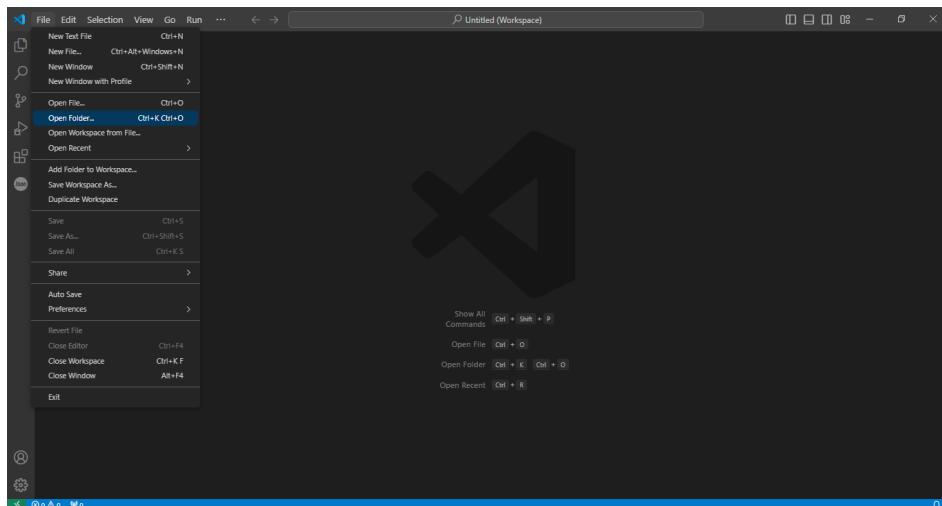


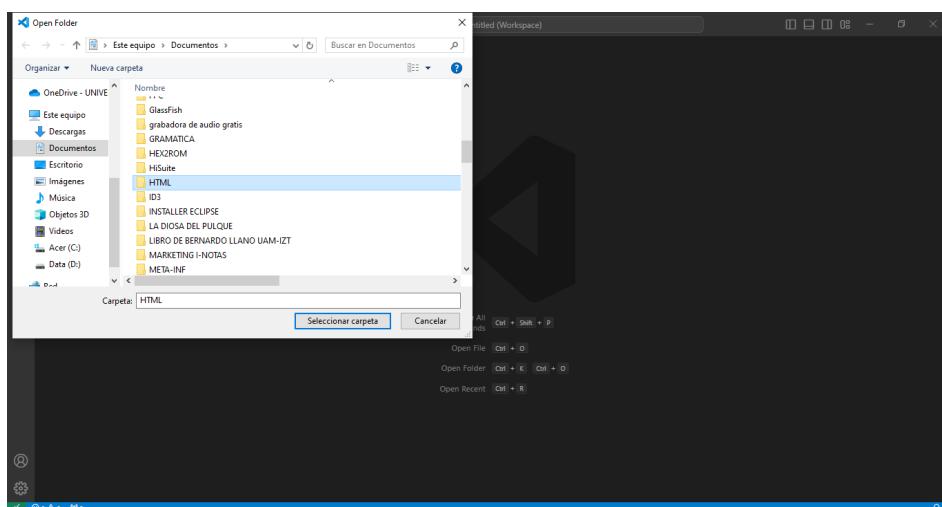
Figura: En el explorador de archivos, haz una carpeta donde guardarás tus documentos.



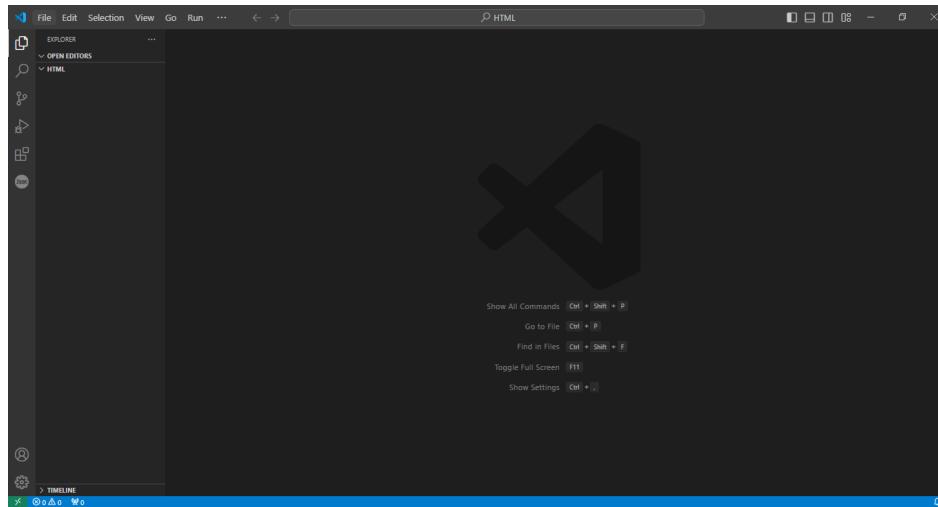
**Figura: En la aplicación de Visual Studio Code (VSC) abrir la pestaña de “File” (en español, se traduce como Archivos).**



**Figura: Dentro de la pestaña de File, dar clic en la opción de “Open Folder” (en español, se traduce como Abrir Carpeta).**



**Figura: Se te abrirá una ventana para ubicar tu carpeta, búscala y seleccionarla con el doble clic o poner la opción que dice “Seleccionar carpeta”.**



**Figura: Dentro del VSC, te debe de aparecer en la parte superior el nombre de tu carpeta, en este caso es “HTML”, así como se muestra.**

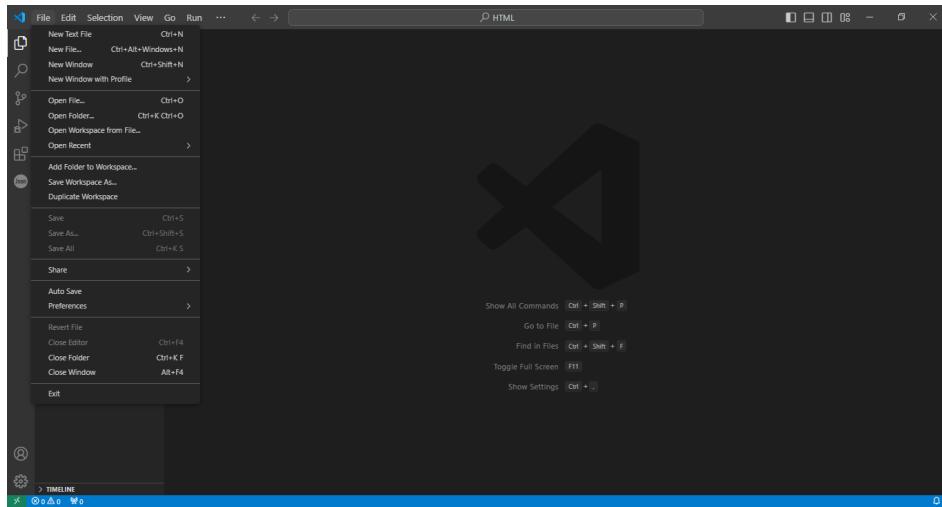
## **REALIZACIÓN DE UN PROGRAMA EN HTML**

En este apartado se van a explicar los pasos (detalladamente) a seguir en la realización de un programa.

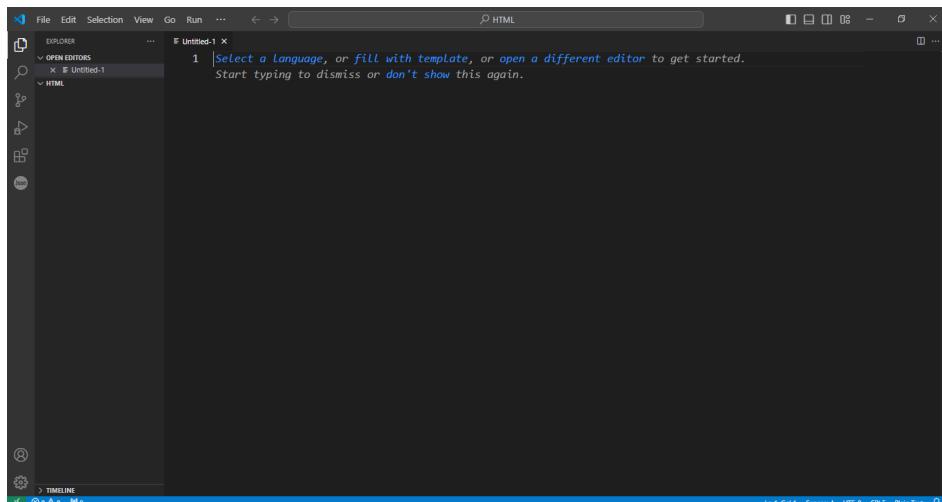
Ciertamente, para comenzar a escribir programas se necesita un editor de código fuente; esto es el Visual Studio Code (VSC).

.....

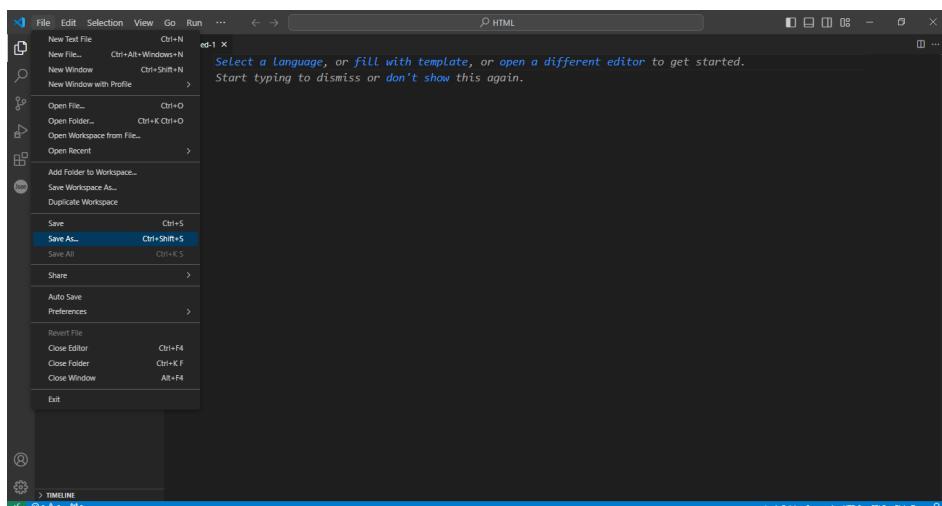
Partiendo de la página de inicio de Visual Studio Code, hacemos clic en la pestaña “File” (en español, se traduce como Archivos) y dar clic en la opción de “New Text File” (en español, se traduce como Nuevo Archivo de Texto) para comenzar a crear tu primer programa. En este nuevo archivo de texto, se estará desarrollando el lenguaje de programación de HTML (ver figura Y) .



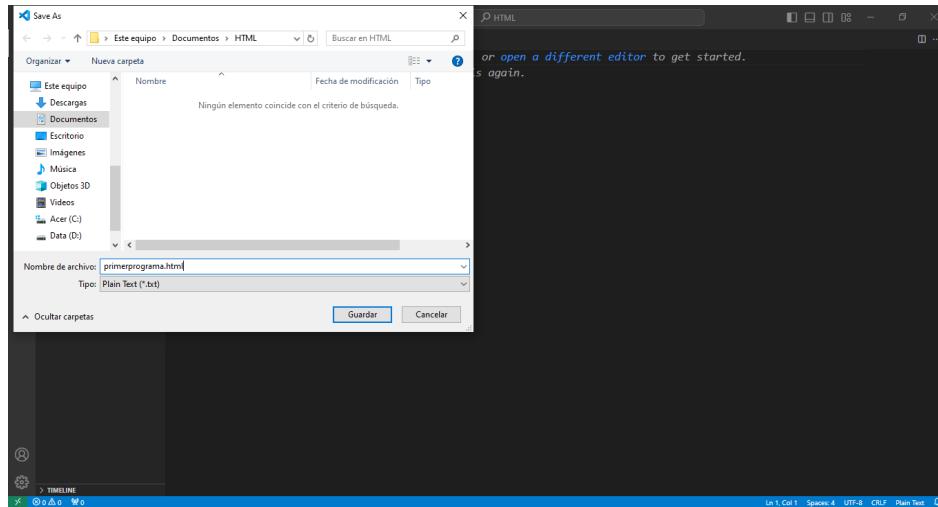
**Figura Y: En la aplicación de VSC, abrir la pestaña de “File” (Archivos) y dar clic en “New Text File” para comenzar tu primer programa.**



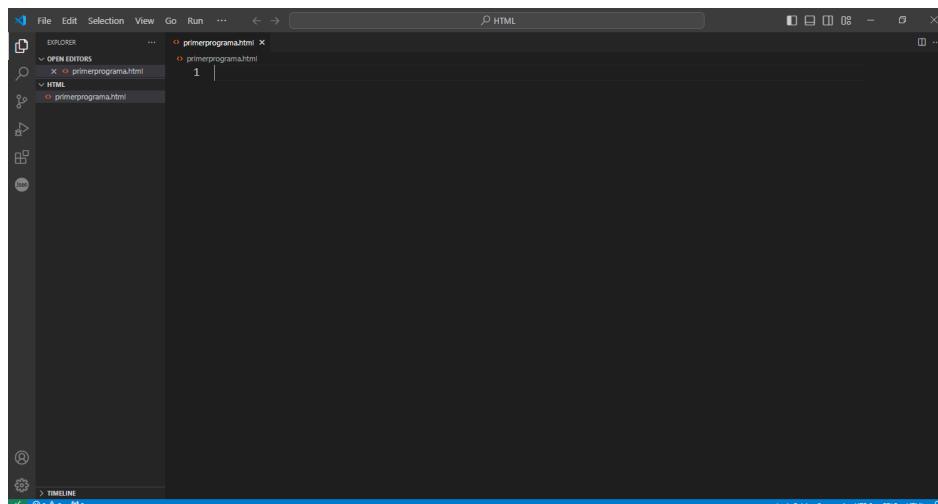
**Figura: En el VSC, se te debe de aparecer un archivo llamado “Untitled”, eso significa que todavía no tiene nombre tu archivo.**



**Figura: En la pestaña de File del VSC, dar clic en donde dice “Save As...” para guardar tu archivo.**



**Figura: Se te abrirá una ventana para que puedas nombrar a tu archivo como quieras, en este caso ponle como nombre “primerprograma”, recuerda poner la extensión de .html para que sea del lenguaje HTML y darle clic en “Guardar”.**



**Figura: Una vez hecho lo anterior, en el VSC se debe aparecer el nombre que pusiste antes o como lo hayas guardado, y si no, le puedes cambiar el nombre en el explorador de archivos, usando el clic derecho y en donde dice “Cambiar nombre”.**

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mi Primer Programa en HTML</title>
</head>
<body>
    <h1>¡Hola, Mundo!</h1>
    <p>Este es mi primer programa en HTML.</p>
</body>
</html>
```

Figura: Dentro de tu archivo que iniciaste, ya comenzarás a escribir tu código.

## CÓDIGO

### ¿Qué hace este programa?

A continuación, vamos a explicar línea por línea que hace este código.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mi Primer Programa en HTML</title>
</head>
<body>
    <h1>¡Hola, Mundo!</h1>
    <p>Este es mi primer programa en HTML.</p>
</body>
</html>
```

Figura: Primer código de “¡Hola, Mundo!” en HTML.

**<! DOCTYPE html >**

En la 1ra línea, se refiere el tipo de documento y la versión de HTML, en este caso es el HTML5.

**<html lang="es" >**

En la 2da línea, se refiere la etiqueta con la cual se abre el documento de HTML y hay que definir el idioma del contenido del archivo, en este caso es el español.

**<head>**

En la 3ra línea, se refiere ...

```
<meta charset="UTF-8">
```

En la 4ta línea, se refiere ...

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

En la 5ta línea, se refiere ...

```
<title>Mi Primer Programa en HTML</title>
```

En la 6ta línea, se refiere ...

```
</head>
```

En la 7ta línea, se refiere ...

```
<body>
```

En la 8ta línea, se refiere ...

```
<h1> ¡Hola, Mundo! </h1>
```

En la 9na línea, se refiere ...

```
<p>Este es mi primer programa en HTML.</p>
```

En la 10ma línea, se refiere ...

```
</body>
```

En la 11° (número ordinal de 11) línea, se refiere ...

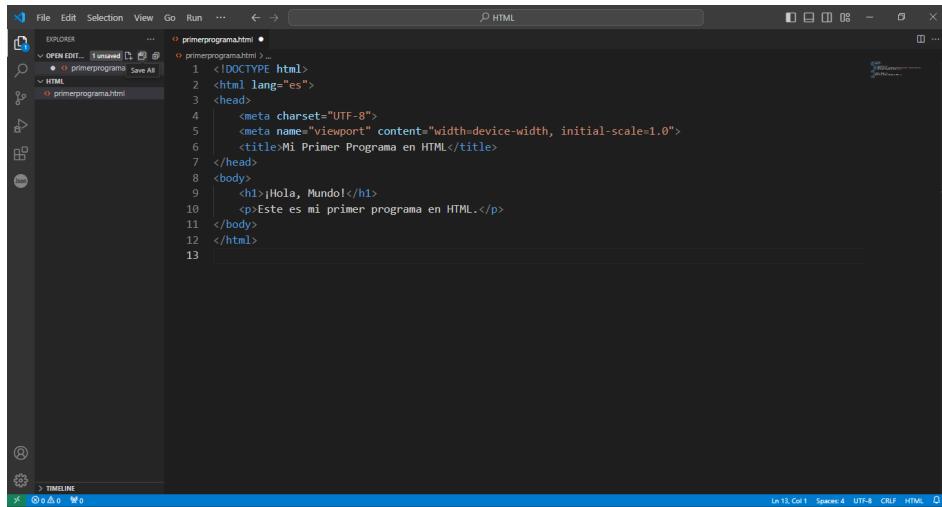
```
</html>
```

En la 12° línea, se refiere ...

