

VSCode

Carrera
Programador
full-stack

Introducción al cuatrimestre

Segundo Cuatrimestre

Programación

- **Programación orientada a Objetos**
 - VSCode y TypeScript
 - Encapsulamiento
 - Composición
 - Herencia
 - Polimorfismo
 - Patrones de Diseño
- **Framework frontend**
 - ReactJs

Segundo Cuatrimestre

FIP

- GIT
- Ciclos de vida de un proyecto
- Metodologías Ágiles
- Scrum
- Diagramas de Clases
- Redes
- Proyecto integrador

VSCode

Carrera
Programador
full-stack

Herramientas

Javascript

- JavaScript es un lenguaje de programación que 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.
- A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java.

NodeJS

Node.js es un entorno de tiempo de ejecución para ejecutar aplicaciones Javascript

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

Abrir una consola (Command Prompt)

En el Command Prompt ejecutar:

- **node --help**

Para chequear la instalación de NodeJS

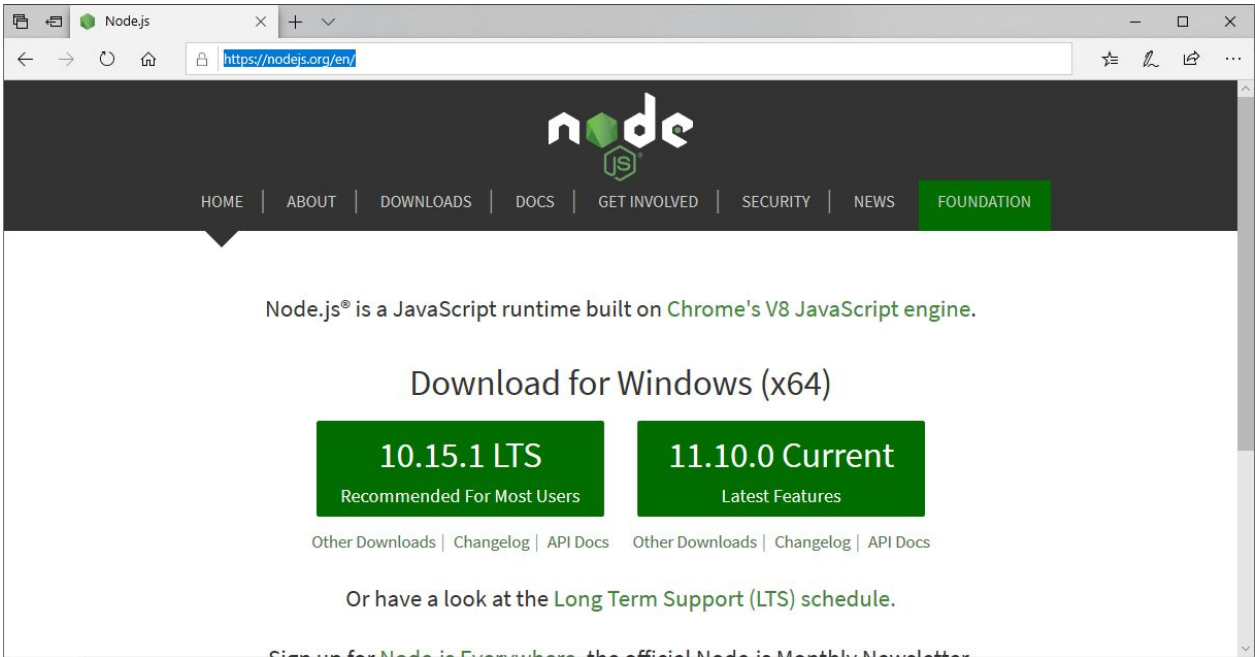
```

Microsoft Windows [Version 10.0.17763.134]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\guillermo.islas>node --help
Usage: node [options] [-e script | script.js | - ] [arguments]
       node inspect script.js [arguments]

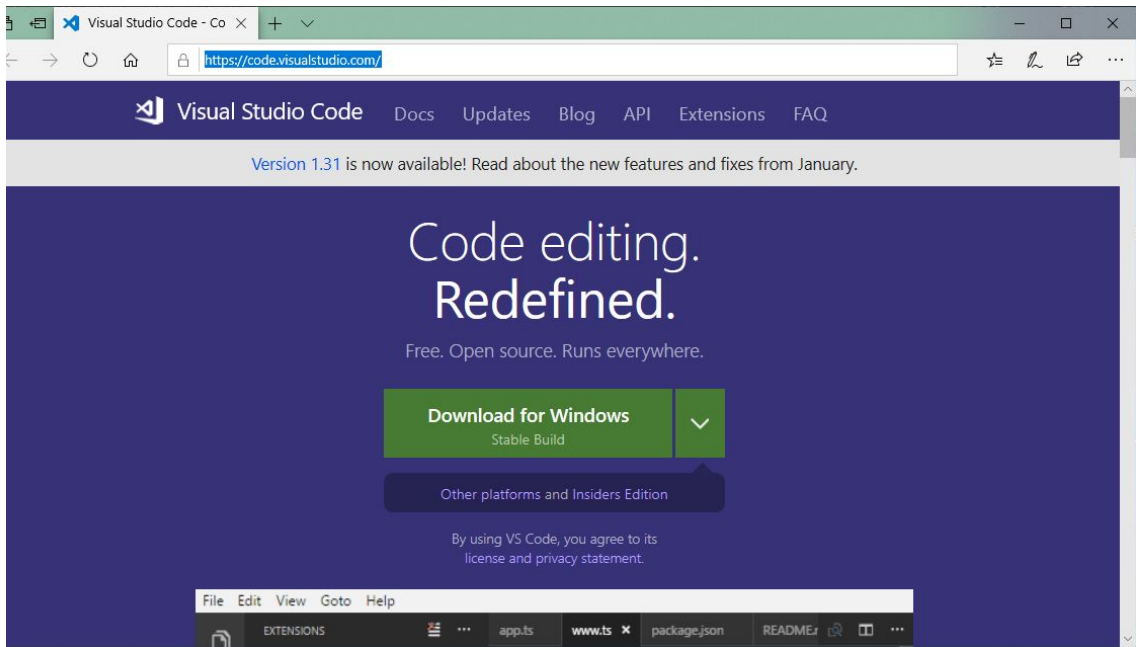
Options:
-
--abort-on-uncaught-exception
--check
--completion-bash
--diagnostic-report-directory=...
--diagnostic-report-filename=...
--diagnostic-report-on-fatalerror
--diagnostic-report-on-signal
--diagnostic-report-signal=...
--diagnostic-report-uncaught-exception
--diagnostic-report-verbose
-e, --eval=...
--experimental-modules
--experimental-policy=...
--experimental-repl-await
--experimental-report
--experimental-vm-modules
-h, --help
--http-parser=...
--icu-data-dir=...
--inspect=[-host:]port
--inspect-brk=[-host:]port
--debug-port, --inspect-port=[host:]port
-i, --interactive
--loader=...
--max-http-header-size=...
--no-deprecation
--no-force-async-hooks-checks
--no-warnings
--openssl-config=...
--pending-deprecation
--preserve-symlinks
--preserve-symlinks-main
-p, --print [...]
--prof-process
--redirect-warnings=...
-r, --require=...
--throw-deprecation
--title=...
--tls-cipher-list=...
script read from stdin (default if no file name is
provided, interactive mode if a tty)
--
--abort-on-uncaught-exception
indicate the end of node options
aborting instead of exiting causes a core file to
be generated for analysis
--check
syntax check script without executing
--completion-bash
print source-able bash completion script
