

# Técnicas de Programación

## Carrera Programador full-stack

*Breve Evolución de las Computadoras*

# Breve Evolución de las Computadoras

## *Primera Generación*

- Década del 50
- Maquinas grandes y costosas
- Construidas con válvulas de vacío



# Breve Evolución de las Computadoras

## *Segunda Generación*

- Se reduce su tamaño y crece el poder de procesamiento
- Empieza a definirse la forma de comunicarse con la computadora (lenguaje)
- Construidas con transistores



# Breve Evolución de las Computadoras

## *Tercera Generación*

- Década del 70
- Se manejan por medio de los lenguajes de control de los sistemas operativos
- Construidas con circuitos integrados



# Breve Evolución de las Computadoras

## *Cuarta Generación*

- Aparecen los microprocesadores
- Surgen las computadoras personales
- Surgen los procesadores de texto, hojas de cálculo, etc.



# Breve Evolución de las Computadoras

## *Quinta Generación*

- Procesamiento en paralelo
- Manejo de lenguaje natural
- Inteligencia artificial



# Técnicas de Programación

## Carrera Programador full-stack

*Introducción al Módulo*

# Técnicas de Programación

## *Objetivo*

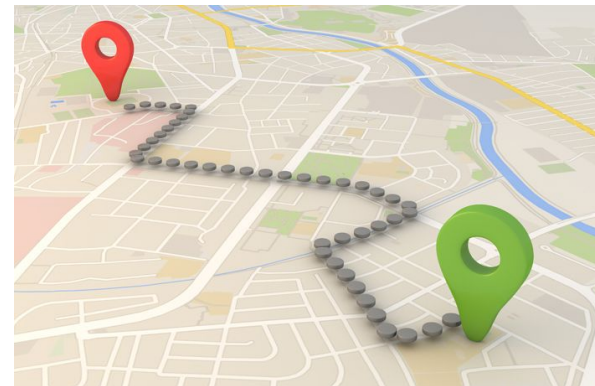
- Interpretar las **especificaciones** de diseño o requisitos de las asignaciones a programar
- Comprendiendo en su contexto inmediato cuál es el **problema a resolver**
- Determinar el **alcance del problema** y convalidar su interpretación a fin de identificar aspectos faltantes.
- **Desarrollar algoritmos** que dan soluciones a los problemas asignados o derivados de los mismos.



# Técnicas de Programación

## *Principales Temas*

- Secuencia
- Condicionales
- Ciclos
- Métodos y parámetros
- Arreglos
- Matrices
- Métodos de ordenamiento



# El Programador Web

La web fue evolucionando rápidamente a nivel de desarrollo, el webmaster multiusos, paso a dividirse en dos grandes y muy diferenciados roles:

- **Diseño**
  - Encargado de hacer los diseños básicos.
  - En ocasiones también se encargaba de animaciones y transiciones.
- **Programación**
  - Realizaba todas las tareas de desarrollo: JavaScript, PHP, Bases de datos, formularios, hosting, etc...

Las webs de entonces no eran muy complejas, gran parte de la lógica se hacía en el servidor y el verdadero reto era lograr los objetivos con la tecnología de la época.

Al evolucionar la web, su complejidad creció exponencialmente. Por eso, la programación se dividió en dos grandes áreas: Front End y Back End.

# El Programador Web

- **Diseñador/Maquetador**
  - Antes con el diseño era suficiente. Luego, el diseñador asume competencias básicas para convertir los diseños en HTML y CSS.
- **Front-End developer**
  - Desarrolladores que asumen las funciones de interacción del lado del cliente (JavaScript) y dejando el servidor. En ocasiones, el diseño quedará fuera de sus competencias.
- **Back-End developer**
  - El desarrollo en el servidor también sufre muchos cambios. Poco a poco, se migrará de proyectos web que basan la mayor parte de su programación en HTML, CSS y JavaScript, desde el servidor a la creación de APIs (especificación y código para que las aplicaciones puedan comunicarse entre ellas).
- **Full Stack Developer**
  - Surge una nueva clase de desarrolladores, que no se encasillan en el back o en el front. Son capaces de adentrarse en ambos mundos y suplir las necesidades de los equipos en estos dos frentes.
  - Cada Full Stack Developer será diferente, cada uno será especialista en unas áreas, y en otras pasará de largo.

# Técnicas de Programación

## Carrera Programador full-stack

*Conceptos Fundamentales*

# Software

- Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación



# ¿Qué es Programar?

- Programar es un arte y el programador debería ser un artesano.
- Las máquinas y los sistemas son geniales haciendo una única cosa, seguir pasos...La responsabilidad de todo programador en relación a las máquinas es ser capaz de guiarlas con las instrucciones más precisas..

