

Técnicas de Programación

CFP Programador full-stack

Introducción al Programa

Acerca del Programa

Módulo	Horas Dictado
Técnicas de Programación	50
Programación de Interfaces WEB	40
Programación Avanzada y Objetos	40
Base de Datos	40
Programación de servidores	40
Práctica, exámenes y repasos	36
Total	246

Técnicas de Programación

CFP Programador full-stack

Introducción al Módulo

Técnicas de Programación

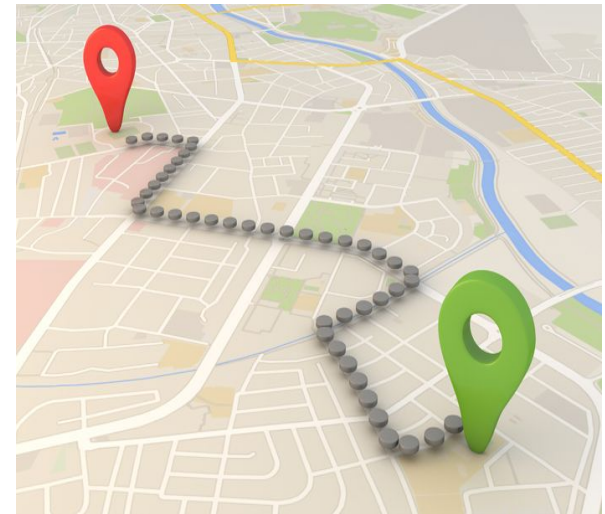
Objetivo

- 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.

Introducción

CFP 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">View gallery</a>
98      </div>
99      </div>
100     </div>
101     <a class="left carousel-control" href="#myCarousel" role="button" data-slide="prev">
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-slide="next">
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">FEATURED CONTENT</div>
116     </div>
117   </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

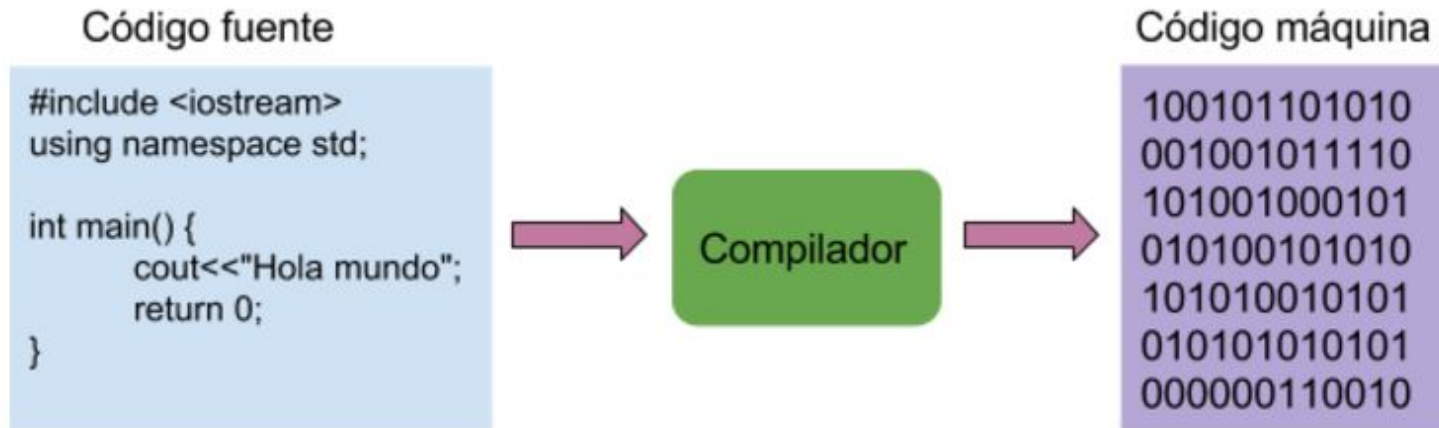
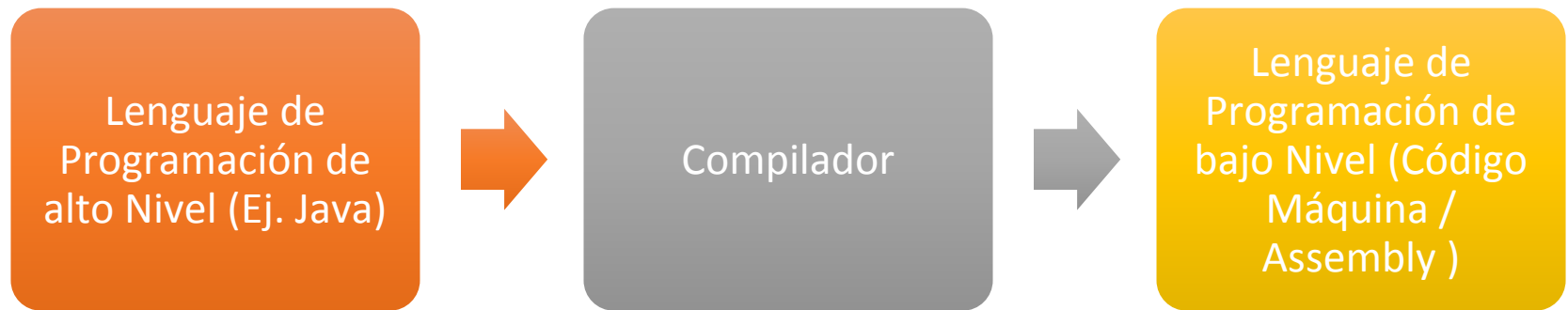
Lenguaje de Máquina

- 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



Sistema Operativo

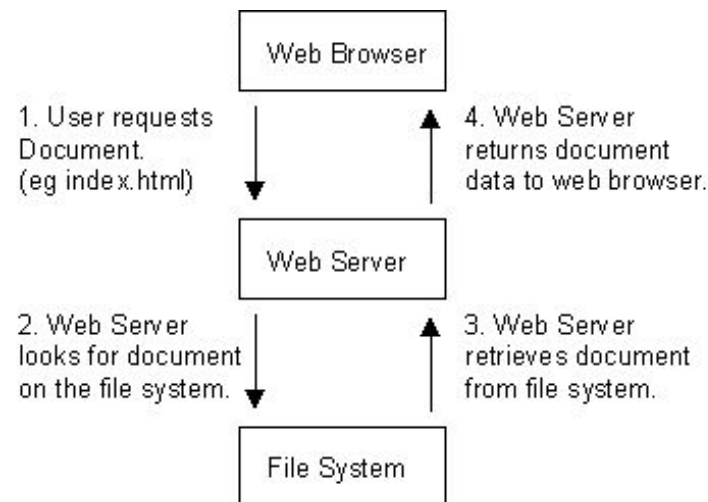
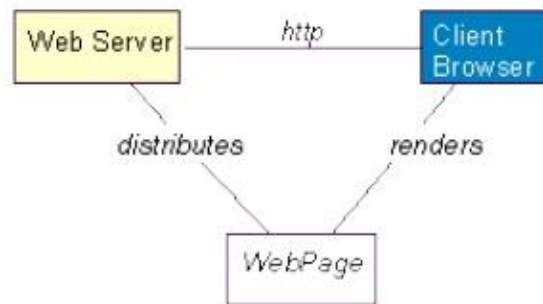
- Gestiona los recursos de **hardware**
- Provee servicios a los **programas de aplicación**



