## ****Looping Using JSON****

JSON stands for JavaScript Object Notation. It’s a light format for storing and transferring data from one place to another. So in looping, it is one of the most commonly used techniques for transporting data that is the array format or in attribute values.

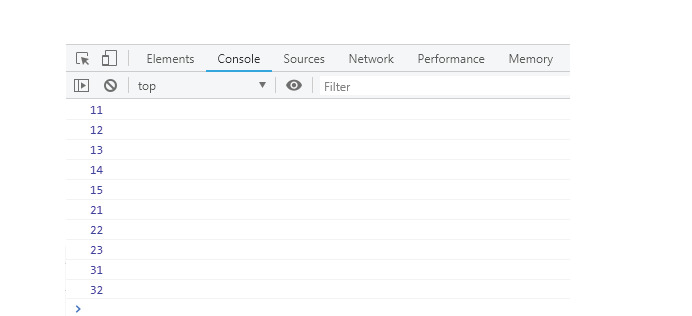
Here’s an example that demonstrates the above concept.

{% code-block language="js" %}  
jsonData ={  
   one: [11, 12, 13, 14, 15],  
   two: [21, 22, 23],  
   three: [31, 32]  
} **‍**{% code-block-end %}  
JavaScript Object Notation also consists of a root, namely, the jsonData. It further contains three nodes that are called, “one”, “two”, and “three”.

Here’s how you can withdraw a piece of information from JSON:

{% code-block language="js" %}  
var json = {  
   one: [11, 12, 13, 14, 15],  
   two: [21, 22, 23],  
   three: [31, 32]  
}  
};  
for(var key in json.jsonData) {  
   for (var key1 in json.jsonData[key]) {  
       console.log(json.jsonData[key][key1])  
   }  
} **‍**{% code-block-end %}  
Above, I used two For In Loops to make use of JSON and extract information from it. One of those loops is the outer loop that runs three times. If you’re wondering why it needs to run three times, that’s because it has to cover the above-mentioned nodes.

The second loop is the inner loop which is meant to cover all the internal values. These internal values are those which are inside the three nodes. Once you run the code, you will get these kinds of results:



### **JSON Explained**

If we go further in depth, there are a few more things you need to know about JSON.  The code that we just ran can also be expressed by including ‘[]’ to contain the above three nodes.

Here’s how it’s done:

{% code-block language="js" %}  
jsonData:  [  
   one: [11, 12, 13, 14, 15],  
   two: [21, 22, 23],  
   three: [31, 32]  
] **‍**{% code-block-end %}  
Then, in the following piece of code, a blend of the two looping techniques of For and For In are used to withdraw all the information from JSON.

{% code-block language="js" %}  
var json = {  
   jsonData:  [  
       {one: [11, 12, 13, 14, 15]},  
       {two: [21, 22, 23]},  
       {three: [31, 32]}  
   ]  
};   
for (var i=0; i<json.jsonData.length; i++) {  
   for (var key in json.jsonData[i]) {  
       for (var j= 0; j<json.jsonData[i][key].length; j++) {  
           console.log(json.jsonData[i][key][j])  
       }  
   }  
} **‍**{% code-block-end %}

### **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 %}

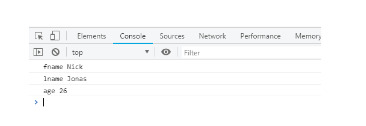
### The ‘For In’ Loop

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:  
{% 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 %}



### 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.

{% 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 %}  
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.  
 {% code-block language="js" %}  
var elements = document.querySelectorAll("a");  
for (var i=0; i<elements.length; i++) {  
   elements[i].style.color = "red";  
} **‍**{% code-block-end %}

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: