# 基础信息

# Vue

## 理解

### Vue不支持IE8以及更低版本的浏览器的原因

是因为Object.defineProperty在es5中无法shim（在旧的环境下，依靠原有的功能无法实现的）的特性

### 框架对比

更抽象一点来看，我们可以把组件区分为两类：一类是偏视图表现的 (presentational)，一类则是偏逻辑的 (logical)。我们推荐在前者中使用模板，在后者中使用 JSX 或渲染函数。这两类组件的比例会根据应用类型的不同有所变化，但整体来说我们发现表现类的组件远远多于逻辑类组件

## Webpack修改

### vue打包后css里-webkit-前缀丢失

步骤一：找到build/webpack.prod.conf.js，注释下面这一段

/\* new OptimizeCSSPlugin({

cssProcessorOptions: config.build.productionSourceMap

? { safe: true, map: { inline: false } }

: { safe: true }

}), \*/

步骤二：在build/utils.js里加入 minimize: true，压缩css

const cssLoader = {

loader: 'css-loader',

options: {

sourceMap: options.sourceMap,

minimize: true

}

}

### Vue打包后图片路径引用失败处理

找到 build->utils.js,在里面加入一句publicPath:'../../'

if (options.extract) {

return ExtractTextPlugin.extract({

use: loaders,

publicPath:'../../',

fallback: 'vue-style-loader'

})

} else {

return ['vue-style-loader'].concat(loaders)

}

### 不打包vue vuex router axios等库

找到build/webpack.prod.conf.js文件，在module.exports中配置

externals: {

'vue': 'Vue',

'vue-router': 'VueRouter',

'vuex': 'Vuex',

'axios': 'axios'

},

Cdn路径

<script src="http://cdn.bootcss.com/vue/2.5.16/vue.min.js"></script>

<script src="http://cdn.bootcss.com/vue-router/3.0.1/vue-router.min.js"></script>

<script src="http://cdn.bootcss.com/vuex/3.0.1/vuex.min.js"></script>

<script src="http://cdn.bootcss.com/axios/0.18.0/axios.min.js"></script>

### 忽略console.Log

Build -> webpack.prop.conf.js

new webpack.optimize.UglifyJsPlugin({

compress:{

warnings: false,

drop\_debugger: true,

drop\_console: true

}

}),

### 优化打包速度

安装：

npm i webpack-parallel-uglify-plugin

修改配置文件build/webpack.prod.conf.js

const ParallelUglifyPlugin = require('webpack-parallel-uglify-plugin');

//....

// 删掉webpack提供的UglifyJS插件

//new UglifyJsPlugin({

// uglifyOptions: {

// compress: {

// warnings: false

// }

// },

// sourceMap: config.build.productionSourceMap,

// parallel: true

//}),

// 增加 webpack-parallel-uglify-plugin来替换

new ParallelUglifyPlugin({

cacheDir: '.cache/',

uglifyJS:{

output: {

comments: false

},

compress: {

warnings: false

}

}

}),

## 模板

name: "",

components: {},

mixins: [ user ],

beforeRouteEnter(to, from, next) {next(vm => {})},

beforeRouteLeave(to, from, next) {next();},

beforeRouteUpdate(to, from, next) {next();},

beforeCreate() {},

data() {return {} },

created() {},

beforeMount() {this.\_init\_page()},

mounted() {},

activated() {},

props: {},

computed:{},

methods: {

async \_init\_page() {

await this.$get\_user\_message()

},

\_load\_more(){

}

},

watch: {/\*'': {handler(val, olval){}, deep: true ,immediate:true} \*/},

destroy() {

}

## 坑点

### 修改prop数据，作为组件内部使用

定义一个局部变量，并用 prop 的值初始化它：

props: ['initialCounter'],

data: function () {

return { counter: this.initialCounter }

}

定义一个计算属性，处理 prop 的值并返回：

props: ['size'],

computed: {

normalizedSize: function () {

return this.size.trim().toLowerCase()