# ¿Qué es Programar?

- Usar vs. Controlar
- Crear Programas
  - Acciones (comandos)
- Solucionar problemas

```
96      .</p>
97      <p><a class="btn btn-lg btn-primary" href="#" role="button">Sign up now</a>
98      </div>
99      </div>
100     </div>
101     <a class="left carousel-control" href="#myCarousel" role="button" data-ride="carousel">
102       <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>
103       <span class="sr-only">Previous</span>
104     </a>
105     <a class="right carousel-control" href="#myCarousel" role="button" data-ride="carousel">
106       <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
107       <span class="sr-only">Next</span>
108     </a>
109   </div><!-- /.carousel -->
110
111   <!--Featured Content Section-->
112   <div class="container">
113     <div class="row">
114       <div class="col-md-4"></div>
115       <div class="col-md-4"></div>
116       <div class="col-md-4"></div>
117     </div>
118   </div>
```

# Lenguajes de Programación

- Lenguaje especial para desarrollar programas
- Hay muchos lenguajes según lo que queramos hacer
  - Desarrollo de aplicaciones y juegos: C, C++, Java
  - Bases de datos: MySQL, SQL
  - Drivers: Assembler, C
  - Web: HTML, JavaScript, Python, PHP





# Lenguajes de Programación

- Los programas están formados por secuencias de **instrucciones**
- Las instrucciones están escritas para que la computadora realice una **tarea específica**
- La secuencia de instrucciones son escritas por un programador usando un lenguaje de programación



# Lenguaje de Programación

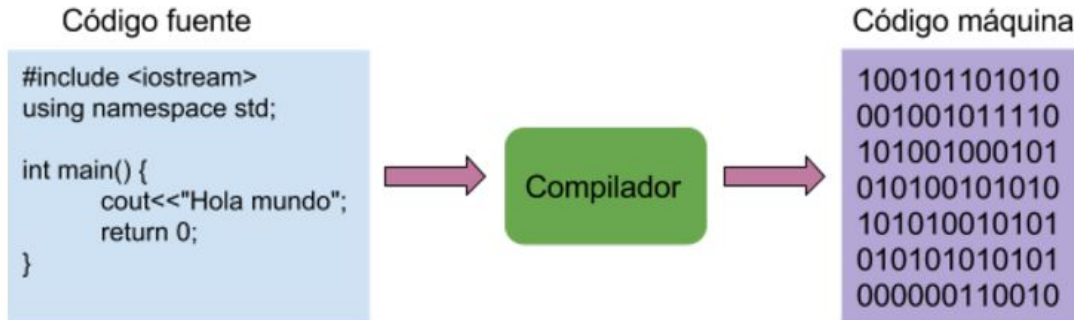
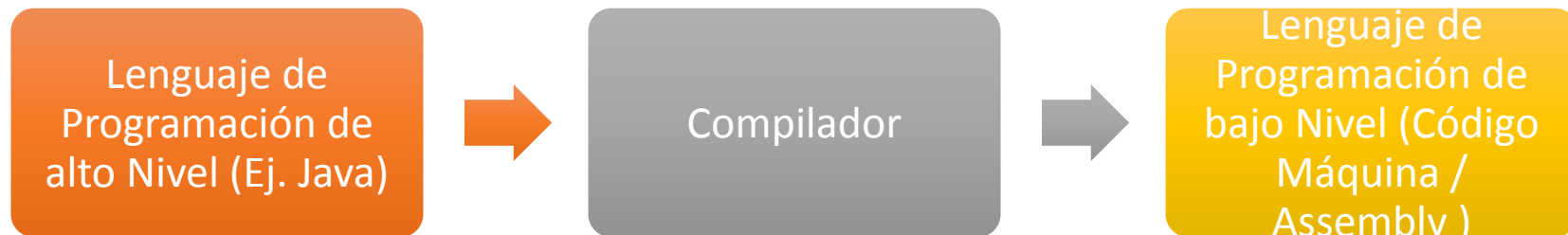
## *Lenguaje de Máquina*

- El lenguaje máquina está compuesto de ceros y unos lo que hace que programar en lenguaje máquina sea un proceso tedioso y sujeto a errores.
- El lenguaje Assembly (ensamblador) hace de traductor entre ese **lenguaje máquina** y uno que es más natural para el humano (**lenguaje de alto nivel**)

Assembly Language	Machine Code
add \$t1, \$t2, \$t3	04CB: 0000 0100 1100 1011
addi \$t2, \$t3, 60	16BC: 0001 0110 1011 1100
and \$t3, \$t1, \$t2	0299: 0000 0010 1001 1001
andi \$t3, \$t1, 5	22C5: 0010 0010 1100 0101
beq \$t1, \$t2, 4	3444: 0011 0100 0100 0100
bne \$t1, \$t2, 4	4444: 0100 0100 0100 0100
j 0x50	F032: 1111 0000 0011 0010
lw \$t1, 16(\$s1)	5A50: 0101 1010 0101 0000
nop	0005: 0000 0000 0000 0101
nor \$t3, \$t1, \$t2	029E: 0000 0010 1001 1110
or \$t3, \$t1, \$t2	029A: 0000 0010 1001 1010
ori \$t3, \$t1, 10	62CA: 0110 0010 1100 1010
ssl \$t2, \$t1, 2	0455: 0000 0100 0101 0101
srl \$t2, \$t1, 1	0457: 0000 0100 0101 0111
sw \$t1, 16(\$t0)	7050: 0111 0000 0101 0000
sub \$t2, \$t1, \$t0	0214: 0000 0010 0001 0100

