

Clase 7:

Lenguajes de

Programación

Índice

Lenguaje de programación:

- Concepto
- Tipos
- Clasificación (bajo, alto y medio)

Paradigmas de la programación

- Clasificación

Lenguaje C.

- Concepto
- Características
- Ventajas

JavaScript:

- Historia.
- Orígenes y concepto.
- Características claves.

Interprete de código y compilador de código:

- Conceptos y diferencias.

El motor de JavaScript: V8.

- Características
- Componentes
- Especificaciones.

Algoritmos:

- Concepto
- Ejemplo practico
- Uso en Programación
- Características claves

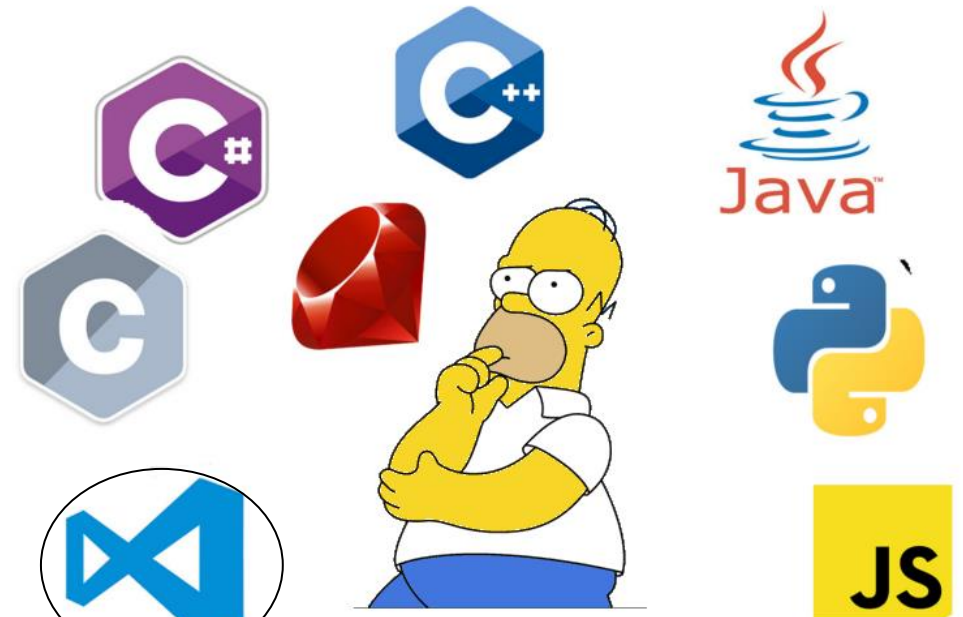


Lenguaje de programación

Lenguaje de Programación

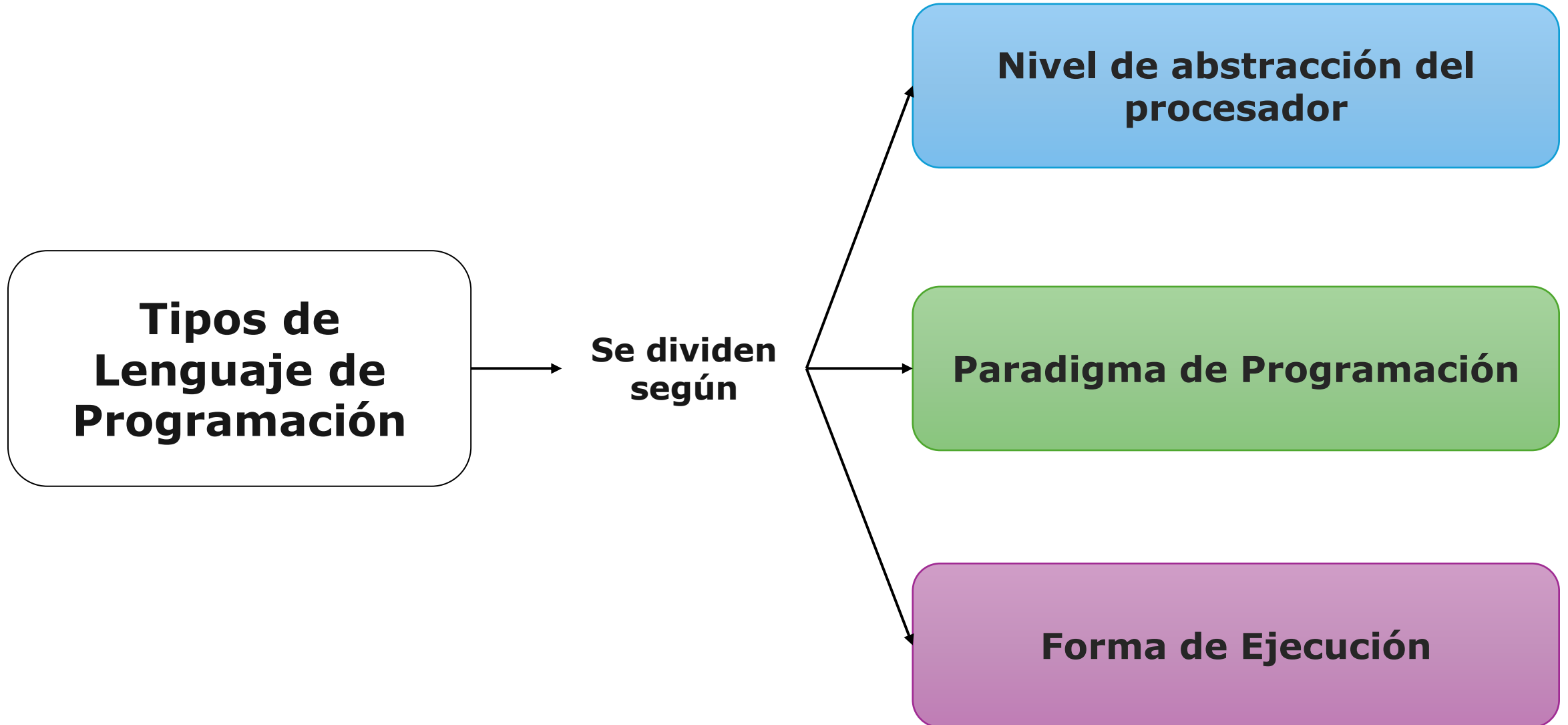
Concepto

- Un lenguaje de programación es un conjunto de reglas, sintaxis y semántica que permiten a los programadores escribir instrucciones que una computadora puede entender y ejecutar.
- Estos lenguajes se utilizan para desarrollar software, aplicaciones, sistemas operativos y realizar tareas específicas dentro de un entorno informático.



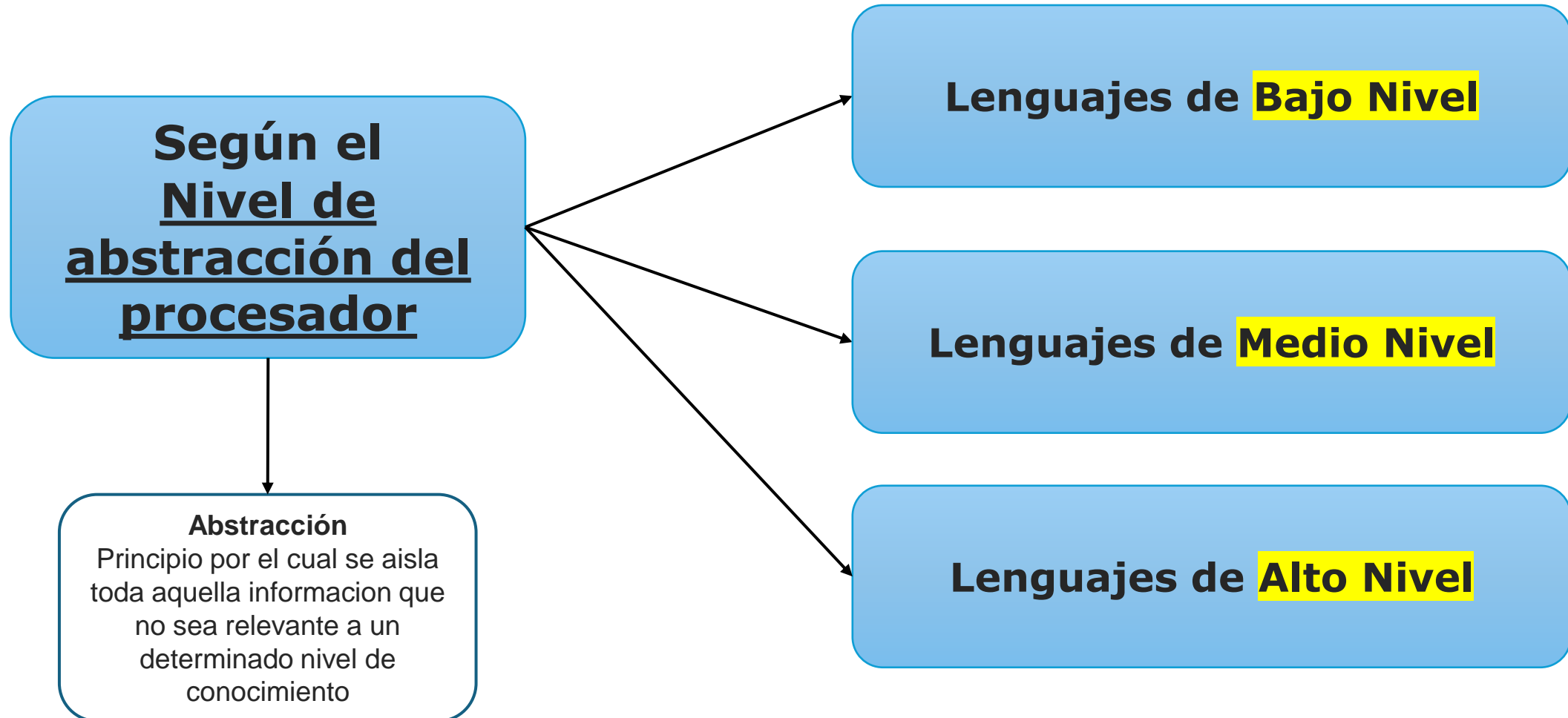
Editor de Código Fuente

Lenguaje de Programación



Lenguaje de Programación

- Clasificación de Lenguajes -



Lenguaje de Programación

- Nivel de abstracción del procesador -

Lenguajes de Bajo Nivel

Es el que proporciona poca o ninguna abstracción del microprocesador de una computadora. Consecuentemente es fácilmente trasladado a lenguaje de máquina. En general se utiliza este tipo de lenguaje para programar controladores (drivers)

Ejemplos:

- **Lenguaje Ensamblador**: Utiliza palabras abreviadas para representar instrucciones en lenguaje máquina. Cada instrucción de ensamblador se traduce directamente a una instrucción de máquina.
 - Ventajas: Mayor control sobre el hardware, más legible que el lenguaje máquina puro.
 - Desventajas: Aún es complicado y específico para cada arquitectura de CPU, lo que dificulta la portabilidad.
- **Lenguaje Máquina**: Es el nivel más bajo de programación, directamente ejecutado por la CPU. Consiste en instrucciones binarias (0s y 1s).
 - Ventajas: Máximo control sobre el hardware y el rendimiento.
 - Desventajas: Muy difícil de escribir, leer y mantener.