--diagnostic-report-directory=...
define custom report pathname. (default: current
working directory of node.js process)
--diagnostic-report-filename=...
define custom report file name. (default:
YYYYMMDD.HHMMSS.PID.SEQUENCE#.txt)
--diagnostic-report-on-fatalerror
generate diagnostic report on fatal (internal)
errors
--diagnostic-report-on-signal
generate diagnostic report upon receiving signals
causes diagnostic report to be produced on provided
signal, unsupported in Windows. (default: SIGUSR2)
--diagnostic-report-uncaught-exception
generate diagnostic report on uncaught exceptions
--diagnostic-report-verbose
verbose option for report generation(true/false).
(default: false)
-e, --eval=...
evaluate script
--experimental-modules
experimental JS Module support and caching modules
--experimental-policy=...
use the specified file as a security policy
--experimental-repl-await
experimental await keyword support in REPL
--experimental-report
enable report generation
--experimental-vm-modules
experimental JS Module support in vm module
--help
print node command line options (currently set)
--http-parser=...
Select which HTTP parser to use; either 'legacy' or
'libhttp' (default: legacy).
--icu-data-dir=...
set ICU data load path to dir (overrides
NODE_ICU_DATA)
--inspect=[-host:]port
activate inspector on host:port (default:
127.0.0.1:9222)
--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
--i, --interactive
always enter the REPL even if stdin does not appear
to be a terminal
--loader=...
(with --experimental-modules) use the specified
file as a custom loader
--max-http-header-size=...
set the maximum size of HTTP headers (default: 8KB)
--no-deprecation
silence deprecation warnings
--no-force-async-hooks-checks
disable checks for async_hooks
--no-warnings
silence all process warnings
--openssl-config=...
load OpenSSL configuration from the specified file
(overrides OPENSSL_CONF)
--pending-deprecation
emit pending deprecation warnings
--preserve-symlinks
preserve symbolic links when resolving
--preserve-symlinks-main
preserve symbolic links when resolving the main
module
-p, --print [...]
evaluate script and print result
--prof-process
process V8 profiler output generated using --prof
--redirect-warnings=...
write warnings to file instead of stderr
-r, --require=...
module to preload (option can be repeated)
--throw-deprecation
throw an exception on deprecations
--title=...
the process title to use on startup
--tls-cipher-list=...
use an alternative default TLS cipher list
  
