TypeScript, Looping through a dictionary

Asked 6 years, 4 months ago Active 22 days ago Viewed 205k times

141

In my code, I have a couple of dictionaries (as suggested here) which is String indexed. Due to this being a bit of an improvised type, I was wondering if there any suggestions on how I would be able to loop through each key (or value, all I need the keys for anyway). Any help appreciated!

```
myDictionary: { [index: string]: any; } = {};
```

9

javascript html5 typescript

edited Aug 7 '18 at 7:36
Jack Miller
2,220 1 25 34

asked Apr 23 '13 at 16:07
ben657
938 2 8 12

Did you try: `for (var key in myDictionary) { }` ? Inside the loop, you'd use `key` to get the key, and `myDictionary[key]` to get the value – Ian Apr 23 '13 at 16:15

@Ian Just tried that, doesn't seem to be working. No errors, but nothing runs within the statement – ben657 Apr 23 '13 at 16:21

1 @Ian Ah sorry, some code elsewhere was messing with it. That works perfectly! Care to make it an answer so I can choose it? – ben657 Apr 23 '13 at 16:38

5 Answers

223

To loop over the key/values, use a `for in` loop:

```
for (let key in myDictionary) {
    let value = myDictionary[key];
    // Use `key` and `value`
```

edited Aug 7 '17 at 18:21

kcpr
550 6 20

answered Apr 23 '13 at 16:42

Ian
41.3k 11 86 102

2 @Yiping Nope, but it helps make it clear for me – Ian Sep 5 '15 at 14:52

5 This isn't safe in general; it needs the hasOwnProperty test (as in Jamie Stark's answer) or something else. – Don Hatch Feb 3 '16 at 19:30

11 @DonHatch It's actually very safe in general. It's not safe in very specific cases (like when you include libraries which modify prototypes incorrectly, or purposely modify prototypes incorrectly). It's up to the developer whether to bloat their code with most-likely-unnecessary checks – Ian Feb 3 '16 at 19:38

4 @Ian The problem is that there are too many libraries which do modify object prototypes. It seems a bad idea to advocate this type of pattern when much better simple alternatives exist, such as `Object.keys(target).forEach(key => { let value = target(key); /* Use key, value here */ });` . If you must show this method, at least mention the risks and better alternatives for those who don't yet know better. – Yona Appletree Apr 13 '16 at 0:08

6 We already have es6 and TypeScript. Why does the problem of writing hasOwnProperty all the time remain unresolved? Even in 2017 and with all attention to JS. I am so disappointed. – Gherman Sep 25 '17 at 13:10

114

@Alcesem had it almost right except that the benefit of that method is that one doesn't need to include the `hasOwnProperty()` guard with `Object.keys` because the iterator will not search further up the prototype chain. So the following is not only safe for dictionaries, but any kind of object in both Typescript and Javascript.

< ES 2017:

```
Object.keys(obj).forEach(key => {
  let value = obj[key];
});
```

>= ES 2017:

```
Object.entries(obj).forEach(
  ([key, value]) => console.log(key, value)
);
```

edited May 10 '18 at 9:45

answered Oct 15 '16 at 5:59

This is the solution I was looking for as it will allow me to sort the keys before iteration – Andrew Oct 18 '16 at 15:57

FYI, this is not supported on TypeScript playground at this time. – Seanny123 Jan 18 '17 at 12:28

1 See here how to make `Object.entries` to work in TypeScript – Alex Klaus Mar 21 '17 at 23:29

49

There is one caveat to the key/value loop that Ian mentioned. If it is possible that the Objects may have attributes attached to their Prototype, and when you use the `in` operator, these attributes will be included. So you will want to make sure that the key is an attribute of your instance, and not of the prototype. Older IEs are known for having `indexof(v)` show up as a key.

```
for (const key in myDictionary) {
    if (myDictionary.hasOwnProperty(key)) {
        let value = myDictionary[key];
    }
}
```

edited Mar 13 '18 at 9:58

Francesco Borzi
18.9k 14 72 119

answered Apr 23 '13 at 17:08

Jamie Starke
5,608 3 19 47

2 Not trying to split hairs, but since we're talking type script, shouldn't you use let key instead of var key? Nice answer thank you. – Luke Dupin Mar 11 '16 at 1:03

3 In fact, with the current version of type script the `key` and `value` can be `const` . – Patrick Desjardins Mar 3 '17 at 5:19

43

How about this?

```
for (let [key, value] of Object.entries(obj)) {
    ...
}
```

answered Jun 21 '17 at 17:50

It requires lib es2017 in tsconfig. See stackoverflow.com/questions/45422573/… – Stéphane Mar 12 at 17:15

0

Shortest way to get all dictionary/object values:

```
Object.keys(dict).map(k => dict[k]);
```

answered Aug 18 at 19:40

k06a

9,030 6 50 89