Lenguaje de Programación

- Nivel de abstracción del procesador -

Lenguajes de Medio Nivel

- Es un lenguaje de programación informática que se encuentran entre los lenguajes de alto nivel y los lenguajes de bajo nivel. Suelen ser clasificados muchas veces de alto nivel, pero permiten ciertos manejos de bajo nivel.
- Son precisos para ciertas aplicaciones como la creación de sistemas operativos, ya que permiten un manejo abstracto (independiente de la máquina, a diferencia del ensamblador), pero sin perder mucho del poder y eficiencia que tienen los lenguajes de bajo nivel.
- Una característica distintiva, por ejemplo, que convierte a C en un lenguaje de medio nivel es que es posible manejar las letras como si fueran números.

Lenguaje de Programación

- Nivel de abstracción del procesador -

Lenguajes de Alto Nivel

- Se caracterizan por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de a la capacidad ejecutora de las máquinas.
- En los primeros lenguajes de alto nivel la limitación era que se orientaban a un área específica y sus instrucciones requerían de una sintaxis predefinida.
- Se clasifican como lenguajes procedimentales.
- Otra limitación de los lenguajes de alto nivel es que se requiere de ciertos conocimientos de programación para realizar las secuencias de instrucciones lógicas.
- Los lenguajes de muy alto nivel se crearon para que el usuario común pudiese solucionar tal problema de procesamiento de datos de una manera más fácil y rápida.

Paradigma de programación

Lenguaje de Programación

¿Que es un Paradigma en Programación?

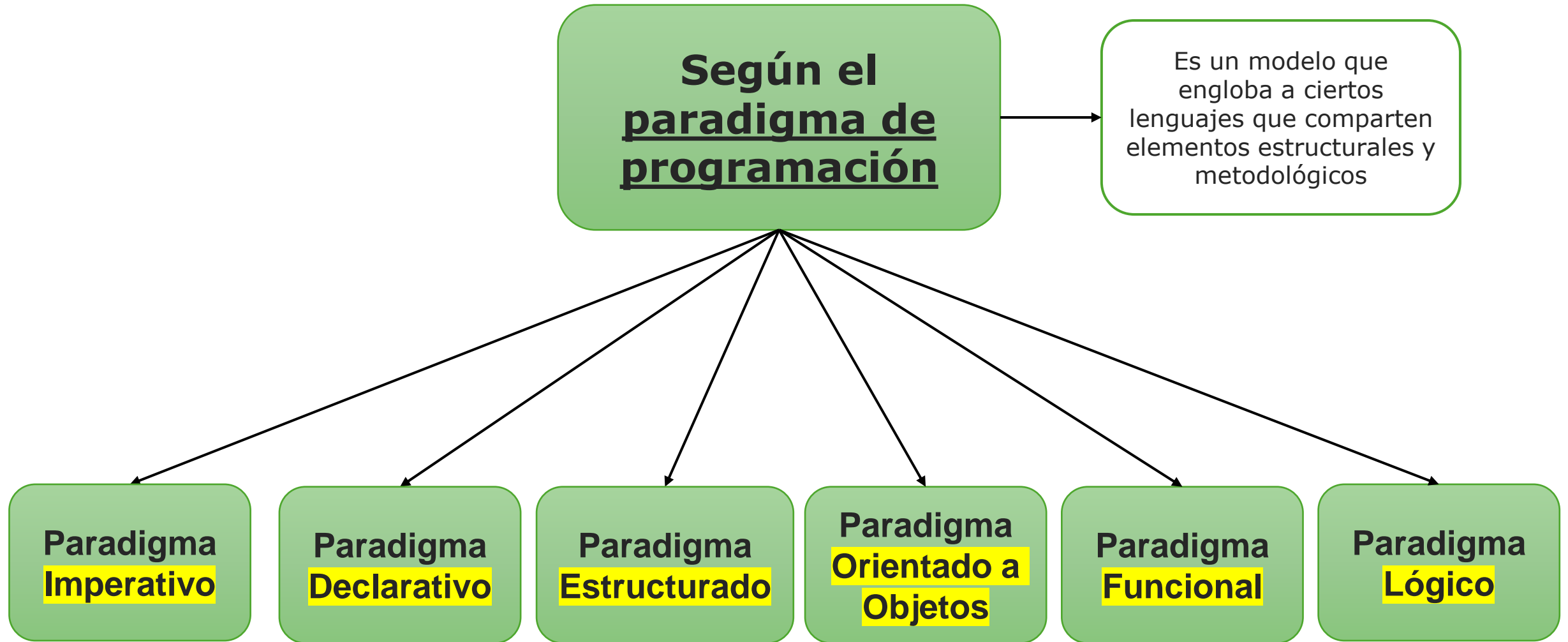
- Un paradigma de programación es un **estilo o enfoque** en la programación que guía la forma en que se estructura y escribe el código.
- Define un **conjunto de principios, conceptos y métodos** que determinan cómo se deben organizar y manipular los datos y las instrucciones dentro de un programa.
- La mejor manera de conocer un paradigma de programación es investigar y programar en un lenguaje característico de ese paradigma. **No hace falta ser un experto.** Solo el hecho de conocerlo nos brinda más herramientas a la hora de desarrollar.



Los paradigmas son los diferentes estilos de usar la programación para resolver un problema.



Lenguaje de Programación

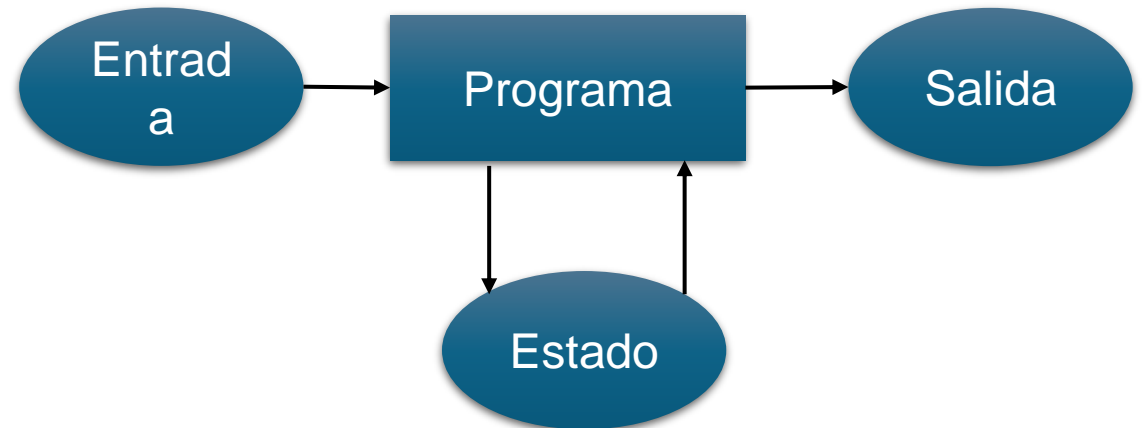


Lenguaje de Programación

- Paradigma de Programación -

Paradigma Imperativo

- Describe la **programación** como una secuencia instrucciones o comandos que cambian el estado de un programa.
- El **código máquina** en general está basado en el paradigma imperativo.
- Su contrario es el paradigma declarativo.
- En este paradigma se incluye el paradigma procedimental (procedural) entre otros

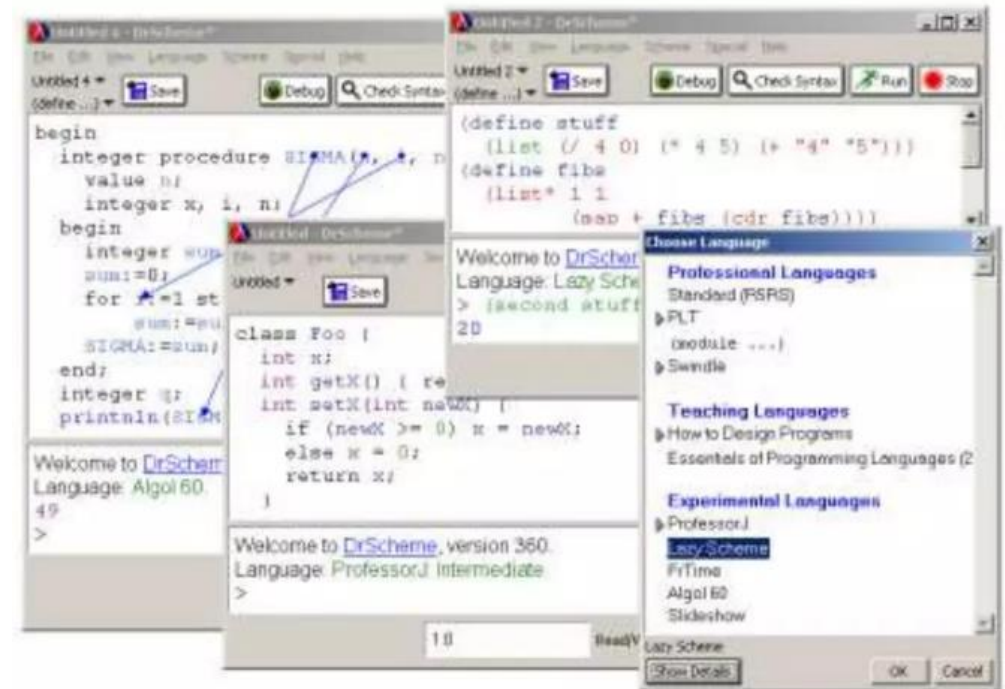


Lenguaje de Programación

- Paradigma de Programación -

Paradigma Declarativo

- No se basa en el **cómo se hace algo** (como se logra un objetivo paso a paso), sino que **describe (declara) cómo es algo**.
- En otras palabras se enfoca en describir las propiedades de la solución buscada dejando indeterminado el algoritmo (conjunto de instrucciones) usado para encontrar esa solución
- Tiene **desventajas en la eficiencia** pero ventajas en la solución de determinados problemas