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

CFP

Programador

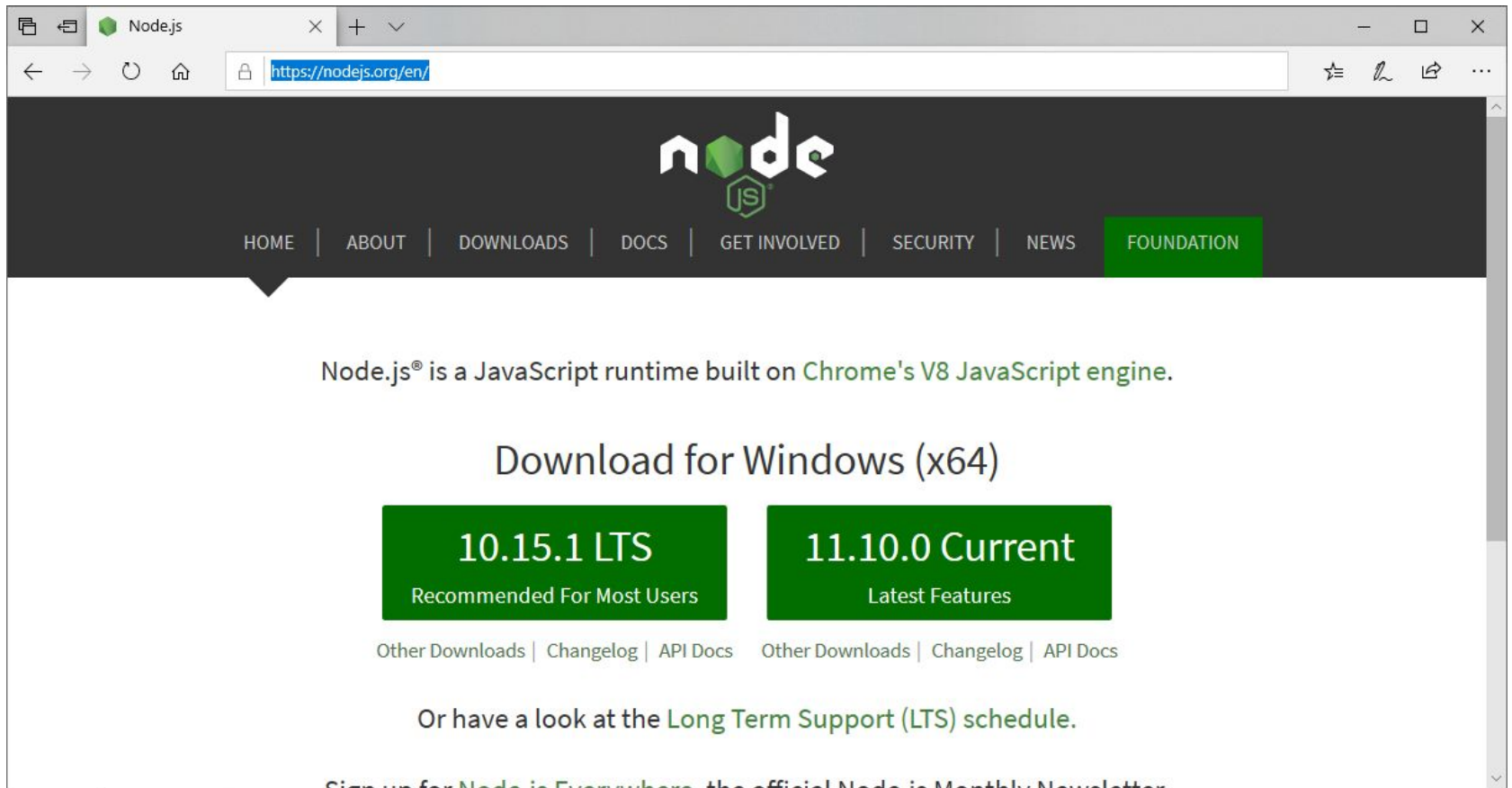
full-stack

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.

Instalación del intérprete/compilador



Instalamos Node.JS: www.nodejs.org

NodeJS

Node.js es una plataforma para crear aplicaciones utilizando JavaScript.

- **Node.js** es para ejecutar scripts 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:
  -                  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
  -c, --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.HHHMMSS.PID.SEQUENCE#.txt)
  --diagnostic-report-on-fatalerror
                    generate diagnostic report on fatal (internal)
                    errors
  --diagnostic-report-on-signal
                    generate diagnostic report upon receiving signals
  --diagnostic-report-signal=...
                    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 ES Modules 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 ES Module support in vm module
  -h, --help         print node command line options (currently set)
  --http-parser=...  Select which HTTP parser to use; either 'legacy' or
                    'llhttp' (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:
                    127.0.0.1:9229)
  --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
  -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
  
```

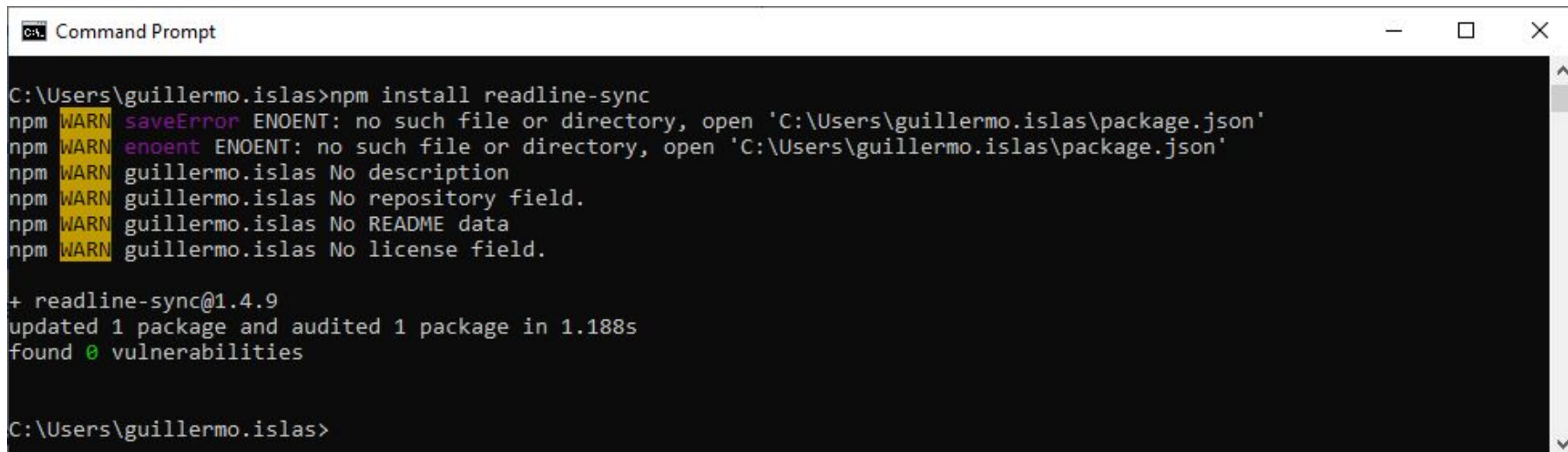
NodeJS - Instalación de paquete “*readline-sync*” usando el comando “*npm*”

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

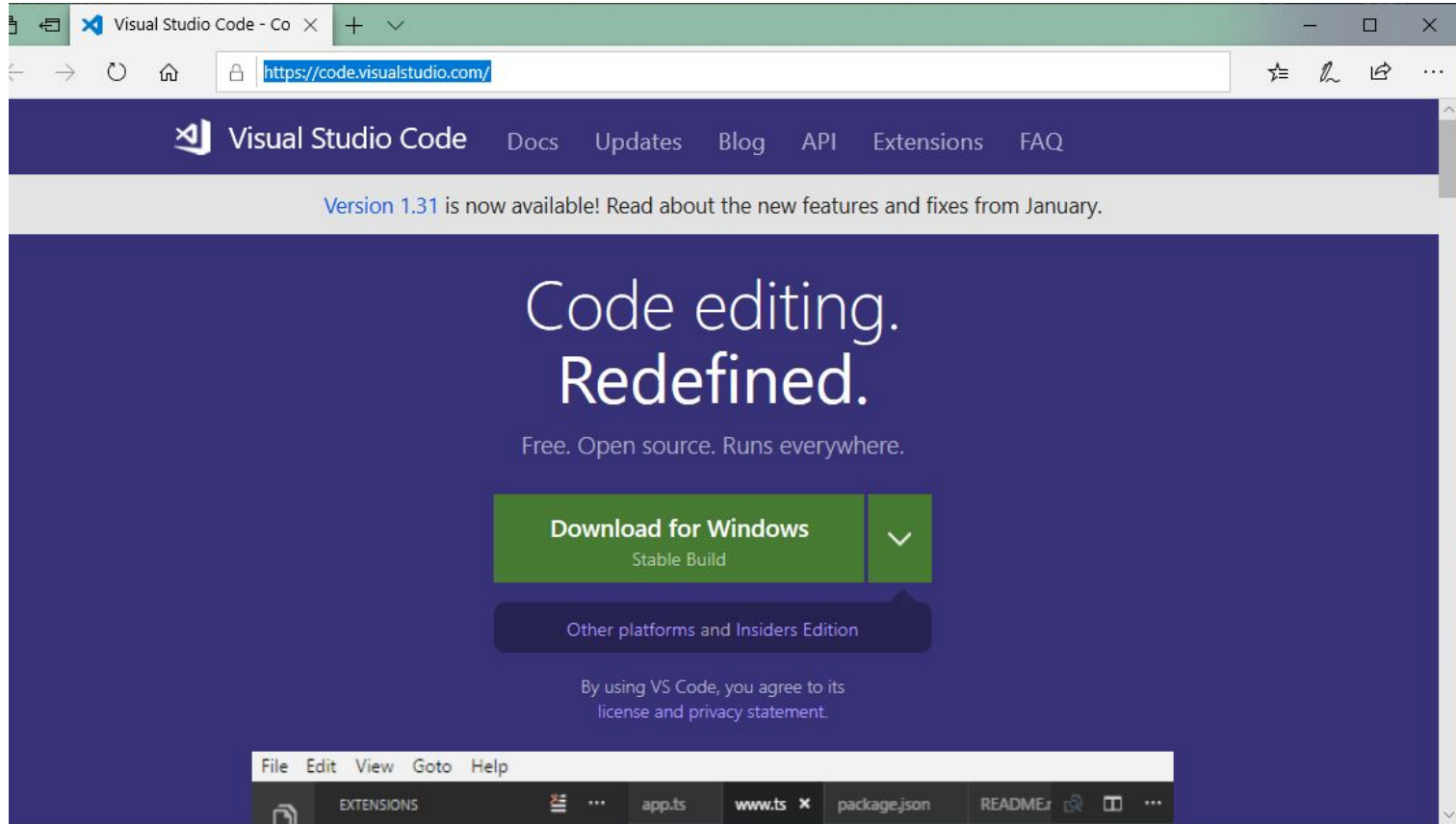