}

}

### 路由改变页面数据不刷新

PS：当前页面跳转到当前页面时，不会执行mouted、activated等生命周期

解决办法：可以在生命周期beforeRouteUpdate (to, from, next) { }中执行方法，页面通过监听‘$route’等方式

### setInterval路由跳转继续运行并没有及时进行销毁

比如一些弹幕，走马灯文字，这类需要定时调用的，路由跳转之后，因为组件已经销毁了，但是setInterval还没有销毁，还在继续后台调用，控制台会不断报错，如果运算量大的话，无法及时清除，会导致严重的页面卡顿。

解决办法：在组件生命周期beforeDestroy停止setInterval

//组件销毁前执行的钩子函数，跟其他生命周期钩子函数的用法相同。

beforeDestroy(){

//我通常是把setInterval()定时器赋值给this实例，然后就可以像下面这么停止。

clearInterval(

this

.intervalId);

},

首先得将定时器定义一个组件变量

### 组件内引入外部js

#### 方法一

缺点：不能在页面加载之前获取到js，导致需要使用js的时候，需要定时器来实现

components: {

['ScriptLink']:{

render(createElement){

return createElement(

'script',

{

attrs:{

type:'text/javascript',src:this.src

}

}

)

},

props:{ type:String,required:true }

}

},

<ScriptLink src="static/js-library/echarts.min.js"></ScriptLink>

#### 方法二

缺点：将引入的js打包成一个js后，会变得比较庞大，使得加载页面的时候回卡顿一小会儿 。手机端比较敏感的（用于单独页面引用（减少首次页面加载速度），以及需要打包才能使用的（比如自己写的一些工具类））

Webpack.base.conf.js中添加

resolve: {

extensions: ['.js', '.vue', '.json'],

alias: {

'vue$': 'vue/dist/vue.esm.js',

'jquery':resolve('static/js-library/jquery.js'),

'echarts':resolve('static/js-library/echarts.min.js'),

'@': resolve('src'),

}

},

### 数据加载时的加载动画

//loading 1 - 不加载 2 - 一直加载 3 - 只加载第一次

let isLoadingAnimate = true ;

if(loading === 3 ){

let urls = sessionStorage.getItem('urls') ? JSON.parse(sessionStorage.getItem('urls')) : [];

if(urls.indexOf(url) === -1){

isLoadingAnimate = true;

urls.push(url);

sessionStorage.setItem('urls',JSON.stringify(urls));

}else{

isLoadingAnimate = false;

}

}

if(loading === 2 || ( loading === 3 && isLoadingAnimate) ){

this.$createMyLoading().show();

interval = 350; // 如果需要加载动画，则延长返回时间，否则接口过快，出现页面闪烁

}

### 安装好后，启动失败

Ps:报错 Module build failed: Error: ENOENT: no such file or directory, scandir

处理办法：npm rebuild node-sass

## 淘宝镜像

--registry=https://registry.npm.taobao.org

# JavaScript

## Shim和polyfill的区别

Shim:指的是一个库，它将一个新的API引入到一个旧的环境中，而且仅靠旧环境中已有的手段实现，shim有时也称之为shiv，比如著名的html5shiv

Polyfill:它就是一个用在浏览器API上的shim，我们的通常做法是先检查浏览器是否支持某个API，如果不支持的话就加载对应的polyfill，然后新旧浏览器都可以使用这个API。

## 图片上传

//选择图片

\_select\_pic(e){

let file = e.target.files[0];

let formData = new FormData();

formData.append('file',file);

this.$upload\_picture(formData).then((res)=>{

console.log(res)

})

}

$upload\_picture(image){

return new Promise((resovle,reject)=>{

let config = { headers:{'Content-Type': 'multipart/form-data'} } ;

let loadPopup = this.$loading({

lock: true,

text: '上传中...',

background: 'rgba(0, 0, 0, 0.5)'

});

axios.post(configs['HTTP\_REQUEST']+'/res/uploadImage.do',image,config).then((response)=>{

setTimeout(()=>{

loadPopup.close()

},800);

resovle(response)

},(error)=>{

this.$message({

message: '上传图片失败！',

type: 'warning'

});

reject(error)

})

})

},

## 多端js工具类的封装

(function(f) {

if (typeof exports === "object" && typeof module !== "undefined") {

module.exports = f();