## ¿CÓMO SE GUARDA?

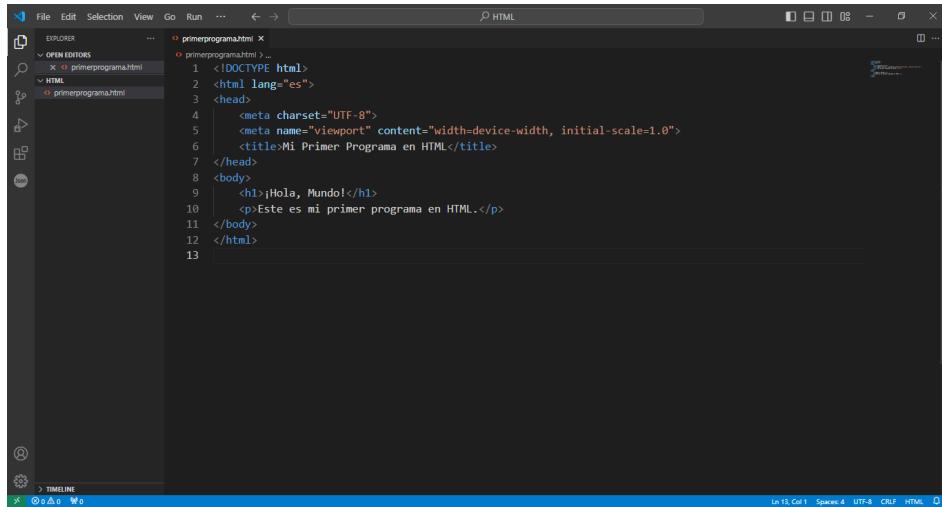
Hay 4 distintas formas de guardar tu programa, estas son las siguientes:

### OPCIÓN 1 - USANDO COMBINACIONES DE TECLAS



```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Mi Primer Programa en HTML</title>
</head>
<body>
<h1>¡Hola, Mundo!</h1>
<p>Este es mi primer programa en HTML.</p>
</body>
</html>
```

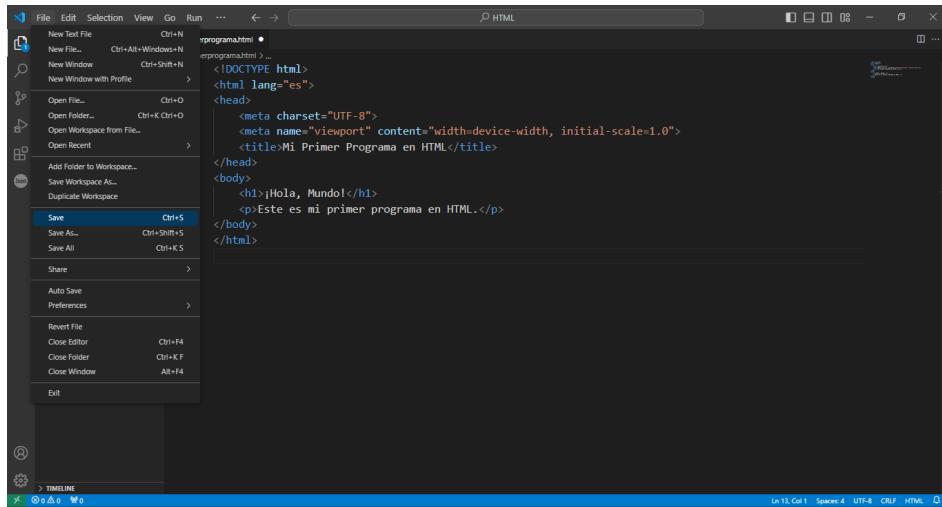
**Figura: Una vez que hayas terminado de escribir tu código, se te va a mostrar un puntito donde está el nombre de tu archivo y usando las combinaciones de teclas Ctrl+S para guardarlo rápido.**



```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Mi Primer Programa en HTML</title>
</head>
<body>
<h1>¡Hola, Mundo!</h1>
<p>Este es mi primer programa en HTML.</p>
</body>
</html>
```

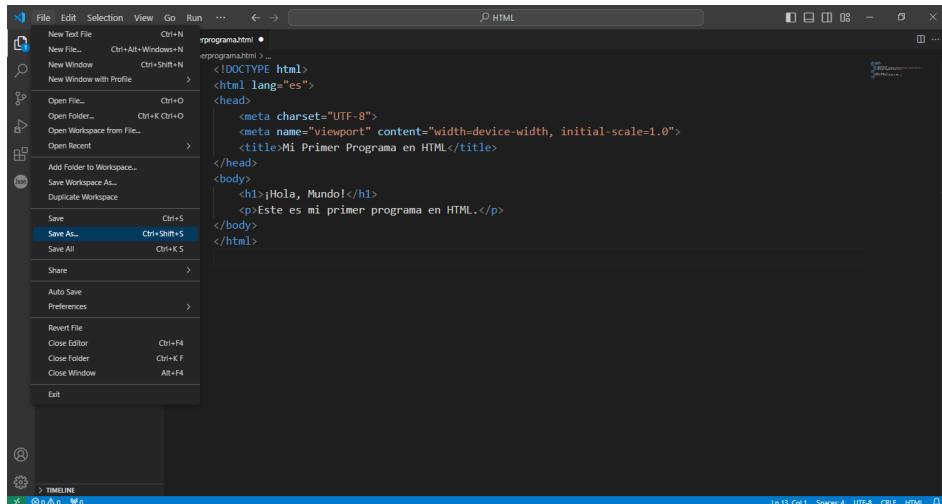
**Figura: Una vez hecho lo anterior, ya se te habrá guardado tu documento y ya no tendrás el puntito.**

## OPCIÓN 2 - USANDO LA OPCIÓN DE SAVE DE LA PESTAÑA FILE DEL VSC

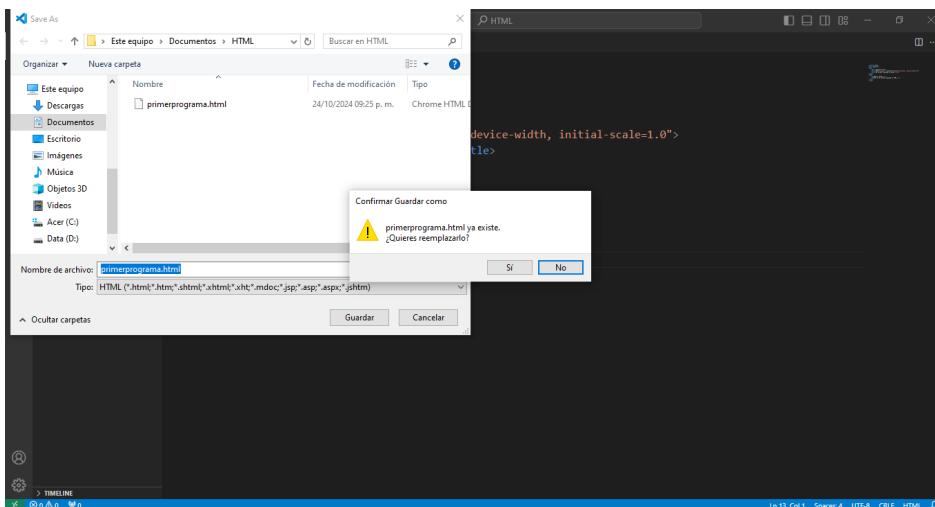


**Figura:** Una vez que hayas terminado de escribir tu código, lo que debes de hacer es irte a la pestaña de File, busca la opción que dice “Save” y da r clic para guardarlo.

### OPCIÓN 3 - USANDO LA OPCIÓN DE SAVE AS... DE LA PESTAÑA FILE DEL VSC

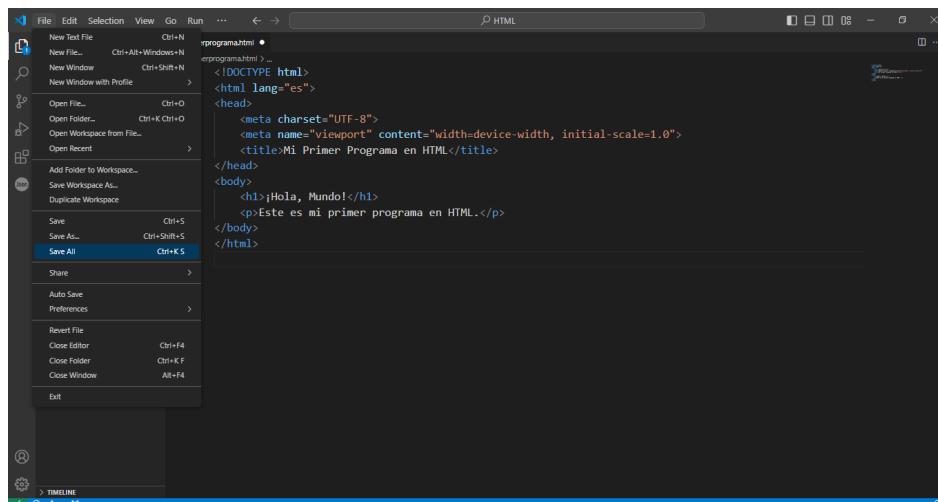


**Figura:** Una vez que hayas terminado de escribir tu código, lo que debes de hacer es irte a la pestaña de File, busca la opción que dice “Save As...”, da r clic en esta y se te abrirá la ventana de Guardar como y ahí le puedes cambiar el nombre o dejar el mismo nombre, después le das clic en Guardar, en este caso, si te sale una advertencia de reemplazarlo, debes de dar clic en la opción de Sí para guardarlo (Ver figura x).



**Figura x:** Ventana de advertencia para reemplazar el archivo, una vez hecho los cambios pertinentes.

#### OPCIÓN 4 - USANDO LA OPCIÓN SAVE ALL DE LA PESTAÑA FILE DEL VSC



**Figura:** Una vez que hayas terminado de escribir tu código, lo que debes de hacer es irte a la pestaña de File, busca la opción que dice “Save All”, da r clic en esta para guardarlo.

Al momento de realizar cualquier opción de guardado ya no te debe de aparecer el puntito que se encuentra en el nombre del archivo, así como se ve en esta figura.

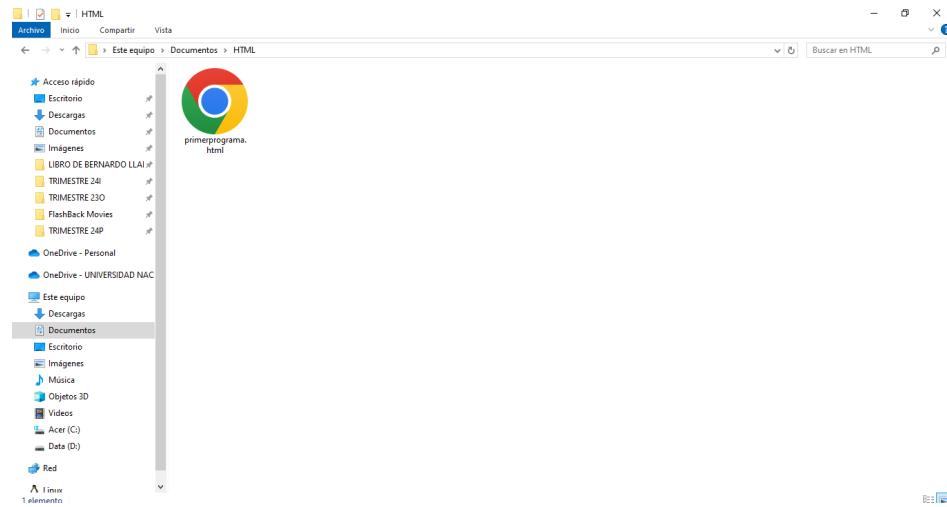
```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Mi Primer Programa en HTML</title>
</head>
<body>
<h1>¡Hola, Mundo!</h1>
<p>Este es mi primer programa en HTML.</p>
</body>
</html>
```

**Figura:** Así se ve cuando hayas guardado tu documento y ya no tendrás el puntito.

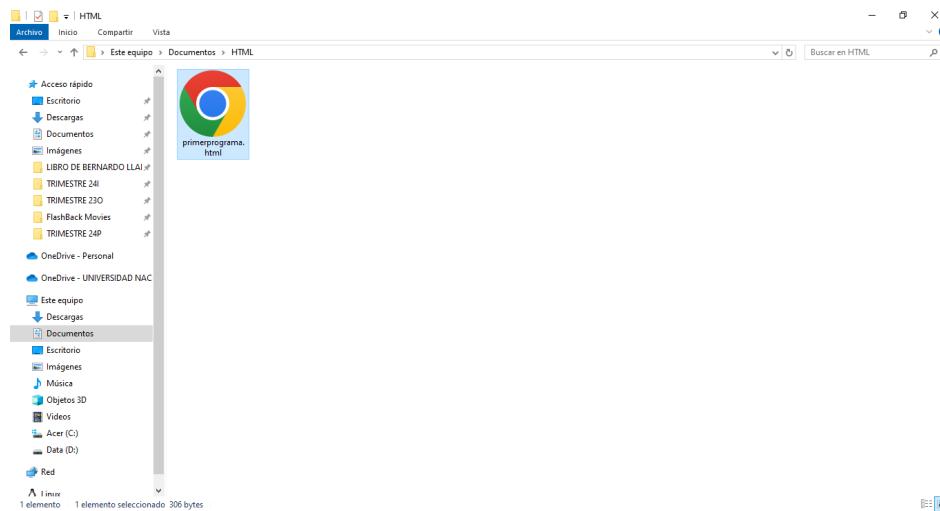
**NOTA:** Recuerda que debe de tener la extensión de .html ya que es para saber el tipo de archivo del lenguaje de HTML.

## EJECUCIÓN

**Revisa en la carpeta de archivos, si se guardó tu primer programa (ver figura X), si no te aparece, pues debes de hacer de nuevo los pasos anteriores.**



**Después dar doble clic para abrirlo**



**Y finalmente te debe de salir así en tu navegador web.**



**Figura: Visualización del programa “Hola Mundo” en HTML.**

## BIBLIOTECA DE FUNCIONES DEL HTML

## SENTENCIAS DE CONTROL DEL HTML

En HTML no se utilizan "sentencias de control" propiamente dichas, ya que HTML es un lenguaje de marcado, no un lenguaje de programación. Aunque HTML no tiene la capacidad de realizar operaciones lógicas o de control de flujo como en otros lenguajes de programación, sí ofrece elementos que permiten organizar y estructurar el contenido de una página web. Por ejemplo, aunque no se pueden crear condiciones directamente en HTML, se puede combinar con JavaScript para implementar estructuras

condicionales que alteren el contenido o el comportamiento de la página. Además, HTML incluye formularios que permiten interactuar con los usuarios y enviar datos, lo cual puede desencadenar ciertas acciones dependiendo de las entradas del usuario.

(IMAGEN)

## ENCUESTA

Responda las siguientes preguntas:

1.¿?

## LENGUAJE CSS (Cascading Style Sheets)

### CONCEPTO

CSS es un lenguaje de programación que sirve para definir cómo se debe mostrar un documento HTML. Esto incluye aspectos visuales como el diseño, los colores, las fuentes y la disposición de los elementos dentro de la página. La principal función de CSS es separar la estructura del contenido (HTML) del estilo visual.



Figura: Logo del lenguaje de CSS.

### CARACTERÍSTICAS DEL CSS

- Diseño y disposición: CSS se encarga de controlar el aspecto visual de una página web, ajustando detalles como colores, tipografía, márgenes, bordes, espaciado y la posición de los elementos en la página.
- Adaptabilidad: Gracias a las media queries, CSS permite que los diseños se ajusten automáticamente a diferentes tamaños de pantalla, lo que garantiza que las páginas se vean bien en dispositivos móviles, tabletas y computadoras de escritorio.
- Flexbox y Grid: Son técnicas avanzadas de CSS para crear diseños más dinámicos y complejos. Flexbox se utiliza para organizar elementos en una sola dirección (horizontal o vertical), mientras que Grid permite distribuir los elementos en una cuadrícula bidimensional (filas y columnas).
- Animaciones y transiciones: CSS también permite agregar efectos visuales como animaciones o transiciones suaves, como cambios en el color, tamaño o posición de los elementos cuando el usuario interactúa con ellos (por ejemplo, al pasar el cursor sobre un botón).

## ESTRUCTURA BÁSICA DE UN DOCUMENTO EN CSS

La estructura básica de un documento CSS se utiliza para definir los estilos visuales de los elementos HTML de una página web. CSS (Cascading Style Sheets) es un lenguaje de hojas de estilo que se emplea para modificar la apariencia de los elementos HTML, como colores, tamaños de texto, márgenes, bordes, etc.

```
/* Esto es un comentario en CSS */

/* Selección de un elemento HTML y definición de sus estilos */
selector {
    propiedad: valor;
    propiedad2: valor2;
}
```

## EXPLICACIÓN DE LA ESTRUCTURA BÁSICA DE UN DOCUMENTO EN CSS

### Comentarios:

- Los comentarios en CSS se escriben entre /\* y \*/. Sirven para agregar notas o descripciones dentro del archivo de estilo, pero no afectan la visualización de la página. Ejemplo: /\* Esto es un comentario \*/.

### Selector:

- El selector indica qué elementos HTML serán afectados por las reglas de estilo. Puede ser el nombre de una etiqueta (por ejemplo, p para párrafos), una clase (.clase), un id (#id), o cualquier otro selector más específico. Ejemplo: p (para seleccionar todos los elementos <p>).

### **Propiedades y valores:**

- **Propiedad:** Define qué aspecto del elemento se va a modificar. Por ejemplo, color, font-size, background-color, etc.
- **Valor:** Define cómo se aplicará la propiedad. Por ejemplo, si la propiedad es color, el valor puede ser red, #ff0000, etc. Ejemplo: color: red;

### **Reglas de estilo:**

- Una regla de estilo se define con un selector, seguido de una serie de propiedades dentro de llaves { }. Cada propiedad está separada por un punto y coma ;.