```
C:\Users\guillermo.islas>npm install readline-sync
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\guillermo.islas\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\guillermo.islas\package.json'
npm WARN guillermo.islas No description
npm WARN guillermo.islas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


Instalación de editor de textos



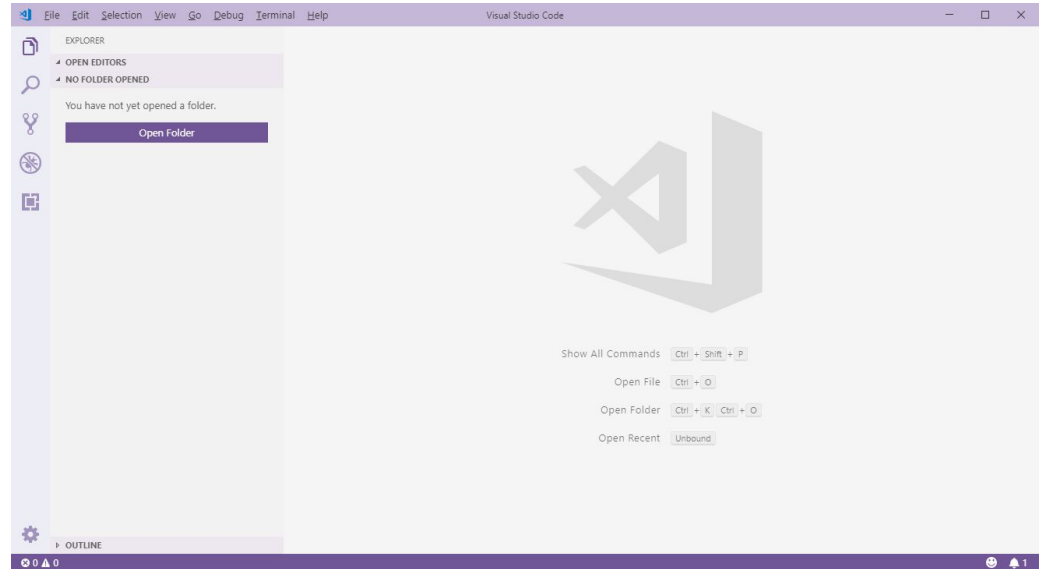
Instalamos Node.JS: <https://code.visualstudio.com/>

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)

