Mocking globals in Jest

Is there any way in Jest to mock global objects, such as `navigator`, or `Image`*? I've pretty much given up on this, and left it up to a series of mockable utility methods. For example:

```
// Utils.js
export isOnline() {
    return navigator.onLine;
}
```

Testing this tiny function is simple, but crufty and not deterministic at all. I can get 75% of the way there, but this is about as far as I can go:

```
// Utils.test.js
it('knows if it is online', () => {
    const { isOnline } = require('path/to/Utils');

    expect(() => isOnline()).not.toThrow();
    expect(typeof isOnline()).toBe('boolean');
});
```

On the other hand, if I am okay with this indirection, I can now access `navigator` via these utilities:

```
// Foo.js
import { isOnline } from './Utils';

export default class Foo {
    doSomethingOnline() {
        if (!isOnline()) throw new Error('Not online');

        /* More implementation */
    }
}
```

...and deterministically test like this...

```
// Foo.test.js
it('throws when offline', () => {
    const Utils = require('../services/Utils');
    Utils.isOnline = jest.fn(() => isOnline);

    const Foo = require('../path/to/Foo').default;
    let foo = new Foo();

    // User is offline -- should fail
    let isOnline = false;
    expect(() => foo.doSomethingOnline()).toThrow();

    // User is online -- should be okay
    isOnline = true;
    expect(() => foo.doSomethingOnline()).not.toThrow();
});
```

Out of all the testing frameworks I've used, Jest feels like the most complete solution, but any time I write awkward code just to make it testable, I feel like my testing tools are letting me down.

Is this the only solution or do I need to add Rewire?

*Don't smirk. `Image` is fantastic for pinging a remote network resource.

javascript unit-testing dependencies jestjs babel-jest

asked Nov 6 '16 at 12:39

Andrew
7,917 9 39 86

1 Answer

As every test run its own environment you can mock globals by just overwrite them. All global vars can be accessed by the `global` name space.

```
global.navigator = {
  onLine: true
```

```
}
```

The overwrite has only effects in your current test and will not effect others. This also a good way to handle `Math.random` or `Date.now`

Note, that through some changes in jsdom it could be possible that you have to mock globals like this:

```
Object.defineProperty(globalObject, key, { value, writable: true });
```

edited Nov 10 '17 at 16:58

answered Nov 7 '16 at 8:02

Andreas Köberle
38.3k 34 149 213

Will `global` be the same as `window` in the browser? – Andrew Nov 7 '16 at 8:07

1 Yes in the the sense that you can set the stuff there. But maybe not all the stuff that is present in `window` is also present in `global`. Thats why I don't use `global.navigator.onLine` cause I'm not sure that there is a `navigator` object in `global`. – Andreas Köberle Nov 7 '16 at 8:27

Be aware that as a general practice not all global properties are overwritable nowadays. Some have writable false and will ignore value change attempts. – Daniel Nalbach Nov 6 '17 at 19:11