} else if (typeof define === "function" && define.amd) {

define([], f);

} else {

var g;

if (typeof window !== "undefined") {

g = window;

} else if (typeof global !== "undefined") {

g = global;

} else if (typeof self !== "undefined") {

g = self;

} else {

g = this;

}

g.Validator = f();

}

})(function () {

})

# 微信

## 分享（ios的坑）

PS：当浏览者从分享窗口进入后，他从分享页按着正常路由跳转到其他页面，其他页面的逻辑中如果有需要获取当前页面url的需要，会发现这个url和正常路由跳转的不一样,导致签名不一致,所以获取不到正确的文案。

解决: 每次路由变化时都重新请求下签名，签名的URL 用第一次进入时的URL就可以解决了

重点：分享路径后面的参数，不能带有中文，并且转码也不行，只能通过其他的方式，调用接口什么的

## Js验签

Ios和安卓的验签有一定的区别，如果只是按照安卓的方式验签，ios的验签会出现不稳定的情况；安卓的url需要编码进行验签，ios只需要进行“#”前面的域名进行验签即可

解决办法：

微信验签接口的调用：

function get\_wx\_sign(href){

return new Promise((resolve,reject)=>{

axios({

url:configs.HTTP\_WX + '/common/js.do?url=' + href

}).then((reponse)=>{

let data = reponse.data;

resolve(data)

},()=>{

reject()

})

})

}

// 设置微信js接口

export function set\_wx\_sign(href){

get\_wx\_sign(href).then((message)=>{

wx.config({

debug: false,

appId: message['appId'],

timestamp: message['timestamp'],

nonceStr:message['nonceStr'] ,

signature:message['signature'],

jsApiList: ['hideAllNonBaseMenuItem','previewImage','uploadImage','chooseImage','getLocalImgData','onMenuShareAppMessage','onMenuShareTimeline','showMenuItems','chooseWXPay']

});

})

}

进入页面之后：

let browser = browserType();

//alert(browser)

let web = window.location.href;

let href = '';

if(browser == 'ios'){

href = web.split('#')[0]; // 专门用于ios的js验签

}

if(browser =='android'){

web = web.split('#')[0]+ '#' + to.fullPath;

href = encodeURIComponent(web)

}

console.log(href)

set\_wx\_sign(href);

wx.ready(()=>{

wx.hideAllNonBaseMenuItem();

})

# 场景

## 登录

若一个接口调用返回004,需要登录，跳转到登录页面，会出现多个接口到达这个页面，而导致不能回到初始页面，需要使用销毁之前的页面，从而达到回到之前页面重新刷新该页面，使用window.location.reload()强制刷新页面，会出现当返回时还会到该页面，即会记录历史页面，交互效果差。

如果一个页面多个接口需要按步加载接口，因为多个接口出现返回004，达到登录页面会出现问题，即不能正常返回；也可以通过路由进入时做个判断，如果出现to.name == from.name，则不进行next()，可以将其拦下

if(to.meta.isLogin && !sessionStorage.getItem('login')){

next({

path: '/MainLogin',

query: { redirect: to.fullPath }

})

}else{

next()

}

sessionStorage.setItem('login',true);

let jumpPath = this.$route.query['redirect'] ? this.$route.query['redirect'] : '/Main';

this.$router.replace({path:jumpPath})

## 页面缓存（用5.4的方法替代更佳）

1.缓存页面的销毁和不销毁记录可能会导致项目有些逻辑问题；

缓存页面：列表（有分页情况），并且到详情页面

beforeRouteLeave(to,from,next){

let keepAliveToPath = from.meta.keepAliveToPath || [] ;

let toName = to.name ;

if(keepAliveToPath.indexOf(toName) == -1){

this.$destroy()

}

next()

}

需要在路由meta中添加keepAliveToPath，记录从哪个页面回来需要做缓存，

2.首先需要在路由配置中meta中添加{ isLoaded:false , isLoad:false, keepAlivePages:['MainUser'] }三个参数，