```

Instalación del intérprete/compilador



Instalamos Node.JS: www.nodejs.org

Instalación de editor de textos



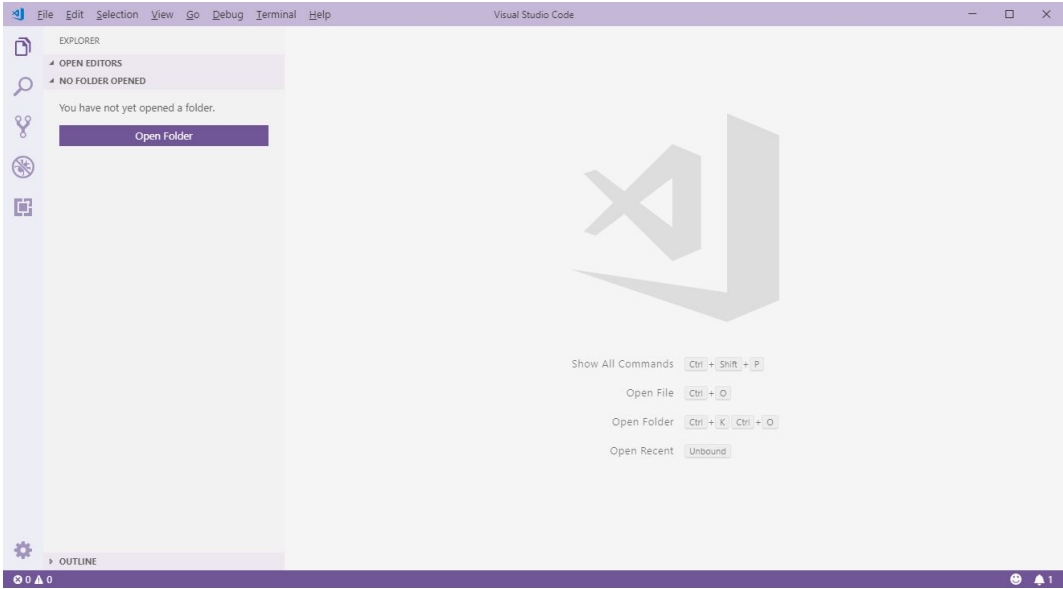
Instalamos Visual: <https://code.visualstudio.com/>

Editor de textos Visual Studio Code

Para hacer código se usan editores de textos.

