

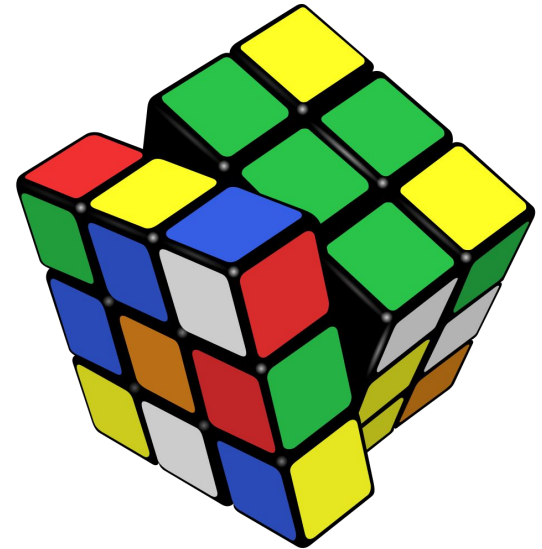
Técnicas de Programación

CFL
Programador
full-stack

Algoritmos Secuenciales

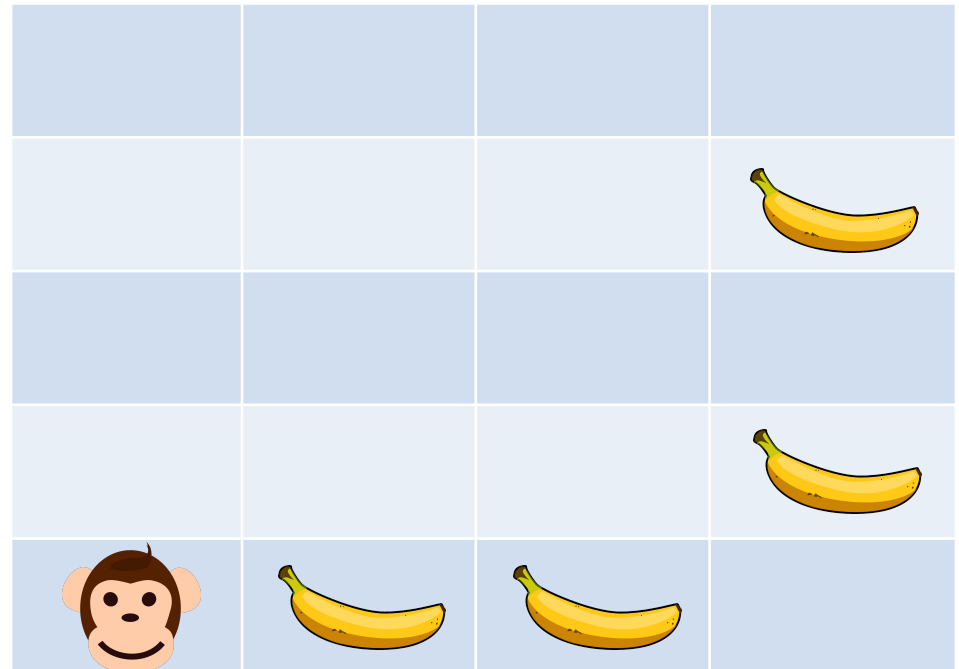
Algoritmos

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



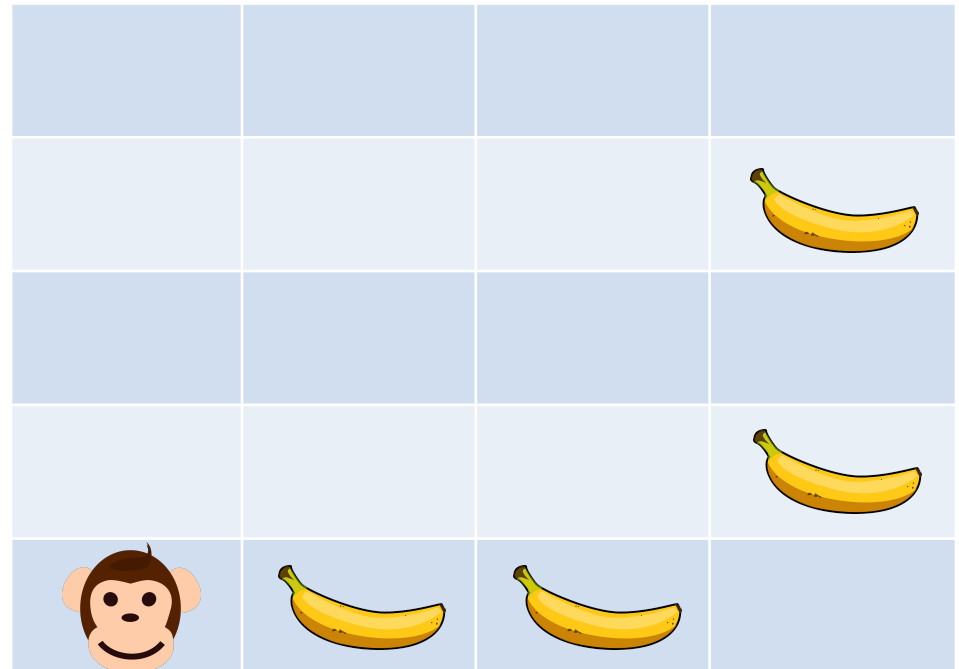
Algoritmos

- 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:**

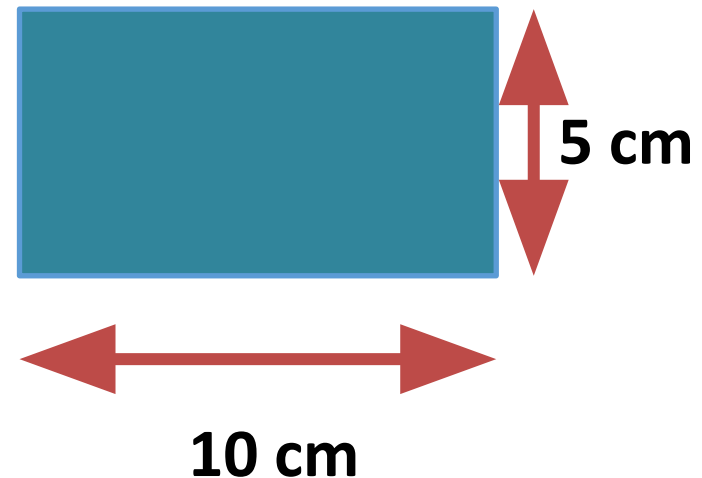
- Dato 1: altura 5 cm
- Dato 2: base 10 cm

- **Proceso:**

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

- **Salida:**

- $5 * 10 = 50$



Técnicas de Programación

CFL Programador full-stack

Algoritmos simples en Javascript (JS)

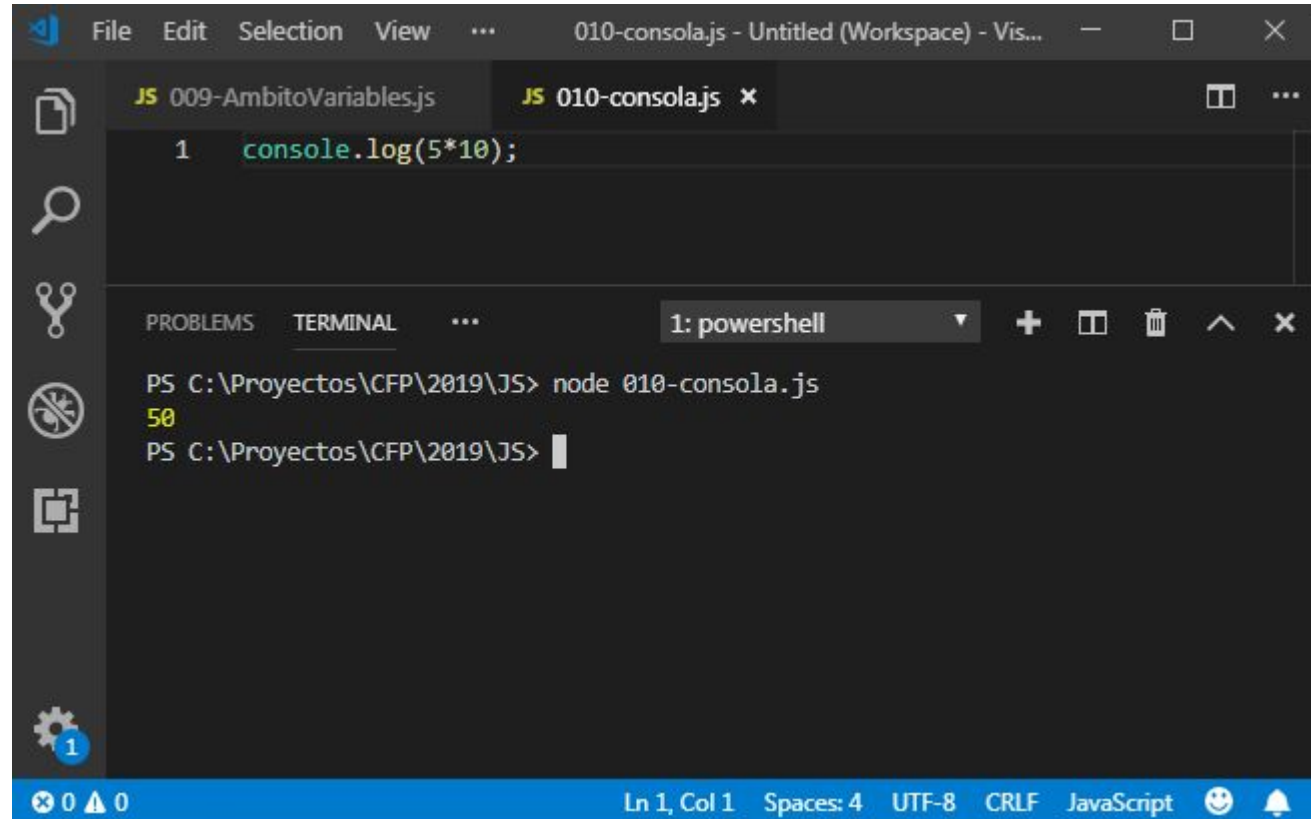
JavaScript - Glosario básico

- **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, let, new, require, return, switch, this, throw, try, typeof, 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', 'JavaScript', and a smiley face icon.

Implementando Algoritmos

Ejercicio: Área del Rectángulo 5x10



Una letra mal escrita puede hacer que la computadora no entienda el programa!

Debo aprender su idioma e incluso ser cuidadoso de escribirlo bien

JavaScript - Sintaxis básica

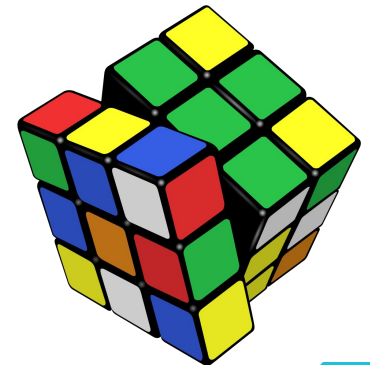
Las normas básicas que definen la sintaxis de JavaScript son las siguientes:

- **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



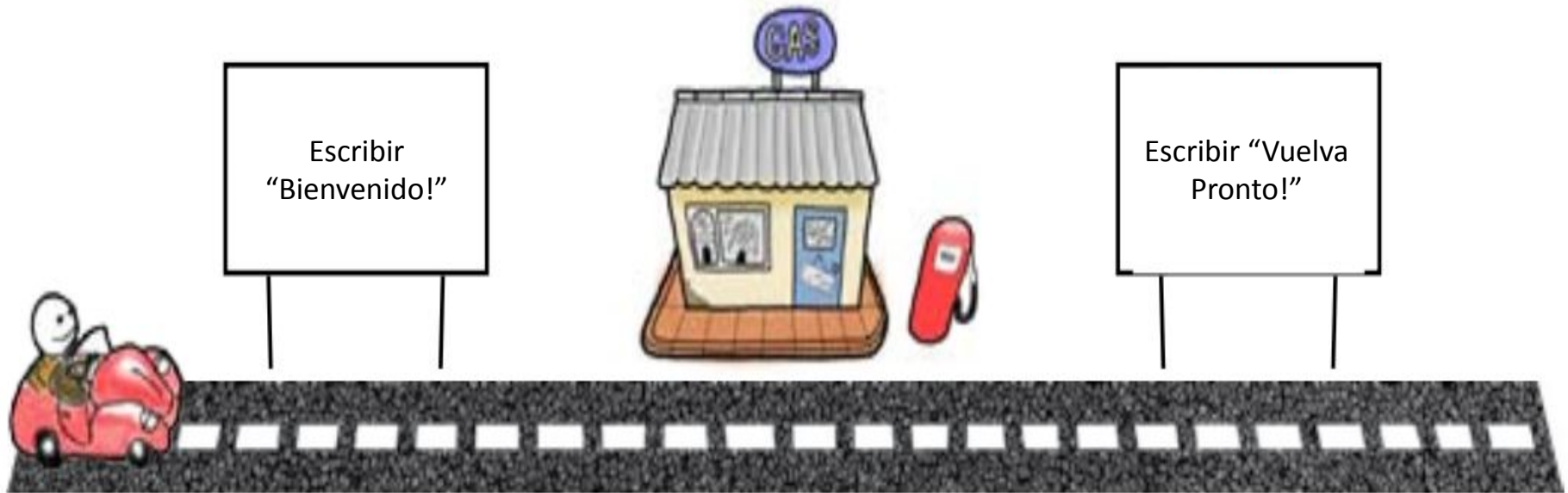
Estructuras de Control

Secuenciales

Selectivas o De
Decisión

Repetitivas

Secuencia

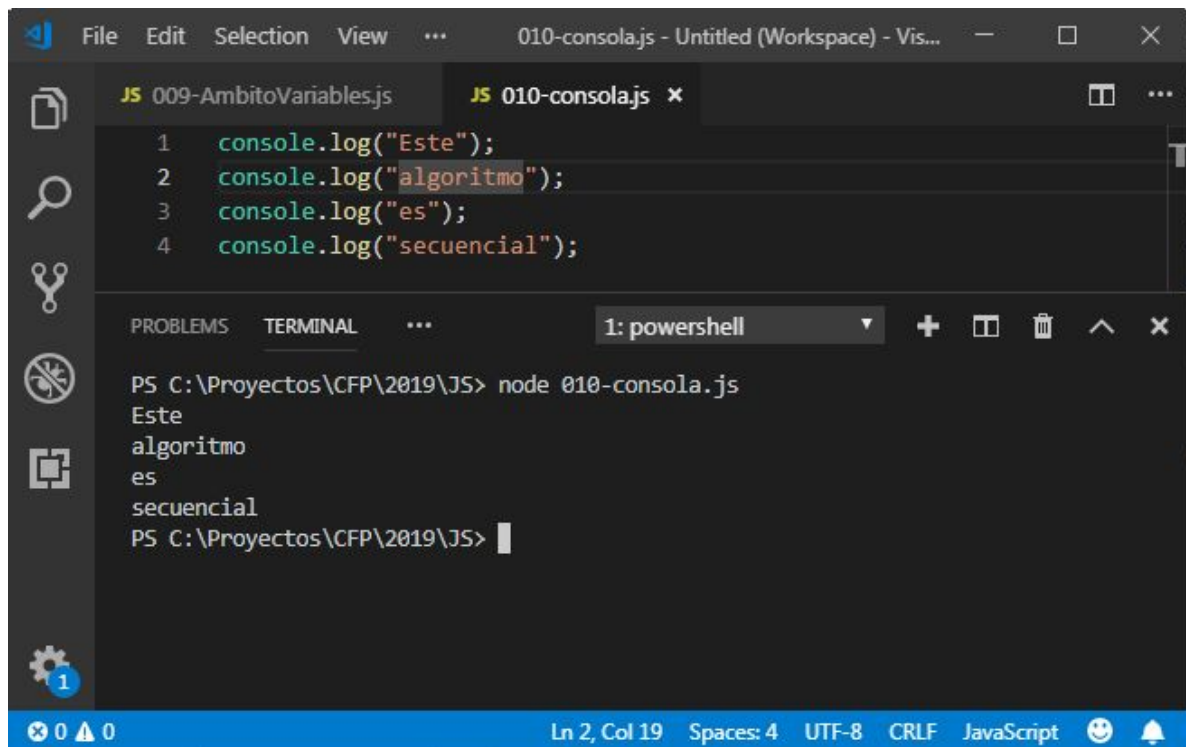


- Un algoritmo es una serie de pasos para resolver un programa
- Los algoritmos más simples son una lista de acciones que se ejecutan en orden

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 command 'node 010-consola.js' executed in a PowerShell window. The output of the command is displayed as follows:

```
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'.

Comparativa code.org