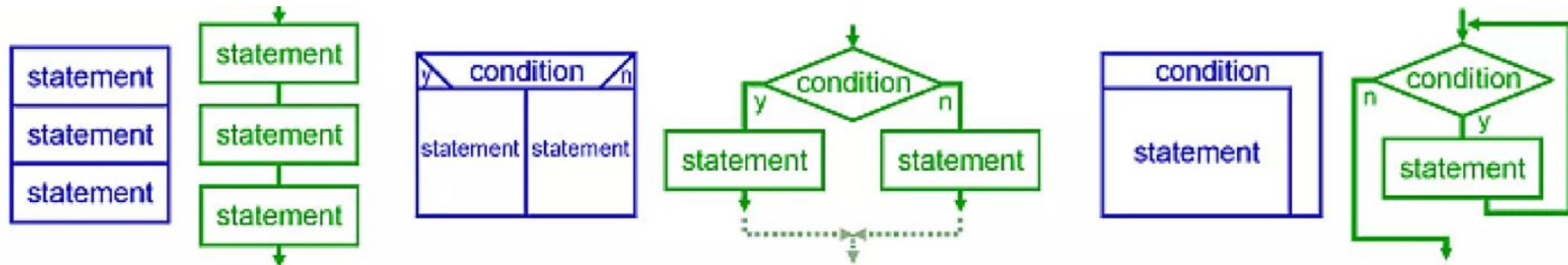


Lenguaje de Programación

- Paradigma de Programación -

Paradigma Estructurado

- Es un enfoque de programación que promueve la mejora de la claridad, calidad y tiempo de desarrollo del software mediante el uso de estructuras de control bien definidas
- La programación estructurada se basa en una metodología de desarrollo de programas llamada refinamientos sucesivos:
 1. Se plantea una operación como un todo y se divide en segmentos más sencillos o de menor complejidad.
 2. Una vez terminado todos los segmentos del programa, se procede a unificar las aplicaciones realizadas por el pool de programadores
- Fue desarrollado para hacer frente a la complejidad y el desorden que se presentaban en los programas escritos en los primeros lenguajes de programación.



Lenguaje de Programación

- Paradigma de Programación -

Paradigma Orientado a Objetos

- La Programación Orientada a Objetos (POO), intenta simular el mundo real a través del significado de objetos que contienen características y funciones.
- Los lenguajes orientados a objetos se clasifican como lenguajes de quinta generación
- Son lenguajes de programación diseñados para resolver problemas utilizando restricciones declarativas y lógica, en lugar de seguir un conjunto de instrucciones previamente definidas

Ejemplo:

Un código representa un carrito de compra

Otro código representa un producto con su precio

Luego, puedo agregarle la responsabilidad al carrito que vaya agregando productos para luego preguntarle el costo total

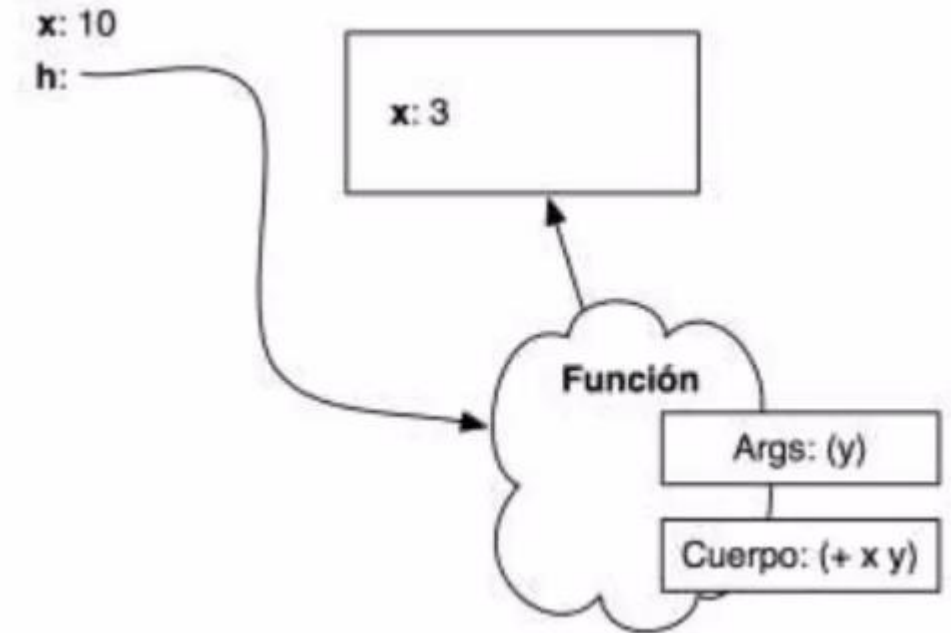


Lenguaje de Programación

- Paradigma de Programación -

Paradigma Funcional

- El paradigma de programación funcional se basa en un concepto muy simple y es el de las **funciones matemáticas**.
- Este paradigma concibe a la computación como la evaluación de funciones matemáticas y evita declarar y cambiar datos.
- En otras palabras, hace hincapie en la **aplicación de las funciones y composición entre ellas**, más que en los cambios de estados y la ejecución secuencial de comandos (como lo hace el paradigma procedimental).
- Permite **resolver ciertos problemas de forma elegante, en cambio** los lenguajes puramente funcionales, evitan los efectos secundarios comunes en otro tipo de programaciones.



Lenguaje de Programación

- Paradigma de Programación -

Paradigma Lógico

- El paradigma lógico de programación es un enfoque que se basa en la **lógica formal para expresar computaciones**.
- En lugar de especificar **cómo** se debe resolver un problema, el programador **describe el problema utilizando hechos, reglas y consultas**.
- La **solución** se encuentra a través de **inferencia lógica**.

Ejemplo de programa lógico

Programa lógico

```
animal(perro).  
animal(gato).  
animal(canguro).  
arbol(palmera).  
flor(margarita).  
  
vegetal(X) :- arbol(X).  
vegetal(X) :- flor(X).
```

Hechos

Reglas

Invocación

```
> vegetal(perro)  
> false  
  
> animal(perro)  
> true  
  
> vegetal(palmera)  
> true
```

Lenguaje de Programación

- Paradigma de Programación -

Multiparadigma... Eso es posible?

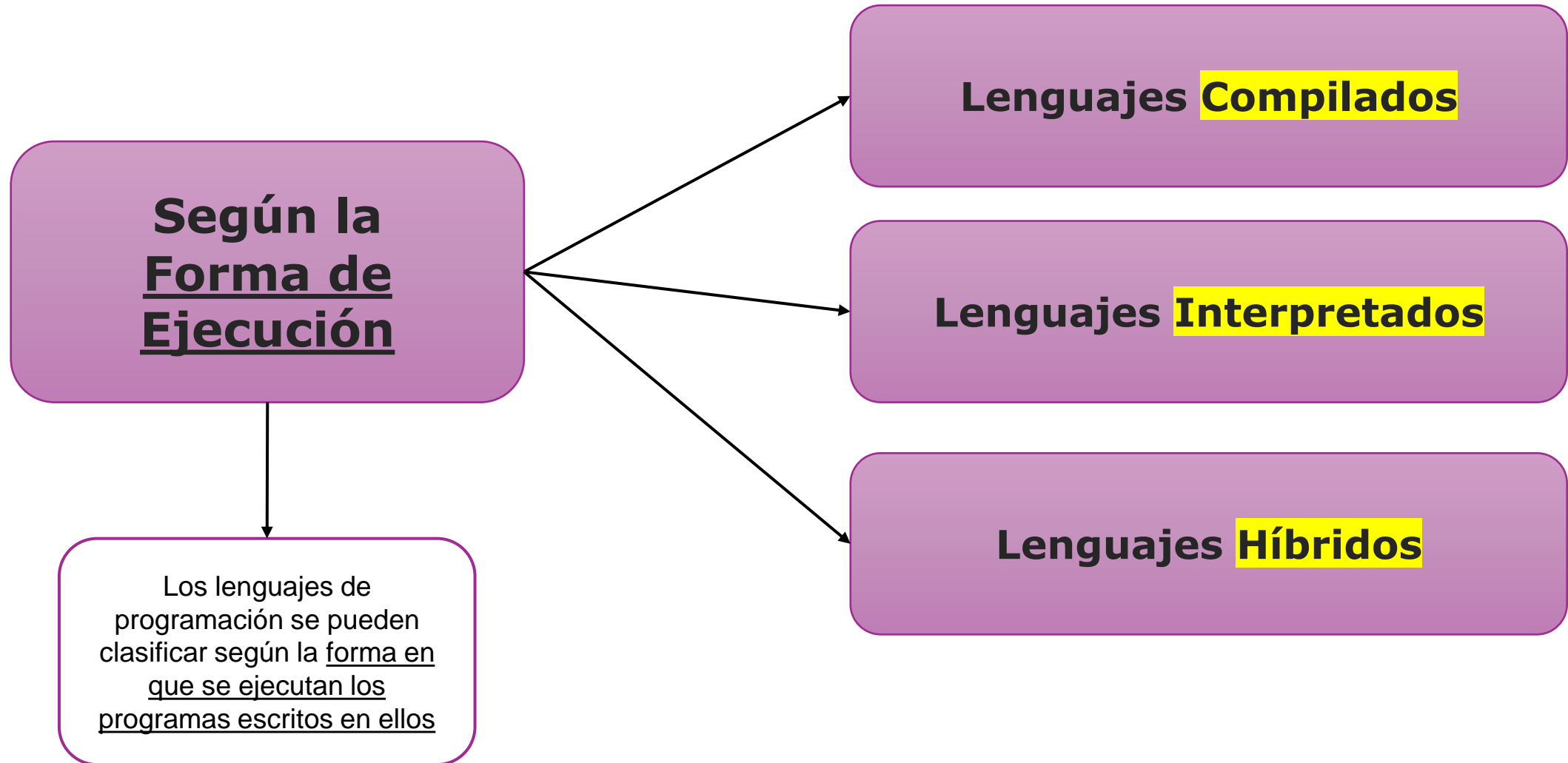
- A lo largo de la evolución de la programación, con nuevos desafíos y paradigmas han habido lenguajes que han modificado su estructura para poder permitir dar soluciones en distintos paradigmas.
- ¿Esto quiere decir que mientras más paradigmas tenga un lenguaje es mejor?
 - No, un lenguaje es una herramienta y hay distintas herramientas para distintas soluciones.
 - Siempre debemos analizar el contexto, tiempos, con que equipo contamos, ¿hay presupuesto? ¿Cuáles son las herramientas que disponemos para trabajar?