# Lenguaje de Programación

## *Compilador*



# Compilación vs. interpretación

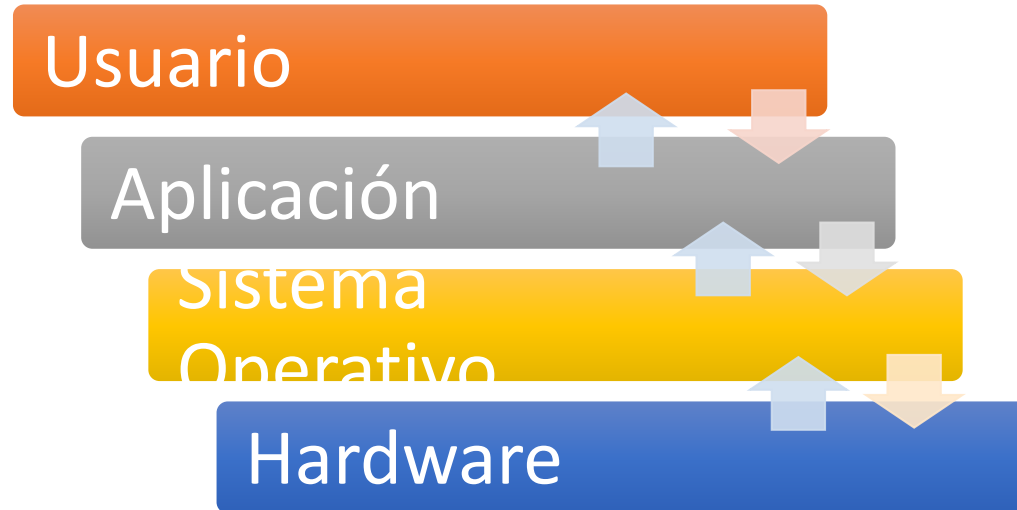
- Un compilador es un programa que transforma el código fuente de un programa a su equivalente en otro lenguaje de programación de más bajo nivel
- Un intérprete es un programa que ejecuta directamente las instrucciones escritas en un lenguaje de programación dado
- Traduce el lenguaje de alto nivel a lenguaje máquina sin generar ningún objeto.

# Compilación vs. interpretación

- Si hay un error, se debe corregir y volver a compilar
- Mayor dificultad al detectar errores.
- Ejecuta más rápido un programa.
- Toma cada línea de código, la analiza y ejecuta al mismo tiempo.
- Más lento que un compilador ya que el intérprete siempre está verificando el código.

# Sistema Operativo

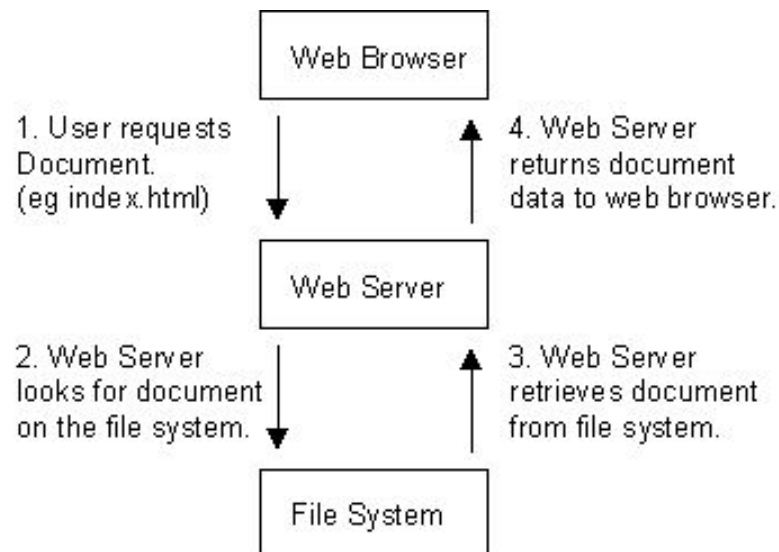
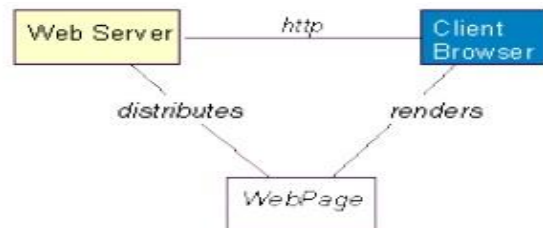
- Gestiona los recursos de **hardware**
- Provee servicios a los **programas de aplicación**



# Lenguaje de Programación

## *Arquitectura WEB*

- La arquitectura de un sitio web tiene 3 componentes principales: un servidor Web, una conexión de red y uno o más clientes (browsers)
- El servidor Web distribuye páginas de información formateada a los clientes que las solicitan. Los requerimientos son hechos a través de una conexión de red, y para ello se usa el protocolo HTTP



