01.在某个路径下打开命令终端，输入mkdir juejin-element-ui创建文件夹juejin-element-ui，并cd juejin-element-ui到根目录下

02.在根目录执行npm init -y创建初始化的package.json文件，同时在同级处创建目录build|webpack.common.js文件，文件内容如下，划线的地方容易忘记

\*\*\*\*\*读懂本文件内容需要先看[CommonJS规范](https://weread.qq.com/web/reader/d1b32290718ff65fd1befcck70e32fb021170efdf2eca12)

const path = require('path');  
const VueLoaderPlugin = require('vue-loader/lib/plugin')  
  
module.exports = {  
 mode: 'production',  
 entry: './src/index.js', # 入口文件  
 output: {  
 path: path.resolve(process.cwd(), './lib'),  
 filename: 'element-ui.common',  
 libraryTarget: 'commonjs2'  
 },  
 module: {  
 rules: [  
 {  
 test: /\.vue$/,  
 loader: 'vue-loader',  
 options: {  
 complierOptions: {  
 preserveWhitespace: false  
 }  
 }  
 }  
 ]  
 },  
 plugins: [  
 new VueLoaderPlugin()  
 ]  
  
}

该文件表明需要安装包'vue-loader， complierOptions根目录下执行以下命令

webpack 只识别 .js 与 .json 文件

 .vue 后缀的文件，都需要找到相关的 loader 处理后才能交给 webpack

npm install webpack@4.14.0 webpack-cli@3.0.8 -D  
npm install vue-loader@15.7.0 vue-template-compiler@2.6.14 -D

03.在package.json同级创建入口文件，目录是src|index.js.文件内容如下。

此文件引入所有组件，

// 导入注册的vue组件  
import Button from '../packages/button/index.js'  
const components = [Button]  
const install = function(Vue) {  
 components.forEach(component => {  
 Vue.component[[1]](#footnote-0)[[2]](#endnote-0)[[3]](#footnote-1)[[4]](#endnote-1)(component.name, component)  
 })  
}  
export default {  
 Button,  
 install,  
}

划线处引入的是js文件，不是button|src|button.vue文件

04.在package.json同级创建组件文件夹packages|button|src

尝试之前的版本说明

|  |  |
| --- | --- |
| Node |  |
| npm |  |
| vue |  |
|  |  |

**Webstorm**

**demo**

1. [↑](#footnote-ref-0)
2. |  |
   | --- |
   | 1. renderContent (h, { node, data, store }) { 2. **var** isAct = data.Type 3. // 添加项目类型识别 IsCooperationProject 4. let isCoo = data.IsCooperationProject && data.Type === 3 ? ' isCoo' : '' 5. let icon = '<span class="tree-icon icon' + isAct + isCoo + '"></span>' 6. Vue.component('tree-child', { 7. template: '<span class="did-name-parent">' + icon + '<span title="' + data.Name + '" class="did-name">' + data.Name + '</span></span>' 8. }) 9. **return** h('tree-child') 10. }, |

   |  |
   | --- |
   | 1. **<div** class="tree-box"**>** 2. **<el-tree** accordion highlight-current ref="proTree" :data="projectData" :default-expand-all="true" :props="defaultProps" node-key="Id" :render-content="renderContent" :filter-node-method="filterNode" @node-click="clickProject"**></el-tree>** 3. **</div>** |

   [↑](#endnote-ref-0)
3. [↑](#footnote-ref-1)
4. |  |
   | --- |
   | //文件上传控件 Vue.component("sapi-upload", {  template: '<div class="sapi-upload">' +  '<el-upload v-if="readonly!=true" :disabled="isReadly" class="sapi-upload-file" :on-change="change" :auto-upload="false" :show-file-list="false" :file-list="fileList">' +  '<button :readonly="isReadly" class="upload-btn el-button el-button--primary">点击上传</button>' +  '</el-upload>' +  '</div>',  data: function() {  return {}  },  props: ["size", "count", "type", "value", "readonly"],  methods: {},  mounted: function() {},  watch: {}  }); |
   |  |