Ejemplo:



En JavaScript se puede escribir código tanto con el paradigma estructurado como con programación orientada a objetos e incluso utilizar el paradigma funcional.

Lenguaje de Programación



Lenguaje de Programación

- Forma de Ejecución -

Lenguajes Compilados

- Los lenguajes compilados son aquellos cuyos programas se traducen completamente a código máquina (binario) mediante un compilador antes de ser ejecutados.
- El resultado de la compilación es un archivo ejecutable independiente del código fuente original.
- Características:
 - ✓ Traducción completa a código máquina antes de la ejecución.
 - ✓ Ejecutables independientes del código fuente.
 - ✓ Errores detectados en la fase de compilación.
 - ✓ Rendimiento superior debido a la optimización realizada durante la compilación.
- Ejemplos de Lenguajes Compilados
 - ✓ **C**: Uno de los lenguajes compilados más conocidos y utilizados.
 - ✓ **C++**: Extensión de C con características de programación orientada a objetos.
 - ✓ **Rust**: Lenguaje moderno que enfatiza la seguridad y el rendimiento.



Lenguaje de Programación

- Forma de Ejecución -

Lenguajes Interpretados

- Los lenguajes interpretados son aquellos cuyos programas se ejecutan directamente a partir del código fuente mediante un intérprete, que traduce y ejecuta el código línea por línea.
- Características:
 - ✓ Ejecución Directa: El código fuente se ejecuta directamente, sin necesidad de compilación previa.
 - ✓ Flexibilidad: Permiten una ejecución más flexible y dinámica.
 - ✓ Errores en Tiempo de Ejecución: Se detectan durante la ejecución, lo que puede hacer más difícil la depuración.
- Ejemplos de Lenguajes Interpretados:
 - ✓ **Python**: Popular en desarrollo web, ciencia de datos y scripting.
 - ✓ **JavaScript**: Lenguaje principal para desarrollo web en el lado del cliente.
 - ✓ **Ruby**: Conocido por su simplicidad y productividad, utilizado en desarrollo web.

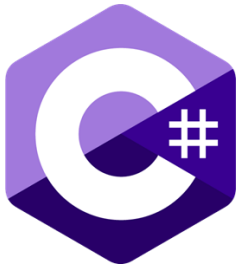


Lenguaje de Programación

- Forma de Ejecución -

Lenguajes Híbridos

- Los lenguajes híbridos combinan características de los lenguajes compilados e interpretados.
- Usan una compilación intermedia para traducir el código fuente a un código intermedio (bytecode) que luego es ejecutado por una máquina virtual o intérprete.
- Características
 - ✓ Compilación a Bytecode: El código fuente se compila a un código intermedio (bytecode) que es independiente de la plataforma.
 - ✓ Ejecución por Máquina Virtual: El bytecode se ejecuta en una máquina virtual que interpreta o compila el bytecode en tiempo de ejecución.
 - ✓ Portabilidad Mejorada: El bytecode puede ejecutarse en cualquier plataforma que tenga la máquina virtual adecuada.
- Ejemplos de Lenguajes Híbridos
 - ✓ **Java**: Compilado a bytecode que se ejecuta en la Máquina Virtual Java (JVM).
 - ✓ **C#**: Compilado a Common Intermediate Language (CIL) que se ejecuta en el Common Language Runtime (CLR).
 - ✓ **Python** (con PyPy o Jython): Puede compilarse a bytecode y ejecutarse en máquinas virtuales específicas



Máquina Virtual:

Es una réplica, en cuanto a comportamiento, de un equipo físico, como una PC, teléfono inteligente o un servidor, etc.

Lenguaje C

Orígenes – Concepto

Lenguaje C

¿Que es?

El lenguaje C es un lenguaje de programación de propósito general, estructurado y de bajo nivel, que fue desarrollado en la década de 1970 por **Dennis Ritchie** en los **Laboratorios Bell**.

Su objetivo principal era **proporcionar una herramienta poderosa y flexible para la creación de sistemas operativos**, y fue utilizado para desarrollar el sistema operativo UNIX, lo que contribuyó a su popularidad y adopción generalizada.



Lenguaje C

Características principales del lenguaje C:

- **Lenguaje de propósito general**: Aunque fue diseñado para la programación de sistemas (especialmente sistemas operativos), su versatilidad le permite ser utilizado en una amplia variedad de aplicaciones, desde software de sistemas hasta programas científicos y de propósito general.
- **Lenguaje estructurado**: C soporta la programación estructurada, lo que implica el uso de estructuras de control como bucles (for, while), condicionales (if, switch), y funciones, que permiten dividir el código en módulos reutilizables y más fáciles de mantener.
- **Cercanía al hardware**: C proporciona acceso directo a la memoria y a las operaciones a nivel de bits, lo que lo hace ideal para tareas que requieren control preciso del hardware. Esta cercanía le otorga eficiencia, haciéndolo adecuado para aplicaciones donde el rendimiento es crucial.
- **Tipado estático**: El lenguaje C es fuertemente tipado, lo que significa que las variables deben declararse con un tipo específico (por ejemplo, int, float, char) y no se pueden cambiar a otros tipos durante la ejecución. Esto ayuda a evitar ciertos errores comunes.
- **Punteros**: Una característica poderosa de C es el uso de punteros, que permiten manipular la dirección de memoria directamente. Esto es útil para optimizar programas, gestionar estructuras de datos dinámicas y acceder a la memoria de forma eficiente.
- **Eficiencia y portabilidad**: C genera código muy eficiente, lo que lo hace ideal para aplicaciones de alto rendimiento. Además, a pesar de ser un lenguaje de bajo nivel, C es portátil, lo que significa que un programa escrito en C puede ser compilado y ejecutado en diversas plataformas con pocas modificaciones.
- **Bibliotecas estándar**: El lenguaje C tiene una biblioteca estándar rica que proporciona funciones para la manipulación de cadenas, matemáticas, manejo de archivos, etc., lo que facilita el desarrollo de aplicaciones complejas sin necesidad de escribir desde cero funcionalidades comunes.

Lenguaje C

Ventajas del lenguaje C:

- **Control del hardware**: Permite un control fino sobre los recursos de la máquina, lo que es esencial en programación de sistemas, controladores y software embebido.
- **Eficiencia**: El código en C tiende a ser rápido y ligero en comparación con otros lenguajes de alto nivel.
- **Base de muchos lenguajes**: Lenguajes modernos como C++, Java, C#, y Python fueron influenciados por la sintaxis y conceptos de C. Aprender C proporciona una base sólida para entender otros lenguajes de programación.
- **Gran comunidad y soporte**: Al ser un lenguaje ampliamente utilizado durante décadas, C cuenta con una gran cantidad de recursos, tutoriales, y bibliotecas que lo hacen accesible.

JAVASCRIPT

Orígenes – Concepto

JAVASCRIPT

Orígenes

- Creado en 1995 por **Brendan Eich** en Netscape.
- Inicialmente llamado **Mocha**, luego **LiveScript**, y finalmente **JavaScript** como estrategia de marketing (para asociarlo con Java).
- **Desarrollado en solo 10 días** para añadir interactividad en las páginas web.
- Estándar ECMAScript introducido en 1997 para evitar fragmentación.
- Competencia con JScript de Microsoft durante la "**guerra de navegadores**".
- Evolución significativa con ECMAScript 3 (1999) y ECMAScript 5 (2009).



JAVASCRIPT

Concepto

- JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript.
- Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

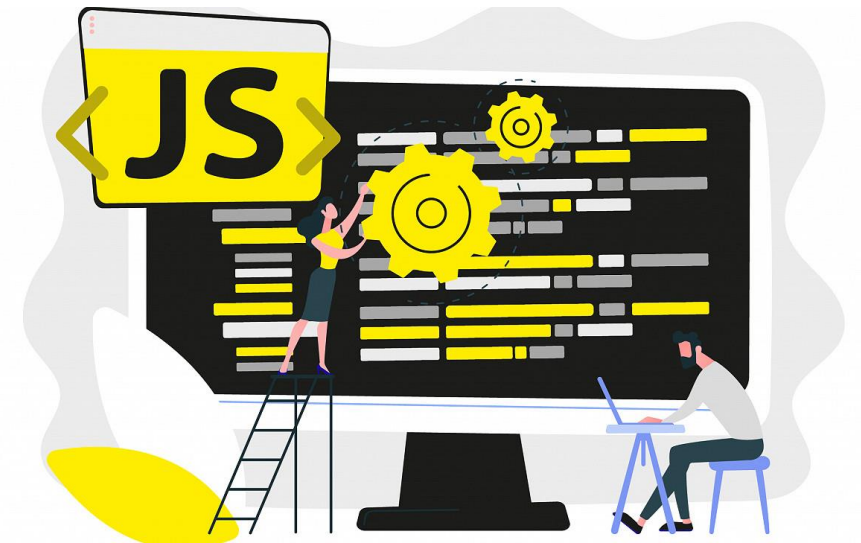
A photograph of a computer screen displaying JavaScript code. The code is written in a dark-themed editor with syntax highlighting. It includes a jQuery function call to attach a 'resize' event, a function named 'cards' that checks the window width and calls 'cardssmallscreen()' or 'cardsbigscreen()', and another function 'cardssmallscreen()' that manipulates a list of cards. Line numbers 40 through 52 are visible on the left side of the code block.

```
40
41 $(function){cards();});
42 $(window).on('resize', function(){cards();});
43 function cards(){
44   var width = $(window).width();
45   if(width < 750){
46     cardssmallscreen();
47   }else{
48     cardsbigscreen();
49   }
50 }
51 function cardssmallscreen(){
52   var cards = $(''.card').length;
```

JAVASCRIPT

Características Claves

- **Lenguaje de Scripting del Lado del Cliente:** JavaScript se ejecuta en el navegador del usuario, permitiendo la interacción y manipulación de elementos HTML y CSS en tiempo real.
- **Interactividad:** Añade comportamiento interactivo a las páginas web, como responder a eventos de usuario (clics, movimientos del ratón, entradas de teclado).
- **Versatilidad:** Aunque inicialmente se diseñó para el navegador, JavaScript ahora también se utiliza en el lado del servidor gracias a Node.js, lo que permite crear aplicaciones completas con un solo lenguaje.
- **Orientado a Objetos y Basado en Prototipos:** Permite la creación de objetos y la reutilización de código mediante herencia prototípica.
- **Lenguaje Interpretado:** No requiere compilación, ya que el navegador interpreta y ejecuta el código directamente.
- **Estandarizado:** Se rige por el estándar ECMAScript, garantizando que el código JavaScript funcione de manera consistente en diferentes navegadores y entornos.



