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

(i)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:‍

>>Using an array

{% code-block language="js" %}

var numbers = [ 10, 20, 30, 40, 50]

for (var i=0; i < numbers.length; i++) {

console.log(numbers[i])

}

‍{% code-block-end %}

--------------------------------------------------------------------------------------------------------------------

(ii) 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.

>>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:

{% code-block language="js" %}

var person = {

fname: "Nick",

lname: "Jonas",

age: 26

};

for (let x in person) {

console.log(x + ": "+ person[x])

}

‍{% code-block-end %}

----------------------------------------------------------------------------------------------------------------------

(iii) The ForEach() Loop

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

{% code-block language="js" %}

var names = ["jerry", "tom", "pluto", "micky", "mini"];

names.forEach(function1);

function function1(currentValue, index) {

console.log("Index in array is: "+index + " :: Value is: "+currentValue);

}

‍{% code-block-end %}

>>Looping through JSON in more detail with examples

We're making a list of books and we enter the title as “Lord of the Rings” and the year as “1994“and the count of pages as “3000”. Similarly, a large number of objects can be created and once it’s done, we will have a large database:

{% code-block language="js" %}

var books = [

{

title: "Lord of the Rings",

year: 1994,

pages: 3000

},

{

title: "1984",

year: 1984,

pages: 2000

},

{

title: "Art of war",

year: 2000,

pages: 5000

}

]

‍{% code-block-end %}

-------------------------------------------------------------------------------------------------------------------------

2. write own resume in Jason

var json = [{

"id" : "Ashwanth21",

"msg" : "For the given JSON iterate over all for loops (for, for in, for of, forEach)",

"task" : "zen portal task",

"mail": "kaviashwanth344@gmail.com"

},

{

"id" : "Ashwanth21",

"msg" : "For the given JSON iterate over all for loops (for, for in, for of, forEach)",

"task" : "zen portal task",

"mail": "kaviashwanth344@gmail.com"

}];

//for loop

for(var i = 0; i < json.length; i++) {

var obj = json[i];

console.log(obj.id);

console.log(obj.msg);

console.log(obj.task);

console.log(obj.mail);

}

//for Each

json.forEach(function(obj) { console.log(obj.msg); });

//for In

for (var key in json) {

if (json.hasOwnProperty(key)) {

console.log(json[key].id);

//console.log(json[key].msg);

}

}

//for Of

let text = "";

for (let x of json[key].id) {

text += x;

}

console.log(text);

---------------------------------------------------------------------------------------------------------

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

>>Window

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, inner Width, inner Height, 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 statements all return the same window object:

window.window

window.window.window

window.window.window.window

>>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:

screen.width

screen.height

screen.availWidth

screen.availHeight

screen.colorDepth

screen.pixelDepth