# Desarrollo de un Programa





# Desarrollo de un Programa

- **Definir el problema**
  - Determinar la información inicial para la elaboración del mismo
- **Análisis del problema**
  - Datos de entrada, de salida, métodos y fórmulas
- **Diseño del algoritmo**
  - Usar las herramientas de representación de algoritmos
- **Codificación**
  - Escribir la solución del problema, en instrucciones detalladas, en un lenguaje reconocible por la computadora
- **Prueba y depuración**
  - Se toman escenarios posibles, validos o inválidos y se corre la secuencia del algoritmo para ver si cumple con los resultados esperados

# Técnicas de Programación

## Carrera Programador full-stack

*Herramientas*

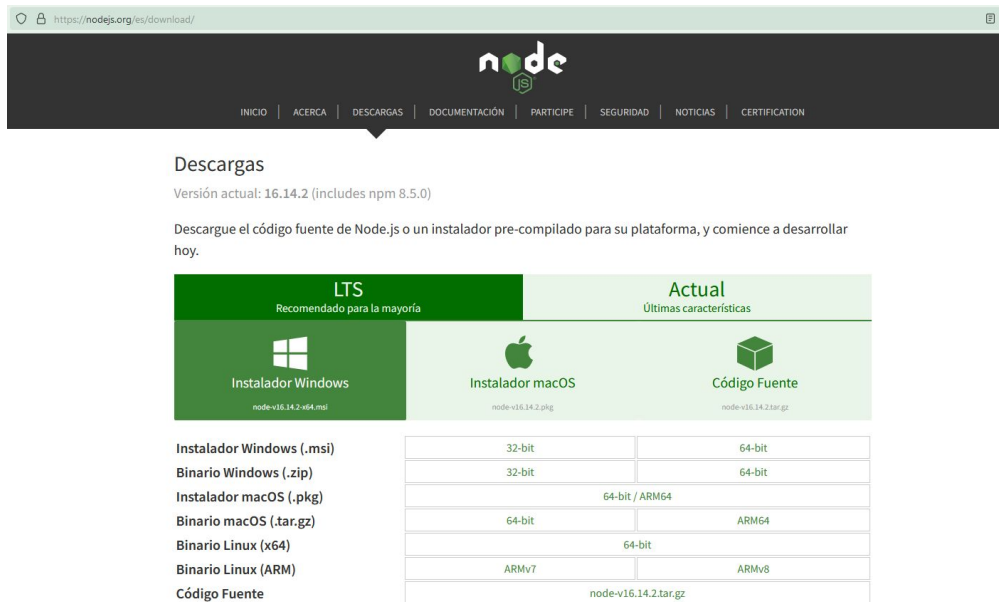
# TypeScript

- TypeScript es un lenguaje de programación basado en JavaScript, el cual nació para crear páginas web dinámicas.
- Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.
- JavaScript es un lenguaje de programación **interpretado**, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.
- TypeScript en cambio, requiere un proceso previo para ser utilizado, ya que realiza una comprobación de **sintaxis** del programa para asegurar su correcta ejecución.




# Características de TypeScript

- Tipos de datos
  - Característica más importante (de ahí el nombre)
  - Código más legible/entendible
  - Más chequeos al momento de desarrollar → mayor seguridad
- Soporte de clases
  - Programación orientada a objetos (en el próximo cuatrimestre)

# Instalación del intérprete/compilador



The screenshot shows the Node.js download page. The header includes the Node.js logo and navigation links: INICIO, ACERCA, DESCARGAS, DOCUMENTACIÓN, PARTICIPE, SEGURIDAD, NOTICIAS, and CERTIFICATION. The main content area is titled 'Descargas' and shows the current version as 16.14.2 (including npm 8.5.0). It offers two main options: 'LTS' (Recommended for most) and 'Actual' (Latest features). Under 'LTS', there is a link for 'Instalador Windows' (node-v16.14.2-x64.msi). Under 'Actual', there are links for 'Instalador macOS' (node-v16.14.2.pkg) and 'Código Fuente' (node-v16.14.2.tar.gz). A table below lists additional download options for various operating systems and architectures.

LTS		Actual	
Recomendado para la mayoría		Últimas características	
 Instalador Windows node-v16.14.2-x64.msi		 Instalador macOS node-v16.14.2.pkg	 Código Fuente node-v16.14.2.tar.gz
Instalador Windows (.msi)		32-bit	64-bit
Binario Windows (.zip)		32-bit	64-bit
Instalador macOS (.pkg)		64-bit / ARM64	
Binario macOS (.tar.gz)		64-bit	ARM64
Binario Linux (x64)		64-bit	
Binario Linux (ARM)		ARMv7	ARMv8
Código Fuente		node-v16.14.2.tar.gz	

Instalamos Node.JS: [www.nodejs.org](https://www.nodejs.org)

# NodeJS

**Node.js** es un entorno de desarrollo en tiempo de ejecución (runtime) para correr aplicaciones **Javascript** (y también **TypeScript**)

- **npm** es el Administrador de paquetes para los módulos Node.js.