Es lo que hacemos en code.org al poner un bloque abajo de otro

//se ejecutan cuando hacemos node archivo.js

```
console.log ("Paso 1") ;
```

```
console.log ("Paso 2") ;
```



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
```

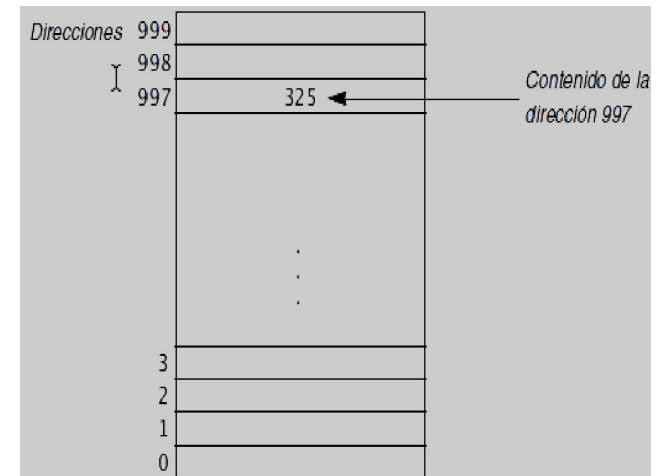
```
console.log ("secuencial") ;
```

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;
```

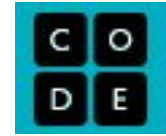
```
let numero_2 = 1;
```

```
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

Comparativa code.org



En este ejemplo de programa de bloques de code.org, la variable es **longitud** y se utiliza, como recordaran, para dibujar casas de distintos tamaño.