JAVASCRIPT



¿Por que es bueno comenzar con JavaScript si quiero estudiar programación?

JavaScript es una excelente opción para comenzar a estudiar programación por varias razones clave:

1. Fácil de aprender y usar

- **Sintaxis sencilla:** JavaScript tiene una sintaxis relativamente fácil de entender, lo que lo convierte en un buen lenguaje para principiantes.
- **Lenguaje interpretado:** No necesitas instalar herramientas complejas. Un navegador y un editor de texto son suficientes para comenzar.

2. Uso en front-end y back-end (full-stack)

- **Front-end:** JavaScript es el lenguaje principal del lado del cliente (front-end). Permite crear interfaces interactivas, manipular elementos HTML y CSS, y mejorar la experiencia del usuario en el navegador.
- **Back-end:** Con el marco Node.js, JavaScript también se puede utilizar en el lado del servidor (back-end). Esto significa que puedes gestionar bases de datos, autenticar usuarios, manejar peticiones HTTP, entre otras tareas.

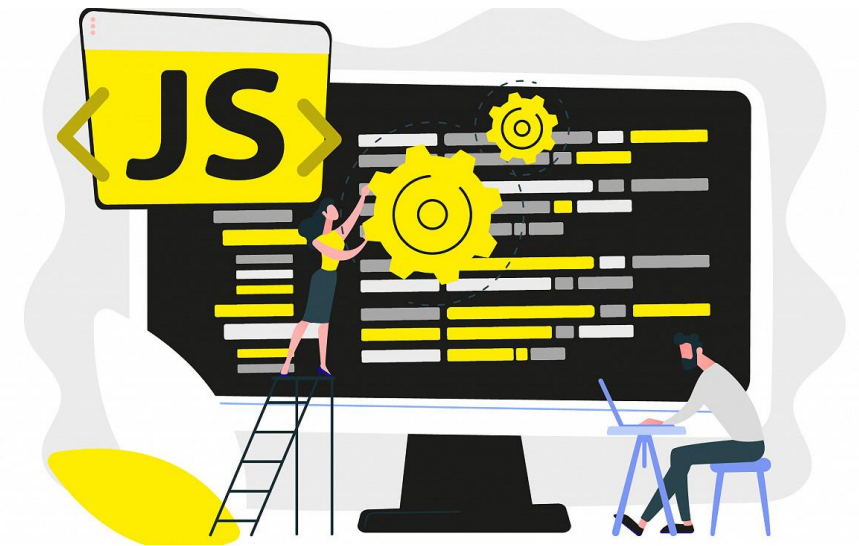
Usar el mismo lenguaje en ambos entornos permite a los desarrolladores trabajar como **full-stack developers**, reduciendo la curva de aprendizaje y facilitando la comunicación entre las partes de un proyecto.

JAVASCRIPT

¿Por que es bueno comenzar con JavaScript si quiero estudiar programación?

3. Gran comunidad y recursos

- **Amplio soporte:** Al ser uno de los lenguajes más populares, hay una gran cantidad de tutoriales, bibliotecas, foros y herramientas para aprender y mejorar tus habilidades.
- **Ecosistema dinámico:** Herramientas como React, Angular, y Vue.js en el front-end, o Express.js en el back-end, han sido construidas sobre JavaScript, lo que lo convierte en un lenguaje esencial en el desarrollo web.



Intérprete de código - Compilador de código

Intérprete de código

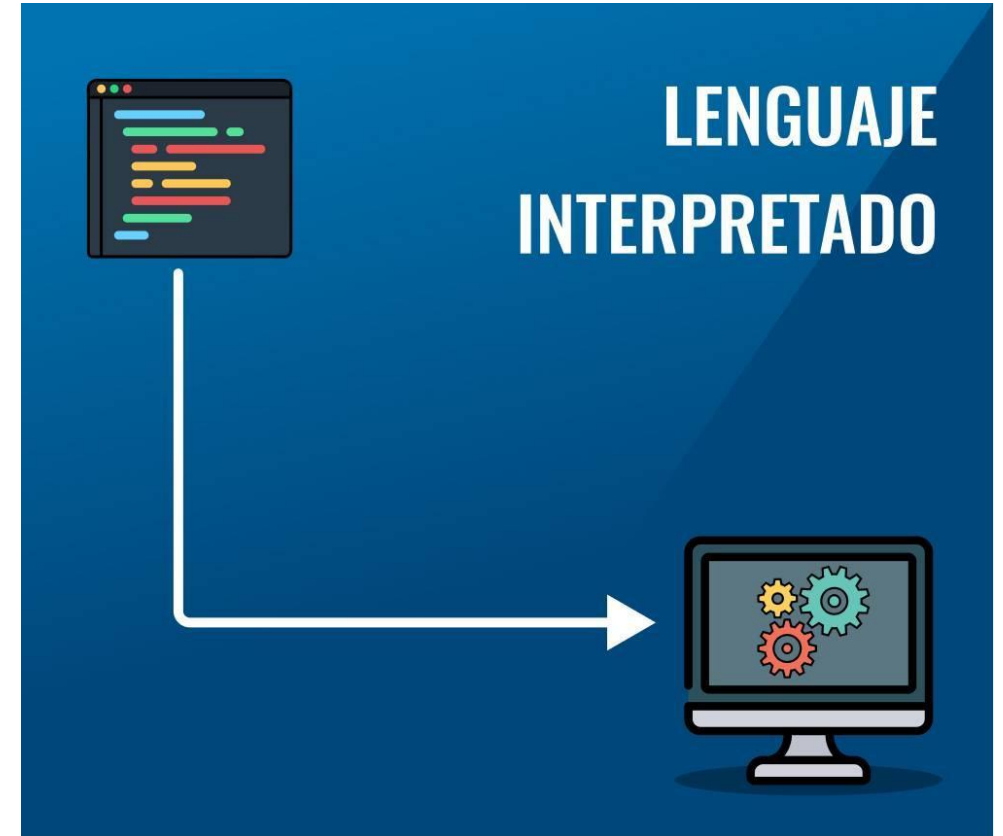
Un **intérprete** es un programa que ejecuta código fuente línea por línea. No traduce el código completo a lenguaje máquina antes de ejecutarlo, sino que lo interpreta y lo ejecuta directamente.

Características

- Ejecución inmediata: Ejecutan el código línea por línea a medida que lo leen.
- No generan archivos ejecutables: No producen un archivo independiente que se pueda ejecutar por sí mismo.
- Buena para desarrollo y pruebas: Permiten una ejecución rápida y fácil de probar el código sin necesidad de una fase de compilación.

Ejemplos:

- Python: El lenguaje Python utiliza un intérprete (CPython es la implementación más común).
- Ruby: Ruby también es un lenguaje interpretado.
- JavaScript: A menudo se interpreta en el navegador web.



Compilador de código

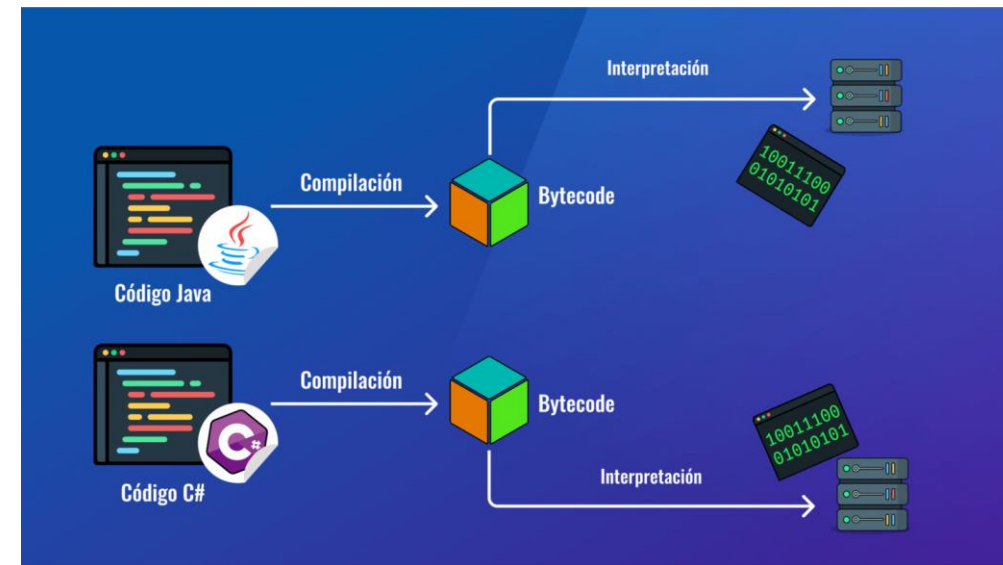
Un **compilador** es un programa que traduce el código fuente completo a lenguaje máquina antes de que se ejecute. El resultado es un archivo binario ejecutable que puede ejecutarse independientemente del código fuente.

