**1.how to compare two JSON have the same properties without order**

**let obj1={name:"person1",age:5};**

**let obj2={age:5,name:"person1"};**

To compare two JSON objects without considering the order of their properties,

you can follow these steps:

1.Create an empty array.

2. Use a for loop to iterate through the properties of the first JSON

object. For each property, push the property name and its

corresponding value in the first JSON object into the array.

3.Convert the array into a JSON object.

4.Repeat steps 2 and 3 for the second JSON object.

5.Compare the two newly created JSON objects.

function that implements this approach:

**javascript :**

function isSameJsonWithoutOrder(obj1, obj2) {

**// Helper function to convert an array of key-value pairs into a JSON object**

function arrayToObject(arr) {

let obj = {};

for (let i = 0; i < arr.length; i += 2) {

obj[arr[i]] = arr[i + 1];

}

return obj;

}

**// Converting objects to arrays of key-value pairs without considering the order**

let arr1 = Object.keys(obj1).sort().map(key => [key, obj1[key]]);

let arr2 = Object.keys(obj2).sort().map(key => [key, obj2[key]]);

**// Converting arrays to JSON objects**

let obj1Sorted = arrayToObject(arr1);

let obj2Sorted = arrayToObject(arr2);

**// Converting JSON objects to strings without spaces, brackets, and commas**

const strObj1Sorted = JSON.stringify(obj1Sorted).replace(/[\s\[\]\,\]+/g, '');

const strObj2Sorted = JSON.stringify(obj2Sorted).replace(/[\s\[\]\,\]+/g, '');

// **Comparing strings**

return strObj1Sorted === strObj2Sorted;

}

**Usage example**:

**javascript**

|  |
| --- |
| let obj1 = { a: 1, b: 2, c: 3 };  let obj2 = { b: 2, a: 1, c: 3 };  let obj3 = { b: 2, a: 1, c: 4 };  console.log(isSameJsonWithoutOrder(obj1, obj2)); // true  console.log(isSameJsonWithoutOrder(obj1, obj3)); // false |

This solution addresses the limitation mentioned earlier by ensuring that both JSON objects have the same structure, meaning they have the same number of properties with the same names and the same values. However, it doesn't consider the order of the properties, which is why { a: 1, b: 2, c: 3 } and { b: 2, a: 1, c: 3 } are considered as the same JSON object.

**2.use the rest countries 'API URL->https://restcountries.com/v3.1/all and display all the country flags in the console ?**

Step 1: Visit the "rest countries" API URL (<https://restcountries.com/v3.1/all>)

to get a JSON array of all countries.

Step 2: Fetch the data from the API URL using the JavaScript Fetch API.

**javascript**

|  |
| --- |
| fetch('https://restcountries.com/v3.1/all') |

Step 3: Convert the response to JSON.

|  |
| --- |
| .then(response => response.json()) |

Step 4: Process the data and display the country flags in the console.

|  |
| --- |
| .then(data => {  data.forEach(country => {  console.log(country.name.common);  });  }) |

code will fetch all the countries' data, and then it will loop through each country. It will log the common name of each country to the console.

**3.use the same rest countries and print all countries names , regions ,sub region and population**

**code**:

Step 1: Fetch the data from the API URL using the JavaScript Fetch API.

javascript

|  |
| --- |
| fetch('https://restcountries.com/v3.1/all') |

Step 2: Convert the response to JSON.

|  |
| --- |
| .then(response => response.json()) |

Step 3: Process the data and display the desired information in the console.

|  |
| --- |
| .then(data => {  data.forEach(country => {  console.log(`Name: ${country.name.common}, Region: ${country.region}, Sub Region:  ${country.subregion},  Population: ${country.population}`);  });  }) |

code will fetch all the countries' data, and then it will loop through each country. It will log the name, region, sub-region, and population of each country to the console.

To retrieve the data and print it to the console, you can use the following code:

|  |
| --- |
| fetch('https://restcountries.com/v3.1/all')  .then(response => response.json())  .then(data => {  data.forEach(country => {  console.log(`Name: ${country.name.common}, Region: ${country.region}, Sub Region: ${country.subregion}, Population: ${country.population}`);  });  })  .catch(error => {  console.error('Error fetching data:', error);  }); |

This code includes a .catch() block to handle any errors that may occur during the fetching and processing of the data.