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

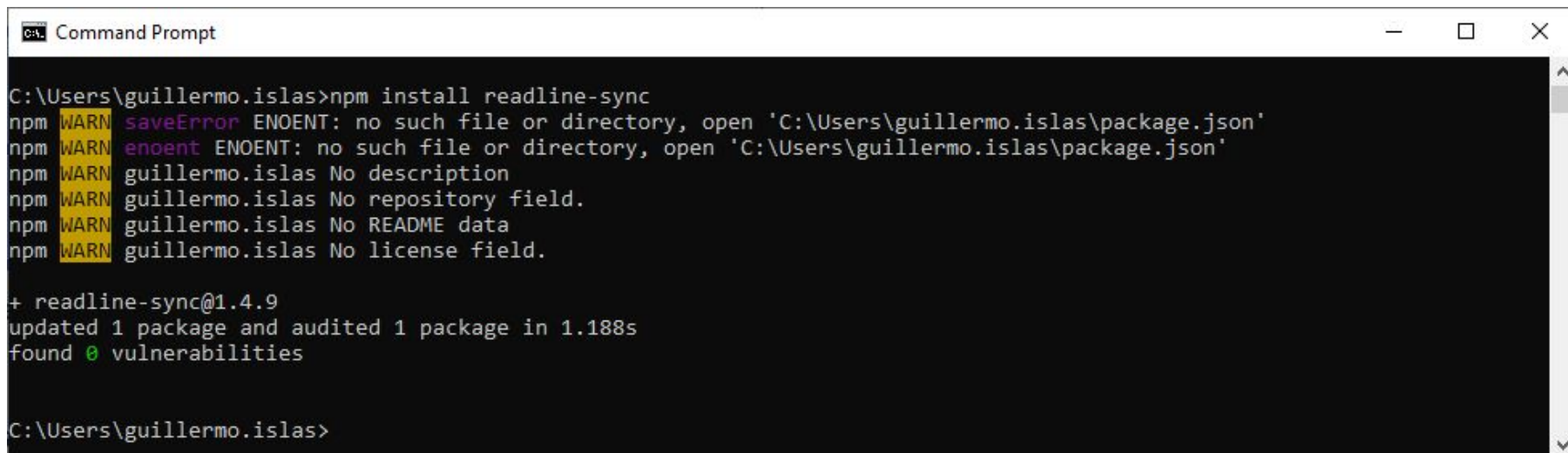
NodeJS - Instalación de paquete “*readline-sync*”
usando el comando “*npm*”

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

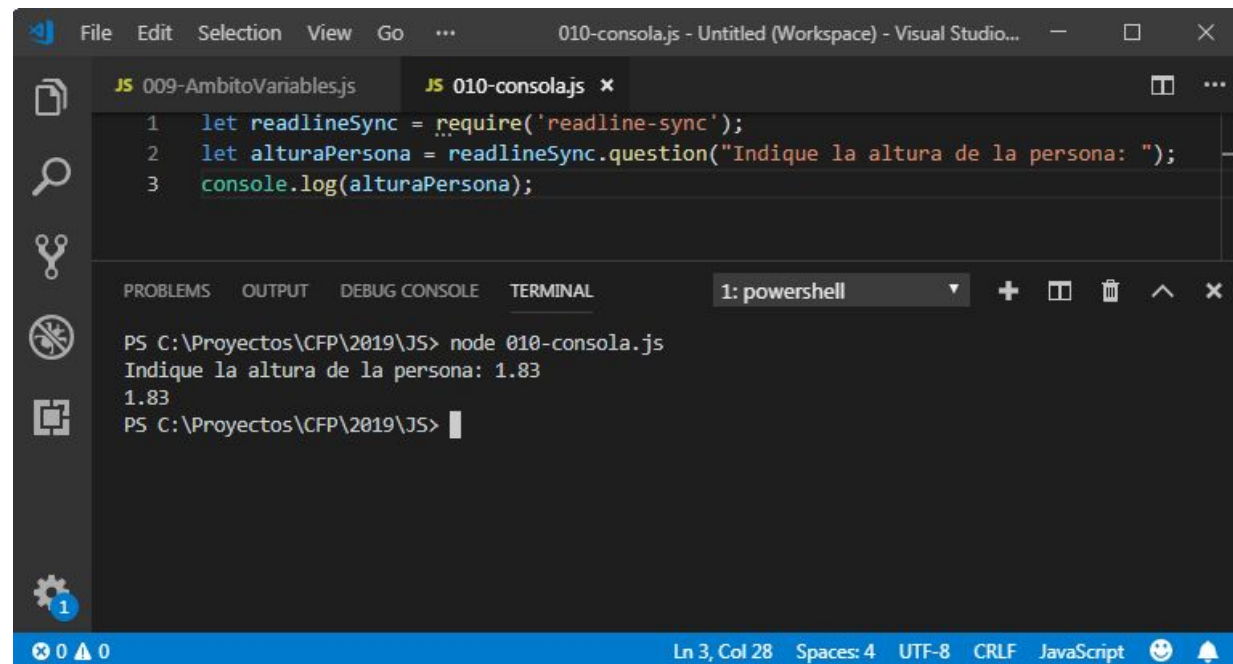
C:\Users\guillermo.islas>
```


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 active tab '010-consola.js' contains 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 is the 'TERMINAL' panel, which shows the command prompt output:

```
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 'Ln 3, Col 28', 'Spaces: 4', 'UTF-8', 'CRLF', and 'JavaScript'.

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

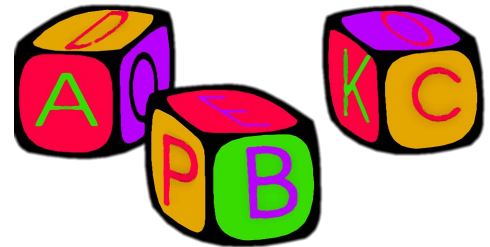
Tipos de Datos Básicos

- **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 qué se encierra con comillas dobles

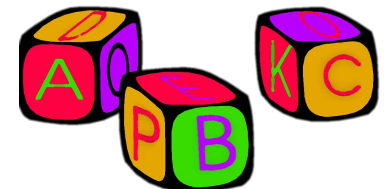
```
let texto1 = "Una frase con 'comillas simples' dentro";
```

// El contenido de texto2 tiene comillas dobles, por lo qué 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
```

- Pero siempre declararemos las variables de forma explícita.

Tipos

- Javascript tiene tipos dinámicos.
- Una misma variable puede cambiar de tipo.
- Puede causar confusiones (y errores que no encuentro durante horas).
- Siempre tengamos cuidado de asignar valores del mismo tipo a una variable.

```
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, el 10 indica la base numerica a utilizar
```

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

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

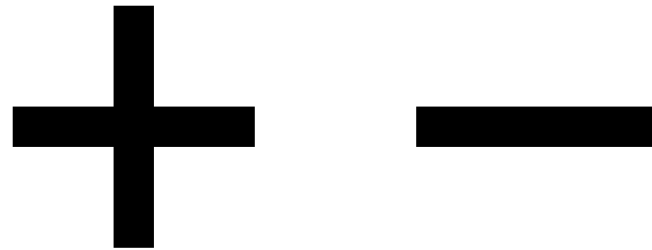
```
(''+nombre+', '+apellido+')
```

```
//se puede escribir de manera mas simple como:
```

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

Operadores

- Son símbolos especiales que sirven para ejecutar una determinada operación, devolviendo el resultado de la misma



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

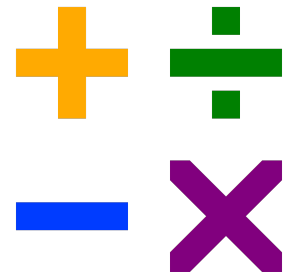
N° Prueba	Entrada		Salida	
	1er Num Ingresado	2do Num Ingresado	Suma	Mensaje
1	20	30	$20+30=50$	El resultado de la suma es: 50
2	15	150	$15+150=165$	El resultado de la suma es: 165
3	130	300	$130+300=430$	El resultado de la suma es: 430

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);
```

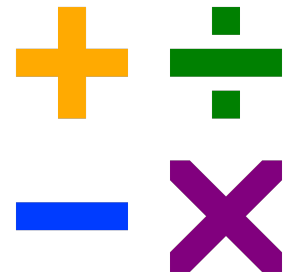


Secuencia

Ejercicio: Suma de Dos Números

- Realizamos la operación y mostramos el resultado

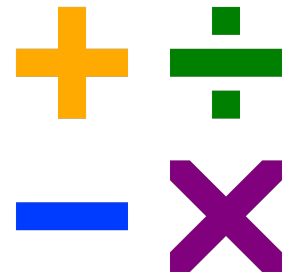
```
let resultado = primerNumero + segundoNumero;  
console.log("El resultado de la suma es:", resultado);
```



