JavaScript for...in Loop

Summary: in this tutorial, you will learn how to use the JavaScript `for...in` loop to iterate over the enumerable properties of an object.

Introduction to JavaScript for...in loop

The `for...in` loop over the enumerable properties that are keyed by strings of an object. Note that a property can be keyed by a string or a symbol.

A property is enumerable when its internal `enumerable` flag is set to `true` .

The `enumerable` flag defaults to `true` when a property is created via a simple assignment or via a property initializer:

```
object.propertyName = value;
```

or

```
let obj = {
    propertyName: value,
    ...
};
```

The following shows the syntax of the `for...in` loop:

```
for(const propertyName in object) {
    // ...
}
```

The `for...in` allows you to access each property and value of an object without knowing the specific name of the property. For example:

```
var person = {
    firstName: 'John',
    lastName: 'Doe',
    ssn: '299-24-2351'
};

for(var prop in person) {
    console.log(prop + ':' + person[prop]);
}
```

Output:

```
firstName:John
lastName:Doe
ssn:299-24-2351
```

In this example, we used the `for...in` loop to iterate over the properties of the person object. We accessed the value of each property using the following syntax:

```
object[property];
```

The for...in loop & Inheritance

When you loop over the properties of an object that inherits from another object, the `for...in` statement goes up in the prototype chain and enumerates inherited properties. Consider the following example:

```
var decoration = {
    color: 'red'
};

var circle = Object.create(decoration);
circle.radius = 10;


for(const prop in circle) {
    console.log(prop);
}
```

Output:

```
radius
color
```

The `circle` object has its own prototype that references the `decoration` object. Therefore, the `for...in` loop displays the properties of the `circle` object and its prototype.

If you want to enumerate only the own properties of an object, you use the `hasOwnProperty()` method:

```
for(const prop in circle) {
    if(circle.hasOwnProperty(prop)) {
        console.log(prop);
    }
}
```

Output:

```
radius
```

The for...in loop and Array

It's good practice to not use the `for...in` to iterate over an array, especially when the order of the array elements is important.

The following example works flawlessly:

```
const items = [10 , 20, 30];
let total = 0;

for(const item in items) {
    total += items[item];
}
console.log(total);
```

However, someone may set a property of the built-in `Array` type in their libraries as follows:

```
Array.prototype.foo = 100;
```

Hence, the `for...in` will not work correctly. For example:

```
// somewhere else
Array.prototype.foo = 100;

const items = [10, 20, 30];
let total = 0;

for (var prop in items) {
  console.log({ prop, value: items[prop] });
```

```
  total += items[prop];
}
console.log(total);
```

Output:

```
{ prop: '0', value: 10 }
{ prop: '1', value: 20 }
{ prop: '2', value: 30 }
{ prop: 'foo', value: 100 }
160
```

Another example:

```
var arr = [];
// set the third element to 3, other elements are `undefined`
arr[2] = 3;

for (let i = 0; i < arr.length; i++) {
    console.log(arr[i]);
}
```

The output shows three elements of the array, which is correct:

```
undefined
undefined
3
```

However, the `for...in` loop ignores the first two elements:

```
for (const key in arr) {
    console.log(arr[key]);
}
```

Output:

```
3
```

The output shows only the third element, not the first two elements.

Summary

- The `for...in` loop iterates over the enumerable properties of an object. It also goes up to the prototype chain and enumerates inherited properties.
- Avoid using `for...in` loop to iterate over elements of an array, especially when the index order is important.