## **EJEMPLO BÁSICO DE CSS**

```
/* Esto es un comentario */  
h1 {  
    color: blue; /* Cambia el color del texto de los encabezados h1 a azul */  
    font-size: 30px; /* Define el tamaño de fuente como 30 píxeles */  
    text-align: center; /* Alinea el texto al centro */  
}  
  
p {  
    color: green; /* Cambia el color del texto de los párrafos a verde */  
    line-height: 1.5; /* Ajusta la altura de la línea para una mejor legibilidad */  
}
```

## **COMENTARIOS EN CSS**

Los comentarios en CSS no afectan al estilo de la página, pero sirven para explicar el código. Se escriben entre /\* y \*/.

```
/* Esto es un comentario */
```

**OBSERVACIÓN:** En CSS, no se utilizan "etiquetas" de la misma manera que en HTML. Sin embargo, en CSS, lo que comúnmente se utiliza son selectores y reglas de estilo para aplicar estilos a los elementos HTML de una página web. Las etiquetas HTML son los elementos que se seleccionan en el archivo CSS para aplicarles estilos.

## **AGREGACIÓN DE UN ARCHIVO CSS EN HTML**

Existen varias formas de aplicar CSS en un documento HTML:

1. **Archivo externo (recomendado):** Enlazas el archivo CSS a tu documento HTML dentro de la etiqueta <head>.

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

2. **Estilos en el documento HTML (dentro de la etiqueta <style>):** Puedes escribir el CSS directamente dentro del archivo HTML, dentro de la etiqueta <style>.

Ejemplo:

```
<style>
  h1 {
    color: blue;
  }
</style>
```

3. **Estilos en línea (no recomendado para mantener el código limpio):** Se aplican directamente a un elemento HTML usando el atributo style.

Ejemplo:

```
<h1 style="color: blue;">Encabezado azul</h1>
```

## IMPORTANCIA DEL CSS

CSS es fundamental para que una página web sea visualmente atractiva y fácil de navegar. Sin CSS, las páginas web tendrán un diseño muy simple y poco profesional, sin ningún estilo visual. Gracias a CSS, los sitios web pueden ser más funcionales y estéticamente agradables, adaptándose a diferentes dispositivos y proporcionando una experiencia de usuario coherente y agradable.

## VENTAJAS DEL CSS

Las ventajas del CSS son las siguientes:

Separar contenido y presentación:

- CSS permite separar la estructura del contenido (HTML) de su presentación (estilo). Esto mejora la organización del código y facilita el mantenimiento de las páginas web. Los cambios de diseño se pueden hacer fácilmente sin modificar el contenido HTML.

Reducción del tamaño del archivo HTML:

- Al utilizar CSS para los estilos, el código HTML se mantiene más limpio y liviano, ya que solo se necesita agregar clases y estilos sin la necesidad de escribir estilos dentro de cada etiqueta HTML.

Diseño consistente en todo el sitio:

- Con CSS, puedes aplicar el mismo diseño a múltiples páginas de un sitio web mediante hojas de estilo externas. Esto asegura una apariencia coherente en todo el sitio y facilita las actualizaciones de diseño globales.

Diseño responsivo:

- CSS permite crear diseños adaptativos que responden a diferentes tamaños de pantalla y dispositivos (como móviles, tabletas y computadoras de escritorio). Esto

es posible gracias a técnicas como las media queries, que ajustan el diseño según las características del dispositivo.

Mejora del rendimiento:

- Al separar el contenido y el diseño, los navegadores pueden almacenar en caché las hojas de estilo, lo que acelera la carga de las páginas y mejora la eficiencia del sitio web. Esto es especialmente útil para sitios grandes o que tienen múltiples páginas con el mismo diseño.

Mayor control sobre la presentación visual:

- CSS ofrece un control muy preciso sobre la presentación de los elementos en la página, permitiendo definir cosas como colores, fuentes, márgenes, bordes, alineación, posicionamiento, animaciones, y mucho más.

Compatibilidad con animaciones y transiciones:

- CSS incluye soporte para animaciones y transiciones, lo que permite agregar efectos visuales sin necesidad de usar JavaScript. Esto mejora la experiencia del usuario al hacer que la interfaz sea más dinámica y atractiva.



## DESVENTAJAS DEL CSS

Las desventajas del CSS son las siguientes:

Curva de aprendizaje:

- Aunque CSS es relativamente fácil de aprender, dominarlo puede ser un desafío debido a las complejidades involucradas en la creación de diseños responsivos, posicionamiento preciso y compatibilidad entre navegadores.

Limitaciones en funcionalidad interactiva:

- CSS no puede proporcionar interactividad avanzada, como la manipulación de datos o la lógica de programación. Para tareas complejas como validación de formularios o interacciones avanzadas, es necesario usar JavaScript junto con CSS.

Problemas de compatibilidad entre navegadores:

- Aunque la mayoría de los navegadores modernos soportan CSS, hay algunas variaciones en cómo se interpretan ciertas propiedades, lo que puede generar inconsistencias en el diseño visual. A veces es necesario aplicar soluciones específicas para cada navegador (por ejemplo, prefijos específicos de navegador).

Difícil de depurar en diseños complejos:

- En proyectos grandes y complejos con muchas hojas de estilo, puede ser difícil localizar y corregir problemas de diseño, especialmente cuando hay reglas CSS que se aplican en cascada y afectan a múltiples elementos.

Limitaciones en el diseño:

- Aunque CSS ofrece muchas posibilidades de diseño, algunos diseños avanzados o específicos pueden ser difíciles de lograr solo con CSS, especialmente cuando se requiere un control muy detallado del comportamiento de los elementos en la página. En tales casos, el uso de JavaScript es inevitable.

Dependencia del HTML para la estructura:

- CSS solo es útil si se utiliza en conjunto con HTML. No tiene sentido usar CSS sin contenido HTML, ya que es un lenguaje de estilo para estructurar y presentar ese contenido. Por lo tanto, CSS no es útil de forma independiente.

Problemas con el mantenimiento en sitios grandes:

- A medida que los proyectos web crecen, la cantidad de CSS también puede aumentar. Sin una estructura adecuada y una gestión cuidadosa de las hojas de estilo, el código puede volverse desordenado y difícil de mantener.



## CREACIÓN DE UN PROGRAMA EN CSS

En este apartado, vamos a comenzar a usar el lenguaje de CSS, pero, primero vamos a crear un programa en CSS usando Visual Studio Code.

## REALIZACIÓN DE UN PROGRAMA EN CSS

En esta sección, se van a exponer los pasos a seguir en la realización de un programa, por medio de un ejemplo. Para poder escribir programas se necesita un editor de código fuente, esto es: Visual Studio Code (Este se encuentra en el apéndice X)

## Creación de tu primer programa

Supongamos que queremos crear una página web sencilla con un encabezado, un párrafo de texto y un botón. A continuación, veremos cómo se puede crear un archivo HTML y un archivo CSS para darle estilo a esa página.

En la carpeta de tu proyecto, haz clic derecho y selecciona Nuevo archivo.

Nómbrase: index.html.

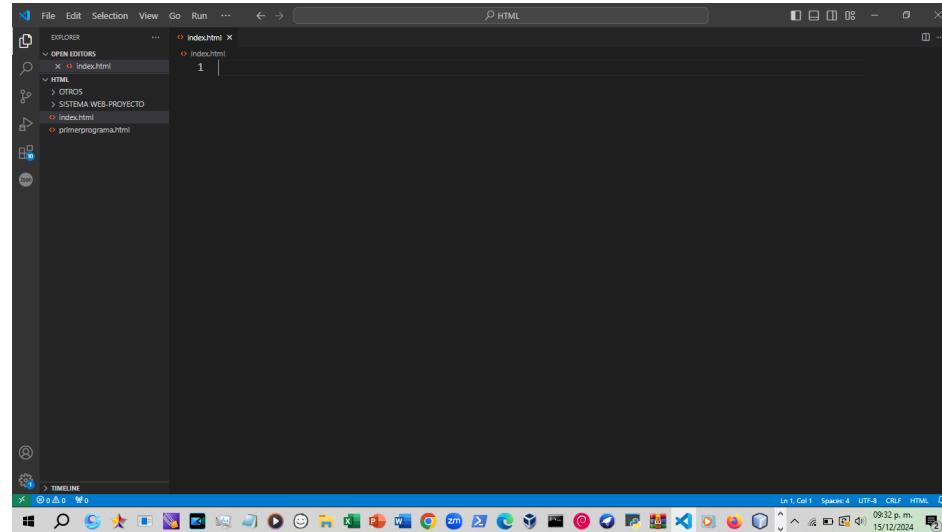


Figura: Creación del archivo index.html.

### Archivo HTML (index.html)

Escribe el siguiente código base para tu archivo HTML:

A screenshot of the Visual Studio Code interface showing the contents of the 'index.html' file. The code is as follows:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Mi página web</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<h1>¡Bienvenidos a mi página web!</h1>
<p>Este es un párrafo de ejemplo.</p>
</body>
</html>
```

The status bar at the bottom shows the date and time: '15/12/2024' and '09:35 p.m.'

Figura x: Código del archivo index.html.

### Archivo CSS (styles.css)

A continuación, creamos el archivo CSS para que este funcione bien y se pueda aplicar a una página HTML.

En la misma carpeta del proyecto, crea otro archivo llamado styles.css.

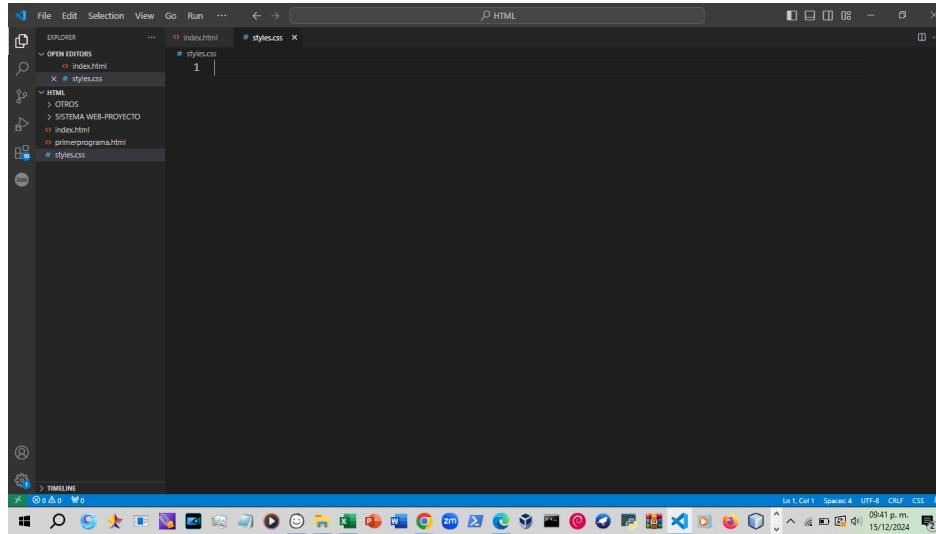


Figura: Creación del archivo styles.css.

Dentro del documento que acabas de crear, escribe las reglas de estilo que deseas aplicar a tu página. Escribe el siguiente código base para tu archivo CSS:

A screenshot of a code editor window titled "HTML". In the left sidebar, under "OPEN EDITORS", there is a file named "index.html" and another file named "styles.css". The "styles.css" file is currently selected and open in the main editor area. The code contains basic styling rules for the body, h1, and p elements. The code is as follows:

```
/* Estilos básicos */
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    color: #333;
}

h1 {
    color: #0077cc;
    text-align: center;
}

p {
    font-size: 18px;
    line-height: 1.5;
    text-align: justify;
    margin: 20px;
}
```

At the bottom of the screen, the Windows taskbar is visible with various application icons.

Figura x: Código del archivo styles.css.

## CONEXIÓN DEL ARCHIVO CSS CON EL ARCHIVO HTML

Puedes revisar que el archivo styles.css esté correctamente enlazado dentro de la etiqueta <head> del archivo index.html, con la siguiente instrucción, esta es:

```
<link rel="stylesheet" href="styles.css">
```

También, debes asegurarte que los archivos que creaste antes deben de estar en la misma carpeta y así se puedan ejecutar sin ningún problema, como se muestra en la siguiente figura, esta es:

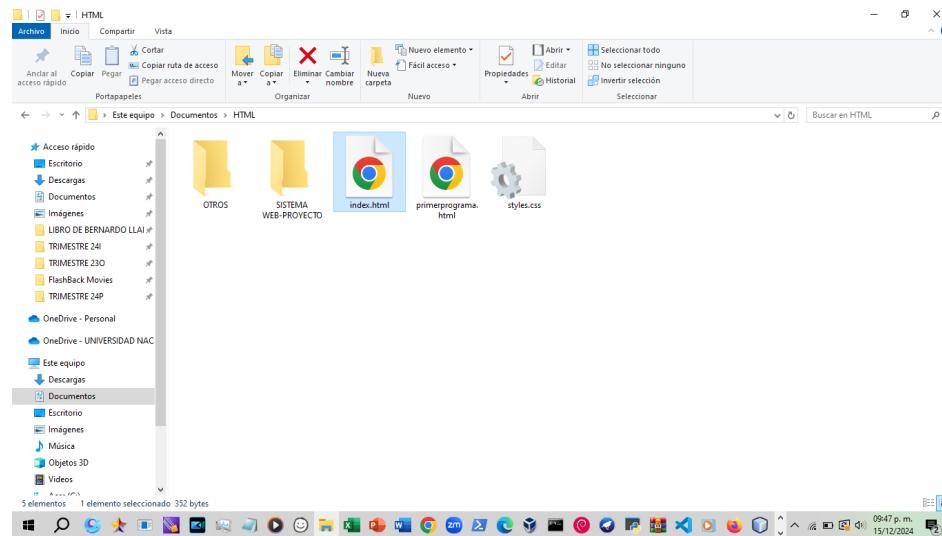


Figura: Revisión de los 2 archivos creados anteriormente dentro de la carpeta.

## REVISIÓN DE LOS CAMBIOS EN EL NAVEGADOR

Una vez que hayas guardado ambos archivos, abre el archivo index.html en tu navegador para que así puedas ver los estilos aplicados a la página.

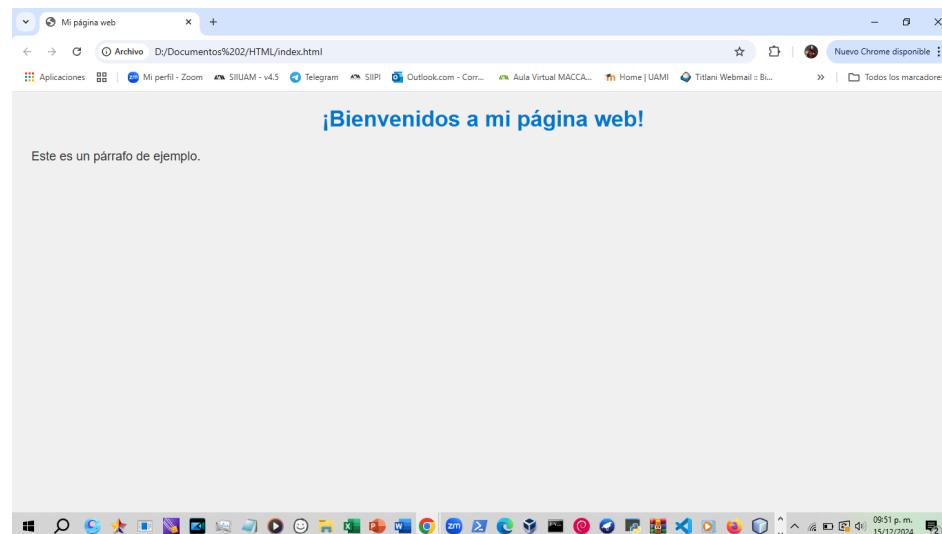


Figura: Visualización del ejemplo de la página web.

Con estos pasos deberías ser capaz de crear un proyecto sencillo en CSS usando Visual Studio Code.

## COLORES EN CSS

Se mencionan algunos ejemplos de los colores comunes en css, esto es con sus valores en formato hexadecimal y su equivalente en RGB, lo podemos ver mejor en la siguiente tabla:

Color	Hexadecimal	RGB (Red, Green, Blue)
Rojo	#FF0000	rgb(255, 0, 0)
Verde	#00FF00	rgb(0, 255, 0)
Azul	#0000FF	rgb(0, 0, 255)
Amarillo	#FFFF00	rgb(255, 255, 0)
Cian (Aqua)	#00FFFF	rgb(0, 255, 255)
Magenta	#FF00FF	rgb(255, 0, 255)
Blanco	#FFFFFF	rgb(255, 255, 255)
Negro	#000000	rgb(0, 0, 0)
Gris Claro	#D3D3D3	rgb(211, 211, 211)
Gris Oscuro	#A9A9A9	rgb(169, 169, 169)
Naranja	#FFA500	rgb(255, 165, 0)
Púrpura	#800080	rgb(128, 0, 128)
Verde Lima	#00FF7F	rgb(0, 255, 127)
Azul Marino	#000080	rgb(0, 0, 128)
Rosa	#FFC0CB	rgb(255, 192, 203)
Marrón	#A52A2A	rgb(165, 42, 42)
Beige	#F5F5DC	rgb(245, 245, 220)

Tabla: Valores de los colores comunes en css con su respectivo formato hexadecimal y RGB.

¿Cómo se puede leer los valores de los colores en css?

- **Hexadecimal:** Este formato está compuesto por 6 caracteres (2 caracteres para el rojo, 2 caracteres para el verde, y 2 caracteres para el azul). Y vemos que cada par de caracteres puede variar entre los 00 y FF (esto es en base 16 porque es hexadecimal).
- **RGB:** Este formato está representado por los 3 colores primarios: el rojo, verde y azul. Lo que vemos es que cada valor de un color puede variar entre los números de 0 y 255 (esto sería como un rango).

Se puede ver mejor con un ejemplo y esto es:

- #FF0000 en hexadecimal es rojo puro, que en RGB es (255, 0, 0).
- #00FF00 en hexadecimal es verde puro, que en RGB es (0, 255, 0).

## BIBLIOTECA DE FUNCIONES DEL CSS

## SENTENCIAS DE CONTROL DEL CSS

CSS no incluye "sentencias de control" como las estructuras de decisión (if, else) o de repetición (for, while), ya que su propósito no es realizar lógica de programación. Sin embargo, CSS dispone de herramientas que permiten modificar y ajustar la apariencia del contenido de forma dinámica o condicional. Por ejemplo, mediante propiedades como las media queries se pueden aplicar estilos diferentes según el tamaño de la pantalla, lo que permite adaptar el diseño de una página a distintos dispositivos, sin necesidad de lógica de programación.

(IMAGEN)

## TRANSICIONES Y ANIMACIONES DEL CSS

## ENCUESTA

**Responda las siguientes preguntas:**

**1.¿?**

# LENGUAJE JAVASCRIPT- JS (JavaScript)

## CONCEPTO

JavaScript es un lenguaje de programación diseñado para agregar interactividad y funciones dinámicas a las páginas web. Es un lenguaje de alto nivel, orientado a objetos y basado en eventos, que se ejecuta directamente en el navegador del usuario.



Figura: Logo del lenguaje de Javascript.

## CARACTERÍSTICAS DEL JS

- Interactividad: JavaScript permite que las páginas web sean interactivas, reaccionando a eventos como clics, desplazamientos del ratón y entradas de texto, lo que posibilita la creación de aplicaciones web dinámicas y con respuesta instantánea.
- Manipulación del DOM: Este lenguaje puede interactuar con el DOM (Modelo de Objetos del Documento), que es la estructura interna de una página HTML. Gracias a esto, JavaScript puede actualizar y modificar el contenido y los elementos de la página sin necesidad de recargarla, proporcionando una experiencia de usuario más fluida.
- Programación asíncrona: JavaScript soporta la programación asíncrona mediante callbacks, promesas y async/await, lo que facilita la ejecución de tareas en segundo plano (como hacer solicitudes a un servidor) sin interrumpir el flujo principal de la página web.
- Acceso a API's: JavaScript puede interactuar con diversas API's (Interfaces de Programación de Aplicaciones) que le permiten realizar tareas complejas, como obtener la ubicación del usuario, gestionar almacenamiento local o comunicarse con servicios web externos.

## ESTRUCTURA BÁSICA DE UN DOCUMENTO EN JS

La estructura básica de un documento en JavaScript (JS) es bastante simple, y su principal función es agregar interactividad y dinámica a una página web. JavaScript se utiliza para manipular los elementos de una página HTML, responder a eventos de los usuarios, y realizar tareas más complejas como validación de formularios o actualización dinámica de contenido.

### Estructura Básica de un Documento en JavaScript:

Un documento JavaScript se puede incluir dentro de una página HTML de varias formas. A continuación te explico cómo se organiza la estructura básica:

#### 1. Incluir JavaScript dentro de una página HTML:

Puedes agregar JavaScript directamente dentro de una página HTML utilizando la etiqueta `<script>`. El código JavaScript puede ir dentro de las etiquetas `<script>` dentro del documento HTML, ya sea en el encabezado (`<head>`) o en el cuerpo (`<body>`).

#### Ejemplo básico de cómo se estructura un documento JavaScript dentro de HTML:

##### Archivo HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Documento en JavaScript</title>
</head>
<body>

    <h1>Mi página con JavaScript</h1>
    <button id="miBoton">Haz clic aquí</button>

    <!-- Script JavaScript -->
    <script>
        // Esto es un comentario en JavaScript

        // Función que se ejecuta cuando se hace clic en el botón
        function saludar() {
            alert("¡Hola, bienvenido a mi página!");
        }

        // Agregar un evento al botón
        document.getElementById("miBoton").addEventListener("click", saludar);
    </script>

</body>
</html>
```