   //文件上传控件  
    Vue.component("sapi-upload", {  
    template: '<div class="sapi-upload">' +  
    '<el-upload v-if="readonly!=true" :disabled="isReadly" class="sapi-upload-file" :on-change="change" :auto-upload="false" :show-file-list="false" :file-list="fileList">' +  
    '<button :readonly="isReadly" class="upload-btn el-button el-button--primary">点击上传</button>' +  
    '</el-upload>' +  
    '<ul class="sapi-upload-list" :class="{\'sapi-upload-list-big\':readonly}">' +  
    '<li v-for="(item,index) in fileList">' +  
    '<p>' +  
    '<span class="hover">下载</span>' +  
    '<span class="hover closeBtn" v-if="readonly!=true" @click="remove(index)">删除</span>' +  
    '</p>' +  
    '<a class="sapi-upload-list-title" :title="item.name" href="javascript:void(0)" @click="preview(item,index)">{{item.name}}</a>' +  
    '<span class="sapi-upload-list-size">{{(item.size?(item.size/1024).toFixed(3):0) + "k"}}</span>' +  
    '</li>' +  
    '</ul>' +  
    '</div>',  
    data: function() {  
    return {  
    fileList: [],  
    isReadly: false  
    }  
    },  
    props: ["size", "count", "type", "value", "readonly"],  
    methods: {  
    remove: function(index) {  
    this.fileList.splice(index, 1);  
    this.isReadly = false;  
    },  
    preview: function(file) {  
    if(file.raw && file.raw.type.search(/^image\//) > -1) {  
     
    }  
     
    window.open(file.url);  
    },  
    change: function(file, fileList) {  
    var datas = this.fileList;  
    for(var i = 0, ii = datas.length; i < ii; i++) {  
    var data = datas[i];  
    if((data.name + data.size) == (file.name + file.size)) {  
    fileList.splice(fileList.length - 1, 1);  
    Vue.msg("不能上传已存在文件");  
    return false;  
    }  
    }  
     
    if(this.type) {  
    var types = this.type.split(",").join("$|") + "$";  
    var reg = new RegExp(types, "ig");  
    if(!reg.test(file.name)) {  
    fileList.splice(fileList.length - 1, 1);  
    Vue.msg("不能上传该格式文件");  
    return false;  
    }  
    }  
     
    if(this.size) {  
    if(this.size \* 1024 < file.size) {  
    fileList.splice(fileList.length - 1, 1);  
    Vue.msg("不能上传大于" + this.size + "k的文件");  
    return false;  
    }  
    }  
     
    if(this.count) {  
    if(this.count == fileList.length) {  
    this.isReadly = true;  
    }  
    }  
     
    this.fileList = fileList.reverse();  
    this.$emit("input", fileList);  
    }  
    },  
    mounted: function() {  
    this.fileList = this.value || [];  
     
    /\*this.isReadly = this.readonly;\*/  
     
    if(this.count && this.fileList.length >= this.count) {  
    this.isReadly = true;  
    }  
     
    /\*setTimeout(function(){  
    var ele=this.$el;  
    ele.style.width=ele.offsetWidth+"px";  
    }.bind(this),300);\*/  
    },  
    watch: {  
    value: function(val) {  
    this.fileList = val || [];  
    }  
    }  
     
    });

   Webstorm

   libraryTarget的几种选择我们来好好分析

   [](https://www.zhihu.com/people/jingwen0_0)

   **[静文](https://www.zhihu.com/people/jingwen0_0" \t "https://zhuanlan.zhihu.com/p/_blank)**

   关注一下？公众号「静文小同学」

   ​关注他

   13 人赞同了该文章

   好久没写库了，发现时代变了

   这一切的起因都是因为那只pm，整个一个奇奇怪怪的需求，然后发现需要写库了，就在我流利的初始化项目时，突然脑抽风的发现，library配置貌似不太对，libraryTarget配置貌似也不太对，所以还是停下来吧，好好分析回顾一下，阿西吧。

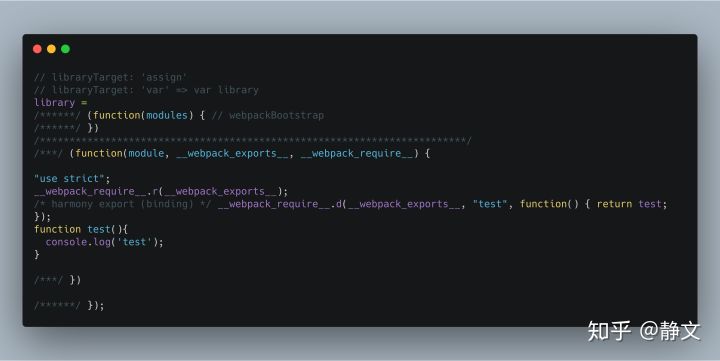
   先来回顾一下，library是指定义一个全局使用的名称变量，libraryTarget是指设置library的暴露方式，是commonjs、commonjs2、umd还是this、var等。