Abrir una consola (Símbolo del sistema o Command Prompt)

En el Command Prompt ejecutar:

- **node --help**

para chequear la instalación de NodeJS

```

C:\Cursos\CFL>node --help
Usage: node [options] [ script.js ] [arguments]
       node inspect [options] [ script.js ] [host:port] [arguments]

Options:
  -                     script read from stdin (default if no file name is provided,
                        interactive mode if a tty)
  --                    Indicate the end of node options
  --abort-on-uncaught-exception
                        aborting instead of exiting causes a core file to be generated
                        for analysis
  -c, --check           syntax check script without executing
  --completion-bash     print source-able bash completion script
  --conditions=...     additional user conditions for conditional exports and imports
  --cpu-prof            Start the V8 CPU profiler on start up, and write the CPU profile
                        to disk before exit. If --cpu-prof-dir is not specified, write
                        the profile to the current working directory.
                        Directory where the V8 profiles generated by --cpu-prof will be
                        placed. Does not affect --prof.
  --cpu-prof-dir=...   specified sampling interval in microseconds for the V8 CPU
                        profile generated with --cpu-prof. (default: 1000)
  --cpu-prof-name=...  specified file name of the V8 CPU profile generated with
                        --cpu-prof
  --diagnostic-dir=... set dir for all output files (default: current working
                        directory)
  --disable-protos=... disable Object.prototype.__proto__
  --disallow-code-generation-from-strings
                        disallow eval and friends
  --enable-source-maps  experimental Source Map V3 support
  -e, --eval=...       evaluate script
  --experimental-abortcontroller
                        experimental AbortController support
  --experimental-import-meta-resolve
                        experimental ES Module import.meta.resolve() support
  --experimental-json-modules
                        experimental JSON interop support for the ES Module loader
  --loader, --experimental-loaders=...
                        use the specified module as a custom loader
  --experimental-policy=...
                        use the specified file as a security policy
  --experimental-repl-await
                        experimental await keyword support in REPL
  --es-module-specifier-resolution, --experimental-specifier-resolution=...
                        Select extension resolution algorithm for es modules; either
                        'explicit' (default) or 'node'
  --experimental-v8-modules
                        experimental ES Module support in v8 module
  --experimental-wasi-unstable-preview1
                        experimental WASI support
  --experimental-wasm-modules
                        experimental ES Module support for webassembly modules
  --force-context-aware
                        disable loading non-context-aware addons
  --frozen-intrinsics
                        experimental frozen intrinsics support
  --heap-prof           Start the V8 heap profiler on start up, and write the heap
                        profile to disk before exit. If --heap-prof-dir is not
                        specified, write the profile to the current working directory.
                        Directory where the V8 heap profiles generated by --heap-prof
                        will be placed.
  --heap-prof-dir=...  specified sampling interval in bytes for the V8 heap profile
                        generated with --heap-prof. (default: 512 * 1024)
  --heap-prof-name=... specified file name of the V8 heap profile generated with
                        --heap-prof
  --heapsnapshot-signal=...
                        Generate heap snapshot on specified signal
  -h, --help           print node command line options (currently set)
  --huge-max-old-generation-size
                        increase default maximum heap size on machines with 16GB memory
                        or more
  --icu-data-dir=...   set ICU data load path to dir (overrides NODE_ICU_DATA) (note:
                        linked-in ICU data is present)
  --input-type=...     set module type for string input
  --insecure-http-parser
                        use an insecure HTTP parser that accepts invalid HTTP headers
  --inspect=[host:]port
                        activate inspector on host:port (default: 127.0.0.1:9229)
  --inspect-brk=[host:]port
                        activate inspector on host:port and break at start of user
                        script
  --debug-port, --inspect-port=[host:]port
                        set host:port for inspector
  --inspect-publish-uid=...
                        comma separated list of destinations for inspector uid(default:
                        stderr, http)
  -i, --interactive    always enter the REPL even if stdin does not appear to be a
                        terminal
  --interned-frames-native-stack
                        help system profiles to translate JavaScript interned frames
  