## EXPLICACIÓN DE LA ESTRUCTURA BÁSICA DE UN DOCUMENTO EN JS

### 1. Etiquetas HTML:

- `<html>`, `<head>`, `<body>`: Son las etiquetas HTML normales que estructuran la página.
- `<h1>`: Encabezado principal de la página.
- `<button>`: Un botón HTML, que será el que el usuario haga clic.

### 2. Elemento `<script>`:

- El código JavaScript está incluido dentro de la etiqueta <script>, que generalmente se coloca antes del cierre de la etiqueta </body>. Esto es para asegurarse de que todos los elementos HTML (como el botón) estén cargados antes de que se ejecute el script.

### 3. Contenido del <script>:

- Comentarios: Los comentarios en JavaScript comienzan con // para comentarios de una sola línea o con /\* y \*/ para comentarios multilínea.
  - Función saludar(): Esta función se ejecuta cuando el usuario hace clic en el botón.
  - Método addEventListener(): Este método se utiliza para escuchar un evento (en este caso, el clic) en un elemento HTML y ejecutar una función cuando ese evento ocurre.

## 2. Archivo Externo de JavaScript:

En lugar de colocar el código JavaScript directamente en el documento HTML, también se puede escribir el código en un archivo externo y luego hacer referencia a ese archivo en el HTML.

### Ejemplo de estructura usando un archivo JavaScript externo:

## Archivo HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Documento en JavaScript</title>
</head>
<body>
  <h1>Mi página con JavaScript</h1>
  <button id="miBoton">Haz clic aquí</button>
  <!-- Referencia al archivo JavaScript externo -->
  <script src="script2.js"></script>
</body>
</html>
```

Así debe de verse como en la figura:

Figura: Código del archivo index2.html

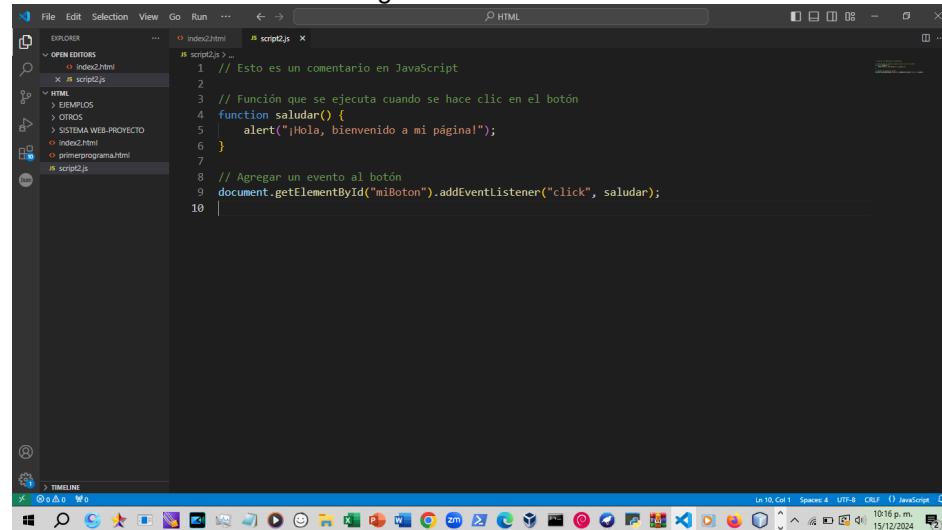
### Archivo JavaScript (script.js):

```
// Esto es un comentario en JavaScript

// Función que se ejecuta cuando se hace clic en el botón
function saludar() {
    alert("¡Hola, bienvenido a mi página!");
}

// Agregar un evento al botón
document.getElementById("miBoton").addEventListener("click", saludar);
```

Así debe de verse como en la figura:



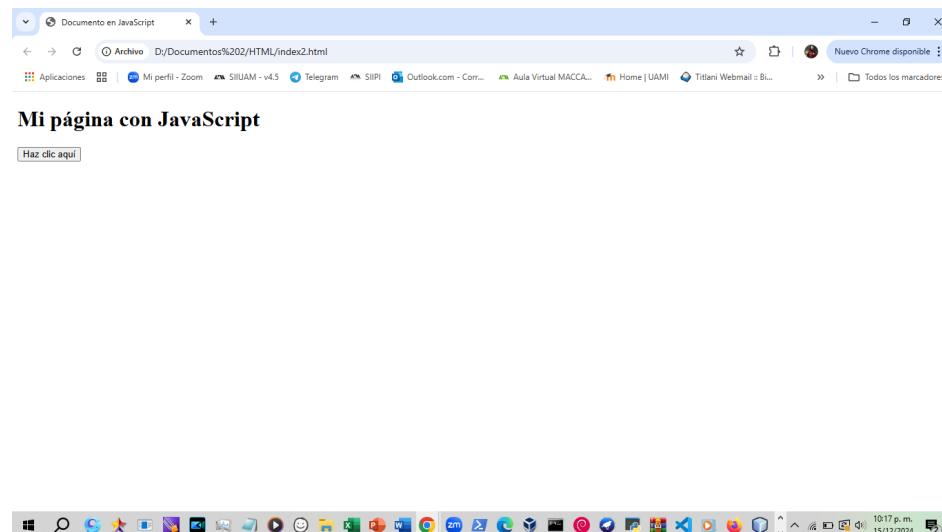
The screenshot shows a code editor window with the following details:

- File Path:** index2.html > script2.js
- Code Content:**

```
1 // Esto es un comentario en JavaScript
2
3 // Función que se ejecuta cuando se hace clic en el botón
4 function saludar() {
5     alert("¡Hola, bienvenido a mi página!");
6 }
7
8 // Agregar un evento al botón
9 document.getElementById("miBoton").addEventListener("click", saludar);
```
- Environment:** The interface includes a sidebar labeled "OPEN EDITORS" with "index2.html" and "script2.js" listed, and a bottom status bar showing "Line 10, Col 1" and "10:16 p.m. 15/12/2024".

Figura: Código del archivo script2.js.

Al momento de ejecutarlo, deberá verse así:



Y una vez que le des clic al botón, te debe aparecer esto:

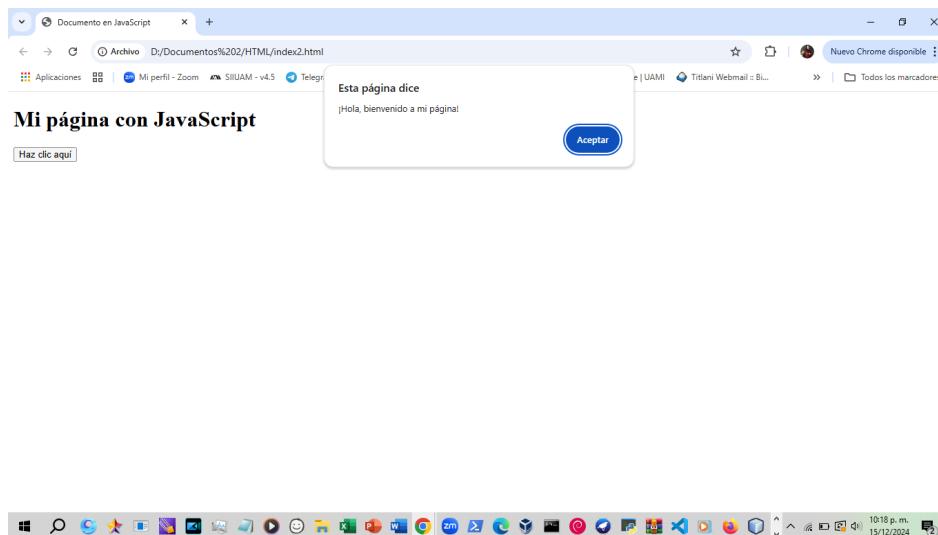


Figura: Visualización del archivo index2.html.

Te debe aparecer la nueva ventana emergente, la cual es:

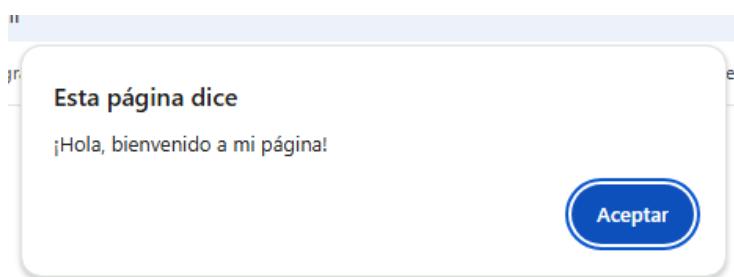


Figura: Visualización de la ventana emergente.

#### Explicación del ejemplo con archivo externo:

- Archivo HTML: El archivo HTML solo contiene la estructura de la página y una referencia al archivo JavaScript mediante la etiqueta `<script src="script.js"></script>`. Esta referencia le indica al navegador que cargue y ejecute el código JavaScript contenido en el archivo script.js.
- Archivo JS externo: El código JavaScript en script.js es similar al código que se incluye dentro del HTML, pero está separado en un archivo independiente. De esta manera, el código es más limpio y fácil de mantener.

El uso de JavaScript en una página web permite agregar interacción, animaciones, validación de formularios, y mucho más, haciendo que la experiencia del usuario sea dinámica y atractiva.

## COMENTARIOS EN JS

## **IMPORTANCIA DEL JS**

JavaScript es esencial para crear aplicaciones web interactivas y dinámicas, ya que permite transformar páginas estáticas en experiencias ricas y atractivas. Sin JavaScript, la web sería mucho menos interactiva y visualmente estática, ya que muchas de las funciones avanzadas y efectos visuales dependen de este lenguaje.

## **VENTAJAS DEL JS**

Las ventajas del JS son las siguientes:

Interactividad en tiempo real:

- JavaScript permite crear interacciones dinámicas en una página web sin tener que recargarla. Puedes responder a eventos de los usuarios, como clics, desplazamientos o teclas presionadas, de manera instantánea.

Compatibilidad en todos los navegadores:

- JavaScript es compatible con todos los navegadores modernos (Chrome, Firefox, Safari, Edge, etc.), lo que permite que las aplicaciones web sean accesibles desde casi cualquier dispositivo.

Funciona del lado del cliente:

- JavaScript se ejecuta directamente en el navegador del usuario, lo que significa que no es necesario hacer múltiples solicitudes al servidor para procesar datos o realizar acciones simples. Esto reduce la carga en el servidor y mejora la velocidad de respuesta de la página.

Versatilidad:

- JavaScript no solo se usa para agregar interactividad en el frontend (el lado del cliente), sino que también es utilizado en el backend (servidor) a través de entornos como Node.js. Esto permite que JavaScript se utilice en toda la pila de desarrollo (frontend y backend).

Gran comunidad y ecosistema:

- JavaScript tiene una enorme comunidad de desarrolladores y un vasto ecosistema de bibliotecas y frameworks (como React, Angular, Vue, Node.js, entre otros). Esto facilita la creación de aplicaciones web avanzadas y acelera el desarrollo.

Asincronía y manejo de eventos:

- JavaScript es excelente para manejar tareas asincrónicas, como las solicitudes a servidores o la ejecución de múltiples acciones simultáneamente sin bloquear la interfaz de usuario. Los mecanismos como callbacks, promesas y async/await permiten trabajar de manera eficiente con operaciones no bloqueantes.

Fácil de aprender y utilizar:

- JavaScript tiene una sintaxis relativamente simple y es un lenguaje de alto nivel. Además, su integración directa con HTML y CSS facilita el aprendizaje para quienes ya están familiarizados con estas tecnologías.

Desarrollo en tiempo real:

- Gracias a herramientas como los "dev tools" (herramientas de desarrollo) en los navegadores, puedes depurar y probar tu código JavaScript en tiempo real, lo que facilita la corrección de errores y la mejora del rendimiento.



## DESVENTAJAS DEL JS

Las desventajas del JS son las siguientes:

Rendimiento en comparación con otros lenguajes:

- Aunque JavaScript ha mejorado mucho con el tiempo, todavía puede ser más lento que lenguajes compilados como C o Java para ciertas tareas, especialmente en operaciones matemáticas complejas o cálculos de alto rendimiento.

Seguridad:

- JavaScript puede ser una puerta de entrada para ataques maliciosos, como cross-site scripting (XSS), si no se valida adecuadamente el contenido proporcionado por los usuarios. Las vulnerabilidades de seguridad pueden surgir si no se implementan buenas prácticas para proteger las aplicaciones.

Dependencia del navegador y diferencias entre navegadores:

- Aunque JavaScript es compatible con todos los navegadores, pueden existir diferencias en la forma en que cada navegador interpreta o ejecuta el código JavaScript. Esto puede llevar a problemas de compatibilidad y requerir soluciones específicas para cada navegador (por ejemplo, prefijos de navegador).

No es adecuado para aplicaciones complejas de bajo nivel:

- JavaScript no es ideal para aplicaciones de bajo nivel que requieren control directo sobre el hardware o la memoria, como aplicaciones de sistemas operativos

o software embebido. En esos casos, lenguajes como C o Rust son más adecuados.

El código puede ser difícil de mantener en proyectos grandes:

- En proyectos grandes, el código JavaScript puede volverse desordenado si no se organiza correctamente. A medida que el tamaño del proyecto crece, puede ser difícil de depurar y mantener si no se sigue una estructura coherente y no se utilizan herramientas adecuadas como frameworks y patrones de diseño.

Problemas con el manejo de errores:

- Aunque JavaScript tiene mecanismos para manejar errores, como try-catch, el manejo de excepciones puede ser más difícil y menos intuitivo en comparación con otros lenguajes. Además, en aplicaciones complejas, la gestión de errores y las promesas pueden complicar el flujo de trabajo.

Inestabilidad debido a actualizaciones frecuentes:

- JavaScript, como parte de su evolución constante, recibe actualizaciones y nuevas características con regularidad. Aunque esto puede ser positivo, también puede generar inestabilidad en el código si no se mantienen las mejores prácticas o si se usan versiones anteriores de los navegadores que no soportan ciertas características nuevas.

Dependencia de bibliotecas y frameworks:

- Para proyectos más grandes, los desarrolladores a menudo recurren a bibliotecas y frameworks como React, Vue o Angular. Si bien estos hacen el desarrollo más rápido y estructurado, también generan una dependencia de herramientas de terceros, lo que puede aumentar la complejidad y el tamaño del proyecto.



## CREACIÓN DE UN PROGRAMA EN JS

En este caso, puedes revisar el apartado de ["ESTRUCTURA BÁSICA DE UN DOCUMENTO EN JS"](#) el cual contiene un ejemplo de un programa de JS dentro de

una página en HTML.

### **TABLA COMPARATIVA DE LAS VENTAJAS Y DESVENTAJAS DESCRIPTAS ANTERIORMENTE DE LOS LENGUAJES DE HTML, CSS Y JS**

De igual manera, se muestra una tabla comparativa de las ventajas y desventajas de los lenguajes HTML, CSS Y JS que hemos mencionado anteriormente.

Lenguajes	Ventajas	Desventajas
HTML	<ul style="list-style-type: none"><li>• Fácil de aprender y utilizar.</li><li>• Estandarizado y compatible con todos los navegadores.</li><li>• Perfecto para estructurar contenido web.</li><li>• Soporta otras tecnologías como CSS y JavaScript.</li><li>• Gratuito y de código abierto.</li></ul>	<ul style="list-style-type: none"><li>• Limitado en diseño y funcionalidad avanzada.</li><li>• No soporta interactividad compleja por sí solo.</li><li>• No adecuado para aplicaciones web completas.</li><li>• Necesita otras tecnologías (CSS y JavaScript) para obtener un diseño atractivo e interacción avanzada.</li><li>• Carece de capacidades de programación lógica.</li></ul>
CSS	<ul style="list-style-type: none"><li>• Separación de contenido y presentación, facilitando mantenimiento y actualización.</li><li>• Reducción del tamaño del archivo HTML, haciendo la página más eficiente.</li></ul>	<ul style="list-style-type: none"><li>• Curva de aprendizaje: Puede ser complicado dominar todas las características y técnicas de CSS.</li><li>• Limitado en interactividad: CSS no maneja la lógica o la manipulación de datos, lo que</li></ul>

	<ul style="list-style-type: none"> <li>• Diseño consistente en todo el sitio, gracias a hojas de estilo externas.</li> <li>• Diseño responsive que se adapta a diferentes dispositivos y tamaños de pantalla.</li> <li>• Mejora del rendimiento con almacenamiento en caché de las hojas de estilo.</li> <li>• Control preciso sobre la apariencia de la página (colores, fuentes, posicionamiento, etc.).</li> <li>• Compatibilidad con animaciones y transiciones para hacer la página más dinámica.</li> </ul>	<ul style="list-style-type: none"> <li>• requiere el uso de JavaScript.</li> <li>• Problemas de compatibilidad entre navegadores pueden generar diferencias visuales.</li> <li>• Difícil de depurar en sitios grandes o con diseños complejos.</li> <li>• Limitaciones en diseño avanzado: Algunos diseños complejos requieren JavaScript o frameworks adicionales.</li> <li>• Dependencia de HTML para que CSS tenga sentido y sea útil.</li> <li>• Problemas con el mantenimiento en proyectos grandes si no se organiza adecuadamente.</li> </ul>
JS	<ul style="list-style-type: none"> <li>• Permite interactividad en tiempo real sin recargar la página.</li> <li>• Compatible con todos los navegadores.</li> <li>• Se ejecuta del lado del cliente, reduciendo la carga en el servidor.</li> <li>• Muy versátil, con uso en frontend y backend (Node.js).</li> </ul>	<ul style="list-style-type: none"> <li>• Menor rendimiento en comparación con lenguajes compilados para operaciones de alto rendimiento.</li> <li>• Vulnerabilidades de seguridad, como XSS.</li> <li>• Diferencias de interpretación entre navegadores.</li> <li>• No ideal para aplicaciones de bajo nivel o</li> </ul>

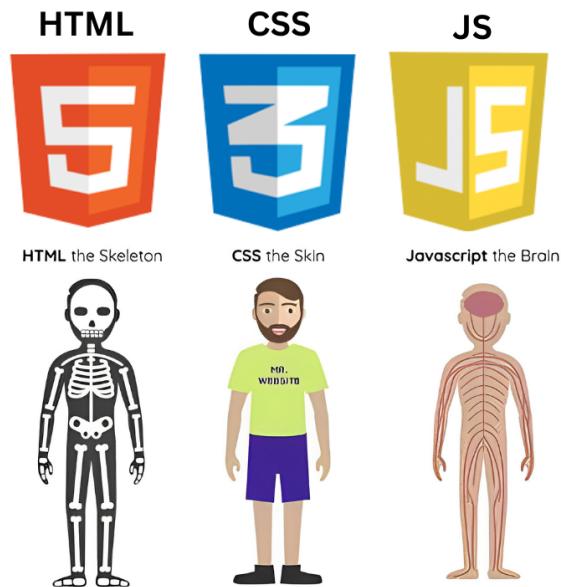
	<ul style="list-style-type: none"> <li>• Gran comunidad y ecosistema de bibliotecas y frameworks.</li> <li>• Manejo eficiente de operaciones asíncronas.</li> <li>• Sintaxis simple y fácil de aprender.</li> <li>• Herramientas de desarrollo para depuración en tiempo real.</li> </ul>	<ul style="list-style-type: none"> <li>• software embebido.</li> <li>• Código difícil de mantener en proyectos grandes sin una estructura adecuada.</li> <li>• Manejo de errores más complejo en comparación con otros lenguajes.</li> <li>• Inestabilidad por actualizaciones frecuentes.</li> <li>• Dependencia de bibliotecas y frameworks que pueden añadir complejidad.</li> </ul>
--	---	---

Figura: Tabla comparativa de las ventajas y desventajas de los lenguajes de HTML, CSS Y JS.

## Resumen

- **HTML: Estructura y contenido.**
- **CSS: Estilo y presentación.**
- **JavaScript: Interactividad y lógica.**

En la siguiente figura nos mostrará la funcionalidad de los distintos lenguajes que se mencionan.



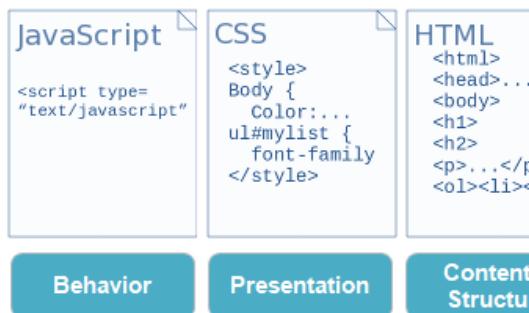
**Figura x. Los lenguajes de programación HTML, CSS Y JS.**

Podemos ver que en la figura lo siguientes aspectos:

HTML sirve para definir la estructura y contenido de una página web. (Esqueleto)

CSS sirve para definir los estilos, presentaciones o apariencias de una página web. (Piel)

Javascript (JS) sirve para definir la interactividad y la lógica de una página web. (Cerebro)



**Figura: Comportamientos de los lenguajes de programación HTML, CSS Y JS.**

Estos 3 lenguajes trabajan juntos para poder crear y elaborar páginas web.



**Figura:** Diferentes lenguajes de programación HTML, CSS Y JS.

## BIBLIOTECA DE FUNCIONES DEL JS

### SENTENCIAS DE CONTROL DEL JS

En JavaScript, las sentencias de control son estructuras que permiten modificar el flujo de ejecución del programa, tomando decisiones o repitiendo ciertas acciones basadas en condiciones específicas. Son fundamentales para hacer que los programas sean más dinámicos y puedan adaptarse a diferentes situaciones. A continuación, te explico las principales sentencias de control en JavaScript:

#### 1. Sentencias Condicionales

Las sentencias condicionales permiten ejecutar diferentes bloques de código dependiendo de si se cumple o no una condición.

**if**

La sentencia if ejecuta un bloque de código sólo si se cumple una condición específica.

#### **javascript - código**