Características

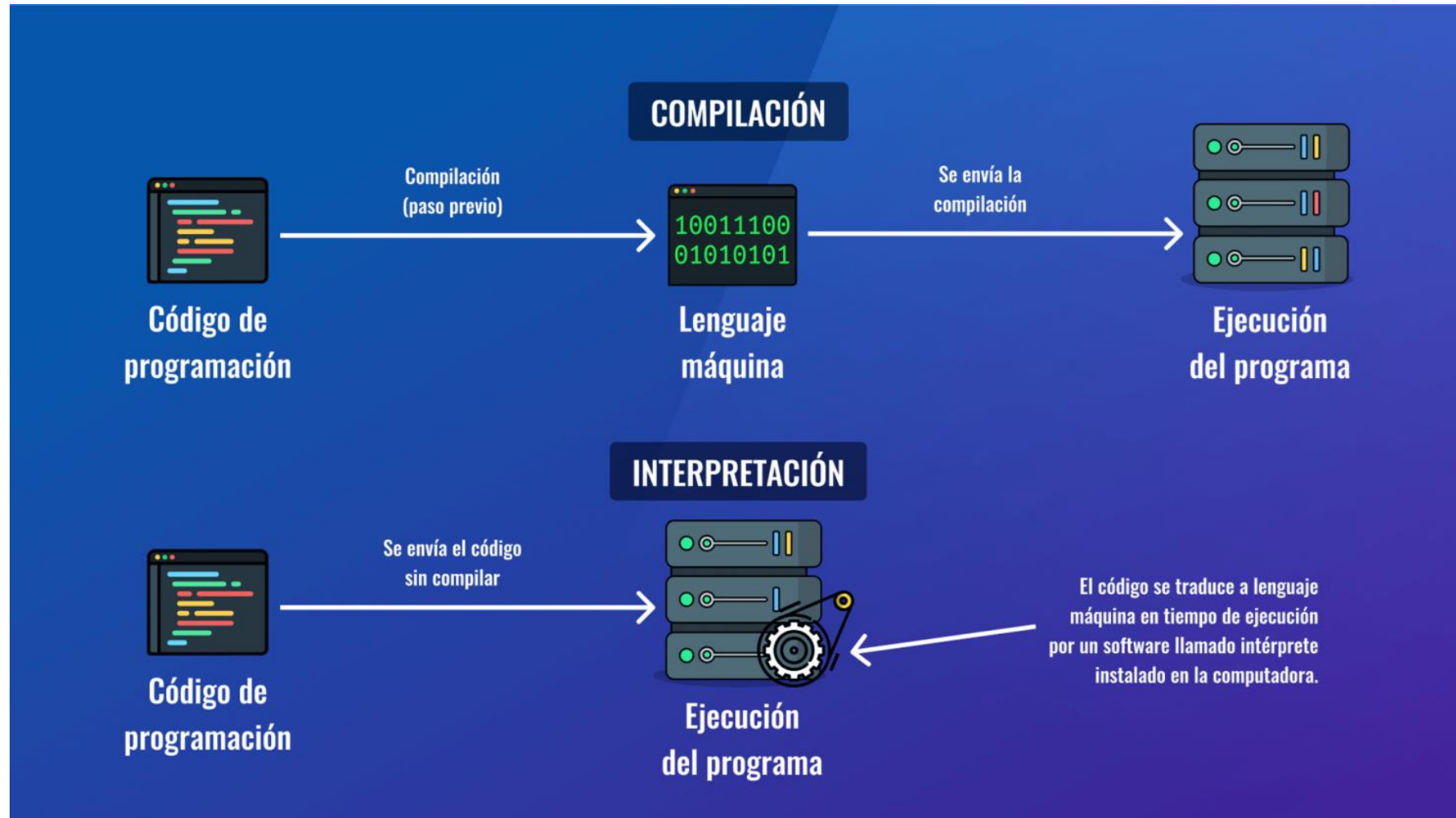
- Traducción completa antes de la ejecución:
- Compilan todo el código fuente a un archivo binario ejecutable.
- Generan archivos ejecutables: Producen un archivo que puede ejecutarse por sí mismo.
- Optimización: Pueden optimizar el código durante la compilación para mejorar la eficiencia

Ejemplos:

- GCC (GNU Compiler Collection): Utilizado para compilar C, C++, y otros lenguajes.
- javac: Compilador de Java que traduce el código fuente a bytecode.
- Clang: Un compilador para lenguajes C, C++, y Objective-C.



Intérprete de código y Compilador de código



El Motor de JavaScript

El Motor de JavaScript

El código JavaScript que escribimos no puede ser entendido por las computadoras.

Es por eso que un motor JavaScript es un programa que convierte el código JavaScript que los desarrolladores escriben en un código de máquina que permite que una computadora realice tareas específicas.

Los motores (o intérpretes) de JavaScript suelen ser desarrollados por proveedores de navegadores web

Ejemplos:

- **V8:** motor JavaScript de código abierto desarrollado por Google para Chrome.
- **SpiderMonkey:** El motor JavaScript de Mozilla Firefox
- **JavascriptCore:** Motor de código abierto desarrollado por Apple para Safari
- **Chakra:** Motor desarrollado para Microsoft Edge original (La última versión de Edge usa V8)



El Motor de JavaScript: V8

- V8 es el motor de JavaScript utilizado por **Google Chrome** y es también el motor que ejecuta el código JavaScript en **Node.js**.
- La integración de V8 en Node.js permite que el código JavaScript se ejecute de manera eficiente y rápida en el servidor, aprovechando las capacidades de compilación y optimización de V8.
- Esta estrecha integración es lo que hace que V8 sea una parte esencial del núcleo de Node.js.
- Para ver todas sus **especificaciones y documentación** nos podemos dirigir a la pagina: <https://v8.dev/>



The screenshot shows the V8 website with a blue header containing the V8 logo and navigation links: Home, Blog, Docs, Tools, JS/Wasm features, and Research. The main content area has a dark background and includes a 'What is V8?' section, a list of 'Latest posts and feature explainers' with links and dates, and a footer with legal information and a large V8 logo.


What is V8?

V8 is Google's open source high-performance JavaScript and WebAssembly engine, written in C++. It is used in Chrome and in Node.js, among others. It implements [ECMAScript](#) and [WebAssembly](#), and runs on Windows, macOS, and Linux systems that use x64, IA-32, or ARM processors. V8 can be embedded into any C++ application.

Latest posts and feature explainers

- 194. [WebAssembly JSPI has a new API](#) 04 June 2024 [WebAssembly](#)
- 193. [The V8 Sandbox](#) 04 April 2024 [security](#)
- 192. [Iterator helpers](#) 27 March 2024 [ECMAScript](#)
- 191. [WebAssembly JSPI is going to origin trial](#) 06 March 2024 [WebAssembly](#)
- 190. [Static Roots: Objects with Compile-Time Constant Addresses](#) 05 February 2024 [JavaScript](#)
- 189. [Import attributes](#) 31 January 2024 [ECMAScript](#)
- 188. [V8 is Faster and Safer than Ever!](#) 14 December 2023 [JavaScript](#) [WebAssembly](#) [security](#) [benchmarks](#)
- 187. [Maglev - V8's Fastest Optimizing JIT](#) 05 December 2023 [JavaScript](#)
- 186. [A new way to bring garbage collected programming languages efficiently to WebAssembly](#) 01 November 2023 [WebAssembly](#)
- 185. [Control-flow Integrity in V8](#) 09 October 2023 [security](#)

More articles can be found in [the blog archive](#) and [the features section](#).

[Branding](#) · [Terms](#) · [Privacy](#) · [Twitter](#) · [Edit this page on GitHub](#)  [Light Theme](#)

Except as otherwise noted, any code samples from the V8 project are licensed under [V8's BSD-style license](#). Other content on this page is licensed under [the Creative Commons Attribution 3.0 License](#).

For details, see [our site policies](#).

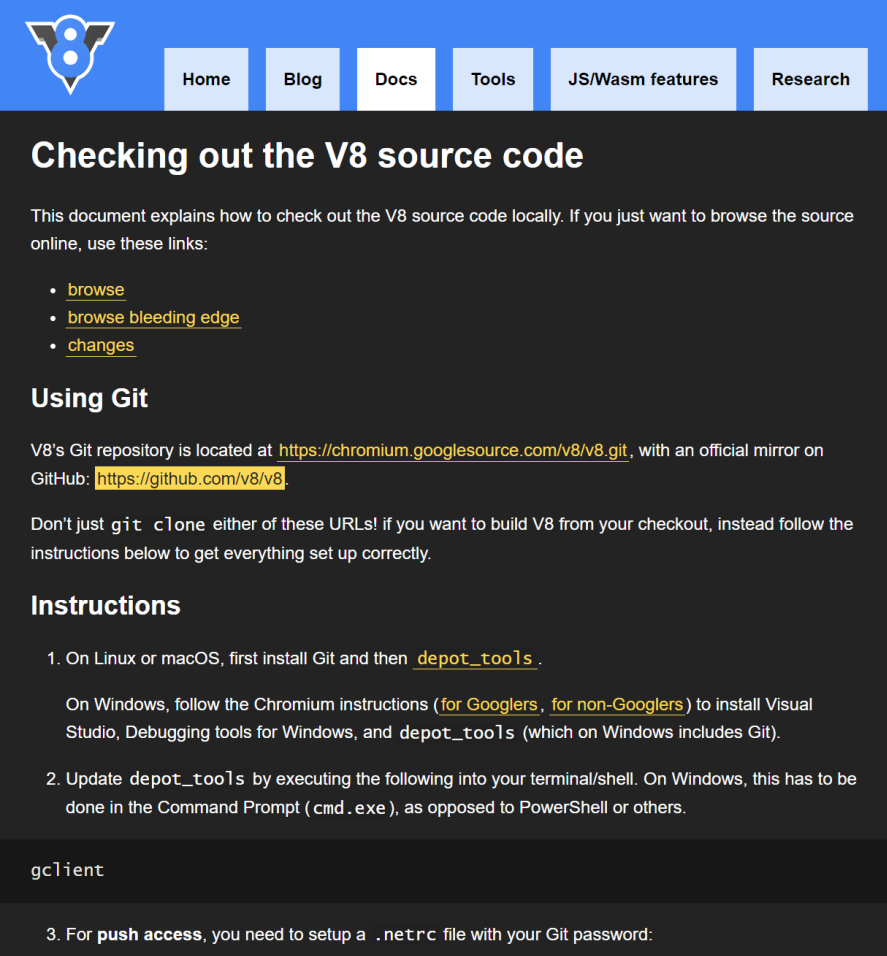
El Motor de JavaScript: V8

Características :

- **Compilación Just-In-Time (JIT):** V8 utiliza técnicas de compilación JIT para convertir el código JavaScript a código máquina nativo antes de ejecutarlo, lo que mejora significativamente el rendimiento.
- **Gestión de memoria eficiente:** V8 incluye un recolector de basura que maneja la memoria automáticamente, liberando recursos que ya no se usan.
- **Optimización agresiva:** V8 aplica varias técnicas de optimización para mejorar la velocidad de ejecución, como la eliminación de código muerto y la inlining (inserción de funciones pequeñas directamente en el código que las llama).
- **Integración con WebAssembly:** V8 también puede ejecutar código WebAssembly, lo que permite la ejecución de código de bajo nivel en el navegador.

