25-Jan-2023

JavaScript

* what is JS?
→ JS is a high level dynamic interpreted programming language.

Q1) Write a program that asks the user for a number n and prints the sum of the numbers 1 to n.

// this only executes in browser console

```
const n = parseInt(prompt("enter the number: "));
let sum = 0;
for(let i=1; i<=n; i++){
    sum += i;
}
console.log(`the sum of the numbers 1 to ${n} is ${sum}`);
```

→ parseInt() converts the string into int value

→ prompt() is used to display a dialog box in the browser, which prompts the user for input.

```
const readline = require('readline-sync');
let n = readline.question("Please enter a
                numbe: ");
let sum = 0;
for(let i = 1; i<=n; i++){
    sum += i;
}
console.log('The sum of 1 to ${n}
                is ${sum}');
```

→ prompt() is not available outside the browser envt, so, we can use 'readline-sync' to get user input

→ we need to install it before using it

by 'npm readline install readline-sync'

* difference between var, let and const

1. var declarations are globally scoped or function scoped while let and const are block scoped.

2. var variables can be updated and re-declared within its scope
- let variables can be updated but not re-declared

ex: let greeting = "hello"; (re-declared)
let greeting = "hi"; ✗

let greeting = "hello"; (updated)
greeting = "hi"; ✓

- const variables can neither be updated nor re-declared.

3. They are all hoisted to the top of their scope. But while var variable are initialized with undefined, let and const variables are not initialized

4. While var and let can be declared without being initialized, const must be initialized during declaration

(2)(a) Modify the previous program such that only multiples of three or five are considered in the sum.

```
const readline = require('readline-sync');
let n = readline.question("Enter the
  number: ");

let sum = 0;
if (let i = 1; i<=n; i++){
    if ((i%3==0)||(i%5==0)){
        sum += i;
    }
}
console.log('The sum of multiples 3 or
  5 from 1 to ${n} is ${sum}');
```

PAGE NO.:
DATE: / /

Q) Make a function that returns "even" or "odd" depending on the number passed to it

parity(1); → "odd"
parity(2); → "even"

```
const readline = require('readline-sync');
let num = readline.question("Enter a num");

function parity(num){
    if(num % 2 ==0){
        return "even";
    }
    else{
        return "odd";
    }
}
console.log(parity(num));
```

Output:

> node parity-fun.js
> enter a num:
> 10
> even.

* difference between "==" and "===".

- "==" is used for comparing two variable but it ignores the datatypes of variables
- "===" is used for comparing two variables but its operator also checks datatype and compares two values.
- "==" return true only if the two operand are equal.
- "===" returns true only if both values are the same for the two variables.

PAGE NO.:
DATE: / /

Q4) Make a function that takes number of flips as parameter and returns the fraction that were heads.

headsRation(10); → 0.7

```
const readline = require('readline-sync');
let numFlips = readline.question("Enter
                number of flips");

function headsRatio(numFlips){
    let headsCount = 0;
  for (let i=0 ; i<numFlips; i++){
      if (Math.random() < 0.5){
          headsCount ++;
      }
  }
    return headsCount / numFlips;
}

console.log(headsRatio(numFlips));
```

→ math.random() returns a random medical decimal number between 0 and 1.
if its less than 0.5, it represents head and if greater than 0.5 it represents tail.

(Q5) Write a program that prints the next 20 leap years

```
let currentYear = new Date().getFullYear();
let count = 0;
while (count < 20){
  if ( currentYear %. 4 ===0 && ((currentYear
      %. 100 !== 0) || (currentYear %. 400 ==0))){

      console.log (currentYear);
        count++;
    }

    currentYear++;
}
```

→ Date() is object and Date().getFullYear() object giver current year ex: 2023

→ while loop to check if the current year is a leap year or not

→ By checting currentYear divisiby by 4 and 400 but not divisible by 100

→ increament the count variable if true.

PAGE NO.:
DATE / /

* difference between != and !== operator.

→ '!=' checks the unequality of two objects without making the type check.
→ it converts the datatype of two operands to one and then compares their value.
ex: 1 != '1' will results false.

→ '!==' operator

In JavaScript, the '!=' operator is the "not equal" operator and compares whether two values are not equal.

The "!==" operator is the strict not equal operator and compares whether two values are not equal and also of the same type.

For example:
```
console.log( 1 != '1' ); // False
console.log( 1 !== '1' ); // true.
```

→ the values 1 and '1' are not equal but since one of them is a number and the other is a string, the != op returns false