// isLoaded - 用于判断是否第一次加载，第一次加载是不管从来自哪个页面（即使是来自的页面需要缓存该页面），

// idLoad - 用于判断页面是否需要缓存

// keepAlivePages - 来自的页面需要缓存当前页面的页面组

在需要缓存的页面中 activated中添加方法

if(!this.$route.meta.isLoaded || (this.$route.meta.isLoaded && !this.$route.meta.isLoad)){

this.$route.meta.isLoaded = true;

this.\_init\_page();

}

3.之前遇到从一个缓存页面到不缓存页面，出现问题：不缓存页面要执行上一个缓存页面的mounted等生命周期内的方法，暂时没有出现多层级页面，暂时没有浮现该情况，有待验证；

## 生命周期

1. beforeRouteEnter：一个一级页面跳转到一个二级页面，出现问题：该生命周期执行两次，导致页面逻辑出错，解决办法，尽量不使用多层级页面，或者不在该生命周期中执行逻辑操作

## 接口调用加载动画

Ps:一些列表接口不需要进行二次加载，会让人烦

//用于判断接口是否是第一次加载，第一次加载才出现加载动画，之后不出现在加载动画

let isFirst = true ;

//debugger

if(isFirstLoading){

let urls = sessionStorage.getItem('urls') ? JSON.parse( sessionStorage.getItem('urls') ) : [];

if( urls.indexOf(url) == -1 ){

urls.push(url);

isFirst = true ;

}else{

isFirst = false

}

sessionStorage.setItem('urls',JSON.stringify(urls))

}

处理办法：在接口调用的时候添加一个参数isFirstLoading，判断该接口是否需要第一次加载需要动画，而之后不需要再出现加载动画

# 软件安装

## Webstorm 破解版

安装好软件之后 找到webstorm.exe.vmoptions、webstorm64.exe.vmoptions文件进行修改

修改首行

-javaagent:C:\AA.SOFT\AA.WEBSTORM\WebStorm 2018.1.5\bin\JetbrainsCrack-2.10-release-enc.jar

JetbrainsCrack-2.10-release-enc.jar

启动软件之后选择中间的选项，并将 -javaagent:C:\AA.SOFT\AA.WEBSTORM\WebStorm 2018.1.5\bin\JetbrainsCrack-2.10-release-enc.jar 复制进去即可

## 配置鼠标右键webtorm快捷打开文件夹

Win+R -> regedit -> HKEY\_CLASSES\_ROOT->Directory->shell->鼠标右键新建 项（K）webstorm->再在webstorm下新建项command->双击command,添加‘数值数据’（C:\AA.SOFT\AA.WEBSTORM\WebStorm 2018.1.5\bin\WebStorm.exe %1）（%1前有空格）

# Git

## 更新提交

Git pull

Git add \*

Git commit –m ‘user’

Git push origin master

## 使用问题

### 问题描述

输入Git add \*时

The following paths are ignor ed by one of your .gitignore files:

SC.WX.iml

build

Use -f if you really want to add them.

### 问题分析

这句提示意思是说我们用git add某个文件的时候，这个文件的路径已经被写入了.gitignore文件当中了，自动被忽略，所以add不上去。

问题通常让人费解的原因在于，我们去工程本目录查看.gitignore文件时，发现根本没有对这个文件进行忽略。

这里我们就遗忘掉了一点是，git存在全局gitignore，这个全局ignore通常配置在我们的.gitconfig文件中。

很多时候并不是我们自己加进去的，而是被其他集成了git版本控制的IDE工具（如Visual Studio）自动加入的。

### 问题解决

首先我们可以通过命令查看全局配置中是否有调用外部（非工程下）的gitignore文件

---------------------

本文来自 爱思考的孩纸总是痛苦滴 的CSDN 博客 ，全文地址请点击：https://blog.csdn.net/lingyanpi/article/details/71724282?utm\_source=copy

# Webstorm

## 快捷键

Ctrl + Shift + U —— 大小写切换

Ctrl + E —— 打开最近操作过的文件

Ctrl + Y —— 删除光标所在行

Ctrl + Alt + L —— 格式化代码

Ctrl + Alt + S —— 打开设置窗口

## 关闭 node\_modules 校验

在 node 项目中存在 node\_modules 目录，每次打开 webstrom 时会校验文件，同样也会校验 node\_modules 中的内容，这会浪费很多时间。

