**Array Methods::**

**1.a.**

**🡪Coding::**

fetch("https://restcountries.com/v3.1/all")

.then(response => {

if (!response.ok) {

throw new Error("Network response was not ok");

}

return response.json();

})

.then(data => {

const asiaCountries = data.filter(country => country.region === "Asia");

console.log("Countries in Asia:", asiaCountries);

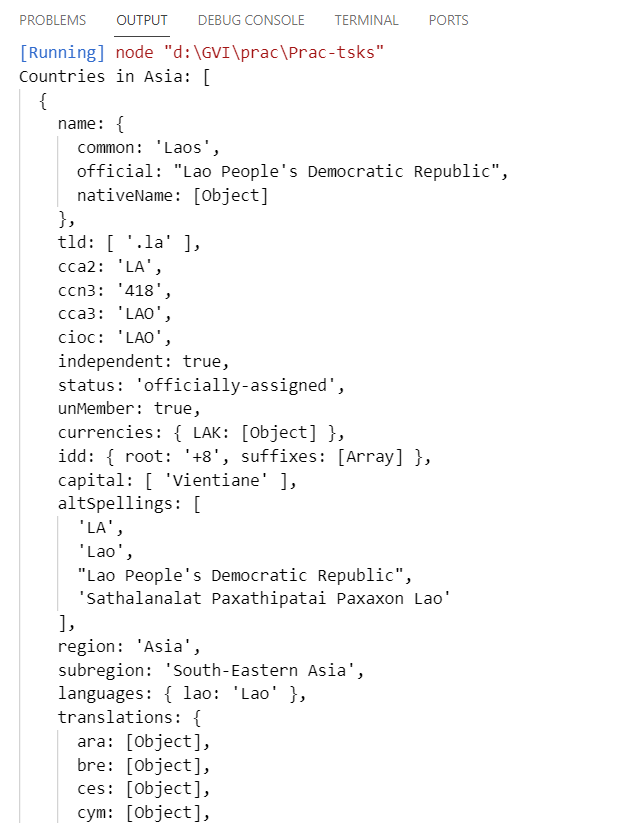
})

.catch(error => console.error("Error fetching data:", error));

**🡪Explanation::**

This code fetches the data from the provided URL, parses it as JSON, then uses the **filter** method to filter out the countries where the **region** property is equal to "Asia". And then it logs the countries from Asia to the console.

**🡪Output::**



**1.b.**

**🡪Coding::**

fetch("https://restcountries.com/v3.1/all")

  .then(response => {

    if (!response.ok) {

      throw new Error("Network response was not ok");

    }

    return response.json();

  })

  .then(data => {

    const countriesWithLowPopulation = data.filter(country => {

      return country.population < 200000;

    });

    console.log("Countries with population less than 2 lakhs:", countriesWithLowPopulation);

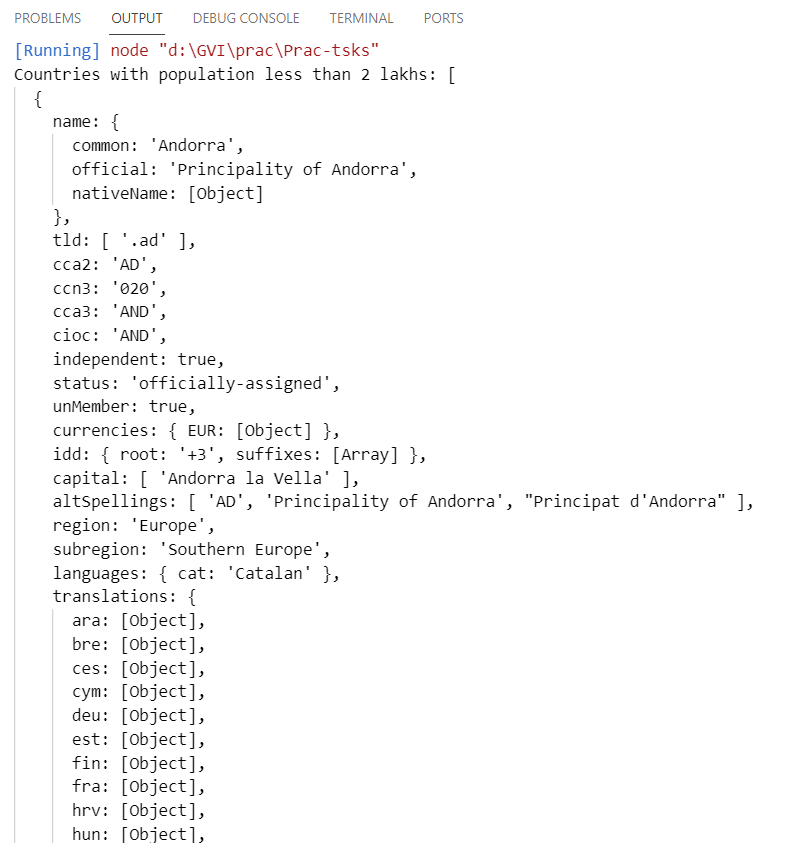
  })

