# 创建默认项目

## 安装：

Yarn global add @vue/cli

Yarn global add

# Vue.config.js 配置

## 基本配置

// 基本路径

publicPath: process.env.NODE\_ENV === 'production' ? './' : './',

// 输出文件目录

outputDir: 'dist', // 默认dist

// 用于嵌套生成的静态资产（js,css,img,fonts）目录

assetsDir: './assets',

// 指定生成的 index.html 的输出路径 (相对于 outputDir)。也可以是一个绝对路径

indexPath: 'index.html', // Default: 'index.html'

// 默认情况下，生成的静态资源在它们的文件名中包含了 hash 以便更好的控制缓存。

filenameHashing: true,

// 构建多页时使用

pages: {

index: {

// index page 的入口

entry: 'src/main.js',//例如：public/index/main.js

// 模板来源

template: 'public/index.html',

// 在 dist/index.html 的输出名称

filename: 'index.html',

// 当使用 title 选项时，

// template 中的 title 标签需要 是 <title><%= htmlWebpackPlugin.options.title %></title>

title: 'sunshiheima',//标题名称在此修改

// 在这个页面中包含的块，默认情况下会包含

// 提取出来的通用 chunk 和 vendor chunk。

chunks: ['chunk-vendors', 'chunk-common', 'index']

},

        // 当使用只有入口的字符串格式时，

        // 模板会被推导为 `public/subpage.html`

        // 并且如果找不到的话，就回退到 `public/index.html`。

        // 输出文件名会被推导为 `subpage.html`。

        //   subpage: 'src/subpage/main.js'

    },

// eslint-loader是否在保存的时候检查

lintOnSave: true,

// 是否使用包含运行时编译器的Vue核心的构建

runtimeCompiler: false,

// 默认情况下 babel-loader 会忽略所有 node\_modules 中的文件。如果你想要通过 Babel 显式转译一个依赖，可以在这个选项中列出来

transpileDependencies: [],

// 如果你不需要生产环境的 source map，可以将其设置为 false 以加速生产环境构建。

productionSourceMap: false,

## devServer

 devServer: {

        // host: 'localhost',设置本地IP地址

        contentBase: path.resolve(\_\_dirname, 'dist'),

        port: 2333,

        https: false,

        open: true,

        // hotOnly: false,

        //    hot:true,//热刷新 一般不需要 另外添加

},

## 第三方插件

 // 第三方插件配置

    configureWebpack: *config* => {

        // let obj=glob.sync(path.join(\_\_dirname, './src/\*\*/rest.scss'));

*config*.plugins.push(

        )

        //  console.log(config.plugins)

    },

## 对css-loader的修改

    css: {

        loaderOptions: {

            sass: {

                /\* 全局变量 \*/

                prependData: `@import "./src/assets/style/main.scss";`

            },

}

## Webpack的其他修改

// 是一个函数，会接收一个基于 webpack-chain 的 ChainableConfig 实例。允许对内部的 webpack 配置进行更细粒度的修改。

    chainWebpack: *config* => {

*config*.module

            .rule('images')

            .use('url-loader')

            .loader('url-loader')

            .tap(*options* => {

                // 修改它的选项...

                return *options*

            })

    },

## 去除多余的css

注意：推荐结合sunshiheima 自定义文件（rest.less/rest.scss + main.scss/main.less +index.less/index.scss） 文件使用. 自成一个体系。

如果不使用rest 文件 没必要去除多余样式，老项目迭代推荐使用；

### 安装

vue: vue add @fullhuman/purgecss

React：yarn add @fullhuman/postcss-purgecss -S

### Vue.config.js 文件引入

注意：@fullhuman/postcss-purgecss 如果不适配postcss 8 **[请将@fullhuman/postcss-purgecss将到3.0.0](mailto:请将@fullhuman/postcss-purgecss将到3.0.0)** 使用

const glob = require('glob-all');

const purgecss = require('@fullhuman/postcss-purgecss');

### 配置使用

purgecss({

         content: glob.sync([

           path.join(\_\_dirname, "./src/\*\*/\*.js"),

           path.join(\_\_dirname, "/\*\*/\*.vue"),

           path.join(\_\_dirname, "./public/\*\*/\*.html"),

         ]),

          /\* 要清除的css 文件 \*/

          // css:[ "./node\_modules/\*\*/\*.css","./src/\*\*/\*.scss"],

          defaultExtractor(*content*) {

             const contentWithoutStyleBlocks = *content*.replace(/<style[^]+?<\/style>/gi, '')

             return contentWithoutStyleBlocks.match(/[A-Za-z0-9-\_/:]\*[A-Za-z0-9-\_/]+/g) || []

            },

            safelist: [

             /-(leave|enter|appear)(|-(to|from|active))$/,

             /^(?!(|.\*?:)cursor-move).+-move$/,

             /^router-link(|-exact)-active$/,

             /data-v-.\*/,

             /^el-/,//不清除element-ui 样式

           ],

            /\*  /如果使用的时 CSS 动画库，例如 animate.css，你可以通过将 keyframes 参数设置为 true 来删除未使用的 keyframes。\*/

            keyframes: true,

            /\* 删除未使用的css变量 \*/

            variables: true,

            /\* 排除文件路径 \*/

            skippedContentGlobs: ["node\_modules/\*\*"],

            /\* 自定义样式 \*/

            // dynamicAttributes: [/^el-/,]

  })

# 接口封装