JavaScript for...of Loop

Summary: in this tutorial, you'll how to use JavaScript `for...of` statement to iterate over iterable objects.

Introduction to the JavaScript for...of loop

ES6 introduced a new statement `for...of` that iterates over an iterable object such as:

- Built-in Array, String, Map, Set, ...
- Array-like objects such as `arguments` or `NodeList`
- User-defined objects that implement the iterator protocol.

The following illustrates the syntax of the `for...of`:

```
for (variable of iterable) {
   // ...
}
```

variable

In each iteration, a property of the iterable object is assigned to the `variable`. You can use `var`, `let`, or `const` to declare the `variable`.

iterable

The iterable is an object whose iterable properties are iterated.

JavaScript for of loop examples

Let's take a look at some examples of using the `for...of` loop.

1) Iterating over arrays

The following example shows how to use the `for...of` to iterate over elements of an array:

```
let scores = [80, 90, 70];

for (let score of scores) {
    score = score + 5;
    console.log(score);
}
```

Output:

```
85
95
75
```

In this example, the `for...of` iterates over every element of the `scores` array. It assigns the element of the `scores` array to the variable `score` in each iteration.

If you don't change the variable inside the loop, you should use the `const` keyword instead of the `let` keyword as follows:

```
let scores = [80, 90, 70];

for (const score of scores) {
    console.log(score);
}
```

Output:

```
80
90
70
```

To access the index of the array elements inside the loop, you can use the `for...of` statement with the `entries()` method of the array.

The `array.entries()` method returns a pair of `[index, element]` in each iteration. For example:

```
let colors = ['Red', 'Green', 'Blue'];

for (const [index, color] of colors.entries()) {
    console.log(`${color} is at index ${index}`);
}
```

Output:

```
Red is at index 0
Green is at index 1
Blue is at index 2
```

In this example, we used the array destructuring to assign the result of the `entries()` method to the `index` and `color` variables in each iteration:

```
const [index, color] of colors.entries()
```

2) In-place object destructuring with for...of

Consider the following example:

```
const ratings = [
    {user: 'John',score: 3},
    {user: 'Jane',score: 4},
    {user: 'David',score: 5},
    {user: 'Peter',score: 2},
];

let sum = 0;
for (const {score} of ratings) {
    sum += score;
}

console.log(`Total scores: ${sum}`); // 14
```

Output:

```
Total scores: 14
```

How it works:

- The `ratings` is an array of objects. Each object has two properties user and score.
- The `for...of` iterate over the `ratings` array and calculate the total scores of all objects.
- The expression `const {score} of ratings` uses object destructing to assign the `score` property of the current iterated element to the `score` variable.

3) Iterating over strings

The following example uses the `for...of` loop to iterate over characters of a string.

```
let str = 'abc';
for (let c of str) {
    console.log(c);
}
```

Output:

```
a
b
c
```

3) Iterating over Map objects

The following example illustrates how to use the `for...of` statement to iterate over a `Map` object.

```
let colors = new Map();

colors.set('red', '#ff0000');
colors.set('green', '#00ff00');
colors.set('blue', '#0000ff');
```

```
for (let color of colors) {
    console.log(color);
}
```

Output:

```
[ 'red', '#ff0000' ]
[ 'green', '#00ff00' ]
[ 'blue', '#0000ff' ]
```

4) Iterating over set objects

The following example shows how to iterate over a `set` object using the `for...of` loop:

```
let nums = new Set([1, 2, 3]);

for (let num of nums) {
    console.log(num);
}
```

for...of vs. for...in

The `for...in` iterates over all enumerable properties of an object. It doesn't iterate over a collection such as `Array`, `Map` or `Set`.

Unlike the `for...in` loop, the `for...of` iterates a collection, rather than an object. In fact, the `for...of` iterates over elements of any collection that has the `[Symbol.iterator]` property.

The following example illustrates the differences between `for...of` and `for...in`

```
let scores = [10,20,30];
scores.message = 'Hi';

console.log("for...in:");
for (let score in scores) {
  console.log(score);
}
```

```
console.log('for...of:');
for (let score of scores) {
  console.log(score);
}
```

Output:

```
for...in:
0
1
2
message
for...of:
10
20
30
```

In this example, the for...in statement iterates over the properties of the scores array:

```
for...in:
0
1
2
message
```

while the for...of iterates over the element of an array:

```
for...of:
10
20
30
```

Summary

- Use the `for...of` loop to iterate over elements of an iterable object.