VSCoode es compatible con los lenguajes JavaScript y TypeScript



Ejecutando un script en VSC

Hola Mundo en JS

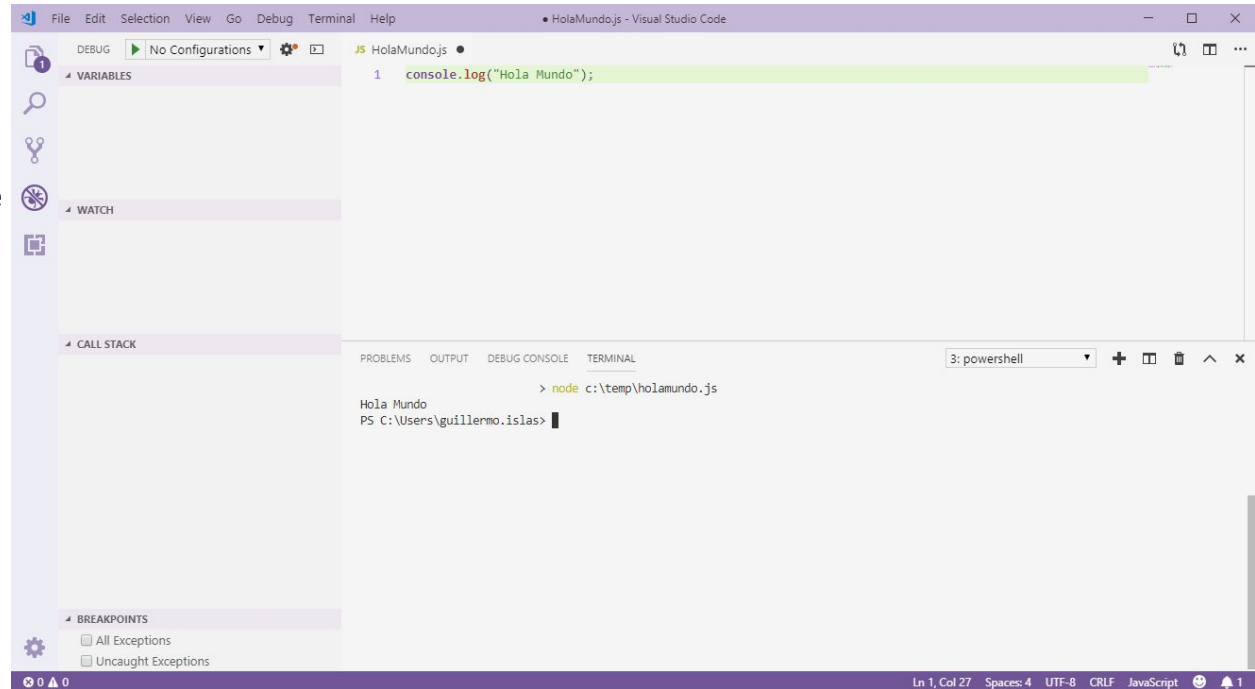
- **console.log()** muestra un mensaje en la consola web (o del intérprete JavaScript)

```
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



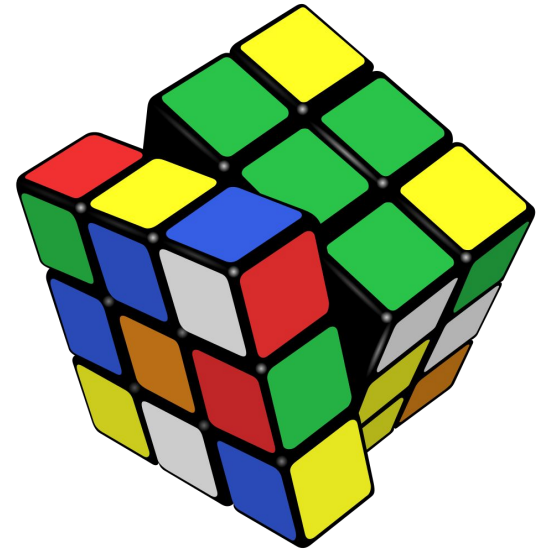
Técnicas de Programación

CFP Programador full-stack

Algoritmos Secuenciales

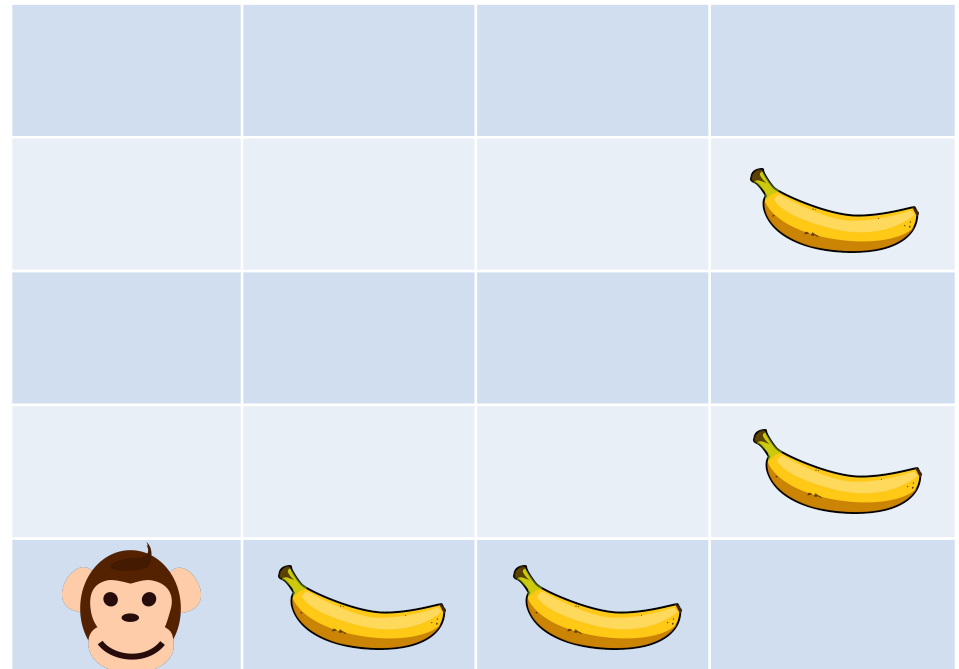
Algoritmos

- Nos ayudan a resolver un problema
- Consisten de pasos lógicamente ordenados
- Dado un conjunto de datos de entrada da un resultado (solución al problema)



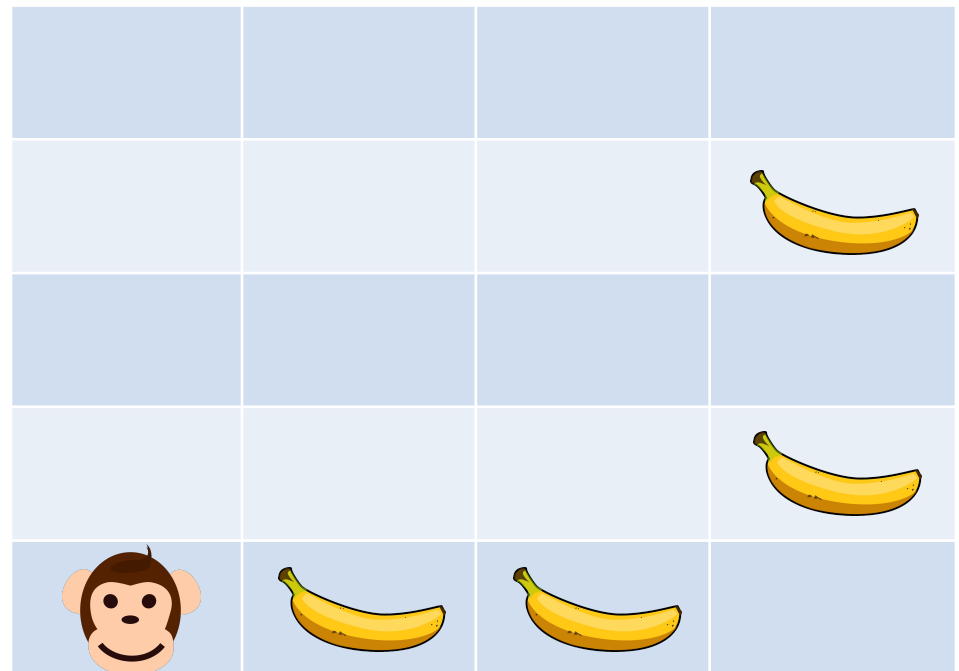
Algoritmos

- Instrucciones
 - Mover arriba
 - Mover abajo
 - Mover derecha
 - Mover izquierda
 - Comer banana



Algoritmos

- Algoritmo
 - Mover derecha
 - Comer banana
 - Mover derecha
 - Comer banana
 - Mover derecha
 - Mover arriba
 - Comer banana
 - Mover arriba
 - Mover arriba
 - Comer banana



Algoritmos

- Debe ser **preciso**
- Debe estar **específicamente definido**
- Debe ser **finito**
- Debe ser **correcto**
- Debe ser **independiente** del lenguaje



Elementos de un Algoritmo

- Calculo del área de un rectángulo

- **Entrada:**

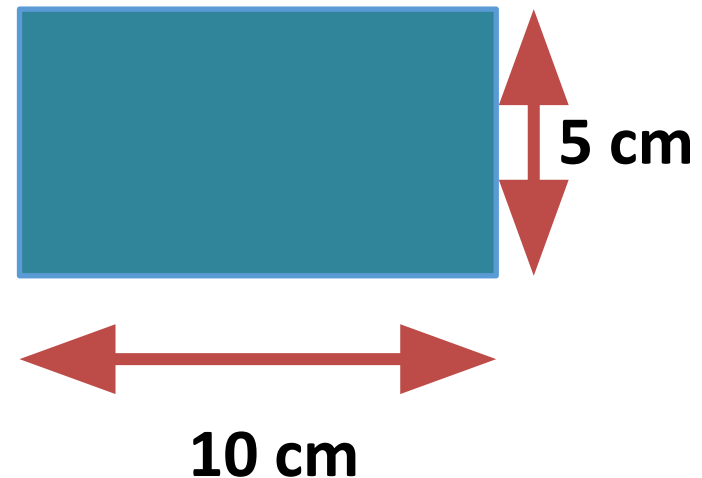
- Dato 1: altura 5 cm
- Dato 2: base 10 cm

- **Proceso:**

- $\text{Área} = \text{base} * \text{altura}$

- **Salida:**

- $5 * 10 = 50$



Técnicas de Programación

CFP Programador full-stack

Algoritmos simples en Javascript (JS)

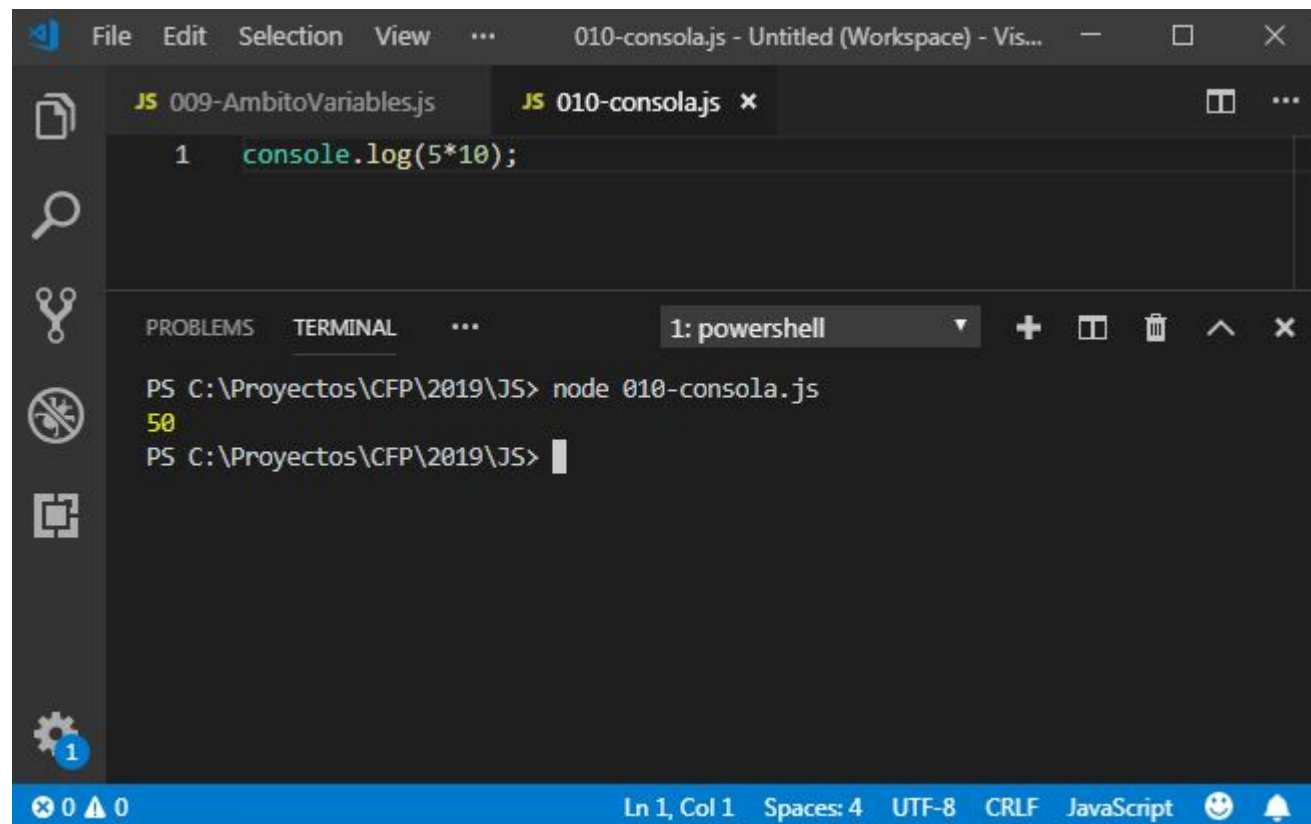
JavaScript - Glosario básico

- **Script:** cada uno de los programas, aplicaciones o trozos de código creados con el lenguaje de programación JavaScript. Unas pocas líneas de código forman un script y un archivo de miles de líneas de JavaScript también se considera un script.
- **Sentencia:** cada una de las instrucciones que forman un script.
- **Palabras reservadas:** son las palabras (en inglés) que se utilizan para construir las sentencias de JavaScript y que por tanto no pueden ser utilizadas libremente. Las palabras actualmente reservadas por JavaScript son: **break, case, catch, continue, default, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, while, with.**

Implementando Algoritmos

Ejercicio: Área del Rectángulo 5x10