```

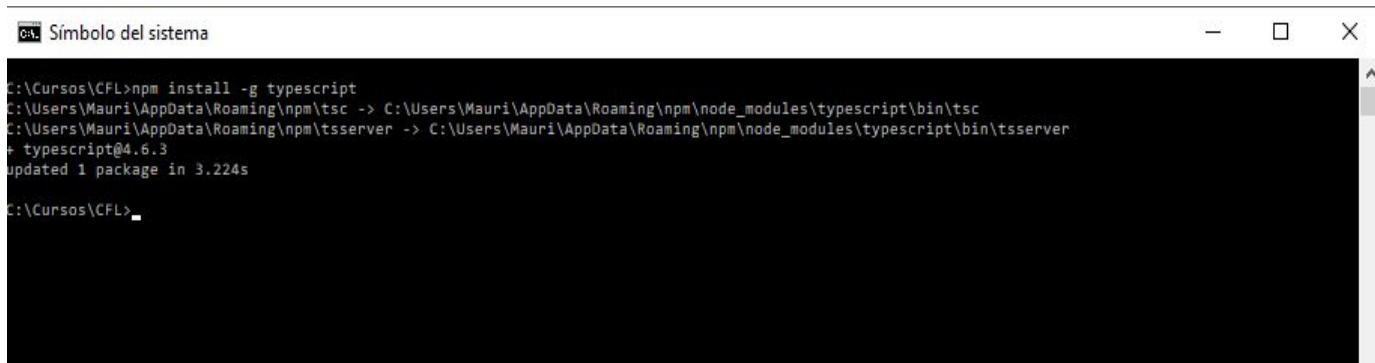
# NodeJS

Instalación del paquete “***typescript***” usando el comando “***npm***”

En el Command Prompt ejecutar:

- **npm install -g typescript**

Junto con el lenguaje, se instala el comando “***tsc***”. Este se encarga de hacer la traducción del lenguaje TypeScript, al lenguaje JavaScript, que es el que puede ejecutar **node.js**



```
Símbolo del sistema

C:\Cursos\CFL>npm install -g typescript
C:\Users\Mauri\AppData\Roaming\npm\tsc -> C:\Users\Mauri\AppData\Roaming\npm\node_modules\typescript\bin\tsc
C:\Users\Mauri\AppData\Roaming\npm\tsserver -> C:\Users\Mauri\AppData\Roaming\npm\node_modules\typescript\bin\tsserver
+ typescript@4.6.3
updated 1 package in 3.224s

C:\Cursos\CFL>
```

# NodeJS

Instalación de paquete “**readline-sync**” usando el comando “**npm**”

En el Command Prompt, vamos a nuestra carpeta de trabajo y ejecutamos:

- **npm install readline-sync**

Este paquete “readline-sync” permite ejecutar de forma interactiva una conversación con el usuario a través de una consola

De esta manera se puede ingresar datos a nuestros scripts

CA: Símbolo del sistema

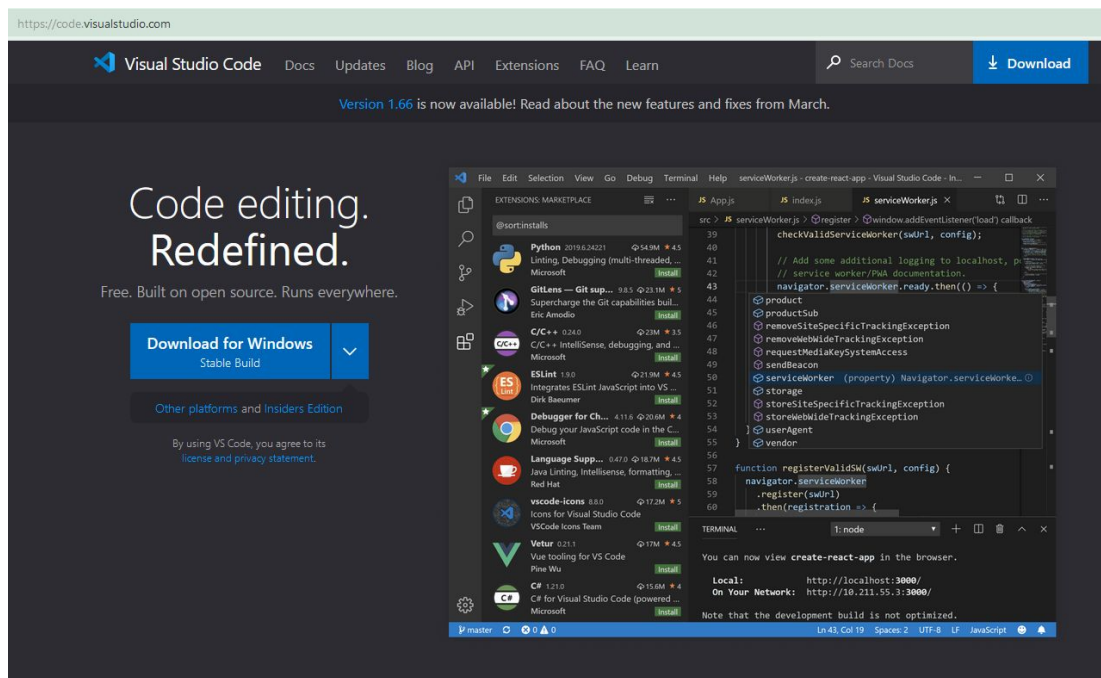
```
C:\Cursos\CFL>npm install readline-sync
npm WARN saveError ENOENT: no such file or directory, open 'C:\Cursos\CFL\package.json'
npm WARN createLockfile only created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\Cursos\CFL\package.json'
npm WARN CFL No description
npm WARN CFL No repository field.
npm WARN CFL No README data
npm WARN CFL No license field.

+ readline-sync@1.4.10
added 1 package from 1 contributor and audited 1 package in 2.328s
found 0 vulnerabilities

C:\Cursos\CFL>
```



