08

Importando seus arquivos JS e peculiaridades de objetos globais

Transcrição

A ideia é trabalharmos com componentes que encapsulam dado e comportamento, no entanto pode ser que precisemos criar algum script para ser aplicado globalmente. O procedimento é o mesmo que adotamos para importar arquivos `css` . Por exemplo, vamos criar o arquivo `alurapic/src/assets/js/teste.js` que exibe um alerta na tela:

```
console.log('Funcionou');
```

COPIAR CÓDIGO

Agora, podemos importá-lo em qualquer módulo. No caso, vamos importá-lo em `alurapic/src/main.js` :

```
// alurapic/src/main.js

import Vue from 'vue'
import App from './App.vue'
import VueResource from 'vue-resource';
import VueRouter from 'vue-router';
import { routes } from './routes';
import './directives/Transform';
import VeeValidate from 'vee-validate';
import msg from './pt_BR';
import 'bootstrap/dist/css/bootstrap.css';
import './assets/css/teste.css';
```

```
import './assets/js/teste.js'; // importando o script!
```

COPIAR CÓDIGO

Reiniciando nosso servidor, o alerta será exibido assim que a aplicação for carregada. Inclusive é possível carregar scripts de módulos baixados através do `npm` assim como fizemos com o `bootstrap` :

```
// alurapic/src/main.js

import Vue from 'vue'
import App from './App.vue'
import VueResource from 'vue-resource';
import VueRouter from 'vue-router';
import { routes } from './routes';
import './directives/Transform';
import VeeValidate from 'vee-validate';
import msg from './pt_BR';
import 'bootstrap/dist/css/bootstrap.css';
import './assets/css/teste.css';
import './assets/js/teste.js';

import 'bootstrap/dist/js/bootstrap.js';
```

COPIAR CÓDIGO

No entanto, uma mensagem de erro será exibida no console do navegador, alegando que Bootstrap precisa de jQuery. Como Bootstrap não é realmente um módulo, não há como Webpack resolver essa dependência automaticamente. Para resolver, podemos baixar o jQuery através do terminal:

```
npm install jquery@3.1.1 --save
```

COPIAR CÓDIGO

Se você já trabalhou alguma vez com o `jQuery` deve lembrar que ele nada mais é do que uma variável global. Nesse caso, ela precisa estar disponível antes da importação do script do bootstrap que fizemos. Não podemos fazer simplesmente um import dele antes do bootstrap que não vai funcionar. Precisamos pedir para que o Webpack **resolva** o jQuery antes dos nossos módulos serem carregados. Fazemos isso através de um plugin. O trecho de código que já já adicionaremos em nosso `webpack.config.js` é esse:

```
plugins: [
      new webpack.ProvidePlugin({
         $: 'jquery/dist/jquery.js',
         jQuery: 'jquery/dist/jquery.js'
      })
  ]
```

COPIAR CÓDIGO

No array de plugins, criamos um instância de `webpack.ProviderPlugin` que recebe como parâmetro um objeto JavaScript. A chave desse objeto é o apelido que queremos dar ao módulo que será carregado, o segundo é o caminho do módulo dentro de `node_modules` . Agora não teremos mais o erro anterior e tanto o jQuery quanto o Bootstrap estarão carregados, mas atenção! Se você procurar jQuery dentro do console do navegador não o encontrará, porque ele foi encapsulado dentro de um módulo pelo Webpack.

Nosso arquivo `webpack.config.js` final fica assim:

```
// alurapic/webpack.config.js

var path = require('path')
var webpack = require('webpack')

module.exports = {
  entry: './src/main.js',
  output: {
```

```
    path: path.resolve(__dirname, './dist'),
    publicPath: '/dist/',
    filename: 'build.js'
  },
  module: {
    rules: [
      {
        test: /\.vue$/,
        loader: 'vue-loader',
        options: {
          loaders: {
            // Since sass-loader (weirdly) has SCSS as its
default parse mode, we map
            // the "scss" and "sass" values for the lang
attribute to the right configs here.
            // other preprocessors should work out of the
box, no loader config like this nessessary.
            'scss': 'vue-style-loader!css-loader!sass-
loader',
            'sass': 'vue-style-loader!css-loader!sass-
loader?indentedSyntax'
          }
          // other vue-loader options go here
        }
      },
      {
        test: /\.js$/,
        loader: 'babel-loader',
        exclude: /node_modules/
      },
      {
        test: /\.(png|jpg|gif|svg)$/,
        loader: 'file-loader',
        options: {
          name: '[name].[ext]?[hash]'
        }
      },
      { test: /\.css$/, loader: 'style-loader!css-loader' },
```

```
      { test: /\.(woff|woff2)(\?v=\d+\.\d+\.\d+)?$/, loader:
'url-loader?limit=10000&mimetype=application/font-woff' },
      { test: /\.ttf(\?v=\d+\.\d+\.\d+)?$/, loader: 'url-
loader?limit=10000&mimetype=application/octet-stream' },
      { test: /\.eot(\?v=\d+\.\d+\.\d+)?$/, loader: 'file-
loader' },
      { test: /\.svg(\?v=\d+\.\d+\.\d+)?$/, loader: 'url-
loader?limit=10000&mimetype=image/svg+xml' }
    ]
  },
  resolve: {
    alias: {
      'vue$': 'vue/dist/vue.common.js'
    }
  },
  devServer: {
    historyApiFallback: true,
    noInfo: true
  },
  performance: {
    hints: false
  },
  devtool: '#eval-source-map',
  plugins: [
        new webpack.ProvidePlugin({
           $: 'jquery/dist/jquery.js',
           jQuery: 'jquery/dist/jquery.js'
        })
    ]
}

if (process.env.NODE_ENV === 'production') {
  module.exports.devtool = '#source-map'
  // http://vue-loader.vuejs.org/en/workflow/production.html
  module.exports.plugins = (module.exports.plugins ||
[]).concat([
    new webpack.DefinePlugin({
      'process.env': {
```

```
        NODE_ENV: '"production"'
      }
    }),
    new webpack.optimize.UglifyJsPlugin({
      sourceMap: true,
      compress: {
        warnings: false
      }
    }),
    new webpack.LoaderOptionsPlugin({
      minimize: true
    })
  ])
}
```

COPIAR CÓDIGO

Com esse extra, vocês ficaram mais seguros com Vue CLI! No entanto, a complexidade de Webpack justificaria um curso só para ele!