```
console.log(5*10);
```



The screenshot shows a code editor with two tabs: '009-AmbitoVariables.js' and '010-consola.js'. The active tab '010-consola.js' contains the code `1 console.log(5*10);`. Below the editor is a terminal window titled '1: powershell'. The terminal shows the command `PS C:\Proyectos\CFP\2019\JS> node 010-consola.js` being executed, followed by the output `50`. The status bar at the bottom indicates 'Ln 1, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', and 'JavaScript'.

JavaScript - Sintaxis básica

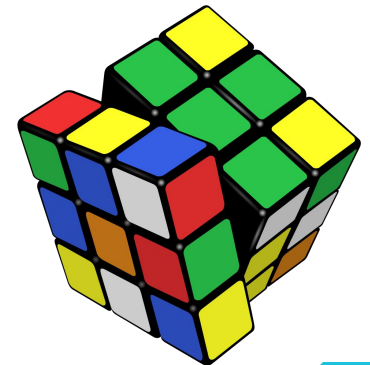
Las normas básicas que definen la sintaxis de JavaScript son las siguientes:

- **No se tienen en cuenta los espacios en blanco y las nuevas líneas**
- **Se distinguen las mayúsculas y minúsculas:** La palabra “console” no es lo mismo que “CONSOLE” ni que “ConSolE”
- **No es necesario terminar cada sentencia con el carácter de punto y coma (;).** Aunque es conveniente hacerlo para usar todos el mismo estilo de código.

Elementos de un Algoritmo

Ejercicios

- Para cada uno de los siguientes puntos describa los **elementos del algoritmo** (entrada, proceso, salida)
 1. Sumar 2 números
 2. Preparar una tarta de frutillas
 3. Tomar la presión sanguínea
 4. Llenar una cajita de huevos con 5 huevos que están dentro de una bolsa



Estructuras de Control

Secuenciales

Selectivas o De
Decisión

Repetitivas

Secuencia

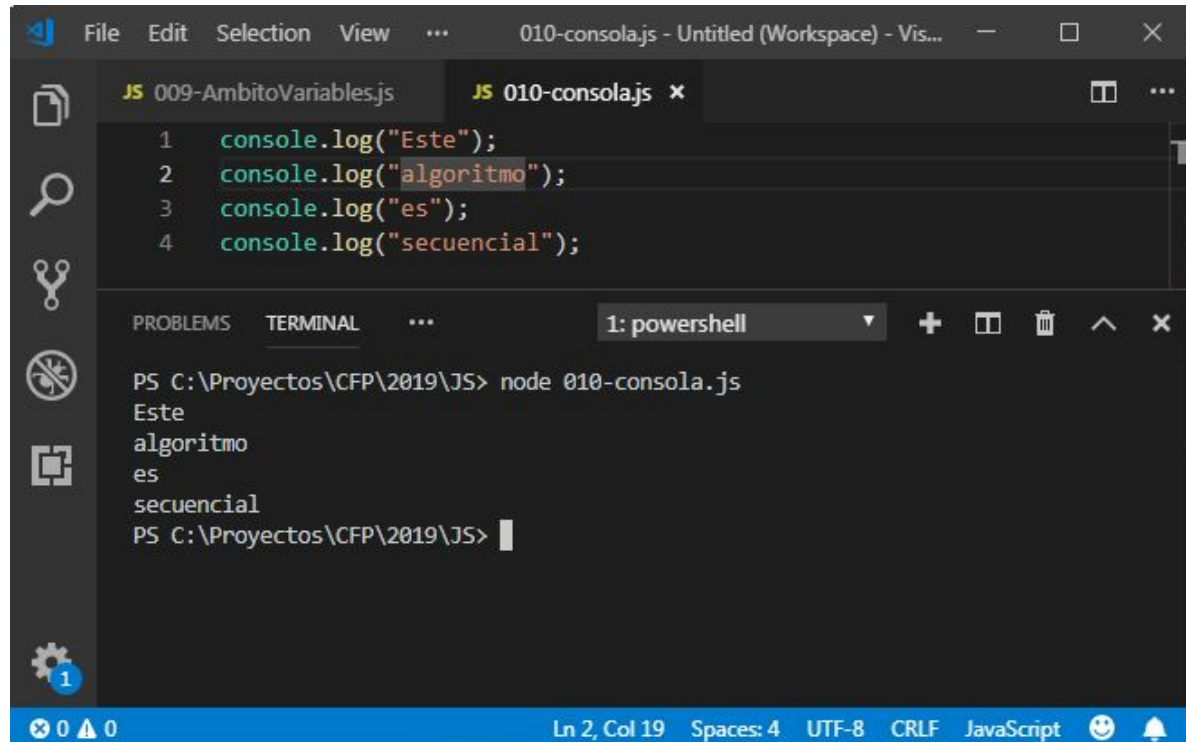


- Un algoritmo es una serie de pasos para resolver un programa
- Los algoritmos más simples son una lista de acciones que se ejecutan en orden

Secuencia

Ejemplo instrucción console.log()