```
let edad = 18;  
  
if (edad >= 18) {  
    console.log("Eres mayor de edad");  
}
```

Explicación: Si la condición edad  $\geq 18$  es verdadera, se ejecuta el bloque dentro del if.

### **else**

El else se utiliza junto con if para ejecutar un bloque de código cuando la condición no se cumple.

#### **javascript - código**

```
let edad = 16;  
  
if (edad >= 18) {  
    console.log("Eres mayor de edad");  
} else {  
    console.log("Eres menor de edad");  
}
```

Explicación: Si la condición edad  $\geq 18$  no se cumple, se ejecuta el bloque dentro de else.

### **else if**

Este comando se usa para verificar múltiples condiciones. Si la primera condición no se cumple, evalúa la siguiente.

#### **javascript - código**

```
let edad = 20;  
  
if (edad < 18) {  
    console.log("Eres menor de edad");  
} else if (edad >= 18 && edad <= 30) {  
    console.log("Eres un adulto joven");  
} else {  
    console.log("Eres un adulto");  
}
```

Explicación: El else if permite manejar varias condiciones, y se ejecuta el bloque que coincida con la condición cumplida.

### **2. Operador ternario (? :)**

El operador ternario es una forma compacta de escribir una sentencia if-else.

#### **javascript - código**

```
let edad = 21;  
let mensaje = (edad >= 18) ? "Eres mayor de edad" : "Eres menor de edad";  
console.log(mensaje);
```

Explicación: Si la condición edad  $\geq 18$  es verdadera, se asigna "Eres mayor de edad"; de lo contrario, se asigna "Eres menor de edad".

### 3. Sentencias de Bucles (Loops)

Los bucles permiten repetir un bloque de código mientras se cumpla una condición o un número determinado de veces.

#### **for**

El bucle for se usa cuando sabes cuántas veces deseas ejecutar un bloque de código.

#### **javascript - código**

```
for (let i = 0; i < 5; i++) {  
    console.log(i);  
}
```

Explicación: El bucle for repite el bloque de código desde  $i = 0$  hasta  $i < 5$ , incrementando  $i$  en cada iteración.

#### **while**

El bucle while ejecuta un bloque de código mientras se cumpla una condición.

#### **javascript - código**

```
let i = 0;  
while (i < 5) {  
    console.log(i);  
    i++;  
}
```

Explicación: El bloque de código se ejecuta mientras  $i < 5$ .

#### **do...while**

Este bucle es similar al while, pero se ejecuta al menos una vez, incluso si la condición es falsa.

#### **javascript - código**

```
let i = 0;  
do {  
    console.log(i);  
    i++;  
} while (i < 5);
```

Explicación: El bloque dentro del do se ejecuta primero y luego evalúa la condición  $i < 5$ .

### 4. Sentencia break

La sentencia break se utiliza para salir de un bucle o de una estructura de control antes de que termine su ciclo.

#### **javascript - código**

```
for (let i = 0; i < 10; i++) {  
    if (i === 5) {  
        break; // Salir del bucle cuando i es igual a 5  
    }  
    console.log(i);  
}
```

Explicación: El bucle se detiene cuando i es igual a 5 debido al break.

## 5. Sentencia continue

La sentencia continue salta la iteración actual de un bucle y continúa con la siguiente.

### javascript - código

```
for (let i = 0; i < 5; i++) {  
    if (i === 3) {  
        continue; // Salta la iteración cuando i es igual a 3  
    }  
    console.log(i);  
}
```

Explicación: Cuando i es igual a 3, el bucle omite esa iteración y pasa a la siguiente.

## 6. Sentencia switch

La sentencia switch permite realizar varias verificaciones de manera más ordenada cuando hay muchas condiciones posibles.

### javascript - código

```
let dia = 3;  
switch (dia) {  
    case 1:  
        console.log("Lunes");  
        break;  
    case 2:  
        console.log("Martes");  
        break;  
    case 3:  
        console.log("Miércoles");  
        break;  
    default:  
        console.log("Día no válido");  
}
```

Explicación: El valor del día se compara con los casos. Si el día es igual a 3, se ejecuta el bloque correspondiente e imprime "Miércoles". Si no coincide con ninguno, se ejecuta el bloque default.

## 7. Sentencia throw

La sentencia throw permite generar un error personalizado en el código.

### **javascript - código**

```
function dividir(a, b) {  
    if (b === 0) {  
        throw "No se puede dividir por cero"; // Lanza un error  
    }  
    return a / b;  
}  
  
