1.For the given JSON iterate over all for loops (for, for in, for of, forEach)

### **The ‘For’ Loop**

The For Loop comes first because of its simplicity and ease of use. It is a very user-friendly kind of loop that runs with a method of using a counter.

The value is first set with an appropriate condition, which is also called ‘initializing a loop’. Next, the terminal or final value is specified. The For Loop makes things really easy when you need to run a set of codes multiple times.

The For Loop is further divided into:****‍****

#### ****1. Using an array****

The following piece of code is a perfect example of how to use a for loop through an array.

var numbers = [ 10, 20, 30, 40, 50]   
for (var i=0; i < numbers.length; i++) {  
   console.log(numbers[i])  
} ****‍****

Here, I have used all the numbers in the form of an array, then printed each of them in a console window. In the same way, you can make a loop through an array of strings.

#### ****2. Making a loop through DOM elements****

The For loop can also be used to alter colors. Consider a situation where you want to choose a particular color for all the anchors of your page. In order to do so, use the following piece of code.

var elements = document.querySelectorAll("a");  
for (var i=0; i<elements.length; i++) {  
   elements[i].style.color = "red";  
} ****‍****

When you look at the above code, things might not seem very clear.

I first used `****document.querySelectorAll("a")`****to get all of my anchors in the array format. Once that was done, the next step looped all the array and changed its color.

Here, we used the color red, but you may choose any color in the code and the anchors of your page will appear that particular color.

In order to better understand this visually, here’s the output of when I ran this code on the W3Schools site:

You can see that the color of the anchors has been changed to red.

### **The ‘For In’ Loop**

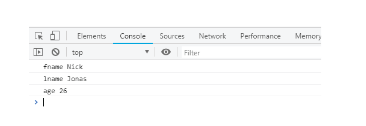
Another way of looping is the For In Loop. Unlike the For Loop, this loop won’t be using a counter. So this makes the whole process even more simple and hassle-free. In fact, the For In Loop is essentially a simplified version of the For Loop.

The following are different ways of looping using the For In technique.

#### ****1. Looping through an Object Property****

Here’s an example; you’ve got an object containing some properties and you need to look up each property and the value that it carries. Here’s how you would use the For In Loop to do so:

**var person = {  
   fname: "Nick",  
   lname: "Jonas",  
   age: 26  
};   
for (let x in person) {  
   console.log(x + ": "+ person[x])  
} **‍****



### ****3. The ForEach() Loop****

This method is used for looping through an array element. Here’s an example of this:

var names = ["jerry", "tom", "pluto", "micky", "mini"];  
names.forEach(function1);  
function function1(currentValue, index) {  
   console.log("Index in array is: "+index + " ::  Value is: "+currentValue);  
} ****‍****

1. Create your own resume data in JSON format

{

"basics": {

"name": "Ishwarya",

"father's name": "Baskaran",

"label": "programmer",

"picture": "",

"email": "sashti0211@gmail.com",

"phone": "+91 8754100591",

"location": {

"address": "ayyankottai",

"pincode": "625221",

"region": "indian",

},

"profiles": [{

"network": "instagram",

"username": "abcdef",

}]

},

"education": [{

"institution": "anna university",

"studytype": "bachelor",

"startdate": "11-08-2017",

"enddate": "14-04-2021",

"cgb": "7.5",

}],

"skills": [{

"name": "web developement",

"level": "begginer",

"keywords": [

"htmt",

"javascript",

]

}],

"languages": [{

"language": "tamil,english,spanish",

"fluency": "tamil,spanish",

}],

}

3.Read about the difference between window, screen and document in javascript

The JavaScript **window object** sits at the top of the JavaScript Object hierarchy and represents the browser window. The window object is supported by all browsers. All global **JavaScript objects** , functions, and variables automatically become members of the window object. The window is the first thing that gets loaded into the **browser** . This window object has the majority of the properties like length, innerWidth, innerHeight, name, if it has been closed, its parents, and more.

The window object represents the current **browsing context** . It holds things like window.location, window.history, window.screen, window.status, or the **window.document** . Each browser tab has its own top-level window object. Each of these windows gets its own separate global object. window.window always refers to window, but **window.parent** and window.top might refer to enclosing windows, giving access to other execution contexts. Moreover, the window property of a window object points to the window object itself. So the following ststements all return the same window object:

## Document

The **Document interface** represents any web page loaded in the browser and serves as an entry point into the web page's content, which is the DOM tree. When an HTML document is loaded into a **web browser** , it becomes a document object. It is the root node of the HTML document. The document actually gets loaded inside the window object and has properties available to it like title, URL, cookie, etc. HTML documents, served with the **"text/html"** content type, also implement the HTMLDocument interface, whereas XML and SVG documents implement the XMLDocument interface.

## Screen

Screen is a small information object about physical **screen dimensions** . It can be used to display screen width, height, colorDepth, pixelDepth etc. It is not mandatory to write **window prefix** with screen object. It can be written without window prefix.

#### Properties:

4.Codekata practice

//Getting input via STDIN

const readline = require("readline");

const inp = readline.createInterface({

input: process.stdin

});

let userInput;

inp.on("line" , (data) => {

userinput = data;

});

inp.on("close" ,() => {

console.log(userinput)

});

//Getting input via STDIN

const readline = require("readline");

const inp = readline.createInterface({

input: process.stdin

});

let userInput = [];

inp.on("line" , (data) => {

userInput.push(data);

});

inp.on("close" ,() => {

console.log(userInput[0]);

console.log(userInput[1]);

});

//Getting input via STDIN

const readline = require("readline");

const inp = readline.createInterface({

input: process.stdin

});

let userInput = [];

inp.on("line" , (data) => {

userInput.push(data);

});

inp.on("close" ,() => {

let temp = ""

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

temp = temp +userInput[i] + " ";

}

console.log(temp)

});

//Getting input via STDIN

const readline = require("readline");

const inp = readline.createInterface({

input: process.stdin

});

let userInput = [];

inp.on("line" , (data) => {

userInput.push(data);

});

inp.on("close" ,() => {

console.log(userInput[0]);

console.log(userInput[1]);

console.log(userInput[2]);

});