```
console.log ("Este");  
console.log ("algoritmo");  
console.log ("es");  
console.log ("secuencial");
```



The screenshot shows a code editor with two tabs: '009-AmbitoVariables.js' and '010-consola.js'. The '010-consola.js' tab is active, displaying the following code:

```
1 console.log("Este");  
2 console.log("algoritmo");  
3 console.log("es");  
4 console.log("secuencial");
```

Below the code editor, the 'TERMINAL' panel is open, showing the output of the code execution in a PowerShell window:

```
PS C:\Proyectos\CFP\2019\JS> node 010-consola.js  
Este  
algoritmo  
es  
secuencial  
PS C:\Proyectos\CFP\2019\JS>
```

The status bar at the bottom indicates the current line and column: 'Ln 2, Col 19', along with other settings like 'Spaces: 4', 'UTF-8', 'CRLF', and 'JavaScript'.

Ejercicio Alumnos

Usando el comando `console.log()` crear las siguientes secuencias

1. Sumar 2 números
2. Preparar una tarta de frutillas
3. Tomar la presión sanguínea
4. Llenar una cajita de huevos con 5 huevos que están dentro de una bolsa

Comentarios

En JS **se pueden incluir comentarios**

Los comentarios se utilizan para añadir información en el código fuente del programa, son aclaraciones para otros programadores (o vos mismo)

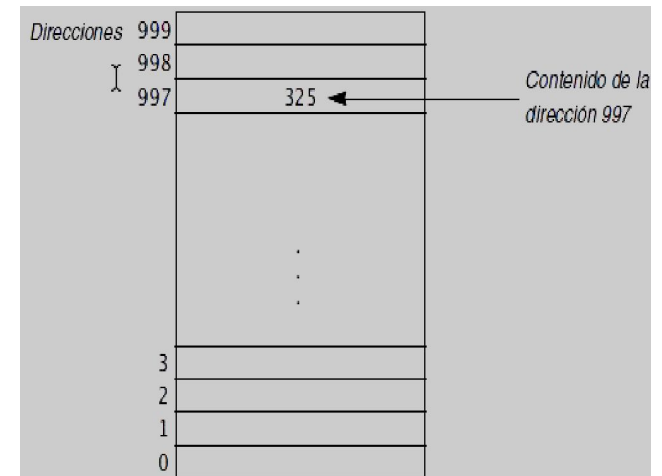
```
/* este programa imprime 4 líneas por la consola,  
Además tiene comentarios */  
console.log ( "Este" ) ;  
//esto es un comentario  
console.log ( "algoritmo" ) ;  
//esto es otro comentario  
console.log ( "es" ) ;  
//esto es un cuarto comentario  
console.log ( "secuencial" ) ;
```

Variables

- Las variables en programación siguen una lógica similar a las variables usadas en otros ámbitos como las matemáticas.
- Una variable es un elemento que se emplea para almacenar y hacer referencia a un valor.
- Gracias a las variables es posible crear "programas genéricos", es decir, programas que funcionan siempre igual independientemente de los valores concretos utilizados.

Variables

- Guarda información (números, letras, etc.)
- Tiene una dirección de memoria
- Tiene un nombre
- Su contenido puede **variar** durante la ejecución del programa



Variables

Ejemplo:

```
resultado = 3 + 1
```

En JavaScript:

```
let numero_1 = 3;
```

```
let numero_2 = 1;
```

```
let resultado = numero_1 + numero_2;
```

Declarar variable: La palabra reservada **let** solamente se debe indicar al definir por primera vez la variable. Se recomienda declarar todas las variables que se vayan a utilizar

Si cuando se declara una variable se le asigna también un valor, se dice que la variable ha sido **inicializada**. Se pueden declarar por una parte y asignarles un valor posteriormente

Variables y Constantes

Antes (JS viejo)

Declarar una variable

```
var nombre = "Pepe";
```

Declarar una constante

No existen

Van a encontrar ejemplos así porque hay mucho código viejo dando vueltas

Ahora

Declarar una variable

