

How to Use Object.keys in JavaScript



Javascript Jeep 🚗

Follow

Nov 1, 2019 · 2 min read



Consider an object:

```
var user = {  
  name: "Jagathish",
```

```
    age: 20  
  }
```

In the `user` object, the `name` and `age` are the keys of the object. Keys are also referred to as object “properties”. We can access the property value by `obj.propertyName` or `obj[propertyName]`.

The `Object.keys()` method returns an array of strings of a given object's own property/key names. The following is what we get for our `user` object:

```
Object.keys(user); // ["name", "user"]
```

Let's look at another example:

```
var user = {  
  name : "Jagathish",  
  age  : 20,  
  getAge() {  
    return this.age;  
  }  
}
```

```
Object.keys(user); // ["name", "age", "getAge"]
```

The key names are returned for all properties, whether it's a function or primitive variable type. The order of key names in the array will be the same as they were in the object.

Syntax

```
Object.keys(obj)
```

Parameter: `obj`

The only parameter the `Object.keys()` function takes is an object itself.

- The object of which the enumerable's own properties are to be returned.
- If we pass an empty object, then it returns an empty array.
- If we don't pass any argument (which is equivalent to passing `undefined`) or if we pass `null`, then it throws an error.

Return value: Array of strings

An array of strings that represent all the enumerable properties of the given object.

```
var array = ['a', 'b', 'c'];

console.log(Object.keys(array)); // ['0', '1', '2']

var funObj = {
  fun : function () {
    ...
  }
}
console.log(Object.keys(funObj)) // ["fun"]
```


When we pass a non-object except `undefined`, it will be coerced to an object.

```
Object.keys(123) // []

Object.keys(123.34) // []

Object.keys("hi") // ["0", "1"]
```

Follow me [JavaScript Jeep](#)  .

Please make a donation [here](#). 80% of your donation is donated to someone needs food . Thanks in advance.