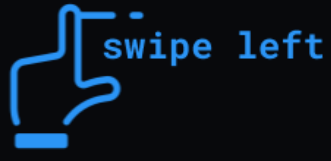
deeecode

LOOPING THROUGH OBJECTS

JS



deeecode

JS

LOOPING THROUGH OBJECTS

For looping through arrays, you have methods like **forEach**, **map** and you can also use **for** statements as each element has an index position in an ordered manner.

How do you loop through objects?

I will show four ways in this post:

- **for...in**: for looping through the keys
- **Object.keys()**: returns the keys in an array
- **Object.values()**: returns the values in an array
- **Object.entries()**: returns an array or arrays

Keep swiping to see more information about each method...

deeecode

JS

LOOPING THROUGH OBJECTS

for...in

This approach is used to loop through the keys of an object.

```
const object = {
  name: "DeeeCode",
  id: "deeecode",
  language: "javascript"
}

for (const key in object) {
  const value = object[key]
  console.log("key: " + key)
  console.log("value: " + value)
}

// key: name
// value: DeeeCode
// key: id
// value: deeecode
// key: language
// value: javascript
```

By looping through the keys, you can get the value using **object[key]**

deeecode

JS

LOOPING THROUGH OBJECTS

Object.keys()

The **keys** method of the Object constructor returns an array of the keys in an object.

```
const object = {
  name: "DeeeCode",
  id: "deeecode",
  language: "javascript"
}

const keys = Object.keys(object)
// [ 'name', 'id', 'language' ]

keys.forEach(function(key) {
  const value = object[key]
  console.log("key: " + key)
  console.log("value: " + value)
})

// key: name
// value: DeeeCode
// key: id
// value: deeecode
// key: language
// value: javascript
```

With the array of keys, you can then loop through the array using any array approaches.

This example shows using the **forEach** method. And you can get the value of the object key also using **object[key]**

swipe left

deeecode

JS

LOOPING THROUGH OBJECTS

Object.values()

The **values** method returns an array of the values in an object (opposite of **keys** method).

```
const object = {
  name: "DeeeCode",
  id: "deeecode",
  language: "javascript"
}

const values = Object.values(object)
// [ 'name', 'id', 'language' ]

values.forEach(function(value) {
  console.log(value)
})

// [ 'DeeeCode',
//   'deeecode',
//   'javascript'
// ]
// DeeeCode
// deeecode
// javascript
```

With the array of values, you can then loop through the array using any array approaches.

Note that you can use a key to get a value directly but you cannot use a value to get a key directly.

swipe left

deeecode

JS

LOOPING THROUGH OBJECTS

Object.entries()

The **entries** method returns an array of subarrays where each subarray has two items: the first one is the **key** and the second one is the **value**.

```
const object = {
  name: "DeeeCode",
  id: "deeecode",
  language: "javascript"
}

const entries = Object.entries(object)
// [
//   [ 'name', 'DeeeCode' ],
//   [ 'id', 'deeecode' ],
//   [ 'language', 'javascript' ]
// ]

entries.forEach(function(entry) {
  const key = entry[0]
  const value = entry[1]

  console.log("key: " + key)
  console.log("value: " + value)
})

// key: name
// value: DeeeCode
// key: id
// value: deeecode
// key: language
// value: javascript
```

Unlike the **keys** and **values** method, **entries** returns the keys and values of an object in subarrays.

Then you can access them using the **0** and **1** index as shown by the left.