try {  
    console.log(dividir(10, 0));  
} catch (error) {  
    console.log(error); // Captura y muestra el error  
}
```

Explicación: Si b es igual a 0, se lanza un error con el mensaje "No se puede dividir por cero".

## **ENCUESTA**

**Responda las siguientes preguntas:**

**1.¿?**

**UNIÓN DE LOS LENGUAJES DE PROGRAMACIÓN HTML, CSS Y JS.**

#### 4. Conclusiones

#### 5. Referencias

<https://es.wikipedia.org/wiki/Inform%C3%A1tica>

file:///D:/Descargas%202/Gran\_Libro\_HTML5\_CSS3\_Javascript.pdf

<https://ceper.uniandes.edu.co/files/2022/10/MANUAL-HTML-Y-CSS.pdf>

<https://www.uv.es/jbosch/PDF/Curso%20de%20HTML.pdf>

#### OTROS

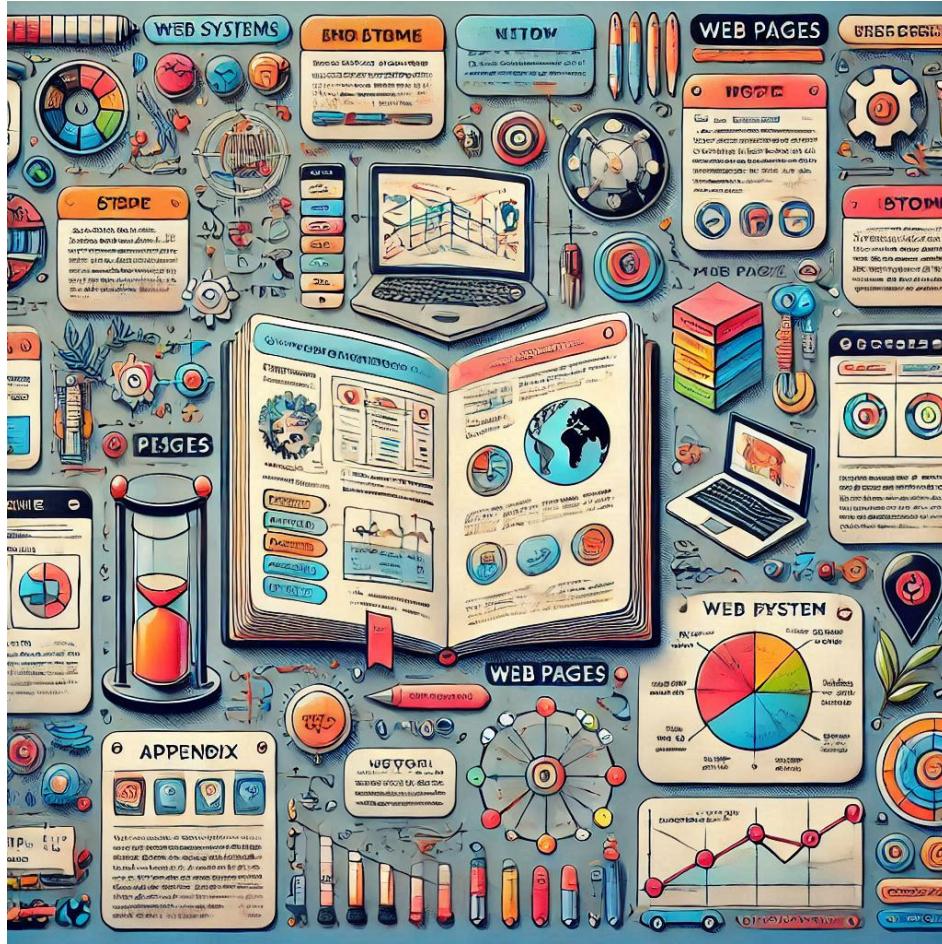
<https://getbootstrap.com/docs/4.0/components/navbar/>

<https://developer.mozilla.org/es/docs/Web/CSS/background-color>

<https://mtechnology.pro/blog/los-mejores-8-libros-de-diseno-de-paginas-web>

[https://www.google.com.mx/books/edition/Construcci%C3%B3n\\_y\\_dise%C3%BAo\\_de\\_p%C3%A1ginas\\_web/0DnCEAAAQBAJ?hl=es-419&gbpv=1&pg=PT6&printsec=frontcover](https://www.google.com.mx/books/edition/Construcci%C3%B3n_y_dise%C3%BAo_de_p%C3%A1ginas_web/0DnCEAAAQBAJ?hl=es-419&gbpv=1&pg=PT6&printsec=frontcover)

## 6. Apéndices



### APÉNDICE A (NOVEDADES DEL LENGUAJE HTML)

En los últimos años, HTML ha evolucionado significativamente, incorporando nuevas características y funcionalidades para mejorar la creación de sitios web. A continuación, se describen algunas de las actualizaciones más importantes:

1. HTML5: La versión más reciente de HTML, conocida como HTML5, introdujo una serie de elementos semánticos que mejoran la accesibilidad y la estructura de las

páginas web. Elementos como `<article>`, `<section>`, `<header>`, `<footer>`, y `<nav>` permiten a los desarrolladores estructurar mejor el contenido, facilitando tanto la comprensión humana como el procesamiento de los motores de búsqueda.

2. Soporte para Multimedia: HTML5 trajo consigo el soporte nativo para audio y video, sin necesidad de complementos adicionales como Flash. Las nuevas etiquetas `<audio>` y `<video>` permiten insertar contenido multimedia directamente en las páginas web, mejorando la experiencia del usuario y la accesibilidad.
3. Elementos de Formularios Mejorados: HTML5 incorpora nuevos tipos de entrada en los formularios, como `email`, `tel`, `date`, y `range`, que ofrecen validaciones automáticas y mejoran la interacción con el usuario. También se añadió el atributo "placeholder" para mostrar un texto de ejemplo en los campos de entrada, y el atributo "required" para asegurar que un campo no se deje vacío antes de enviar el formulario.
4. API's y Almacenamiento Local: HTML5 incluyó varias API's como la API de Geolocalización, la API de almacenamiento local (`localStorage`) y la API de arrastrar y soltar (`drag-and-drop`). Estas tecnologías permiten una interacción más rica entre el navegador y el usuario, como el almacenamiento de datos en el navegador para su uso posterior sin necesidad de una conexión constante a la red.
5. Compatibilidad con Aplicaciones Web: HTML5 permite la creación de aplicaciones web más robustas y con funcionalidad similar a las aplicaciones nativas. Características como el "modo offline" y el "manifesto de aplicaciones web" permiten a los usuarios acceder a las aplicaciones sin conexión a Internet, lo que abre nuevas posibilidades para la creación de experiencias interactivas.
6. Mejoras en el Rendimiento y la Seguridad: HTML5 también ha mejorado la seguridad y el rendimiento al incorporar nuevas directivas de seguridad, como el uso de Content Security Policy (CSP) para prevenir ataques de tipo XSS (Cross-site Scripting). Además, la optimización en el manejo de gráficos y animaciones ha facilitado la creación de contenido dinámico, como juegos o interfaces de usuario interactivas.



En resumen, las novedades introducidas en HTML, especialmente a partir de la versión HTML5, han transformado la manera en que se diseñan y desarrollan sitios web. Estas innovaciones no solo han mejorado la funcionalidad y la experiencia de usuario, sino que también han simplificado y acelerado el proceso de desarrollo web, permitiendo la creación de aplicaciones web más sofisticadas y dinámicas.

## APÉNDICE B (NOVEDADES DEL LENGUAJE CSS)

A lo largo de los años, CSS ha evolucionado de ser una herramienta básica para estilizar páginas web a un lenguaje muy robusto y versátil, capaz de proporcionar experiencias visuales complejas. A continuación se presentan algunas de las novedades más relevantes en las últimas versiones de CSS:

1. CSS Grid Layout: Una de las innovaciones más importantes en CSS es la introducción de CSS Grid Layout. Esta herramienta permite crear diseños bidimensionales más complejos, facilitando la disposición de los elementos en filas y columnas de manera flexible y eficiente. Gracias a Grid, los desarrolladores pueden crear interfaces mucho más dinámicas sin la necesidad de recurrir a frameworks adicionales.

2. Flexbox: Flexbox es otra novedad que ha simplificado la alineación y distribución de elementos dentro de un contenedor. Este sistema de diseño unidimensional permite distribuir el espacio de manera eficiente entre los elementos de una página, garantizando que se ajusten de forma automática al tamaño disponible, mejorando la responsividad de las interfaces.
3. Propiedades de Transición y Animación: CSS ha mejorado significativamente en el ámbito de las animaciones. Con la incorporación de las propiedades transition y animation, ahora es posible aplicar cambios de estilo suaves y controlados a los elementos cuando se produce algún evento, como el pasar el cursor sobre un objeto. Esto permite a los diseñadores crear interacciones visuales más ricas y dinámicas sin necesidad de JavaScript.
4. Media Queries Mejoradas: Las media queries han sido ampliadas y mejoradas para brindar un control más preciso sobre el diseño responsive. Con las nuevas características, los diseñadores pueden aplicar diferentes estilos dependiendo de factores como el tamaño de la pantalla, la resolución y la orientación del dispositivo. Esto facilita la creación de sitios web que se adaptan perfectamente a cualquier tipo de dispositivo, ya sea un teléfono móvil, tablet o escritorio.
5. Variables en CSS: Una de las incorporaciones más prácticas en CSS es el soporte para variables, también conocidas como Custom Properties. Estas permiten a los desarrolladores definir valores reutilizables para propiedades de estilo, lo que mejora la mantenibilidad del código y facilita la actualización de diseños. Al igual que en los lenguajes de programación, las variables CSS hacen que el código sea más limpio y flexible.
6. Sombras y Efectos Avanzados: CSS ha añadido nuevas propiedades como box-shadow y text-shadow, que permiten crear efectos de sombra en los elementos sin necesidad de imágenes externas. Esto otorga a los diseñadores un mayor control sobre los efectos visuales, mejorando la estética y la profundidad de las páginas web.
7. Soporte para Tipografía Avanzada: Con las mejoras en la tipografía de CSS, ahora es posible utilizar fuentes personalizadas de manera más sencilla y mejorar el control sobre el espaciado y alineación del texto. Propiedades como letter-spacing, line-height, y text-align permiten un ajuste más preciso de la presentación del texto, mientras que las fuentes externas pueden ser cargadas fácilmente a través de @font-face.
8. Funciones de Filtros y Efectos Visuales: Con la llegada de los filtros CSS, los diseñadores pueden aplicar efectos visuales complejos como desenfoques, transformaciones y cambios de color directamente a los elementos sin necesidad de software de edición gráfica. Propiedades como filter: blur() o filter: grayscale() permiten añadir efectos interesantes de forma sencilla.