## 取消勾选不常用的插件

webstorm 中可以集成很多插件，这些插件也会影响运行速度，有的插件你可能压根都没听过，更不会使用，可以取消勾选

# Vscode

## 安装的实在的插件

Auto Close Tag – 自动闭合HTML/XML标签

Auto Rename Tag - 自动完成另一侧标签的同步修改

Beautify –格式化代码，值得注意的是，beautify插件支持自定义格式化代码规则，需要在自定义配置中添加

"vetur.format.defaultFormatter.html": "js-beautify-html",

为了覆盖

"vetur.format.defaultFormatter.html": "none",

Braket Pair Colorizer -给括号加上不同的颜色，便于区分不同的区块，使用者可以定义不同括号类型和不同颜色

HTML CSS Support - 智能提示CSS类名以及id ，需要在自定义配置中添加

"editor.parameterHints": true,

"editor.quickSuggestions": {

    "other": true,

    "comments": true,

    "strings": true

HTML Snippets - 智能提示HTML标签，以及标签含义

JavaScript(ES6) code snippets - ES6语法智能提示，以及快速输入，不仅仅支持.js，还支持.ts，.jsx，.tsx，.html，.vue，省去了配置其支持各种包含js代码文件的时间

Material Icon Theme - 个人认为最好的vscode图标主题，支持更换不同色系的图标，值得点出的是，该插件更新极其频繁，基本和vscode更新频率保持一致

open in browser - vscode不像IDE一样能够直接在浏览器中打开html，而该插件支持快捷键与鼠标右键快速在浏览器中打开html文件，支持自定义打开指定的浏览器，包括：Firefox，Chrome，Opera，IE以及Safari

Path Intellisense - 自动提示文件路径，支持各种快速引入文件

Vetur - Vue多功能集成插件，包括：语法高亮，智能提示，emmet，错误提示，格式化，自动补全，debugger。vscode官方钦定Vue插件，Vue开发者必备

VueHelper – vue的语法的提示

Chinese – 汉化编辑器

Rest Client – 发起http请求

Easy Less – 自动编译less文件

Cssrem – px自动转化为rem

Code Runner – 代码直接运行，鼠标右键即可 run code 运行

Vue 2 Snippets – vue代码提示

Htmltagwrap – 添加html标签 （选中需要包裹的元素，alt + w ）

## 添加的配置

### Vue文件格式化

PS：template中不能格式化

处理办法：在用户设置中添加"vetur.format.defaultFormatter.html": "js-beautify-html",覆盖"vetur.format.defaultFormatter.html": "none ",

### 我的css类名习惯用 - 横线连字符，可是在vscode每次双击只能选中其中一个单词，不能选中整个连字符

处理办法：在配置中添加，并去掉‘-’

"editor.wordSeparators": "`~!@#$%^&\*()-=+[{]}\\|;:'\",.<>/?",

### 不让新开文件覆盖旧文件

添加 "workbench.editor.enablePreview": false 覆盖 "workbench.editor.enablePreview": true

### 全局的搜索

Ps：默认没有全局搜索功能，

处理办法：添加

"serach.exclude":{//

"system/":true,

"!/system/\*/.ps\*":true

},

### 过滤打开文件的类型

"files.exclude": {

"\*\*/.idea": true,

"\*\*/.vscode": true,

"\*\*/dist": true,

"\*\*/node\_modules": true

},

### 关联文件的配置

files.associations 配置文件关联，比如

“files.associations”: {

“\*.vue”: “vue”,

“\*.wxss”: “css”

}

任何wxss后缀的文件会被认为是css文件 ，然后vscode会用css规则去显示代码风格，给出加载对应文件格式插件

# Html

## Input flie在Android和ios的区别

<input type="file" accept="video/\*" >

<input type="file" accept="audio/\*" >

<input type="file" accept="image/\*" multiple >

Ios不能调用audio

Ios 调用图片有时会出问题

## Vue单页应用在ios浏览器（非微信内置）上的路由表现

Document.title的设置，切换路由时也会出现title不正确的问题，原因是由于ios的路由切换时并没有真正的切换，导致标题不能正常显示