For.. of Loop vs For.. in Loop (When to use which?) | JavaScript Tutorial In Hindi #56

As we have studied loops in the previous lecture, we know that loops are used in programming to automate repetitive tasks. JavaScript works in very different ways. It provides us with several ways of doing something. There are several loop statements in JavaScript like while, do-while, forEach. However, in most cases, everyone tends to use for statements because we usually deal with looping scenarios which consist of some conditions with counters. In addition to the **for** loop, there are two other types of iteration methods of for, which are: **for..in** and **for..of**. The main topic of today's tutorial is for..in and for..of loops. Before getting in the detail of these loops, let us revise the for loop.

for loop:-

It is the most basic loop in every programming language. The for loop syntax takes three parameters; a variable declaration, an expression that will evaluate before each iteration, and the expression to be evaluated at the end of each iteration. Here is the syntax of for loop:

```
for (initialization; condition; final expression) {
    // code to be executed
}
```

For example, the for loop will console.log each item in an array.

```
const myarray = ['abcd', 'efgh', 'i', 'eeb12'];
for (let i = 0; i < myarray.length; i++) {
console.log(array[i]);
} // Result: abcd,efgh,I,eeb12
```

The for...in loop:-

The for...in loop is an iteration method for iterating over "enumerable" properties of an object. This loop applies to all objects that have these properties. Here enumerable means an array or object or strings. If we are using a for...in loop over an object, it will give us the value to each key in the object. Here is the syntax of the **for...in loop:**

```
for (variable in enumerable) {
        // do stuff
}
```

Here is an example to loop through and console.log all the values in the object,

```
const obj = {
        first: 134,
        second: 255,
        third: 367,
        fourth: 433
}
for (const key in obj) {
        console.log( obj[key] )
} // Result: 134, 255, 367, 433
```

For...in loops do not have any specific order for execution. If we want the guarantee order on an object. Then specify the index in the objects or put the keys in an array. As we know, a character in a string has an index. Therefore, similar to Arrays, the indexes are enumerable properties that happens to be integers. Here is an example of a for..in loop with strings:

```
const str = 'Lets Code';
for (let index in str) {
console.log(str[index])
} //Result: L, e, t, s C, o, d, e
```

For...Of Loop:-

To iterate over objects like arrays and strings, we can use the for...of statement. This statement is a newer feature of ECMAScript 6. The **for...of** loop does not work with objects because they are not "iterable". This iteration method is a more reliable way of looping through an Array in sequence. In this example of a for...of loop, we will print each item in the array to the console.

```
let students = [ "Mark", "Harry", "Joe" ];
// Print out each type of shark
```

```
for (let std of students) {
   console.log(std);
} //Result: Mark, Harry, Joe
```

Comparison:-

In the following table, we are comparing for..in loop and for..of loop.

	for_in	for_of
Applies to	Enumerable Properties	Iterable Collections
Objects	Yes	No
Arrays	Yes, but not advised	Yes
Strings	Yes, but not advised	Yes

Which loop should we use? and When?

Each type of loop is useful in a different scenario:

- If indexes are needed while accessing an array of indexes related logical stuff are there, the for loop is a good choice.
- If there is a need to access keys/properties regardless of the sequence, use *a for-in* loop.
- If there is a need to iterate through items of an iterable, then the *for-of*loop is the right choice.

Code as described/written in the video

```
console.log("This is tutorial 56 on for-in and for-of loops");

let people = ["Harry", "Rohan", "SkillF", "Shubham", "Vikrant"];


// **********For in loop*******

// Traditional for loop:
// for (let index = 0; index < people.length; index++) {
//     const element = people[index];
//     console.log(element);
// }


// 1. ITERATING AN OBJECT
let obj = {
    name: "Harry",
```

```
    language: "JavaScript",
    hobbies: "Android app development"
}
// console.log(obj);
// 1.1 Iterating an object using Traditional for loop:
// for (let index = 0; index < Object.keys(obj).length; index++) {
//     const element = obj[Object.keys(obj)[index]];
//     console.log(element);
// }

// 1.2 Iterating an object using for-in loop:
for (let key in obj){
    console.log(obj[key]);
}

// 2. ITERATING A STRING
// We can use for in with strings to loop through all the characters
myString = "This is my string";
for (let char in myString){
    console.log(myString[char]);
}

// Quiz: Use traditional for loop to iterate through the same string


// **********For of loop**********
console.log("**********For of loop starts here**********")
people = ["Harry", "Rohan", "SkillF", "Shubham", "Vikrant"];

for(let name of people){
    console.log(name);
}

for(let name of myString){
    console.log(name);
}
```

Previous

Next

CodeWithHarry