Ejercicios

0.6 Estructuras de control

Introducción

Primero vamos a aprender a hacer que nuestro programa nos permita introducir datos. Para ello, vamos a instalar la librería *readline-sync* que nos hará la tarea más sencilla. Ejecutaremos la siguiente línea en la línea de comandos dentro del directorio donde vamos a escribir el ejercicio:

```
npm i readline-sync
```

Eso instalará la librería creando los archivos *"package"* y la carpeta *"node_modules"*.

Para usar la librería podemos escribir las siguientes líneas en el archivo *ejercicio.js*:

```
1  const readLine = require("readline-sync");
2
3  const num = readLine.question("Introduce un número: ");
4
5  console.log("El número es:", num);
```

Con la línea 1, vamos a traer la librería readline-sync que nos permitirá interactuar con la consola.
Con la línea 3 Utilizaremos el método question para leer por la consola.
Con la línea 5 escribiremos por la consola el número que hemos introducido.

El método question realmente recoge un string, por lo que podremos escribir lo que queramos.

Para ejecutar el código, escribiremos en la consola:

```
node ejercicio.js
```

Si queremos asegurarnos de que el valor introducido es numérico, podemos continuar el programa así:

```
1  const readLine = require("readline-sync");
2
3  const input = readLine.question("Introduce un número: ");
4
5  const num = parseInt(input);
6  if (isNaN(num)) {
7    console.log("No has introducido un número");
8  } else {
9    console.log("El número es:", num);
10 }
```

Ahora, la línea 5 traduce el string input al número num. Si input no es un número, parseInt devolverá un valor especial en JavaScript, NaN (Not a Number).

Ejercicios

0.6 Estructuras de control

Entrega

- Cada ejercicio deberá entregarse en un archivo JavaScript (.js) subido a la plataforma del curso.

- La entrega deberá ir acompañada de al menos un diagrama de flujo en formato imagen (se recomienda usar un software de diagramas como Draw.io o LucidChart) que represente el código de la solución de un ejercicio. El ejercicio a escoger se deja a criterio del alumno. Si se entregan más diagramas todos serán corregidos, pero al menos debe entregarse uno. Si no se entrega ningún diagrama la entrega se dará por insatisfecha.

Evaluación

Se evaluarán los siguientes puntos:
- Que el código funcione (que no de error).
- Que el código satisfaga el enunciado.
- Que el código no tenga redundancias
- Que el código esté correctamente indentado.
- Que el diagrama sea claro, correcto, y represente el código fielmente.

Ejercicios

Ejercicio 1

Escribe un programa que pida al usuario un número e imprima por pantalla si el número es **par** o **impar** y si es **positivo** o **negativo**. Si la entrada no es un número, se deberá pintar "No es un número".

Ejemplos:

```
$> node ejercicio1.js

Introduce un número: -3

El número es negativo e impar

$>
```

```
$> node ejercicio1.js

Introduce un número: patata

No es un número

$>
```

Ejercicios

0.6 Estructuras de control

Ejercicio 2

Escribe un programa que pida al usuario tres números y que pinte por pantalla si al menos uno de ellos es par.

Ejemplos:

```
$> node ejercicio2.js

Introduce un número: 10
Introduce otro número: 3
Introduce otro número: 0

Hay al menos un número par

$>
```

```
$> node ejercicio2.js

Introduce un número: 1
Introduce otro número: 3
Introduce otro número: 77

No hay números pares

$>
```

Ejercicio 3

Escribe un programa que tome un número como entrada y calcule su factorial.

Ejemplo:

```
$> node ejercicio3.js

Introduce un número: 10

El factorial de 10 es 3628800

$>
```

Ejercicio 4

Escribe un programa que tome un número como entrada e imprima la tabla de multiplicar de ese número del 1 al 10.

Ejemplo:

```
$> node ejercicio4.js

Introduce un número: 3

3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

$>
```

Ejercicio 5

Escribe un programa que tome una cadena como entrada y cuente el número de vocales (a, e, i, o, u) en la cadena.

Ejemplo:

```
$> node ejercicio5.js

Introduce una frase: Juntos podremos romper un iceberg

El número de vocales es 11

$>
```

Ejercicio 6

Escribe un programa que pida al usuario un año y que imprima por pantalla si dicho año es bisiesto o no.

Ver algoritmo computacional

Ejemplo:

```
$> node ejercicio6.js

Introduce un año: 1024

El año es bisiesto

$>
```

Ejercicio 7

Escribe un programa que reciba un número del usuario y pinte un triángulo de asteriscos cuya altura sea el número recibido.

Ejemplo:

```
$> node ejercicio7.js

Introduce la altura del triángulo: 5

    *
   ***
  *****
 *******
*********

$>
```

Ejercicio 8

Escribe un programa que reciba dos números por consola que serán la base y la altura de un rectángulo de asteriscos que se deberá pintar por pantalla.

Ejemplo:

```
$> node ejercicio8.js

Introduce la altura del rectángulo: 5
Introduce la anchura del rectángulo: 7

*******
*     *
*     *
*     *
*******

$>
```

Ejercicio 9

Escribe un programa que solicite un número al usuario (del 0 al 10) y devuelva el número escrito en letras. Si el valor introducido no está dentro del rango 0-10 o no es un número se deberá volver a solicitar un número.

Ejemplo:

```
$> node ejercicio9.js

Introduce un número del 0 al 10: ornitorrinco

Introduce un número del 0 al 10: 8

Ocho

$>
```

Ejercicio 10

Escribe un programa que genere un número aleatorio del 1 al 10.
Haz que el programa pregunte al usuario números hasta que acierte el número aleatorio generado.
El programa debe mantener una cuenta de cuántos intentos han hecho falta hasta acertar el número. Cuando el usuario acierte el número aleatorio, el programa mostrará por pantalla el número de intentos.

Ayuda: Para generar un número aleatorio, puedes usar la siguiente línea de código:

```
1  const randomNumber = Math.floor(Math.random() * 10) + 1;
```

Ejemplo:

```
$> node ejercicio10.js

He pensado un número del 1 al 10. ¡Intenta adivinarlo!

Adivina: 1

¡Fallo!

Adivina: 3

¡Correcto! Has necesitado 2 intentos
```


```
$>
```