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

1.For..

The **for statement** creates a loop that consists of three optional expressions, enclosed in parentheses and separated by semicolons, followed by a statement (usually a [block statement](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/block)) to be executed in the loop.

Syntax

For ([initialization]; [condition]; [final-expression])

Statement.

Initialization:

variable declaration evaluated once before the loop begins. Typically used to initialize a counter variable. This expression may optionally declare new variables with var or let keywords. Variables declared with var are not local to the loop, i.e. they are in the same scope the for loop is in. Variables declared with let are local to the statement. Result expression can be decleared.

Condition:

An expression to be evaluated before each loop iteration. If this expression [evaluates to true](https://developer.mozilla.org/en-US/docs/Glossary/Truthy), statement is executed. This conditional test is optional. If omitted, the condition always evaluates to true. If the expression [evaluates to false](https://developer.mozilla.org/en-US/docs/Glossary/Falsy), execution skips to the first expression following the for construct.

Final expression:

An expression to be evaluated at the end of each loop iteration. This occurs before the next evaluation of condition. Generally used to update or increment the counter variable.

Statement:

To execute multiple statements within the loop, use a [block](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/block) statement ({ ... }) to group those statements. To execute no statement within the loop, use an [empty](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/Empty) statement (;).

Example using for:

for (let i = 0; i < 9; i++) {

console.log(i);

// more statements

}

2.for..in:

The **for...in statement** iterates over all [enumerable properties](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Enumerability_and_ownership_of_properties) of an object that are keyed by strings (ignoring ones keyed by [Symbol](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Symbol)s), including inherited enumerable properties.

Syntax:

For (variable in object) {

Statement

}

Variable:

A different property name is assigned to Variable on each iteration.

Object:

Object whose non-symbol enumerable property are iterated over.

A for...in loop only iterates over enumerable, non-Symbol properties. Objects created from built–in constructors like Array and Object have inherited non–enumerable properties from Object.prototype and String.prototype, such as [String](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String)'s [indexOf()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/indexOf) method or [Object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object)'s [toString()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/toString) method.

For.. in loop can be added deleted and the loop iterates over the object in an arbitrary can delete which one cannot be opened.

Example:

var obj = {a: 1, b: 2, c: 3};

for (const prop in obj) {

console.log(`obj.${prop} = ${obj[prop]}`);

}

// Output:

// "obj.a = 1"

// "obj.b = 2"

// "obj.c = 3"

3.For…of:

The **for...of statement** creates a loop iterating over [iterable objects](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Iteration_protocols" \l "the_iterable_protocol), including: built-in [String](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String), [Array](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array), array-like objects (e.g., [arguments](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/arguments) or [NodeList](https://developer.mozilla.org/en-US/docs/Web/API/NodeList)), [TypedArray](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/TypedArray), [Map](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map), [Set](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set), and user-defined iterables. It invokes a custom iteration hook with statements to be executed for the value of each distinct property of the object.

Syntax:

For (variable of iterable) {

Statement

}

Variable:

On each iteration a value of a different property is assigned to variable.variable may be declared with const, let or var.

Iterable

Object whose iterable property are literated.

Example:

const iterable = [10, 20, 30];

for (const value of iterable) {

console.log(value);

}

// 10

// 20

// 30

4.For each

Definition and Usage

The forEach() method calls a function for each element in an array.

The forEach() method is not executed for empty elements.

Syntax

*array*.forEach(*function(currentValue, index, arr), thisValue*)

Parameters

|  |  |
| --- | --- |
| *function()* | Required. A function to run for each array element. |
| *currentValue* | Required. The value of the current element. |
| *index* | Optional. The index of the current element. |
| *arr* | Optional. The array of the current element. |
| *thisValue* | Optional. Default undefined. A value passed to the function as its this value. |

Return Value

|  |
| --- |
| undefined |