Using vue-template-loader with Vue.js to Compile HTML Templates

alligator.io/vuejs/vue-template-loader

Most people familiar with Angular 2+ know that in order to compile HTML templates one simply needs to add a template url in the component's TypeScript file and be done with it. With Vue, the recommendation is to instead use template tags to build your template's markup in the vast majority of cases.

We can use vue-template-loader if we want to use Vue with the Angular-way of building templates. Since **vue-template-loader** supports `vue-class-component` we can use decorators on classes for class-styled components.

vue-template-loader compiles HTML into individual render functions in the respective TypeScript or JavaScript files.

Installation

We'll need a typical seed `Vue.js` project , along with `webpack` dependencies.

Install `vue-template-loader` using `yarn` or `npm` like this:

```
# yarn
$ yarn add vue-template-loader

# npm
$ npm install vue-template-loader
```

webpack Configuration for JavaScript

Now we can integrate **vue-template-loader** using `webpack` .

Add `vue-template-loader` as a rule in your webpack config file:

webpack.config.js

```
module.exports = {
  module: {
    rules: [
        {
          test: /\.html$/,
          loader: 'vue-template-loader',

          exclude: /index.html/,
        }
    ]
  }
}
```

Rendering the assets used in our HTML file like with processing the *src* attribute of tags can specified with options:

webpack.config.js

```
module.exports = {
  module: {
    rules: [
      {
        test: /\.html$/,
        loader: 'vue-template-loader',


        exclude: /index.html/,
        options: {
          transformToRequire: {
            img: 'src'
          }
        }
      }
    ]
  }
}
```

Please note that for the above options to work we also need to add a loader to handle the image files (see file-loader).

TypeScript configuration

If we want to use **vue-template-loader** with TypeScript we need to have the `tsloader` and `typescript` dependencies installed in the project along with `webpack`.

vue-template-loader is used the same way in webpack's config for both JavaScript and TypeScript.

The only addition will be in the *typings* folder of our project. We need to add the following shim in the *typings* folder to make TypeScript understand *.vue* files:

```
declare module '*.vue' {
  import Vue from 'vue';
  export default Vue;
}


declare module '*.html' {
  import Vue, { ComponentOptions } from 'vue';

  interface WithRender {
    <V extends Vue>(options: ComponentOptions<V>): ComponentOptions<V>
    <V extends typeof Vue>(component: V): V
  }

  const withRender: WithRender
  export = withRender
}
```

Usage in Javascript / Typescript Files

Now, let’s create an example with a template file that we’ll call `nest.html`:

nest.html

```
<div class="nest">
  <p>{{ text }}</p>
  <button type="button" @click="baz()">Click Me!</button>
</div>
```

Let’s add a `nest.js` file corresponding to `nest.html`. We can use **vue-template-loader** with or without class decorators when using *es6* with *Vue*:

nest.js

```
import withRender from './nest.html';

export default withRender({
  data () {
    return {
      text: 'I\'m an alligator'
    };
  },
  methods: {
    baz () {
      console.log('Clicked!');
    };
  };
});
```

nest.js

```
import Vue from 'vue';
import Component from 'vue-class-component';
import WithRender from './nest.html';

@WithRender
@Component
export default class Nest extends Vue {
  text = 'I\'m an alligator!';

  baz() {
    console.log('Clicked!');
  }
}
```

It can also be used in TypeScript like this:

nest.ts

```
import Vue from 'vue';
import { Component } from 'vue-property-decorator';
import WithRender from './nest.html';

@WithRender
@Component({})
export default class NestComponent extends Vue {
  data(){
    return {
      text: 'I\'m an alligator!'
    }
  };

  baz(){
    console.log('clicked!');
  }
};
```

Conclusion

Using **vue-template-loader** provides great support for TypeScript, and can also decrease the number of files to be compiled as it eliminates *.vue* files. Lastly, it can be really easy to understand for people coming from an Angular background.