En conclusión, las novedades de CSS han permitido a los desarrolladores web crear interfaces más sofisticadas, interactivas y visualmente atractivas. Las nuevas propiedades y técnicas no sólo simplifican el desarrollo de diseños complejos, sino que también mejoran la experiencia de usuario y la flexibilidad en el diseño responsive. Gracias a estas innovaciones, CSS continúa siendo una herramienta indispensable en el diseño web moderno.

## APÉNDICE C (NOVEDADES DEL LENGUAJE JAVASCRIPT)

JavaScript ha sido un pilar fundamental en el desarrollo web moderno, y con el tiempo, ha incorporado nuevas características que mejoran su capacidad, eficiencia y versatilidad. A continuación, se presentan algunas de las principales novedades en JavaScript:

1. ES6 (ECMAScript 2015) y versiones posteriores: Una de las actualizaciones más significativas para JavaScript fue la introducción de ES6 (ECMAScript 2015), que trajo consigo numerosas características nuevas que simplificaron el desarrollo y mejoran la legibilidad del código. Entre las novedades más destacadas se encuentran las clases, que introdujeron una forma más sencilla y comprensible de trabajar con objetos y herencia en comparación con las funciones constructores tradicionales.

2. Let y Const: Antes de ES6, JavaScript solo disponía de la palabra clave var para declarar variables. Con ES6, se introdujeron let y const, que proporcionan un control más preciso sobre el alcance y la mutabilidad de las variables. let permite declarar variables con un alcance de bloque, mientras que const se utiliza para declarar constantes cuyo valor no puede ser reasignado.
3. Funciones Arrow (Arrow Functions): Las arrow functions son una nueva forma de definir funciones en JavaScript. Utilizando una sintaxis más corta, este tipo de funciones también ofrece un comportamiento especial respecto al contexto de this, ya que no lo vinculan al objeto que las invoca, sino que lo heredan del entorno donde se crearon. Esto resuelve algunos problemas relacionados con el manejo de this en funciones tradicionales.
4. Promesas y Async/Await: Antes de la introducción de las promesas en ES6, JavaScript manejaba la asíncronía mediante callbacks, lo que a menudo llevaba a problemas como el "callback hell". Las promesas proporcionan una forma más clara y estructurada de manejar operaciones asíncronas, y con la llegada de async/await en versiones posteriores, se simplificó aún más el trabajo con código asíncrono, haciendo que se escribiera de una manera más sincrónica y fácil de entender.
5. Desestructuración: La desestructuración es una característica que permite extraer valores de arrays u objetos y asignarlos a variables de manera más compacta. Esta característica hace que el código sea más limpio y eficiente, y se aplica tanto a los objetos como a los arreglos. Por ejemplo, en lugar de acceder a una propiedad de un objeto de forma tradicional, se puede hacer directamente con la sintaxis de desestructuración.
6. Spread y Rest Operator: El spread operator (...) se utiliza para expandir elementos de un array o propiedades de un objeto, lo que facilita la clonación de objetos y arreglos o la combinación de múltiples elementos. Por otro lado, el rest operator se utiliza para agrupar varios elementos en un solo array, permitiendo una gestión más flexible de los parámetros en funciones y objetos.
7. Módulos (Modules): JavaScript ha incorporado un sistema de módulos nativo que permite dividir el código en archivos separados. Esto facilita la organización del código, favorece la reutilización y mejora la mantenibilidad del proyecto. Con las instrucciones import y export, los desarrolladores pueden importar y exportar funciones, objetos o clases entre diferentes archivos.
8. Operador de Encadenamiento Opcional (Optional Chaining): Introducido en versiones más recientes, el operador de encadenamiento opcional permite acceder a propiedades de objetos profundamente anidados sin necesidad de verificar explícitamente si cada nivel del objeto es null o undefined. Esto reduce la necesidad de escribir verificaciones redundantes y hace que el código sea más limpio y menos propenso a errores.
9. Nullish Coalescing Operator: Este operador es una forma más precisa de manejar valores nulos o indefinidos. A diferencia del operador OR (||), que retorna el primer valor verdadero, el operador nullish coalescing solo devuelve el segundo operando si el primer operando es null o undefined, lo que evita problemas al trabajar con valores falsy como 0, NaN, o cadenas vacías.

10. Collections (Map, Set, WeakMap, WeakSet): JavaScript ha introducido nuevas estructuras de datos como Map, Set, WeakMap, y WeakSet. Estas permiten trabajar de manera más eficiente con colecciones de datos, ofreciendo ventajas en términos de búsqueda, almacenamiento y eliminación de elementos en comparación con los objetos y arrays tradicionales.



En resumen, las novedades más recientes de JavaScript han mejorado sustancialmente la sintaxis, la eficiencia y la capacidad del lenguaje para manejar operaciones complejas, especialmente en el contexto de la programación asíncrona y el desarrollo de aplicaciones modernas. Estas mejoras permiten escribir código más limpio, fácil de mantener y con un rendimiento más optimizado, facilitando el desarrollo de aplicaciones web ricas e interactivas.

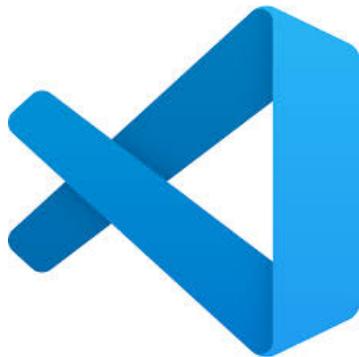
## APÉNDICE D (EDITOR DE CÓDIGO FUENTE-VSC)

### VISUAL STUDIO CODE (VSC)

El Visual Studio Code (VSC) es un software, además, es un editor de código fuente gratuito desarrollado por Microsoft, este se puede utilizar en Windows, Ubuntu o MacOS (ya sea el de tu preferencia o el mejor que se te acomode).

Esta herramienta es de suma importancia ya que para los programadores te permite escribir y editar los códigos de los programas que vayas haciendo.

**Logo**



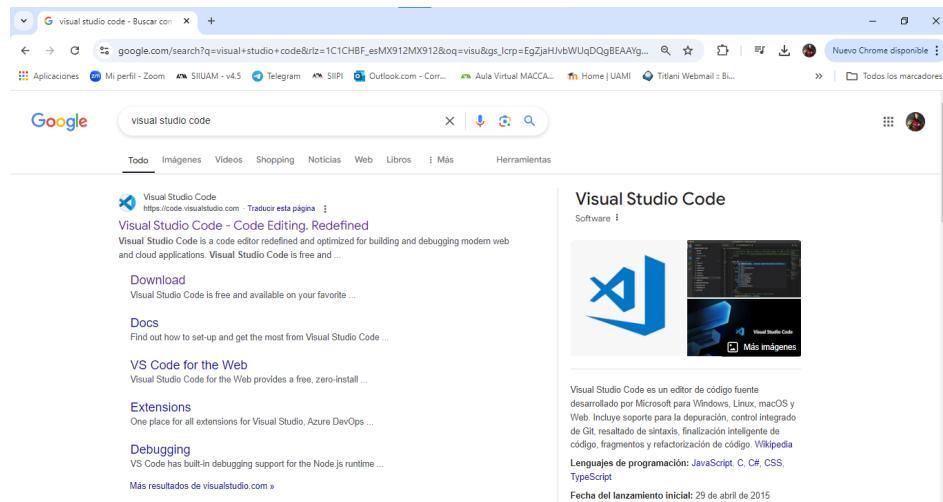
**Figura X: Logo del programa de Visual Studio Code (VSC).**

## INSTALACIÓN

A continuación, se indica cómo se realiza.

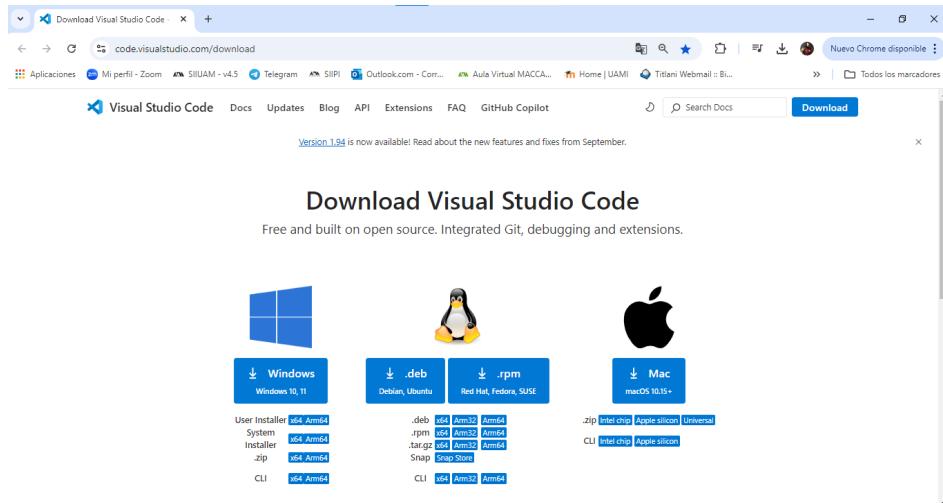
1. Abrir el navegador web (puede ser el de tu preferencia, ya sea Google Chrome, Microsoft Edge, Mozilla Firefox, Brave, etc) y buscar visual studio code y darle Enter o ↵ del teclado o en el ícono de la lupa (Buscar), después darle clic al vínculo o link (URL) de Download (Descargar) para abrir la página oficial para poder descargar la aplicación (ver figura X).

Link: <https://code.visualstudio.com/download>



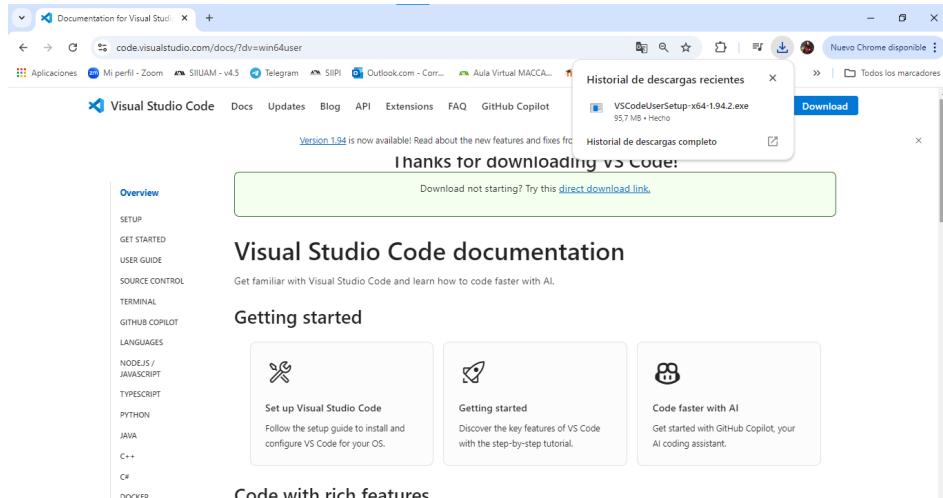
**Figura X: Navegador web de Google Chrome para descargar la aplicación.**

2. Una vez adentro de la página, te aparecerán las distintas formas de poder descargar la aplicación. Aquí debes saber qué tipo de máquina es la que vas a estar utilizando, ya sea Windows, Ubuntu o MacOs (esto lo puedes revisar en la configuración de tu máquina y ver los detalles) (ver figura X).

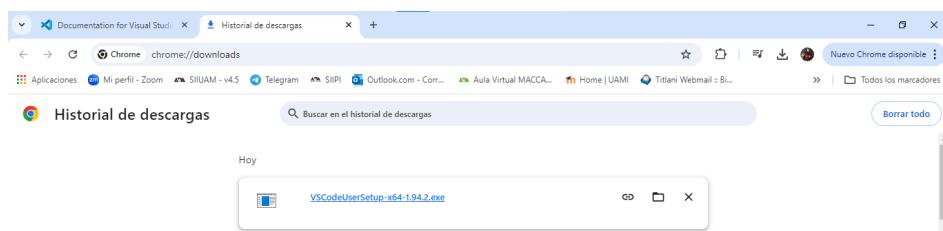


**Figura X: Distintas opciones de descargar Visual Studio Code (VSC) en tu ordenador o tu computadora.**

3. Despues se te va a descargar el paquete de instalación de Visual Studio Code con la extensión de .exe (esto lo puedes ver en la figura X ), lo puedes ver bien usando la combinación de teclas Ctrl + J, esto es para abrir el historial de las descargas que hagas (ver figura X ).

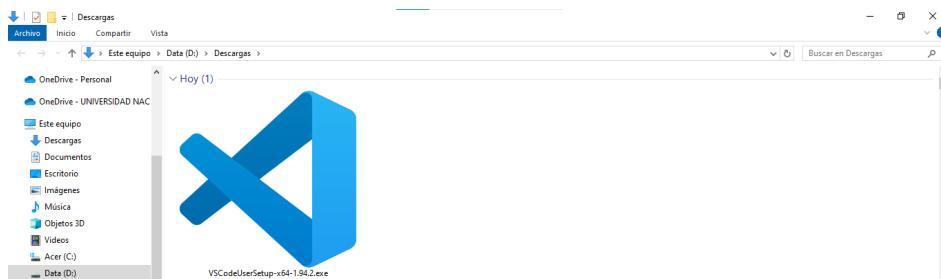


**Figura X: Descarga completa del paquete de instalación de Visual Studio Code (VSC).**



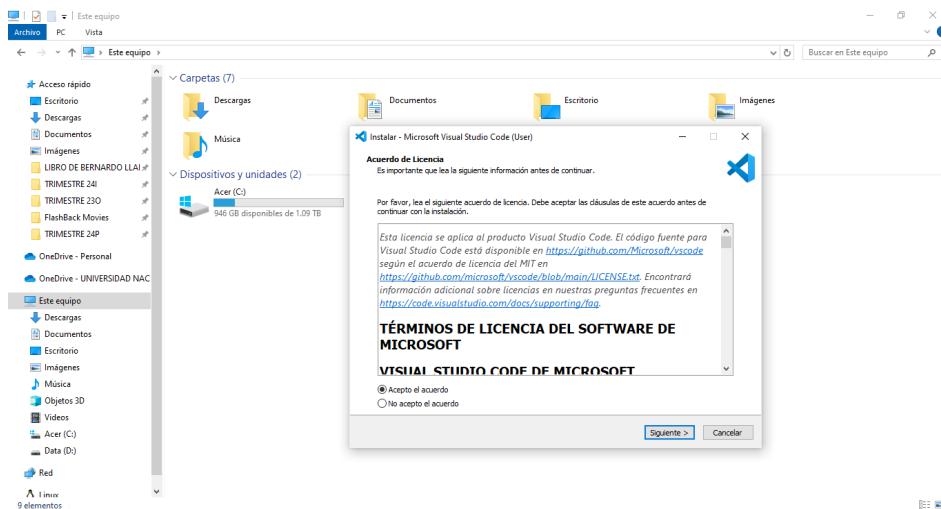
**Figura X: Historial de las descargas que vayas haciendo.**

4. En la carpeta de archivos, busca en la carpeta de Downloads o Descargas para buscar lo que descargaste, en este caso el archivo de .exe del paquete de instalación de Visual Studio Code (VSC) y darle clic al archivo para que se empiece a ejecutar (ver figura X ).