Componentes:

- **Parser:** Convierte el código JavaScript en una representación intermedia llamada Abstract Syntax Tree (AST).
- **Ignition:** Es el intérprete de V8 que ejecuta el bytecode generado a partir del AST.
- **TurboFan:** Es el compilador de optimización de V8 que convierte el bytecode en código máquina altamente optimizado.



The screenshot shows the V8 source code documentation page. At the top is a blue header with the V8 logo and navigation links: Home, Blog, Docs, Tools, JS/Wasm features, and Research. The main content area has a dark background. The title 'Checking out the V8 source code' is in white. Below it, a paragraph explains how to check out the source code locally or online. A list of links includes 'browse', 'browse bleeding edge', and 'changes'. The 'Using Git' section provides the Git repository URL and a mirror on GitHub. It also includes instructions on how to build V8 from the checkout. The 'Instructions' section lists steps for installing Git and depot_tools on Linux/macOS and Windows. A code block shows the command 'gclient'. The final step mentions setting up a .netrc file for push access.

Checking out the V8 source code

This document explains how to check out the V8 source code locally. If you just want to browse the source online, use these links:

- [browse](#)
- [browse bleeding edge](#)
- [changes](#)

Using Git

V8's Git repository is located at <https://chromium.googlesource.com/v8/v8.git>, with an official mirror on GitHub: <https://github.com/v8/v8>.

Don't just `git clone` either of these URLs! If you want to build V8 from your checkout, instead follow the instructions below to get everything set up correctly.

Instructions

1. On Linux or macOS, first install Git and then [depot_tools](#).

On Windows, follow the Chromium instructions ([for Googlers](#), [for non-Googlers](#)) to install Visual Studio, Debugging tools for Windows, and `depot_tools` (which on Windows includes Git).

2. Update `depot_tools` by executing the following into your terminal/shell. On Windows, this has to be done in the Command Prompt (`cmd.exe`), as opposed to PowerShell or others.

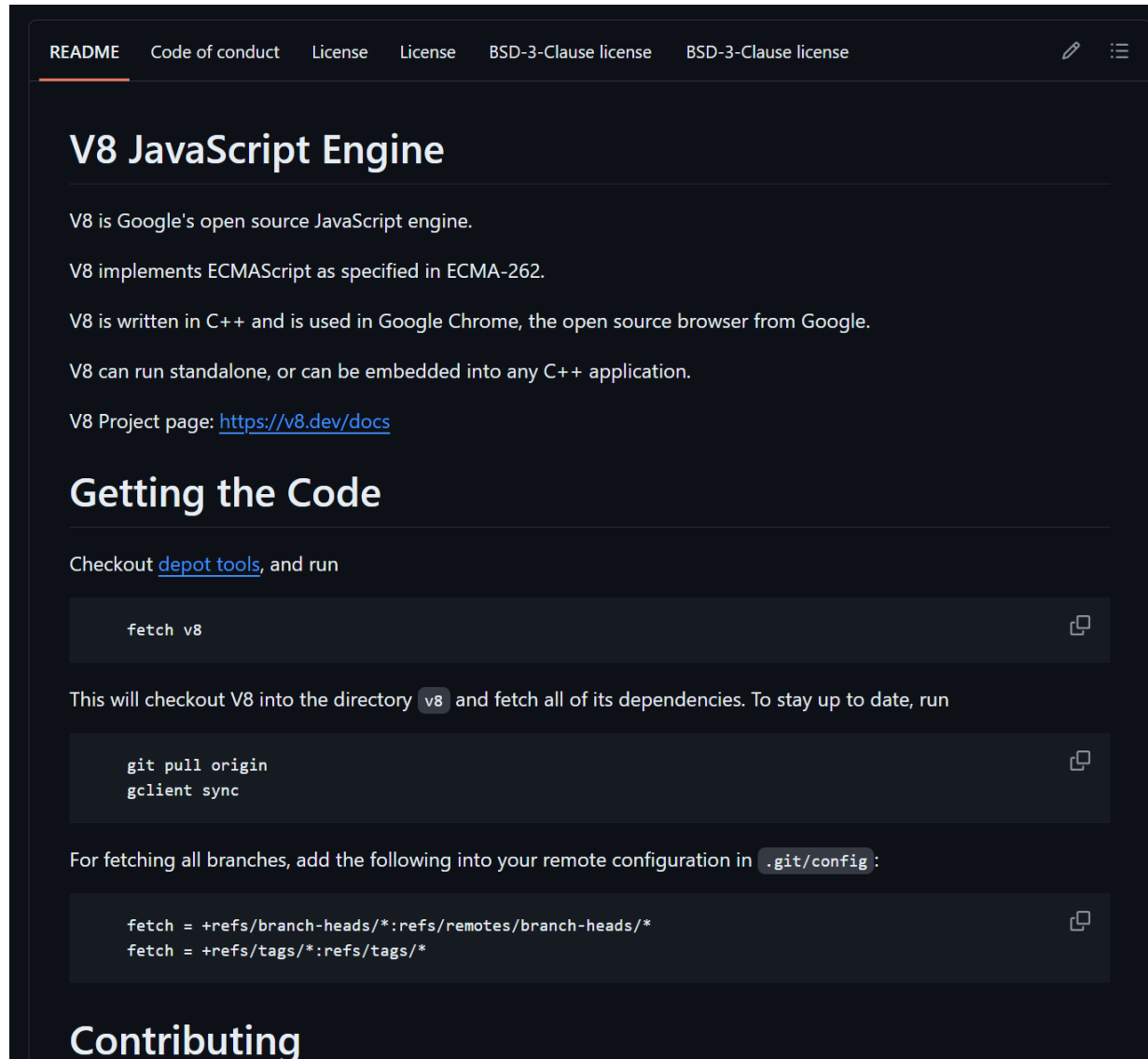
```
gclient
```

3. For **push access**, you need to setup a `.netrc` file with your Git password:

El Motor de JavaScript: V8

Especificaciones:

- El motor V8 de Chrome de Google se encuentra en el núcleo de NodeJs.
- Al incorporar V8 en su propia aplicación C++, puede escribir código C++ que se ejecuta cuando un usuario escribe código JavaScript.
- Puede agregar nuevas características al mismo JavaScript.
- Dado que C++ es ideal para operaciones de nivel inferior como manejo de archivos, conexiones de bases de datos y operaciones de red, al incorporar V8 en su propio programa, C++ tiene el poder de agregar toda esa funcionalidad en JavaScript.
- EL programa C++ del que estamos hablando es NodeJs (No olvidar)



The image is a screenshot of the V8 JavaScript Engine README page on GitHub. The page has a dark theme. At the top, there are navigation links: README (highlighted), Code of conduct, License, License, BSD-3-Clause license, and BSD-3-Clause license. The main heading is "V8 JavaScript Engine". Below it, the text says: "V8 is Google's open source JavaScript engine.", "V8 implements ECMAScript as specified in ECMA-262.", "V8 is written in C++ and is used in Google Chrome, the open source browser from Google.", "V8 can run standalone, or can be embedded into any C++ application.", and "V8 Project page: <https://v8.dev/docs>". The next section is "Getting the Code". It says "Checkout [depot tools](#), and run" followed by a code block:

```
fetch v8
```

. Below this, it says "This will checkout V8 into the directory `v8` and fetch all of its dependencies. To stay up to date, run" followed by a code block:

```
git pull origin
gclient sync
```

. Then it says "For fetching all branches, add the following into your remote configuration in `.git/config`:" followed by a code block:

```
fetch = +refs/branch-heads/*:refs/remotes/branch-heads/*
fetch = +refs/tags/*:refs/tags/*
```

. The final section is "Contributing".

README Code of conduct License License BSD-3-Clause license BSD-3-Clause license

V8 JavaScript Engine

V8 is Google's open source JavaScript engine.

V8 implements ECMAScript as specified in ECMA-262.

V8 is written in C++ and is used in Google Chrome, the open source browser from Google.

V8 can run standalone, or can be embedded into any C++ application.

V8 Project page: <https://v8.dev/docs>

Getting the Code

Checkout [depot tools](#), and run

```
fetch v8
```

This will checkout V8 into the directory `v8` and fetch all of its dependencies. To stay up to date, run

```
git pull origin
gclient sync
```

For fetching all branches, add the following into your remote configuration in `.git/config`:

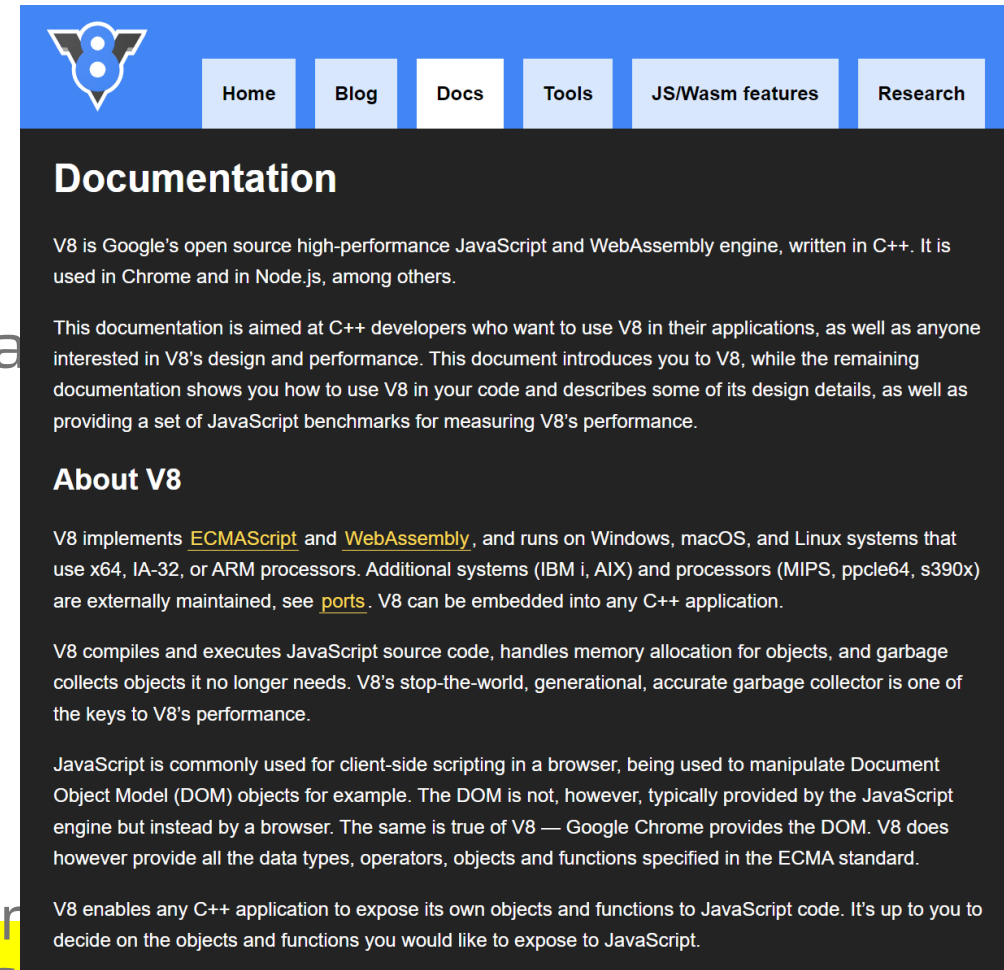
```
fetch = +refs/branch-heads/*:refs/remotes/branch-heads/*
fetch = +refs/tags/*:refs/tags/*
```