   变量定义：

   libraryTarget:“assign”，暴露一个未定义的library设置的变量。在node环境不支持。

   libraryTarget:“var”，暴露一个用var 定义的library设置的变量。在node环境下不支持。

   这两个属性，都是在全局创建一个变量，只有定义与未定义的区别，并且并不能在nodejs中得的支持，并且存在变量冲突的可能性。

   观察一下libraryTarget:“assign” or “var”产出的代码，只是创建了一个library里设置的变量，然后通过webpack的方法做一个导出。

   对象定义：

   libraryTarget:“window”，在window对象上定一个library设置的变量。在node环境下不支持。

   libraryTarget:“global”，在global对象上定义一个library设置的变量。受target属性影响，当target为默认值web时，会在window上注册，如果你想在global上注册，必须修改target为node。

   libraryTarget:“this”，在当前this对象上定义一个library设置的变量，如果this是window，就在window。在node的环境中，如果没指定require赋值的变量，并不会在指向global。

   这三个属性的特征都是在公共对象上export出你的方法函数。特点无非是减少了变量冲突的可能性，但是依旧没有解决问题，只有global模式支持在node环境中，还必须设置target为node不然也是不支持的。

   观察一下libraryTarget:“window” or “global” or “this”产出的代码，在相应的libraryTarget设置的对象上，创建一个library设置的变量，然后根据webpack的方法做一个导出。global有一个注意点，那就是target配置。如果没有写那默认是在window对象上注册。

   模块定义

   libraryTarget:“commonjs”，在export对象上定义library设置的变量。在node中支持，浏览器中不支持。

   libraryTarget:“commonjs2”，直接用module.export导出export，会忽略library设置的变量。在node中支持，在浏览器中不支持。

   libraryTarget:“amd”，在define方法上定义library设置的变量，不能用script直接引用，必须通过第三方模块RequireJS来时用

   这两个属性，是符合模块化规范的产物，commonjs是在export上定义library设置的变量，commonjs2是用module.export直接export的。amd的依赖前置方案在浏览器、node中都必须额外引入RequireJS来使用。

   commonjs/commonjs2，查看源码的话，commonjs是在exports中创建一个library设置的变量，而commonjs2是直接把方法export到module.exports中，这就是为什么commonjs2会忽略library的原因。

   amd那就是在define的方法中设置了library，这样就能符合RequireJS的使用规范。

   兼容的模块化定义

   libraryTarget:“umd”，该方案支持commonjs、commonjs2、amd，可以在浏览器、node中通用。

   但是如果你想做到这一点，必须要额外设置，umdNamedDefine: true，globalObject: ‘this’，umdNamedDefine为设置amd前置名称使用library设置的变量，globalObject为改变全局指向。这样就能保证你的库在node和浏览器中通用了。当然便捷的引入一定会带来一定的冗余，这就看你如何取舍了。

   umd的核心无法是webpackUniversalModuleDefinition中的判断，这里会做环境判断，去使其能够使用，这里有必要注意两个关注点，那就是window对象，这个对象在node中是无法使用的，并且会报错。而且define([var])的var为空值这也会在RequireJS中报错。如果你需要在node、RequireJS中使用，那么就需要额外配置umdNamedDefine、globalObject了。

   这样，你的define和window就符合要求了。

   最后建议，如果目标明确，我只是兼容nodejs，那么选择commonjs/commonjs2，如果只兼容浏览器，那就选择暴露变量的方式，如果想通用，那就选择umd的方式，对于不同的情况做多种处理方式，是非常明智的选择。

   最后附上表格一张，仅供参考。

   [点击奇奇怪怪的地方，留下了一点文字。](https://link.zhihu.com/?target=https://mp.weixin.qq.com/mp/profile_ext?action=home&__biz=MzAxMDM1ODg3NQ==&scene=124%23wechat_redirect" \t "https://zhuanlan.zhihu.com/p/_blank)

   编辑于 2021-03-03 16:44 [↑](#endnote-ref-1)