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

VSCode es compatible con los lenguajes JavaScript y TypeScript



Ejecutando un script en VSC

Hola Mundo en JS

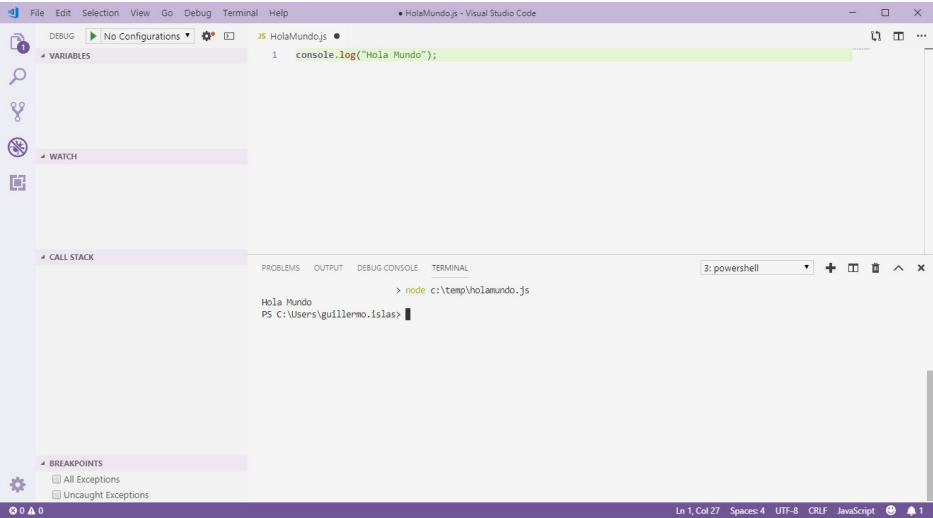
- `console.log()` muestra un mensaje en la consola web (o del intérprete JavaScript)

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

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

```
node c:\temp\HolaMundo.js
```

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



VSCode

Carrera
Programador
full-stack

Ingreso de datos

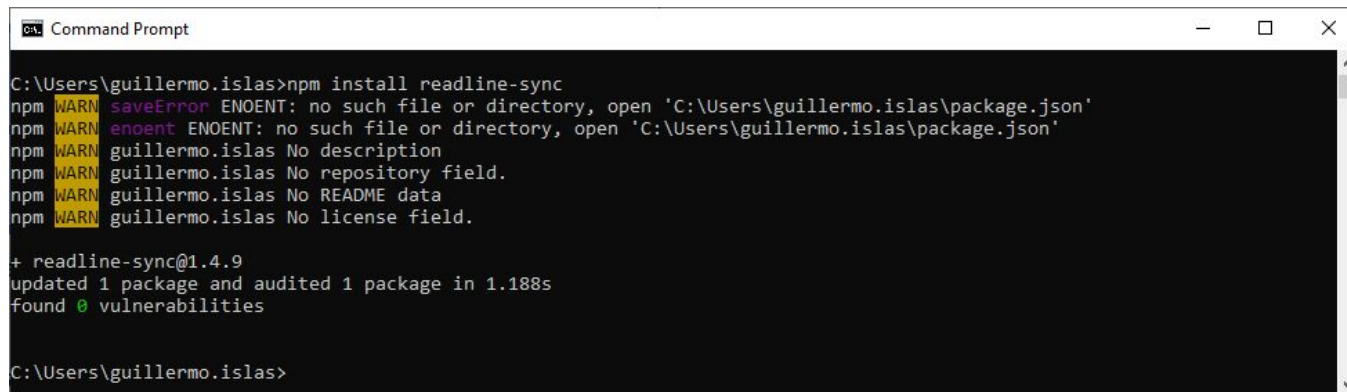
NodeJS - Instalación de paquete “*readline-sync*” usando el comando “*npm*”