# Instalación de editor de textos



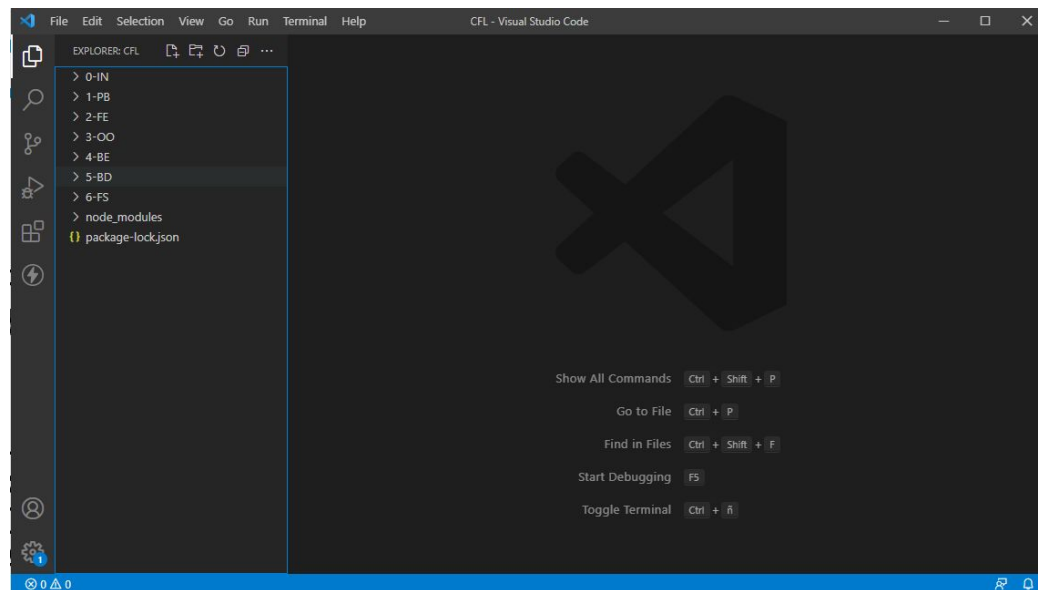
Instalamos Visual Studio Code: <https://code.visualstudio.com/>

# Editor de textos Visual Studio Code

Para escribir código se usan editores de textos.

Un Editor de texto especial para código se llama IDE (Integrated Development Environment - Ambiente de Desarrollo Integrado)

VSCode es compatible con los lenguajes JavaScript y TypeScript



# Nuestro Primer Programa

Recuerden que vamos a tener una carpeta por módulo y dentro además una carpeta por clase. Entonces dentro de nuestra **carpeta general del curso** vamos a crear una subcarpeta para la **primera clase**. Por ejemplo:

```
C:\cursos\CFS>mkdir clase_1  
C:\cursos\CFS>cd clase_1  
C:\cursos\CFS\clase_1>_
```

# Ejecutando un script en VSC

## *Hola Mundo en JS*

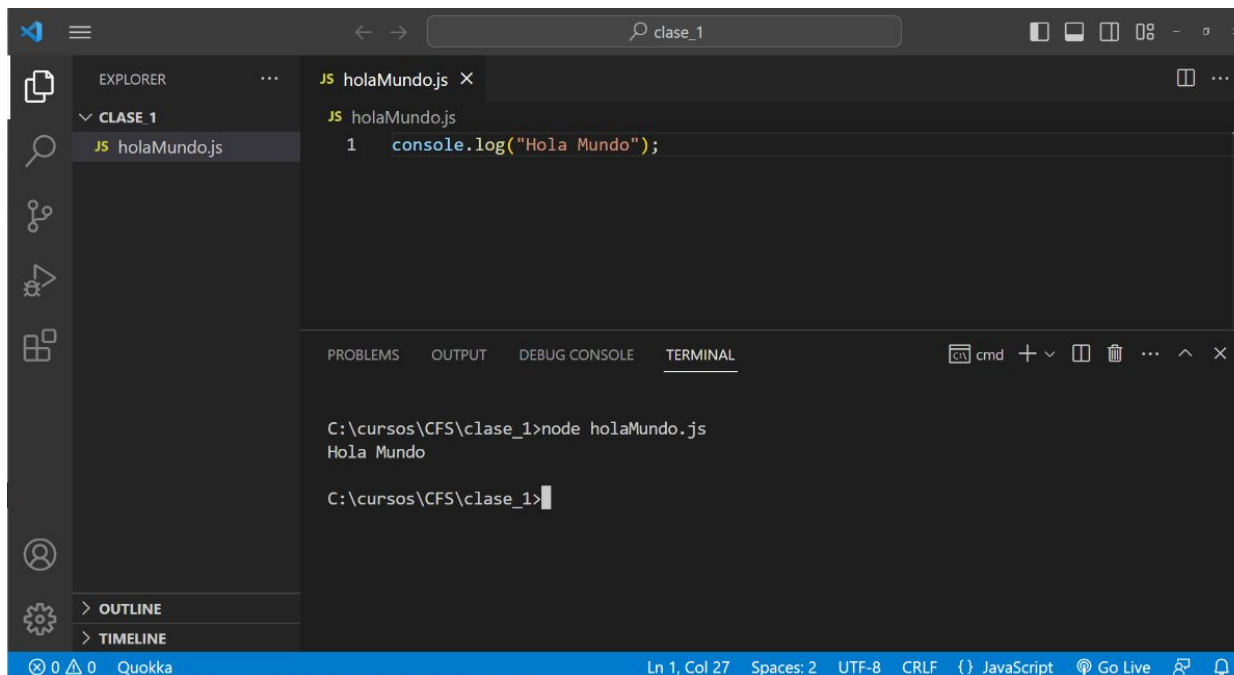
- **console.log()** muestra un mensaje en la consola

```
console.log("Hola Mundo");
```

