Array.prototype.values() is undefined - using Babel to transpile ES6 in NodeJS environment

My setup is:

```
$ babel --version
6.3.15 (babel-core 6.3.15)
$ node --version
v5.1.0
```

Using Webstorm 11 IDE in case that is important too.

I am using Babel(js) to transpile the following ES6, and have set up some logging to verify:

```
Array.from([ 'a', 'b' ].keys());
Array.from([ 'a', 'b' ].values());  // TypeError: ["a","b"].values is not a function
Array.from([ 'a', 'b' ].entries());
```

Can verify this v.quickly:

```
Array.prototype.values === undefined) // true
```

Note that **keys**, and **entries** both exist.

Any ideas what the likely cause is? (Have I missed a special options flag or something on Babel to switch on support of this feature?). Thanks for any help, and shall continue to check documentation etc in meantime.

javascript node.js ecmascript-6 babeljs

asked Dec 8 '15 at 15:01
arcseldon
12.6k 3 55 71

4 You have to manually require the polyfill babeljs.io/docs/usage/polyfill – Paolo Moretti Dec 8 '15 at 15:11

Thanks Paolo, taking a look now. you may wish to put your comment into a one line answer so i can credit you if it works. – arcseldon Dec 8 '15 at 15:14

1 Just in case of another missing feature, you should take a look at kangax.github.io/compat-table/es6 ;) – juliobetta Dec 8 '15 at 15:21

Excellent, thank you, this is the solution. Mocha tests all passing now - mystery solved. Cheers! – arcseldon Dec 8 '15 at 15:22

@juliobetta - thanks, I had that link open, but didn't fully understand the meaning of "Babel + Core JS". Great suggestion all the same. – arcseldon Dec 8 '15 at 15:24

2 Answers

Providing an answer for completeness. BabelJS requires an extra polyfill package to extend it with some extra ES6+ features - such as the one in this question.

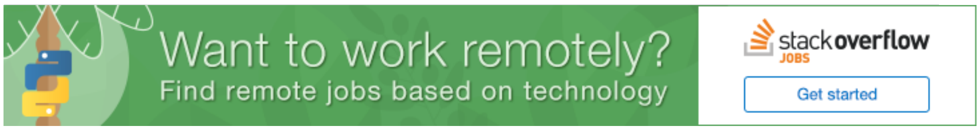
```
npm install babel-polyfill --save
```

Then insert the following require statement towards the top of the affected module to obtain required (generator) behaviour:

```
require("babel-polyfill");
```

This should be all you need, just importing the module adds required polyfill

answered Jan 11 '16 at 15:44
arcseldon
12.6k 3 55 71



I know, you accepted the answer, but I'd highly recommend to use **core-js** standard library for node.js.

With this library you'll forget seeing any problems on JS features support. It includes polyfills for ECMAScript 5, ECMAScript 6: promises, symbols, collections, iterators, typed arrays, ECMAScript 7+ proposals, setImmediate, etc. Some additional features such as dictionaries or extended partial application.

It's easy to install dependency and to use it:

```
npm i core-js --save
```

Then within your project, use it this way, to include all the features it support:

```
// Without global namespace pollution
var core = require('core-js/library');
```

Or like this, if you want to include only specific features (in your case you was missing *Array.prototype.values()*):

```
require('core-js/fn/array/values');
```

This lib is a "must-have" to me for every project, after I discovered it for myself. Package description can be found at official page: https://www.npmjs.com/package/core-js

answered Mar 2 '16 at 15:15
Farside
3,462 2 19 40

thanks for this answer. I have upvoted it on basis it looks like a good option. I realise that babel leans on this library internally, so for now I shall leave the accepted answer as-is. If more voters choose this answer then I'd be happy to assign this as a correct answer. – arcseldon Mar 2 '16 at 16:25

Feel free to upvote my question too ;) – arcseldon Mar 2 '16 at 16:26

2 As arcseldon already said, babel-polyfill uses core-js itself and it loads all the additional runtimes needed (such as regenerator). – Felix Kling Mar 2 '16 at 17:31