**Figura X: Carpeta de archivos donde se encuentra la descarga que acabas de hacer.**

5. Para poder abrirlo, tienes 3 opciones, las cuales son: le puedes dar doble clic o lo puedes seleccionarlo y darle Enter con el teclado o lo puedes seleccionarlo y dar clic derecho para que se te abra el menú de las opciones y le das clic en “Abrir”. Despues, se te abrirá una ventana emergente en la cual se comenzará a instalar el programa que acabas de descargar y tienes que leer el **Acuerdo de Licencia** para poder hacer la instalación (ver figura X).



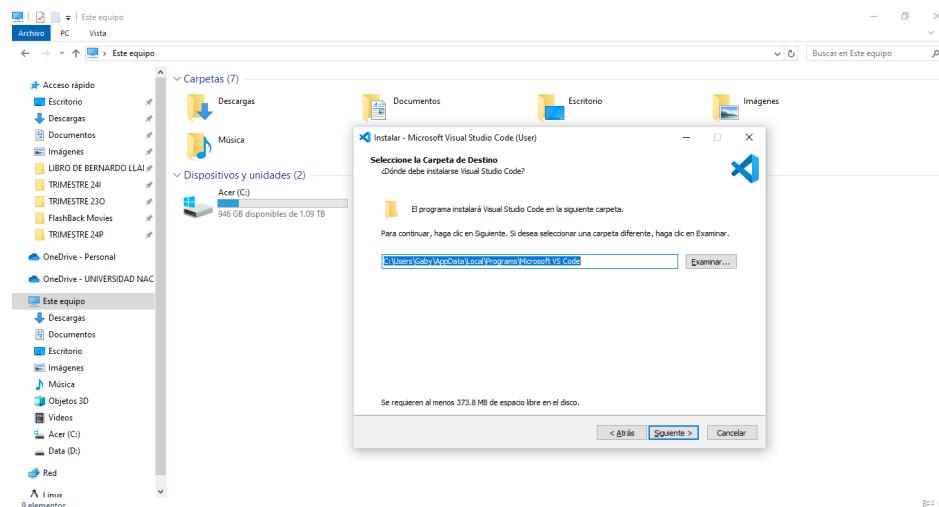
**Figura X: Ventana emergente de la instalación del programa de Microsoft Visual Studio Code.**

6. En esta parte, le hice un zoom a la ventana emergente de la instalación para poder verlo bien y entender cómo se hace (ver figura X-ZOOM).

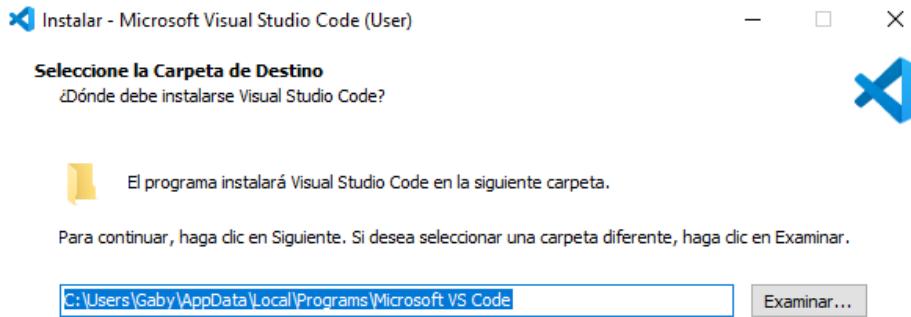


**Figura X:** Ventana del Acuerdo de Licencia del VSC, en este debes de leerlo, darle clic en la opción de “Acepto el acuerdo” y darle clic en “Siguiente” para continuar con la instalación, usando zoom.

7. Asimismo, debes continuar con el proceso de la instalación, y revisar lo que se te vaya pidiendo, aquí debes de **Seleccionar la Carpeta de Destino** donde se instalará VSC y debes procurar revisar bien la ruta donde se encuentra la carpeta (ver figura X y ver figura X-ZOOM).



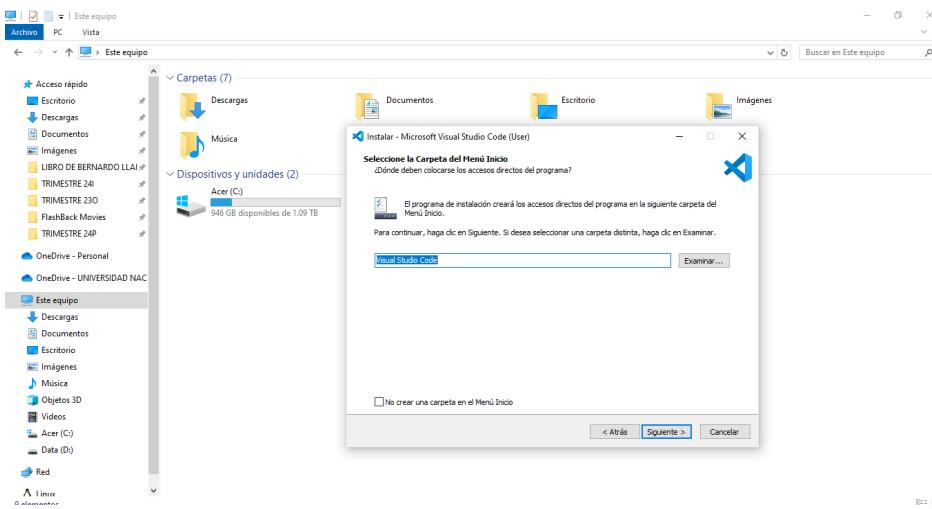
**Figura X:** Selección de la Carpeta de Destino donde se instalará el programa de VSC usando la ruta establecida en tu computadora.



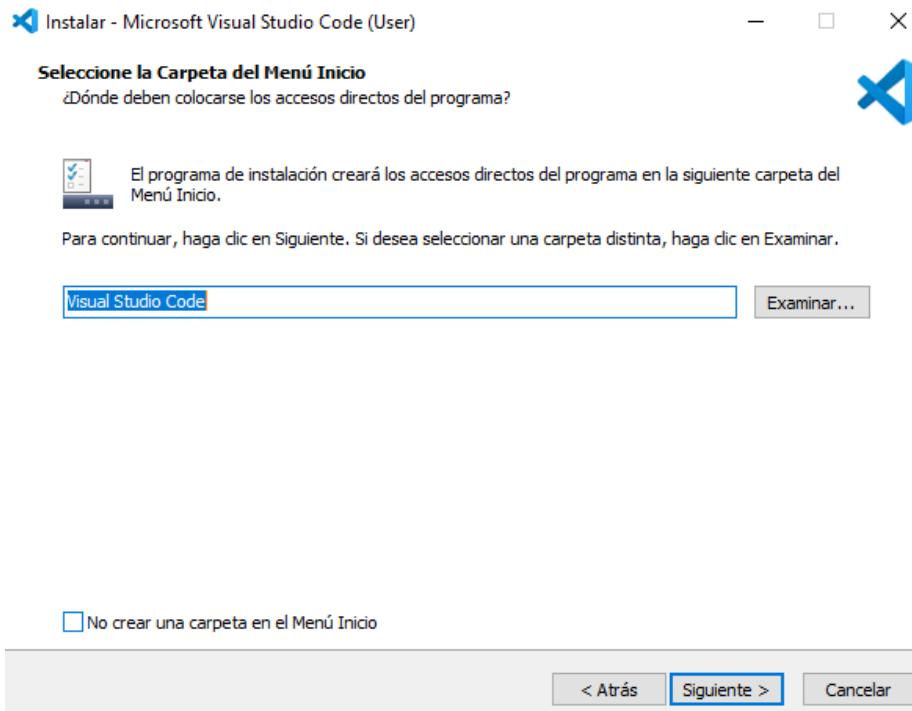
**Figura X: Selección de la Carpeta de Destino donde se instalará el programa de VSC usando la ruta establecida en tu computadora usando zoom para verlo bien.**

**NOTA: Si no sabes cómo revisar la ruta, lo que debes hacer es dejarla como está, o sea, sin moverle a nada y darle clic en “Siguiente” para continuar con la instalación.**

8. Luego, se te aparecerá que debes **Seleccionar la Carpeta del Menú Inicio** del programa, es decir, que son los accesos directos que contiene el programa, en este caso, **NO** hay que moverla nada y darle clic en la opción de “Siguiente” (ver figura X y ver figura X-ZOOM).

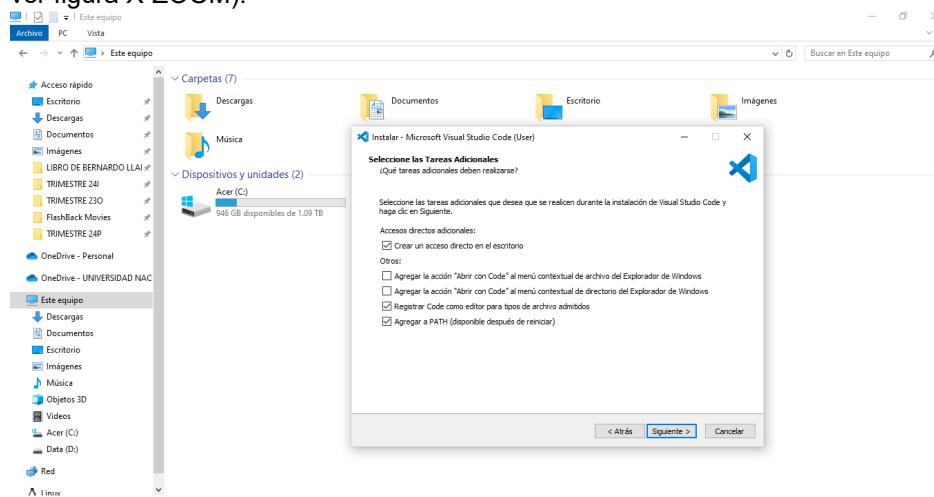


**Figura X: Ventana emergente de la Selección de la Carpeta del Menú Inicio del programa VSC y darle clic en “Siguiente”.**



**Figura X:** Ventana emergente de la Selección de la Carpeta del Menú Inicio del programa VSC usando zoom y darle clic en “Siguiente”.

9. En adición, se te aparecerá la **Selección de las Tareas Adicionales** del programa, dicho de otro modo, revisar el listado de las otras cosas que quieras que se hagan durante la instalación de VSC, pero, si usted no lo sabe, lo que debe hacer es dejarlo así y darle clic en la opción de “Siguiente” (ver figura X-ZOOM).



**Figura X:** Ventana emergente de la Selección de las Tareas Adicionales del programa VSC y darle clic en “Siguiente”.

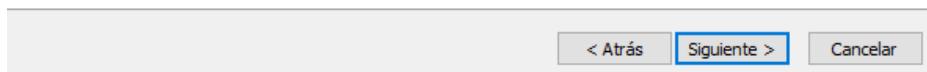
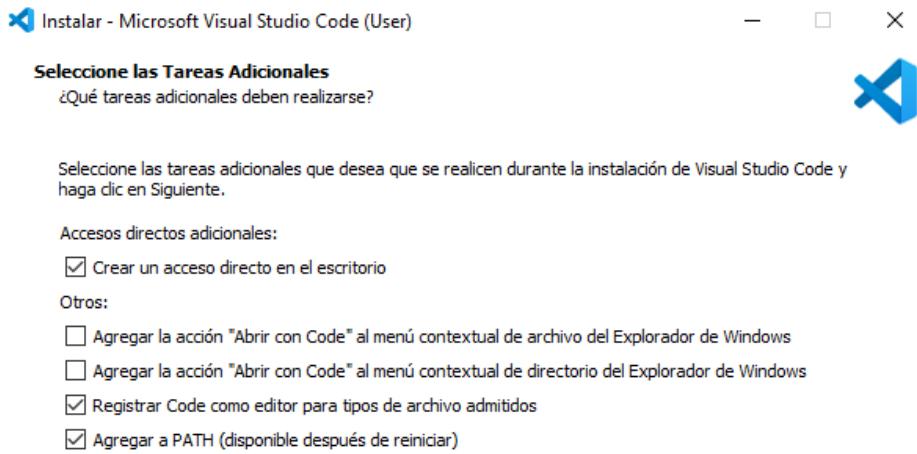


Figura X: Ventana emergente de la Selección de las Tareas Adicionales del programa VSC y darle clic en “Siguiente”, usando zoom.

10. Seguidamente, se te aparecerá lo de **Listo para Instalar**, esto quiere decir que el programa está listo o preparado para poder iniciar la instalación de Visual Studio Code en el sistema, aquí verás los detalles de la configuración, en esto se encuentran la Carpeta de Destino, la Carpeta del Menú Inicio y las Tareas Adicionales que hicimos anteriormente. En esta parte, si usted desea volver a revisar las cosas o cambiar algo, le puede dar clic en la opción de “Atrás”, y si no, si usted ya está seguro que quiere continuar, le puede dar clic la opción de “Instalar” para que siga la instalación (ver figura X y ver figura X-ZOOM).

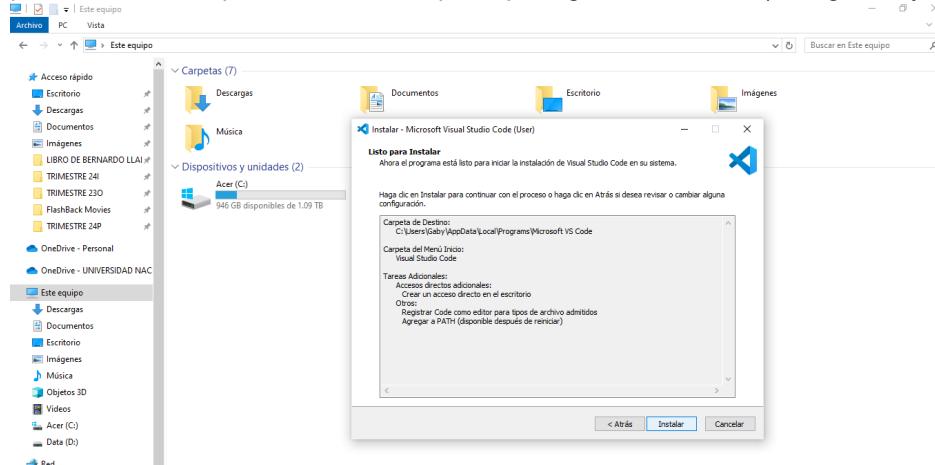
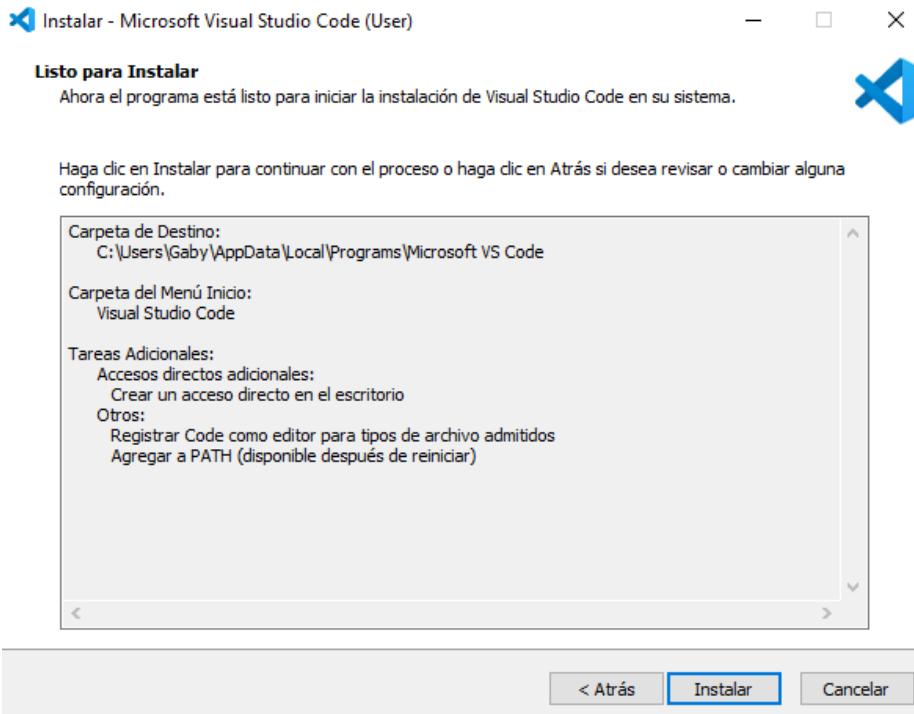
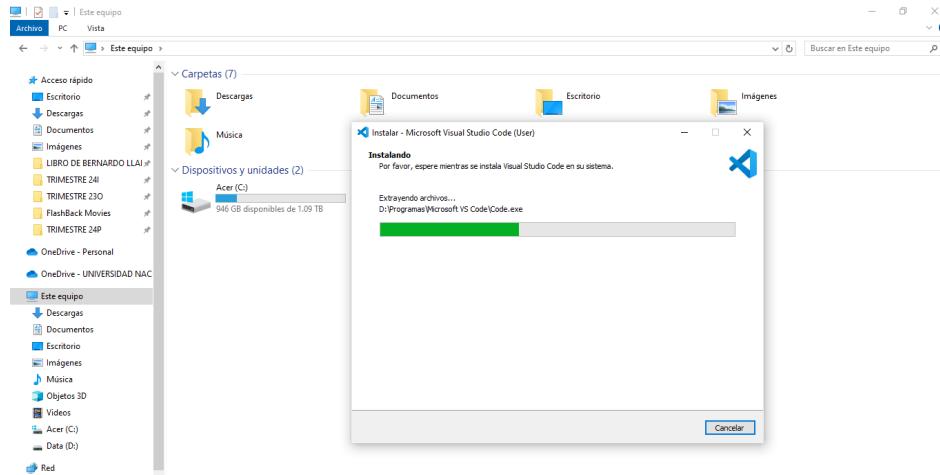


Figura X: Ventana emergente de Listo para Instalar el programa VSC y darle clic en “Instalar”.



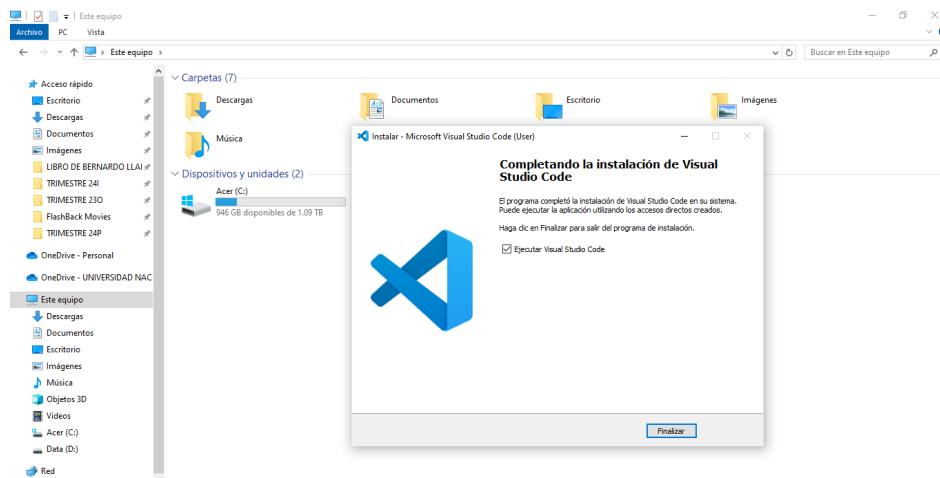
**Figura X:** Ventana emergente de Listo para Instalar el programa VSC y darle clic en “Instalar”, usando zoom.

11. Aquí debe de esperar a que se vaya instalando el programa de Visual Studio Code en su sistema que vaya a usar, esto puede demorar en unos minutos o incluso pueden ser horas, esto depende del espacio que tenga disponible en su ordenador o computadora (ver figura X).

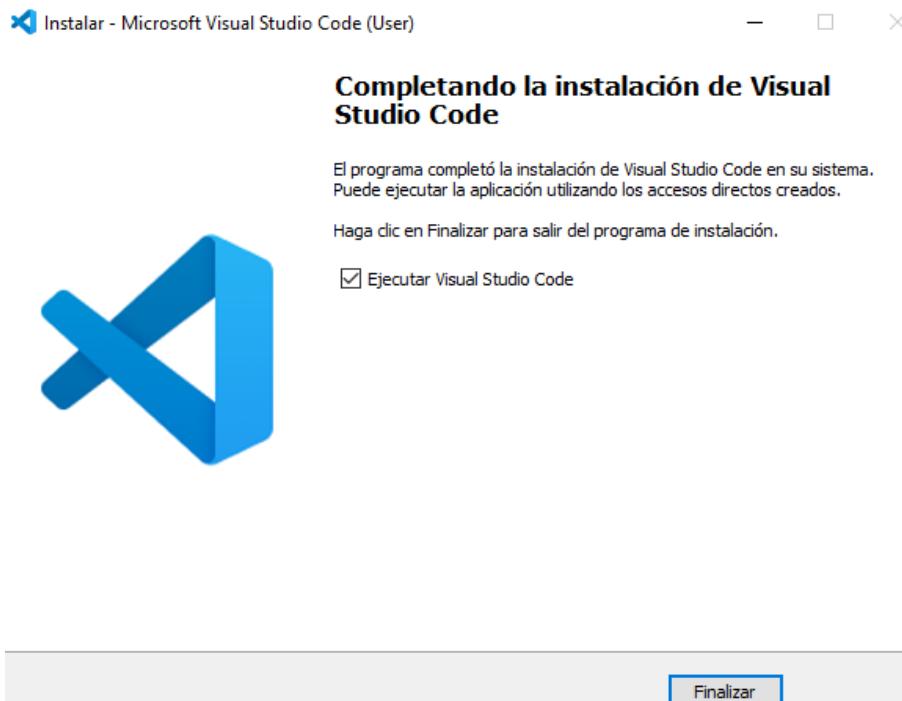


**Figura X:** ¿Tiempo de espera? de la Instalación de Visual Studio Code en el sistema.

12. Una vez que se haya completado correctamente la instalación de Visual Studio Code en el sistema, debe hacer clic en la opción de “Finalizar” para poder salir del programa de instalación (ver figura X y ver figura X-ZOOM).



**Figura X:**Ventana emergente que muestra la instalación completa del programa VSC en el sistema que está usando y le das clic en “Finalizar” para salir.



**Figura X:**Ventana emergente que muestra la instalación completa del programa VSC en el sistema que está usando y le das clic en “Finalizar” para salir, usando zoom.

**NOTA:** Si más tarde usted quiere cambiar o modificar la instalación, lo que debe hacer es que tiene que volver a ejecutar el paquete de instalación de VSC con la extensión de .exe y realizar los cambios que desee o que necesite.

#### PÁGINA DE INICIO O INICIACIÓN DE LA APLICACIÓN DE VISUAL STUDIO CODE (VSC)

Una vez terminada la instalación, en tu Escritorio, busca el ícono de la aplicación de Visual Studio Code (VSC) (ver figura X), para poder abrirlo, tienes 3 opciones, las cuales son: le puedes dar doble clic o lo puedes seleccionarlo y darle Enter con el teclado o lo puedes seleccionarlo y dar clic derecho para que se te abra el menú de las opciones y le das clic en “Abrir”, y así verás la iniciación o la página de inicio de este, así como se muestra en la figura (ver figura X).



Figura X: Ícono de la aplicación de Visual Studio Code (VSC) en tu Escritorio de tu sistema.

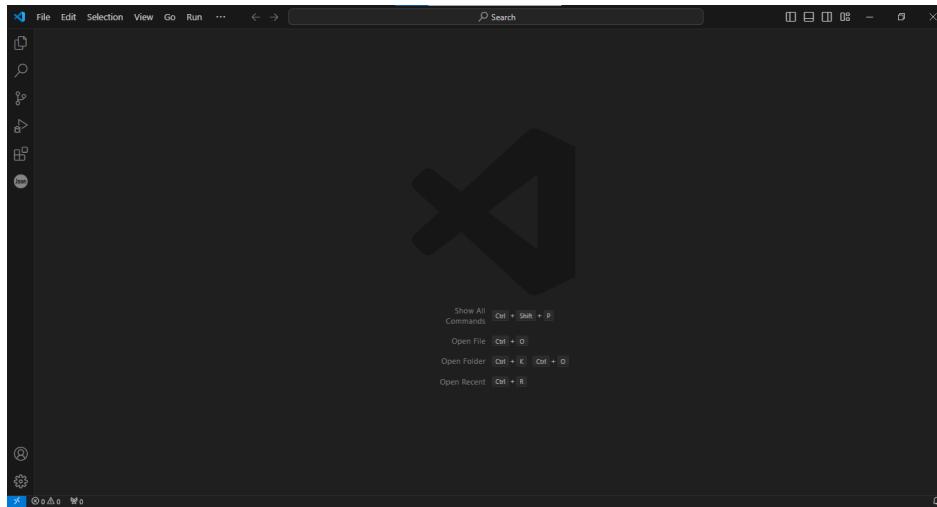


Figura X: Iniciación del aplicación de Visual Studio Code (VSC).

Después, se te abrirá la Bienvenida que te da el VIsual Studio Code, en este te aparecen las opciones de cómo vas a empezar, es decir, de cómo crear y abrir un archivo, abrir una carpeta, clonar un repositorio Github y conectar con otros entornos, etc. Además, se te aparecen tus archivos recientes los cuales son los archivos o carpetas que puedes volver a usar y se encuentran los Walkthroughs (en español, se traduce como los “tutoriales”), aquí son las guías que da el VSC para poder empezar a programar (ver figura X).

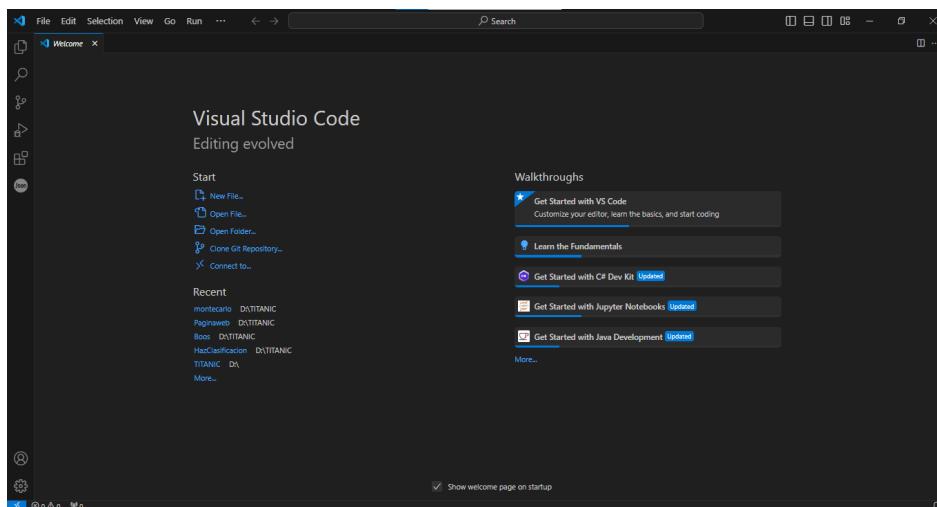


Figura X: Bienvenida al Visual Studio Code (VSC).

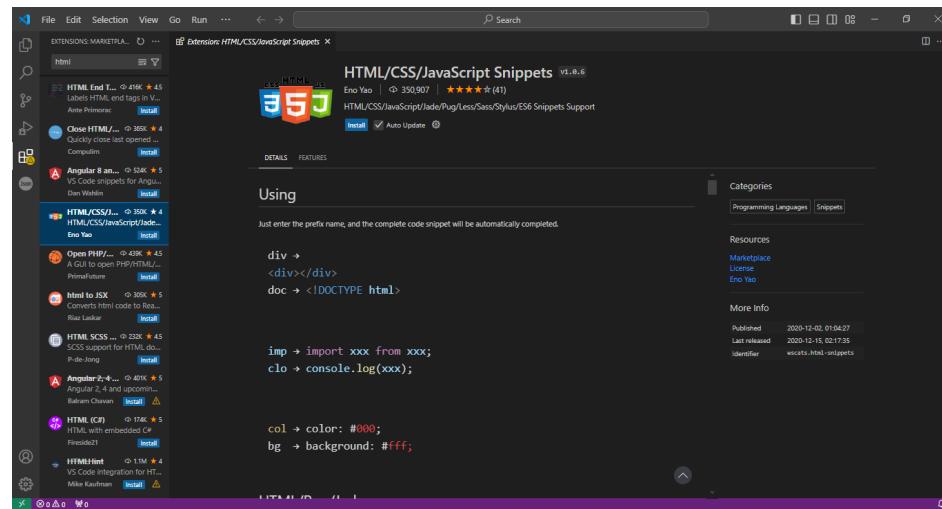
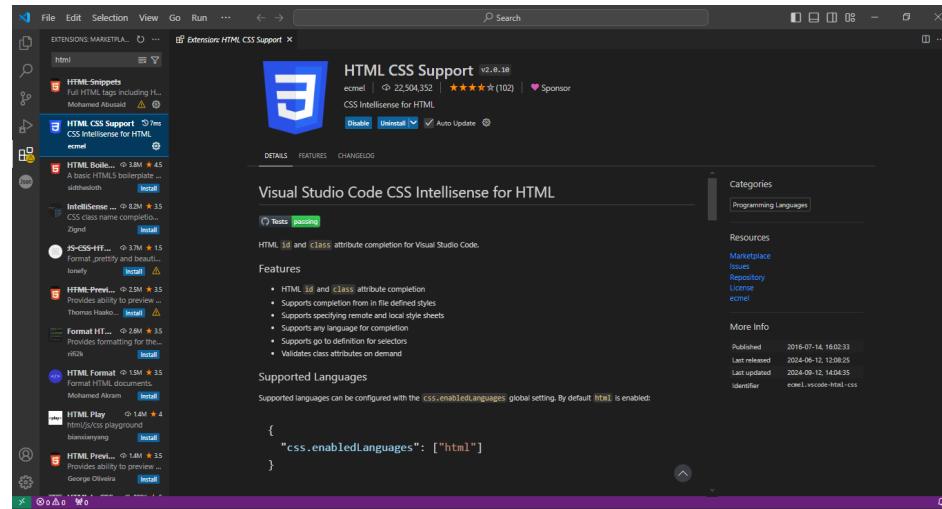
## EXTENSIONES

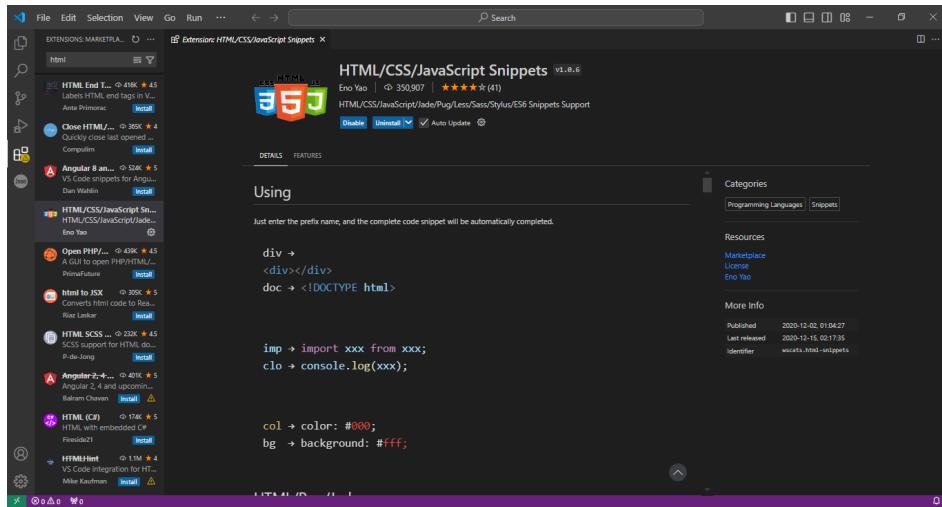
Las extensiones o complementos son herramientas adicionales o paquetes de código que se ejecutan dentro del Visual Studio Code. Estas son diseñadas para ayudarnos con el lenguaje de programación

que vayamos a utilizar, ya sea por ejemplo para Python, Haskell, Prolog, C, entre otros.  
que cuentan Visual Studio Code.....

Hay que instalar las extensiones que dicen de HTML , CSS, JS.

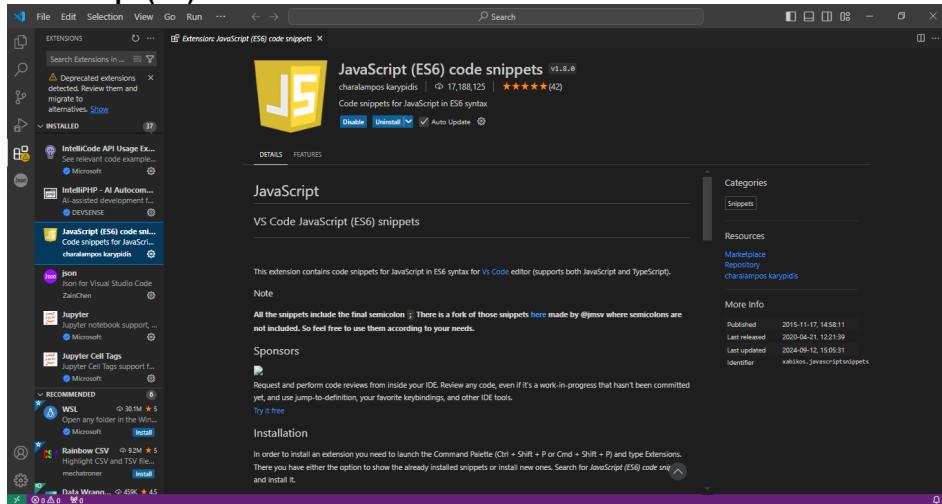
## HTML





## CSS

### JavaScript (JS)



Una vez que termines de instalar las extensiones de cada lenguaje que necesites, ahora sí podrás realizar tus primeros códigos dentro del VSC (Esto lo puedes ver en los apartados de HTML, CSS y JS, que se encuentran antes).

## 7. ÍNDICE DE LAS PALABRAS

ALGORITMO, PP.X  
ARCHIVO, PP.X  
CÓDIGO, PP.X

...  
 LENGUAJE, PP.X

....

[1] Programa informático: Conjunto de instrucciones ejecutables, escritas con un lenguaje de programación que permite al usuario cumplir una función específica o realizar una tarea concreta.