```
let nombre = "Pepe";
```

Declarar una constante

```
const cantDados = 2;
```

Usar "let" es mejor, después veremos porque

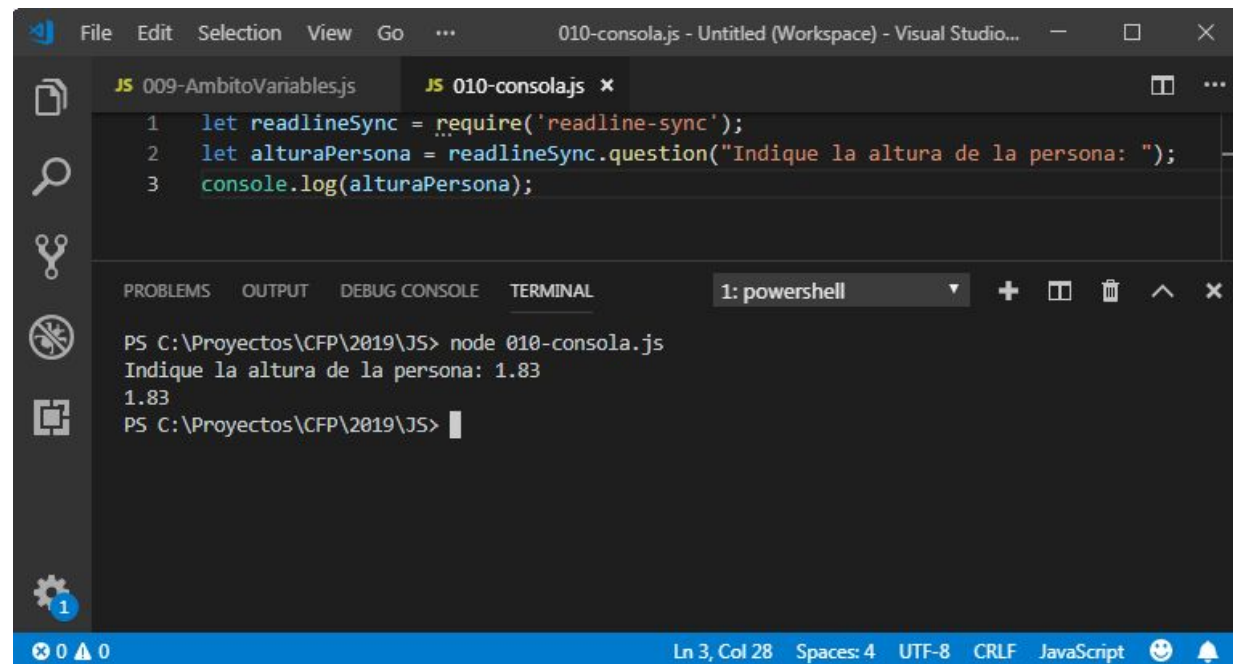


Ingreso de datos

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 interface. The editor has two tabs: '009-AmbitoVariables.js' and '010-consola.js'. The '010-consola.js' tab is active, displaying the following JavaScript code:

```
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. The prompt is '1: powershell'. The command executed is 'PS C:\Proyectos\CFP\2019\JS> node 010-consola.js'. The output shows the prompt 'Indique la altura de la persona: 1.83' followed by the value '1.83' being printed. The prompt returns to 'PS C:\Proyectos\CFP\2019\JS> '.

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

Tipado de Variables - Tipos

- El **tipado estático** nos obliga a definir desde el principio el tipo de una variable. Lenguajes con tipado estático son C++, Java, C#, etc.
- El **tipado dinámico** nos da la facilidad de no definir los tipos al declarar una variable, algunos ejemplos son PHP, JavaScript, Groovy, Python, entre otros.

Variables

Tipos de Datos Básicos

- Aunque todas las variables de JavaScript se crean de la misma forma, la forma en la que se les asigna un valor depende del tipo de valor que se quiere almacenar (números, textos, etc.)
- **Numérico (Number):**
 - Números tanto enteros (integer) como decimales (float)
 - Para separar decimales se utiliza el punto
 - Ejemplos: 12, 0, -2.3, 3.14

```
let iva = 16;      // variable tipo entero  
let total = 234.65; // variable tipo decimal
```



Variables

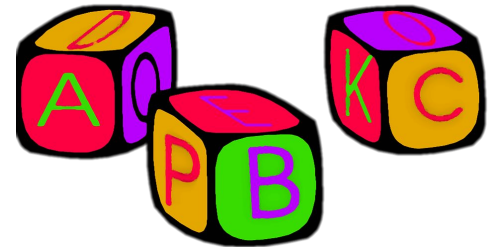
Tipos de Datos Básicos

- **Lógico (Boolean):**

- Sólo puede tomar dos valores: **true** o **false**
- No se puede utilizar para almacenar números y tampoco permite guardar cadenas de texto.

```
let clienteRegistrado = false;
```

```
let ivaIncluido = true;
```



Variables

Tipos de Datos Básicos

• Texto (String):

- Caracteres o cadenas de caracteres encerrados entre comillas (dobles o simples)

```
let mensaje = "Bienvenido a nuestro sitio web";  
let nombreProducto = 'Producto ABC';  
let letraSeleccionada = 'c';
```

- Si el propio texto contiene comillas simples o dobles, la estrategia que se sigue es la de encerrar el texto con las comillas (simples o dobles) que no utilice el texto

// El contenido de texto1 tiene comillas simples, por lo que se encierra con comillas dobles

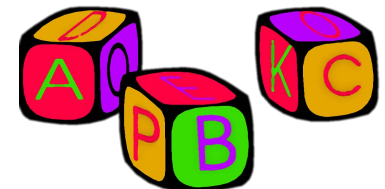
```
let texto1 = "Una frase con 'comillas simples' dentro";
```

// El contenido de texto2 tiene comillas dobles, por lo que se encierra con comillas simples

```
let texto2 = 'Una frase con "comillas dobles" dentro';
```

- A veces las cadenas de texto contienen tanto comillas simples como dobles. Además existen otros caracteres (tabulador, ENTER, etc.) especiales. JavaScript permite caracteres especiales dentro de una cadena de texto con el "mecanismo de escape" de los caracteres problemáticos.

```
let texto1 = 'Una frase con \'comillas simples\' dentro';  
let texto2 = "Una frase con \"comillas dobles\" dentro";  
let texto3 = "Una frase con \n Una nueva línea dentro";  
let texto4 = "Una frase con \t Un tabulador dentro";  
let texto5 = "Una frase con \\ Una barra inclinada dentro";
```



Variables

Tipos de Datos Básicos

- **NULO (Null):**

- El valor null es un literal de Javascript que representa un valor nulo o "vacío".

```
console.log(typeof 42);
```

```
// expected output: "number"
```

- **Indefinido (Undefined):**

- Una variable a la cual no se le haya asignado valor tiene entonces el valor undefined.

```
console.log(typeof 'blubber');
```

```
// expected output: "string"
```

- **Determinación del tipo usando el operador typeof**

```
console.log(typeof true);
```

```
// expected output: "boolean"
```

```
console.log(typeof declaredButUndefinedVariable);
```

```
// expected output: "undefined";
```

Variables

- Las variables pueden declararse de forma implícita.
- La primera vez que uso una variable se declara “automáticamente”.

```
numero = 2; //declaró variable número mágicamente
```

```
let nombre = “Pepe”;
```

```
...
```

```
nombbre = “Juan”; //error, tipeo mal la variable
```

```
//crea una variable global nueva
```

```
alert(nombre) //imprime Pepe
```

Tipos

- Javascript tiene tipos dinámicos.
- Una misma variable puede cambiar de tipo.
- Puede causar confusiones (y errores que no encuentro durante horas).

```
let nombre = "Pepe"; //nombre es un string
```

```
...
```

```
nombre = 2; //nombre es un int (cambia tipo)
```

NO HACER
ESTO

Tipos de Datos

- String
- Number
- Boolean
- Null
- Undefined
- Object
 - Function
 - Array
 - Date
 - Expresiones Regulares (RegExp)

Conversión de tipos

- **Cuidado** con los **tipos**, son dinámicos y no saber de qué tipo es una variable puede cambiar el resultado.

```
5 == "5">//true
```

```
"1" + 2 + 3;//"123"
```

```
//Conversion manual de tipos
```

```
parseInt("1", 10) + 2 + 3; //6
```

- **ES6** introduce una nueva forma de trabajar con Strings

```
(''+nombre+', '+apellido+')
```

```
//es lo mismo que:
```

```
`(${nombre}, ${apellido})`
```

Operadores

Operador	Significado	Ejemplo
=	Asignación	nombre="Juan"
+	Suma	total = cant1 + cant2
-	Resta	stock = disp - venta
*	Multiplicación	area = base * altura
/	División	porc = 100 * parte / total
^	Potenciación	sup = 3.41 * radio ^ 2
% ó MOD	Resto de la división entera	resto = num MOD div