En el Command Prompt ejecutar:

- **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



```
Command Prompt

C:\Users\guillermo.islas>npm install readline-sync
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\guillermo.islas\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\guillermo.islas\package.json'
npm WARN guillermo.islas No description
npm WARN guillermo.islas No repository field.
npm WARN guillermo.islas No README data
npm WARN guillermo.islas No license field.

+ readline-sync@1.4.9
updated 1 package and audited 1 package in 1.188s
found 0 vulnerabilities

C:\Users\guillermo.islas>
```

Ingreso de datos

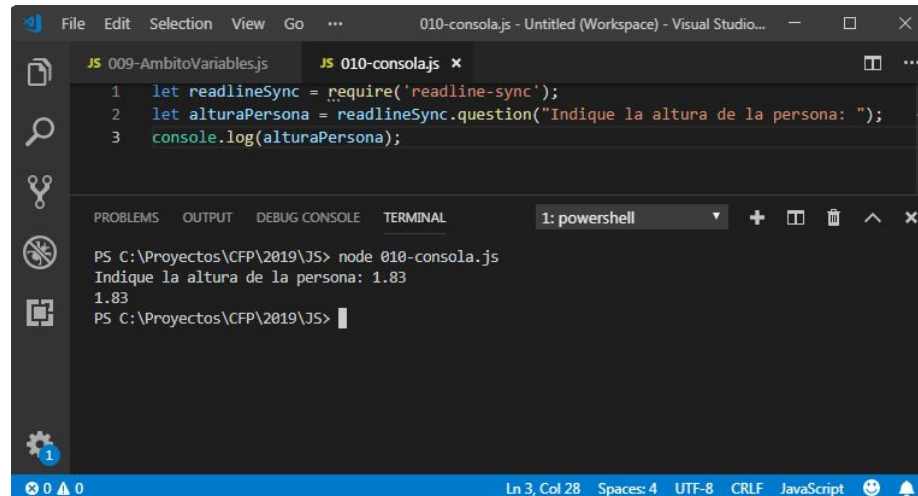
Ejemplo instrucción readline-sync()

```
let readlineSync = require('readline-sync');
```

```
let alturaPersona = readlineSync.question("Indique la altura de la persona: ");
```

```
console.log(alturaPersona);
```

Esta instrucción nos permite ingresar datos al script desde teclado



The screenshot shows the Visual Studio Code editor with a file named '010-consola.js'. The code in the editor is:

```
1 let readlineSync = require('readline-sync');
2 let alturaPersona = readlineSync.question("Indique la altura de la persona: ");
3 console.log(alturaPersona);
```

Below the editor, the TERMINAL panel is open, showing the execution of the script using Node.js:

```
PS C:\Proyectos\CFP\2019\JS> node 010-consola.js
Indique la altura de la persona: 1.83
1.83
PS C:\Proyectos\CFP\2019\JS> |
```

The status bar at the bottom indicates the current position is Line 3, Column 28, with 4 spaces, UTF-8 encoding, CRLF line endings, and the JavaScript language mode.

VSCode

CFL

**Programador
full-stack**

Ejercicios en clase

Ejercicio

Modificar el primer script de “Hola Mundo” para que:

- El mensaje que se muestra al usuario se almacene en una variable llamada mensaje y el funcionamiento del script sea el mismo.

Modificar el ejemplo de secuencia:

- Qué cada mensaje se almacene en una variable a mostrar por consola y que el funcionamiento del script sea el mismo

Modificar el ejemplo de base por altura

- Almacenar la base y la altura en variables y el resultado y que el funcionamiento del script sea el mismo