Testing CSS-in-JS

Popular CSS-in-JS libraries like styled-components or emotion can also be tested with the help of jest-styled-components or @emotion/jest respectively. These packages improve Jest's built-in snapshot testing experience and are a great way to help avoid unintended changes to your website's UI. Please refer to your package's documentation to see if it also offers testing capabilities.

Snapshot serializers like `jest-styled-components` or `@emotion/jest` modify the standard output to a more meaningful and readable snapshot, e.g. by removing unnecessary information or displaying data in another format. Which ultimately leads to more comparable and effective snapshot tests.

By default snapshots of your styled components show the generated class names (which you didn't set) and no styling information. When changing the styles you'll only see the diff of some cryptic class names (hashes). That's why you should use the above-mentioned *snapshot serializers*. They remove the hashes and format the CSS in style elements.

For this example you'll use emotion. The testing utilities of emotion and glamor are largely based on jest-styled-components so they have a similar usage. Please have a look at the testing section of your library to follow along.

Installation

```
npm install --save-dev @emotion/jest @emotion/babel-plugin
```

As Gatsby's emotion plugin is using `@emotion/babel-plugin` under the hood you'll also need to install it so that Jest can use it.

If you followed along with the Unit testing guide you'll have the file `jest-preprocess.js` at the root of your project. Open that file and add the plugin:

```
const babelOptions = {
~  presets: ["babel-preset-gatsby", "@emotion/babel-preset-css-prop"],
+  plugins: [
+    "emotion",
+  ],
}
```

```
module.exports = require("babel-jest").createTransformer(babelOptions)
```

In order to tell Jest to use the serializer you'll need to create the file `setup-test-env.js` which will be run automatically before every test. Create the file `setup-test-env.js` at the root of your project. Insert this code into it:

```
import { createSerializer } from "@emotion/jest"
import * as emotion from "@emotion/react"

expect.addSnapshotSerializer(createSerializer(emotion))
```

Lastly you need to tell Jest where to find this file. Open your `package.json` and add this entry to your `"jest"` section:

```
"jest": {
  "setupFilesAfterEnv": [`<rootDir>/setup-test-env.js`]
}
```

Usage

In this example you'll use `react-test-renderer` but you can also use @testing-library/react or any other appropriate library. Because you created the `setup-test-env.js` file you can write your unit tests like you used to do. But now you'll also get the styling information!

```
import React from "react"
import styled from "react-emotion"
import renderer from "react-test-renderer"

const Button = styled.div`
  color: hotpink;
`

test("Button renders correctly", () => {
  expect(
    renderer.create(<Button>This is hotpink.</Button>).toJSON()
  ).toMatchSnapshot()
})
```

The resulting snapshot will look like this:

```
// Jest Snapshot v1, https://goo.gl/fbAQLP

exports[`Button renders correctly 1`] = `
.emotion-0 {
  color: hotpink;
}
```

```
<div
  className="emotion-0 emotion-1"
>
  This is hotpink.
</div>
`
```

If your styled component depends on `theme` via `ThemeProvider` you'll have two options:

- Wrap all your components with the `ThemeProvider`
- Use API helpers (have a look at the library's documentation, e.g. styled-components or emotion)

And this is where snapshots tests really shine. If you change, e.g. the primary color in your theme file you'll see which components get affected by this change. This way you can catch unintended changes to the style of your components.

This example uses the first option:

```
import React from "react"
import { ThemeProvider, withTheme } from "@emotion/react"
import renderer from "react-test-renderer"

const theme = {
  maxWidth: "1450px",
}

const Wrapper = styled.section`
  max-width: ${props => props.theme.maxWidth};
`

test("Wrapper renders correctly", () => {
  expect(
    renderer
      .create(
        <ThemeProvider theme={theme}>
          <Wrapper>Content.</Wrapper>
        </ThemeProvider>
      )
      .toJSON()
  ).toMatchSnapshot()
})
```

The resulting snapshot will look like this:

```
// Jest Snapshot v1, https://goo.gl/fbAQLP

exports[`Wrapper renders correctly 1`] = `
```

```
.emotion-0 {
  max-width: 1450px;
}

<section
  className="emotion-0 emotion-1"
>
  Content
</div>
`
```