- Grabar el archivo con nombre y extensión “holaMundo.js”
- Abrir la solapa Terminal
- Ejecutar el comando:

```
node holaMundo.js
```

- Este comando permite ejecutar nuestros scripts desde el path donde los almacenamos



# Ejecutando un script en VSC

## *Hola Mundo en TS*

- **console.log()** muestra un mensaje en la consola

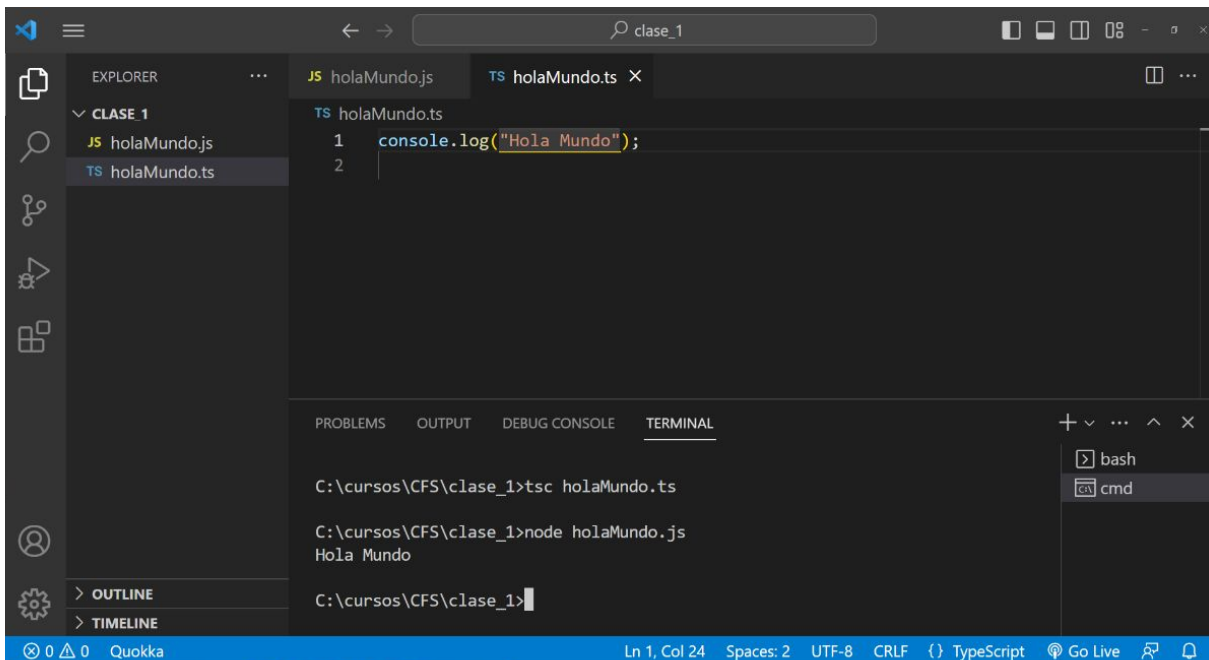
```
console.log("Hola Mundo");
```

- Grabar el archivo con nombre y extensión “holaMundo.ts”
- Abrir la solapa Terminal
- Ejecutar los comandos:

```
tsc holaMundo.ts (compila)
```

```
node holaMundo.js (corre)
```

- Este comando permite ejecutar nuestros scripts desde el path donde los almacenamos



# Palabras reservadas

En los lenguajes informáticos, una palabra reservada es una palabra que tiene un significado gramatical especial para ese lenguaje y no puede ser utilizada como un identificador de objetos en códigos del mismo, como pueden ser las variables.

# Ejercicio de clase

Crear un archivo nuevo que se denomine ejercicio1.ts e imprimir por consola los pasos para:

- a) Cocinar una torta
  - b) Dirigirse desde su domicilio hasta el supermercado
  - c) Crear un posteo en facebook
  - d) O cualquier otro procedimiento que el alumno desee
- (Mínimo 10 instrucciones)