Secuencia

Prueba de Escritorio

- Una prueba de escritorio consiste en analizar (antes de hacer el algoritmo) cuál debe ser el resultado dada la entrada del algoritmo

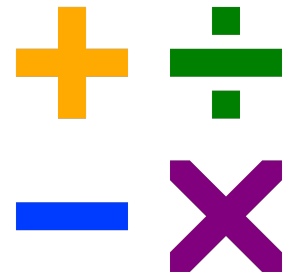
N° Prueba	Entrada		Salida	
	1er Num Ingresado	2do Num Ingresado	Suma	Mensaje
1	20	30	$20+30=50$	El resultado de la suma es: 50
2	15	150	$15+150=165$	El resultado de la suma es: 165
3	130	300	$130+300=430$	El resultado de la suma es: 430

Secuencia

Ejercicio: Suma de Dos Números

- Leemos los números desde el teclado y los guardamos en las variables

```
let readlineSync = require('readline-sync');  
let primerNumero = readlineSync.questionInt("Ingrese el primer número: ");  
console.log("el primer número es", primerNumero);  
let segundoNumero = readlineSync.questionInt("Ingrese el segundo número: ");  
console.log("el segundo número es", segundoNumero);
```

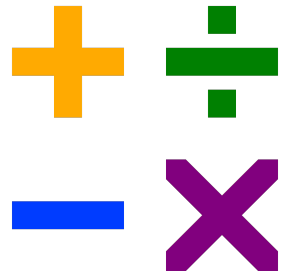


Secuencia

Ejercicio: Suma de Dos Números

- Realizamos la operación y mostramos el resultado

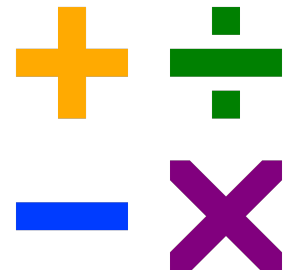
```
let resultado = primerNumero + segundoNumero;  
console.log("El resultado de la suma es:", resultado);
```



Secuencia

Ejercicio: Suma de Dos Números

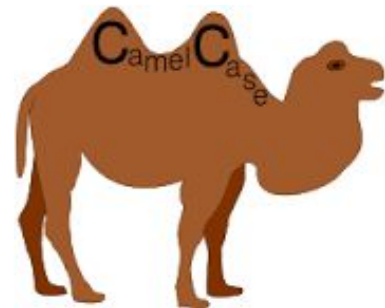
```
let readlineSync = require('readline-sync');  
let primerNumero = readlineSync.questionInt("Ingrese el primer número: ");  
console.log("el primer número es", primerNumero);  
let segundoNumero = readlineSync.questionInt("Ingrese el segundo número: ");  
console.log("el segundo número es", segundoNumero);  
let resultado = primerNumero + segundoNumero;  
console.log("El resultado de la suma es:", resultado);
```



Variables

Buenas Prácticas

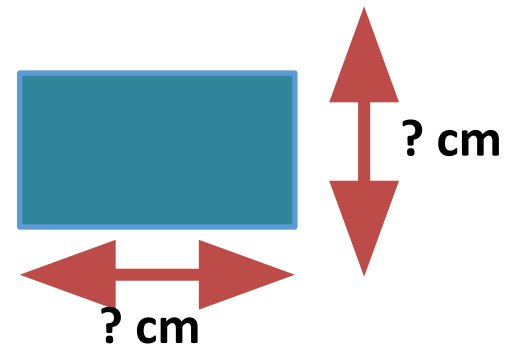
- Los nombres de las variables deben ser representativos
 - La falta de buenos nombres hace a nuestro programa muy difícil de entender y leer por nosotros y otros desarrolladores
- El nombre de una variable siempre comienza con una letra minúscula
- Si son varias palabras, se escribe en mayúsculas la primera letra de cada palabra (excepto la primera palabra)
 - Ejemplos: primerNumero, resultadoDeLaSuma



Secuencia

Ejercicio: Área del Rectángulo

- Volvamos a implementar el proceso que calcula el área de un rectángulo pero **para cualquier base o altura**
 - El usuario debe ingresar la base y altura por teclado
 - El área debe guardarse en una variable
 - El resultado debe ser mostrado por pantalla

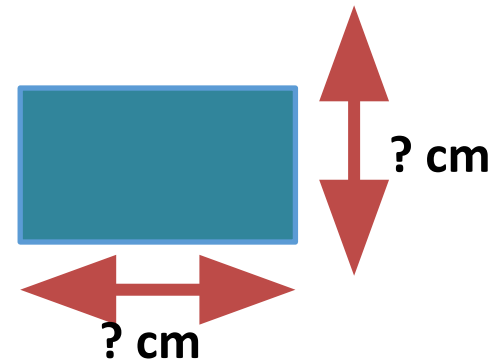


Secuencia

Ejercicio: Área del Rectángulo

- Leemos la base y la altura desde el teclado y las guardamos en las variables

```
let readlineSync = require('readline-sync');  
let base = readlineSync.questionInt("Ingrese la base: ");  
let altura = readlineSync.questionInt("Ingrese la altura: ");
```

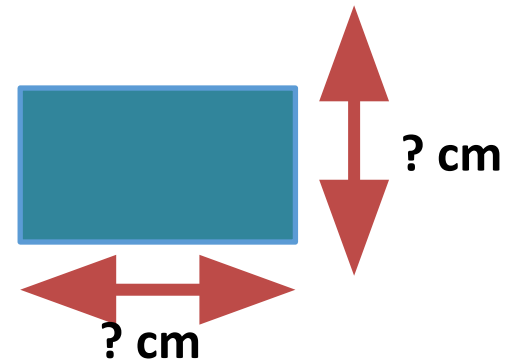


Secuencia

Ejercicio: Área del Rectángulo

- Calculamos el área y mostramos el resultado

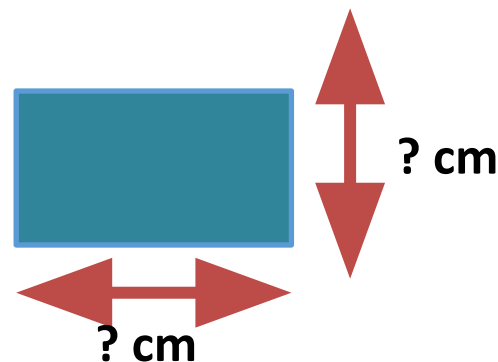
```
area = base * altura;  
console.log("El área es: ", area);
```



Secuencia

Ejercicio: Área del Rectángulo

```
let readlineSync = require('readline-sync');  
let base = readlineSync.questionInt("Ingrese la base: ");  
let altura = readlineSync.questionInt("Ingrese la altura: ");  
area = base * altura;  
console.log("El área es: ", area);
```



Técnicas de Programación

CFP Programador full-stack

Introducción (Profundización)

Secuencia

Ejercicio: Calculo de Descuento

- Implemente un algoritmo que calcule y muestre por pantalla el precio final de un producto después de aplicarle un descuento
 - El precio inicial del producto es \$450,50
 - El descuento a aplicar es del 10%. Recuerde que puede obtener el 10% de un valor multiplicado por 0,1
 - El precio y el descuento deben ser guardados en variables (no ingresados por teclado)

10% OFF

Estructuras de Control

Problema: Autos de Carrera

- En una prueba, un piloto tiene que completar 4 vueltas
- Se necesita un programa que permita ingresar por teclado el tiempo de cada vuelta
- El programa debe retornar el tiempo total y el promedio de vuelta

