Testing MessageList with Message Component

In this lesson, we will test components with Deep Rendering.

WE'LL COVER THE FOLLOWING

Testing Components Using Deep Rendering

Step 1:

To test MessageList with Deep Rendering, we just need to use `mount` instead of `shallowMount` from the previously created `test/MessageList.test.js`:

```
require('./check-versions')()

process.env.NODE_ENV = 'production'

var ora = require('ora')
var rm = require('rimraf')
var path = require('path')
var chalk = require('chalk')
var webpack = require('webpack')
var config = require('../config')
var webpackConfig = require('./webpack.prod.conf')

var spinner = ora('building for production...')
spinner.start()

rm(path.join(config.build.assetsRoot, config.build.assetsSubDirectory), err => {
  if (err) throw err
  webpack(webpackConfig, function (err, stats) {
    spinner.stop()
    if (err) throw err
    process.stdout.write(stats.toString({
      colors: true,
      modules: false,
      children: false,
      chunks: false,
      chunkModules: false
```

```
    }) + '\n\n')

    console.log(chalk.cyan('  Build complete.\n'))
    console.log(chalk.yellow(
      '  Tip: built files are meant to be served over an HTTP server.\n' +
      '  Opening index.html over file:// won\'t work.\n'
    ))
  })
})
```

> Have you noticed the `beforeEach` thing at line number 7? That's a clean way to create a clean component before each test, which is very important in unit testing since it defines that the tests shouldn't depend on each other.

Both `mount` and `shallowMount` use exactly the same API; the difference is in the rendering.

Step 2:

Now press the `RUN` button.

If the test fails because of Snapshot's mismatch for `MessageList.test.js.` then to regenerate them, run it with the `-u` option:

```
npm t -- -u
```

Now if you open and inspect `test/__snapshots__/MessageList.test.js.snap`, you'll see `class="message"`, meaning the component has been rendered.

MessageList.test.js.snap

```
exports[`MessageList.test.js has the expected html structure 1`] = `
<ul>
  <li class="message">
    Cat
  </li>
</ul>
`;
```

> **Avoid deep rendering when there might be side effects**, since children component hooks such as `created` and `mount` will be triggered, and there may be HTTP calls or other side effects that we don't want to be

called.

If you want to verify what we've discussed, add a `console.log` to the `Message.vue` component in the `created` hook.

```
require('./check-versions')()

process.env.NODE_ENV = 'production'

var ora = require('ora')
var rm = require('rimraf')
var path = require('path')
var chalk = require('chalk')
var webpack = require('webpack')
var config = require('../config')
var webpackConfig = require('./webpack.prod.conf')

var spinner = ora('building for production...')
spinner.start()

rm(path.join(config.build.assetsRoot, config.build.assetsSubDirectory), err => {
  if (err) throw err
  webpack(webpackConfig, function (err, stats) {
    spinner.stop()
    if (err) throw err
    process.stdout.write(stats.toString({
      colors: true,
      modules: false,
      children: false,
      chunks: false,
      chunkModules: false
    }) + '\n\n')

    console.log(chalk.cyan('  Build complete.\n'))
    console.log(chalk.yellow(
      '  Tip: built files are meant to be served over an HTTP server.\n' +
      '  Opening index.html over file:// won\'t work.\n'
    ))
  })
})
```

Now, if you press the `RUN` button, you'll see the `"CREATED!"` text in the terminal output. So, be cautious.

Let's test a running project of what we have done so far in the next lesson.