Secuencia

Ejercicio: Suma de Dos Números

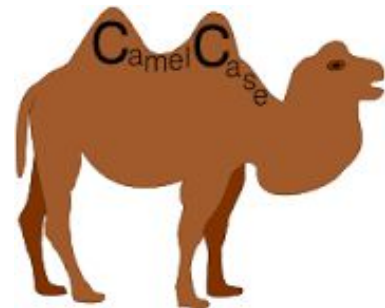
```
let readlineSync = require('readline-sync');  
let primerNumero = readlineSync.questionInt("Ingrese el primer número: ");  
console.log("el primer número es", primerNumero);  
let segundoNumero = readlineSync.questionInt("Ingrese el segundo número: ");  
console.log("el segundo número es", segundoNumero);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



Variables

Buenas Prácticas

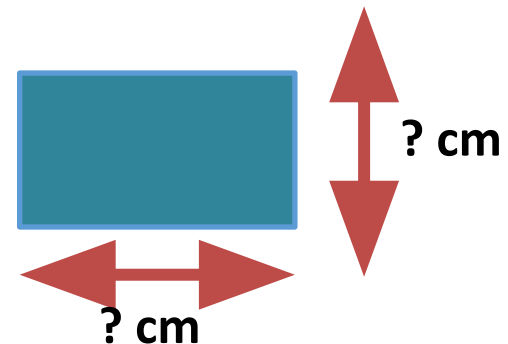
- Los nombres de las variables deben ser representativos
 - La falta de buenos nombres hace a nuestro programa muy difícil de entender y leer por nosotros y otros desarrolladores
- El nombre de una variable siempre comienza con una letra minúscula
- Si son varias palabras, se escribe en mayúsculas la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

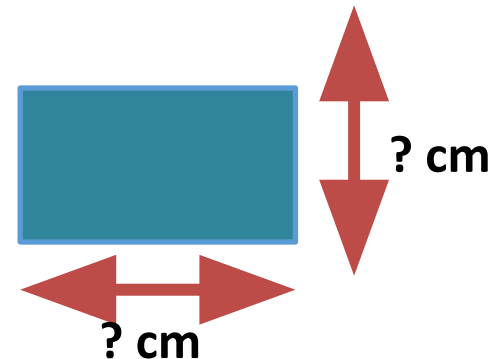
Ejercicio: Área del Rectángulo

- 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: ");
```

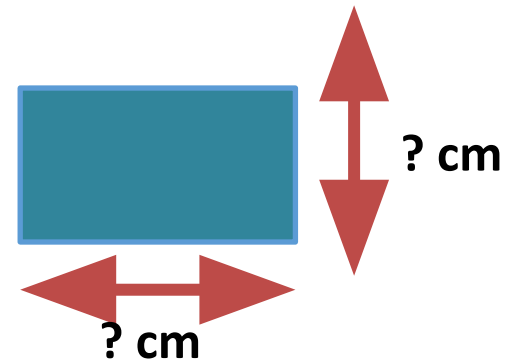


Secuencia

Ejercicio: Área del Rectángulo

- Calculamos el área y mostramos el resultado

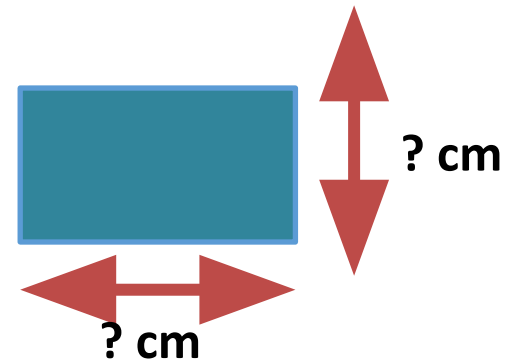
```
area = base * altura;  
console.log("El área es: ", area);
```



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

CFL Programador full-stack

Introducción (Profundización)

Secuencia

Ejercicio: Calculo de Descuento

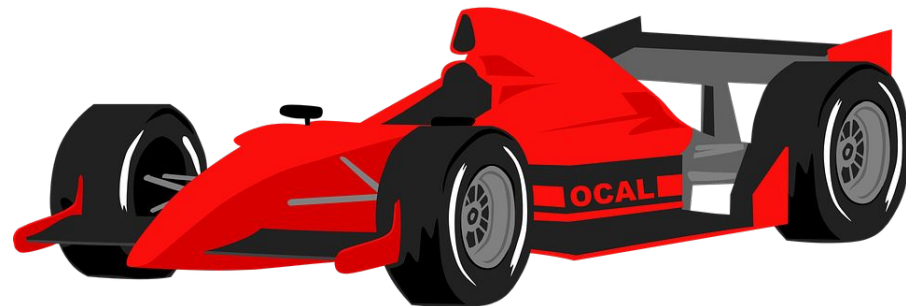
- Implemente un algoritmo que calcule y muestre por pantalla el precio final de un producto después de aplicarle un descuento
 - El precio inicial del producto es \$450,50
 - El descuento a aplicar es del 10%. Recuerde que puede obtener el 10% de un valor multiplicado por 0,1
 - El precio y el descuento 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



Técnicas de Programación

CFL Programador full-stack

Introducción (Resolución)