Contributing

El Motor de JavaScript: V8

Resumen:

- Un motor de JavaScript es un programa que ejecuta código JavaScript.
- En 2008, Google creó su propio motor de JavaScript llamado V8.
- V8 está escrito en C++ y se puede usar de forma independiente o se puede integrar en otro programa en C++
- Eso le permite escribir programas que hacen todo lo que V8 puede hacer y más.
- Esto significa que tu programa puede ejecutar ECMAScript y características adicionales que elijas incorporar
- Por ejemplo, funciones que están disponibles en C++ pero no en JavaScript. Esta es la idea detrás de V8.



The screenshot shows the V8 Documentation website. The header is blue with the V8 logo on the left and navigation links: Home, Blog, Docs, Tools, JS/Wasm features, and Research. The main content area is dark gray. The 'Documentation' section includes an introductory paragraph about V8 being Google's open-source engine, followed by a paragraph about the documentation's target audience. The 'About V8' section describes the engine's implementation of ECMAScript and WebAssembly, its supported platforms, and its garbage collection. A final paragraph explains how V8 integrates with JavaScript in browsers and C++ applications.

Documentation

V8 is Google's open source high-performance JavaScript and WebAssembly engine, written in C++. It is used in Chrome and in Node.js, among others.

This documentation is aimed at C++ developers who want to use V8 in their applications, as well as anyone interested in V8's design and performance. This document introduces you to V8, while the remaining documentation shows you how to use V8 in your code and describes some of its design details, as well as providing a set of JavaScript benchmarks for measuring V8's performance.

About V8

V8 implements [ECMAScript](#) and [WebAssembly](#), and runs on Windows, macOS, and Linux systems that use x64, IA-32, or ARM processors. Additional systems (IBM i, AIX) and processors (MIPS, ppc64, s390x) are externally maintained, see [ports](#). V8 can be embedded into any C++ application.

V8 compiles and executes JavaScript source code, handles memory allocation for objects, and garbage collects objects it no longer needs. V8's stop-the-world, generational, accurate garbage collector is one of the keys to V8's performance.

JavaScript is commonly used for client-side scripting in a browser, being used to manipulate Document Object Model (DOM) objects for example. The DOM is not, however, typically provided by the JavaScript engine but instead by a browser. The same is true of V8 — Google Chrome provides the DOM. V8 does however provide all the data types, operators, objects and functions specified in the ECMA standard.

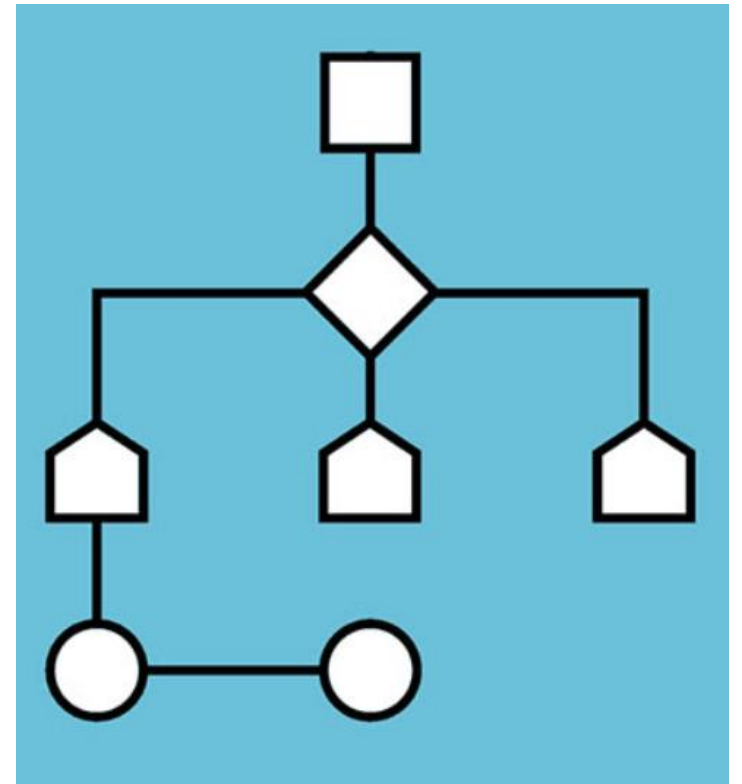
V8 enables any C++ application to expose its own objects and functions to JavaScript code. It's up to you to decide on the objects and functions you would like to expose to JavaScript.

Algoritmos

¿Qué es un Algoritmo?

Concepto

- La palabra **algoritmo** viene del sobrenombre del célebre matemático árabe del siglo I Muhammad ben Musa Al-Khwarizmi.
- Un algoritmo es un conjunto de instrucciones secuenciales que permiten la resolución de un problema.



Algoritmos | Ejemplo práctico

Realizar un algoritmo que permita la suma de dos números, y que imprima en pantalla el resultado indicando si es par o impar.

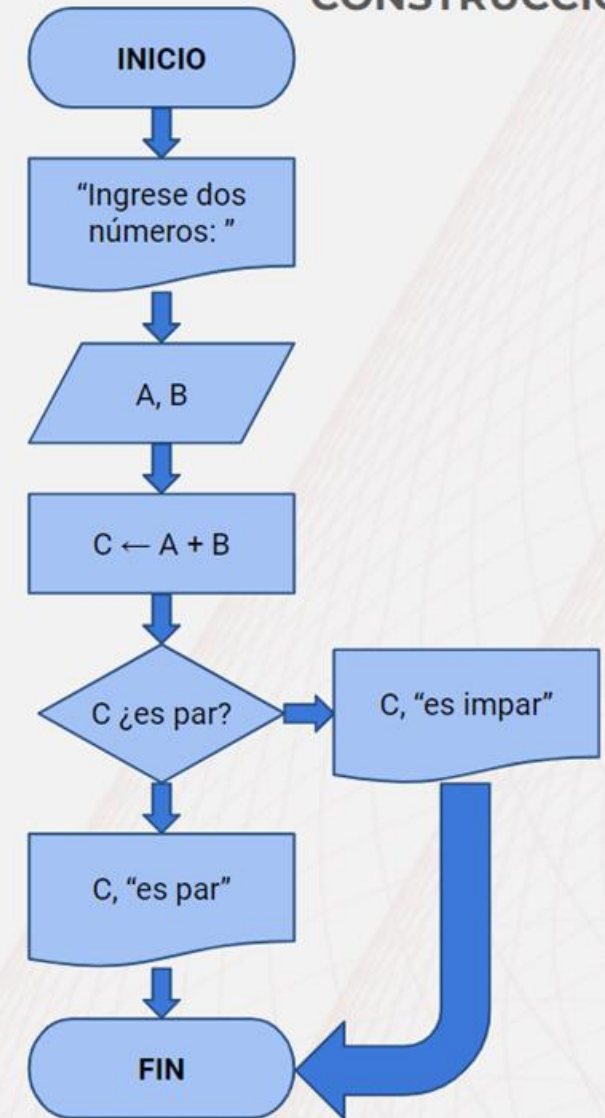
ANÁLISIS

- 2 números.
- 2 variables de entrada.
- 1 variable.
- $C = A + B$

PRUEBA DE ESCRITORIO

1. INICIO
2. 3, 5
3. $A = 3, B = 5$
4. $C = 3 + 5$
5. Verdadero
6. 8 es par
7. FIN

CONSTRUCCIÓN



¿Qué es un Algoritmo?

¿Por qué usamos algoritmos en la programación?

- Usamos algoritmos en la programación porque son esenciales para resolver problemas de manera eficiente y estructurada.
- Un algoritmo es simplemente una **secuencia de pasos bien definidos que resuelven un problema específico o ejecutan una tarea.**



Algoritmo

Características claves de los Algoritmos

1. **Resolución eficiente de problemas:** Los algoritmos nos permiten encontrar soluciones óptimas para problemas específicos. Un buen algoritmo puede ahorrar tiempo y recursos al ejecutar tareas de manera más rápida y con menos consumo de memoria.
2. **Claridad y estructura:** Un algoritmo organiza el proceso de resolución de un problema en pasos lógicos, lo que facilita la comprensión y depuración del código. Permite dividir problemas complejos en partes más manejables y comprensibles.
3. **Reutilización y adaptación:** Una vez que se diseña un buen algoritmo, se puede reutilizar en múltiples contextos. Por ejemplo, los algoritmos de búsqueda y ordenación se aplican en muchas áreas, desde bases de datos hasta procesamiento de datos.
4. **Escalabilidad:** Los algoritmos ayudan a que los programas sean escalables. Es decir, permiten manejar grandes cantidades de datos o usuarios sin perder eficiencia. Un algoritmo mal diseñado puede funcionar bien en pequeñas cantidades de datos, pero ser ineficiente a gran escala.
5. **Toma de decisiones automatizada:** Los algoritmos permiten que las computadoras tomen decisiones de manera automática, siguiendo reglas predefinidas. Esto es esencial en tareas como la inteligencia artificial, donde los algoritmos determinan cómo responder o actuar en función de la información disponible.



**Momento de
poner a prueba
lo aprendido!**