JavaScript Intro: Variables, I/O, operatios, data types

Video

JavaScript Variables Exercise

Required Practice

For this practice, we are going to identify and operate the different data types and structures of javascript. Create a document specifying the answers to each part of this practice.

PART I:

For each of the following variables we need to identify the data type:

1. let age = 25;
2. const name = "John";
3. var isStudent = true;
4. let salary = 1500.75;
5. const fruits = ["apple", "banana", "orange"];
6. let person = { name: "Alice", age: 30, isEmployed: true };

PART II:

Now, it is time to write a function **sumNumbers** that takes an array of elements as input, returns and prints to the console the sum of all numeric elements in the array, the non-numeric elements should be ignored.

For example:

sumNumbers([10, "hello", 5, true, 8]) // should return 23

Additional Resources

[Data types and structures]()

[Variables]()

JS Intro: Conditionals, loops

Video

Conditionals - Loops Exercise

Required Practice

PART I

For this practice, we are going to write a JavaScript program that analyzes a given range of numbers and performs the following tasks:

For each number in the range from 1 to 20 (inclusive):

- If the number is even, print the number followed by " is even."
- If the number is odd, print the number followed by " is odd."
- If the number is divisible by 5, print the number followed by " is divisible by 5."

We should take into account the following instructions:

- Create a loop that iterates through the numbers from 1 to 20.

- Use conditional statements to determine if a number is even, odd, or divisible by 5.
- Print out the appropriate message based on the conditions.

PART II

Now that we are able to join conditionals and loops together, create a program to determine if a given number is a prime number. Remember a number is considered prime when it is divisible only by itself and by 1.

Tip: Use the Module operator to determine when a number is divisible by another.

Additional Resources

Loops and iteration

Conditional statement

JS Intro: Functions, Collections

Video

Functions and Collections Exercise

Required Practice

For this practice we are going to implement the use of functions and collections in javascript to convert different temperatures from celsius to fahrenheit:

- It is necessary to define and implement the function **celsiusToFahrenheit()** which takes an array of temperatures in Celsius and converts the given Celsius temperature to Fahrenheit.
- On the other hand, the **.map()** method should be implemented to apply this function to each temperature in the **celsiusTemperatures** array, creating a new array **fahrenheitTemperatures** with the converted values.
- Finally, the original and converted temperature arrays will be printed to the console.
- Make sure to use multiple functions on this practice, one to convert from celsius to fahrenheit, and another one to get the initial values, call the celsius to fahrenheit function, and print the results.

Additional Resources

Keyed collections

Functions

Extra exercises

JS Intro: Promises

Video

Promises Exercise

Complementary Practice

For this practice, we are going to create a simple promise-based that simulates fetching data from a server using a promise, taking into account the following instructions:

1. We need to define a function named **fetchData** that takes a parameter url, representing the URL from which to fetch data.
2. Once the **fetchData** function is created, we create a new promise and within its constructor we use the setTimeout function to simulate a delay of 2 seconds..
3. After the delay, let's randomly resolve the promise with a mock data object or reject it with a mock error object.
4. Outside the Promise constructor, return the created promise.
5. Finally, let’s call the **fetchData** function with a URL and handle both the resolved and rejected cases using **.then()** and **.catch()**.

The URL, and the mock data can be defined with an API URL, as **https://swapi.dev/** , optionally, add the use of a library like Axios, or Chai HTTP, to successfully retrieve the data via API services.

Additional Resources

Promises

Promises challenges