  .catch(error => console.error("Error fetching data:", error.message));

**🡪Explanation::**

This code fetches the data from the provided URL, parses it as JSON, then uses the **filter** method to filter out the countries with a population of less than 200,000. And then it logs the filtered countries to the console. If there's an error during the fetch or JSON parsing, it will be caught and logged to the console.

**🡪Output::**



**1.c.**

**🡪Coding::**

fetch("https://restcountries.com/v3.1/all")

.then(response => {

if (!response.ok) {

throw new Error("Network response was not ok");

}

return response.json();

})

.then(data => {

data.forEach(country => {

const { name, capital, flags } = country;

console.log("Name:", name.common);

console.log("Capital:", capital[0]);

console.log("Flag:", flags.svg);

console.log("--------------------");

});

})

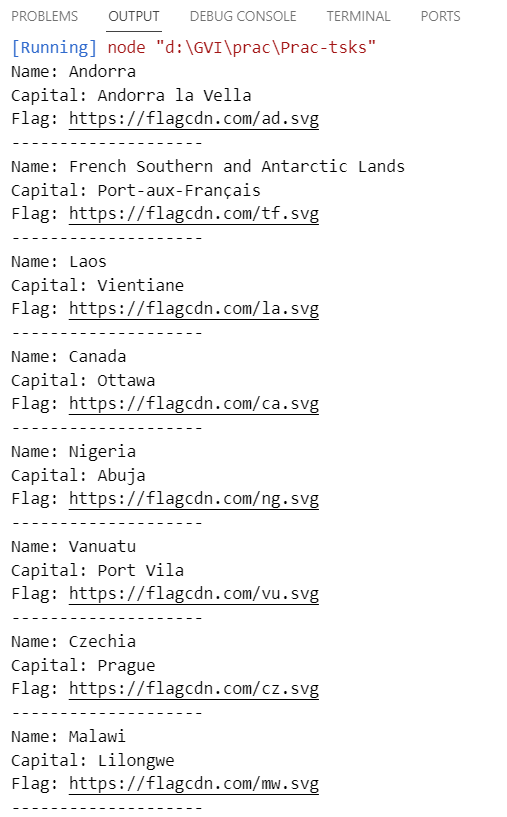
.catch(error => console.error("Error fetching data:", error.message));

**🡪Explanation::**

In this code:

* We fetch the data from the provided URL and parse it as JSON.
* Then, we iterate over each country object using the **forEach** method.
* Inside the loop, we destructure the country object to extract the **name**, **capital**, and **flags** properties.
* We then log the name, capital (assuming it's an array and taking the first element), and the flag's SVG URL to the console for each country.

**🡪Output::**



**1.d.**

**🡪Coding::**

fetch("https://restcountries.com/v3.1/all")

.then(response => {

if (!response.ok) {

throw new Error("Network response was not ok");

}

return response.json();

})

.then(data => {

const totalPopulation = data.reduce((accumulator, country) => {

return accumulator + (country.population || 0); // Using 0 as default value if population is missing

}, 0);

console.log("Total population of all countries:", totalPopulation);

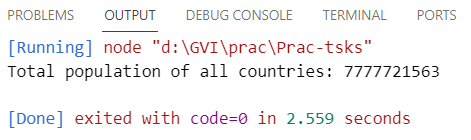
})

.catch(error => console.error("Error fetching data:", error.message));

**🡪Explanation::**

This code fetches the data from the provided URL, parses it as JSON, then uses the **reduce** method to iterate over each country object. Inside the reducer function, I accumulate the population of each country to calculate the total population. If the population data is missing for any country, we consider it as 0. Finally, I log the total population of all countries to the console.

**🡪Output::**



**1.e.**

**🡪Coding::**

fetch("https://restcountries.com/v3.1/all")

.then(response => {

if (!response.ok) {

throw new Error("Network response was not ok");

}

return response.json();

})

.then(data => {

const countriesUsingUSD = data.filter(country => {

const currencies = country.currencies;

return currencies && currencies.USD;

});

console.log("Countries using US dollars as currency:", countriesUsingUSD);

})

.catch(error => console.error("Error fetching data:", error.message));

**🡪Explanation::**

In the above coding,

* I check if the **currencies** property of each country object includes the code "USD".
* Thus filter out the countries that use US dollars as their currency.