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

Secuencia

Ejercicio: Calculo de Descuento

Definir
variables



Calcular
descuento



Calcular
precio final

Secuencia

Ejercicio: Calculo de Descuento

```
p=450.5;  
variable1=0.1;  
descuento=p*variable1;  
precioFinal=p-descuento;  
console.log(precioFinal);
```



Las variables no fueron definidas

10% OFF

Secuencia

Ejercicio: Calculo de Descuento

```
let p=450.5;  
let variable1=0.1;  
let descuento=p*variable1;  
let precioFinal=p-descuento;  
console.log(precioFinal);
```



Los nombres de las variables no son representativos

10% OFF

Secuencia

Ejercicio: Calculo de Descuento

```
let precioProducto=450.5;  
let porcentajeDescuento=0.1;  
let descuento=precioProducto*porcentajeDescuento;  
let precioFinal=precioProducto-descuento;  
console.log(precioFinal);
```

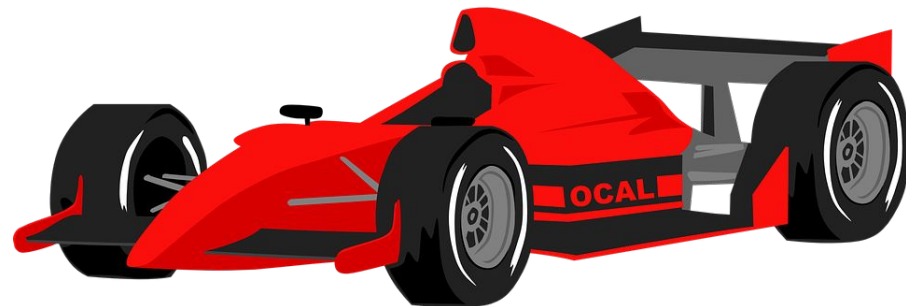


10% OFF

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

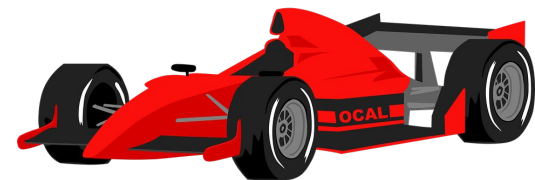


Estructuras de Control

Problema: Autos de Carrera

```
let readlineSync = require('readline-sync');  
let vuelta1 = readlineSync.questionInt("Ingrese la vuelta 1: ");  
let vuelta2 = readlineSync.questionInt("Ingrese la vuelta 2: ");  
let vuelta3 = readlineSync.questionInt("Ingrese la vuelta 3: ");  
let vuelta3 = readlineSync.questionInt("Ingrese la vuelta 4: ");  
let tiempoTotal=vuelta1+vuelta2+vuelta3+vuelta4;  
console.log("Tiempo total; ", tiempoTotal);  
console.log("Promedio de vuelta: ", (vuelta1+vuelta2+vuelta3+vuelta4)/4);
```

**vuelta4 nunca fue
inicializada (en la suma
contiene basura)**

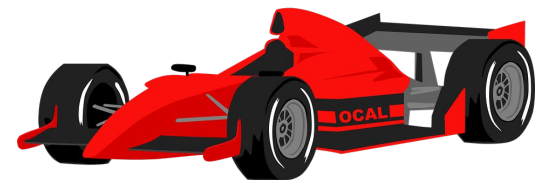


Estructuras de Control

Problema: Autos de Carrera

```
let readlineSync = require('readline-sync');  
let vuelta1 = readlineSync.questionInt("Ingrese la vuelta 1: ");  
let vuelta2 = readlineSync.questionInt("Ingrese la vuelta 2: ");  
let vuelta3 = readlineSync.questionInt("Ingrese la vuelta 3: ");  
let vuelta4 = readlineSync.questionInt("Ingrese la vuelta 4: ");  
let tiempoTotal=vuelta1+vuelta2+vuelta3+vuelta4;  
console.log("Tiempo total; ", tiempoTotal);  
console.log("Promedio de vuelta: ", (vuelta1+vuelta2+vuelta3+vuelta4)/4);
```

**El calculo ya se realizó y
su resultado esta en
tiempoTotal**



Estructuras de Control

Problema: Autos de Carrera

```
let readlineSync = require('readline-sync');  
let vuelta1 = readlineSync.questionInt("Ingrese la vuelta 1: ");  
let vuelta2 = readlineSync.questionInt("Ingrese la vuelta 2: ");  
let vuelta3 = readlineSync.questionInt("Ingrese la vuelta 3: ");  
let vuelta4 = readlineSync.questionInt("Ingrese la vuelta 4: ");  
let tiempoTotal=vuelta1+vuelta2+vuelta3+vuelta4;  
console.log("Tiempo total; ", tiempoTotal);  
console.log("Promedio de vuelta: ", tiempoTotal/4);
```

