Webpack Aliases

In this lesson, we will be exploring Webpack aliases.

WE'LL COVER THE FOLLOWING

The module managers we have in the JavaScript community, mainly ES Modules and CommonJS, don't support project-based paths. They only support relative paths for our own modules and paths for the `node_modules` folder. When a project grows a bit, it's common to see paths such as:

```
import SomeComponent from "../../../../components/SomeComponent";
```

Luckily, we have different ways to cope with this in a way that we can define aliases for folders relative to the project root, so we could write the above line like:

```
import SomeComponent from "@/components/SomeComponent";
```

The `@` here is an arbitrary character to define the root project; you can define your own. Let's see what solutions we have to apply to module aliasing. Let's start from where we left off in the last chapter.

How to Set Up Webpack Aliases?

Webpack aliases are very simple to set up. You just need to add a `resolve.alias` property in your webpack configuration. If you take a look at the `build/webpack.base.conf.js`, it has already defined this:

build/webpack.base.conf.js

```
module.exports = {
  // ...
  resolve: {
    extensions: [".js", ".vue", ".json"],
    alias: {
      vue$: "vue/dist/vue.esm.js"
    }
  }
};
```

Taking this as an entry point, we can add a simple alias that points to the `src` folder and uses that as the root:

```
module.exports = {
  // ...
  resolve: {
    extensions: [".js", ".vue", ".json"],
    alias: {
      vue$: "vue/dist/vue.esm.js",
      "@": path.join(__dirname, "..", "src")
    }
  }
};
```

Just with this, we can access anything taking the root project as the `@` symbol. Let's go to `src/App.vue` and change the reference to those two components:

```
import MessageList from "@/components/MessageList";
import Message from "@/components/Message";
// ...
```

We still haven't configured Jest to do so. So let's go to `package.json` where the Jest config is, and add `"@/([^\\.]*)$": "<rootDir>/src/$1"` to `moduleNameMapper`:

```
{
  "jest": {
    "moduleNameMapper": {
      "@(.*)$": "<rootDir>/src/$1",
      "^vue$": "vue/dist/vue.common.js"
    }
  }
}
```

Explanation

Let's explain it:

- `@(.*)$`: Whatever starts with `@`, continues with (`(.*)$`) till the end of the string, grouping it by using the parenthesis
- `<rootDir>/src/$1`: `<rootDir>` is a special word of Jest, meaning the root directory. Then we map it to the `src`, and with `$1` we append the whatever clause from the `(.*)` statement.

For example, `@/components/MessageList` will be mapped to `../src/components/MessageList` when you're importing it from the `src` or `test` folders.

That's really it. Now you can update your `App.test.js` file to use the alias as well since it's usable from within the tests:

```
import { shallowMount } from "@vue/test-utils";
import App from "@/App";
// ...
```

It will work for both `.vue` and `.js` files.

Now that we have learned about aliases, let's deal with multiple aliases.
