# 前端开发规范

目 录

[前端开发规范 1](#_Toc512677110)

[一、 前端开发流程 2](#_Toc512677111)

[二、 浏览器兼容 2](#_Toc512677112)

[二、开发工具及安装说明（说明可百度） 2](#_Toc512677113)

[三、脚手架搭建与运行 3](#_Toc512677114)

[四、数据存储 8](#_Toc512677115)

[五、包、文件命名、ReadMe规范 8](#_Toc512677116)

[六、注释编写规范 8](#_Toc512677117)

[七、HTML编写规范 9](#_Toc512677118)

[八、CSS编写规范 10](#_Toc512677119)

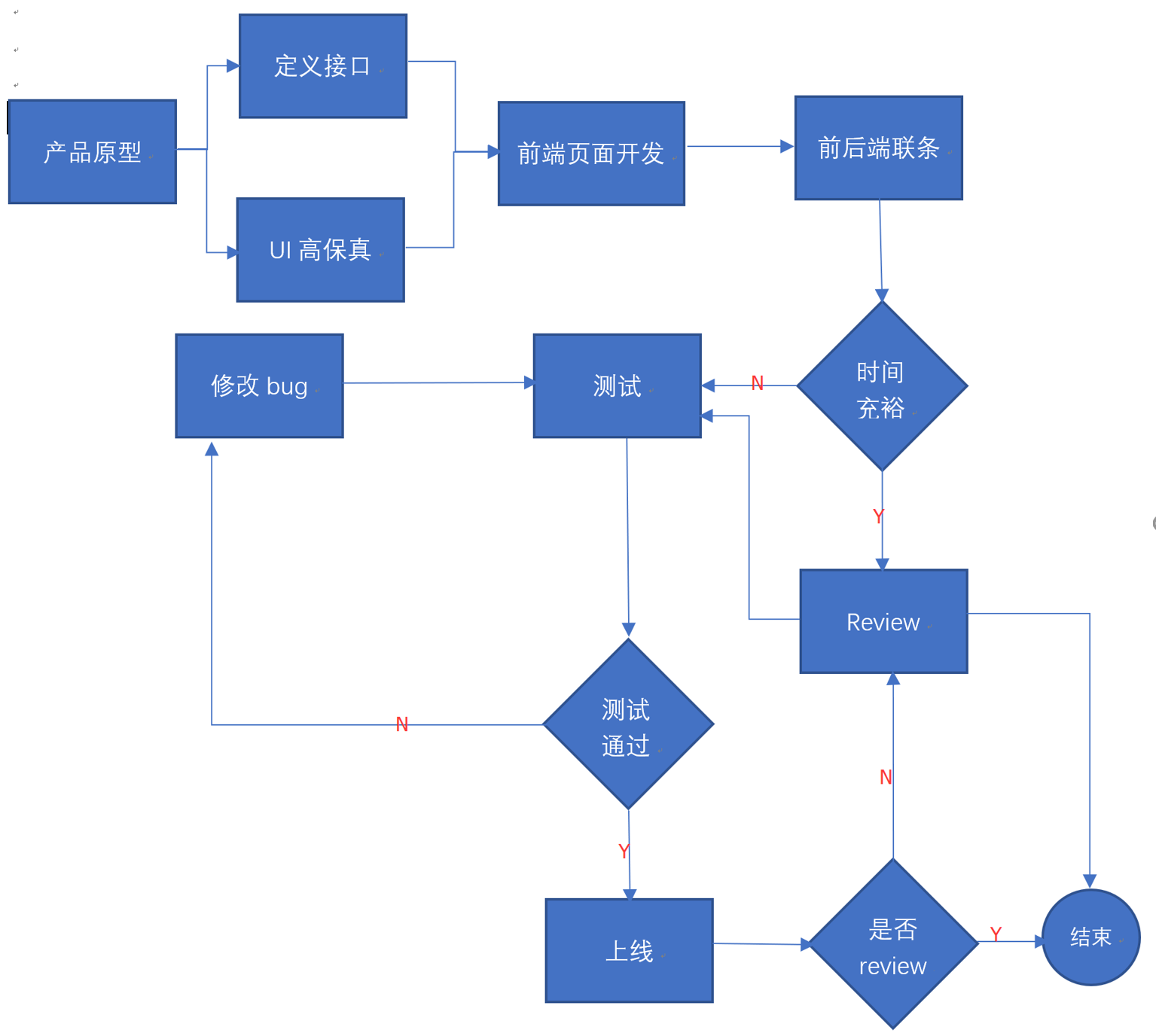
[九、图片处理规范 12](#_Toc512677120)

[十、前端JS规范 12](#_Toc512677121)

[十一、组件及组件说明规范 16](#_Toc512677122)

[十二、api文档 16](#_Toc512677123)

### 前端开发流程



### 浏览器兼容

IE8以上版本。因为IE8无法模拟ES5特性

### 二、开发工具及安装说明（说明可百度）

开发工具下载路径：

源码管理工具：

TortoiseSVN-1.9.7.27907-x64-svn-1.9.7.msi

LanguagePack\_1.9.7.27907-x64-zh\_CN.msi

环境部署相关：

运行环境：node-v8.9.3-x64.msi

sass编译：rubyinstaller-devkit-2.5.1-1-x64.exe

安装依赖包工具：cnpm

开发工具：Atom

原型及设计文件相关：

原型工具：AxureRP\_Setup\_Setup.exe

Axure\_RP\_8 CHS\_Patch.exe

设计工具：Adobe Fireworks CS6

photoshop

辅助工具：

文本比较器：BCompare-zh-3.3.8.16340.exe

取色器、标尺、坐标轴：picpick\_inst.exe

安装说明：

1 安装node

2 淘宝镜像 npm install -g cnpm --registry=https://registry.npm.taobao.org 参考 http://npm.taobao.org/

3 代码依赖模块安装 cnpm i 运行代码

4 统一编辑器atom 安装

5 编辑器插件安装 language-vue atom-beauty 网络不好可参考离线安装http://blog.csdn.net/a454213722/article/details/53888142

### 三、脚手架搭建与运行

1. 基础插件使用：

Axios：用来处理http请求

node-sass：用来处理scss文件

sass-loader：scss文件加载器

vue：vue项目核心库

vue-router：用来处理路由

vuex：用来处理全局变量及父子组件传值

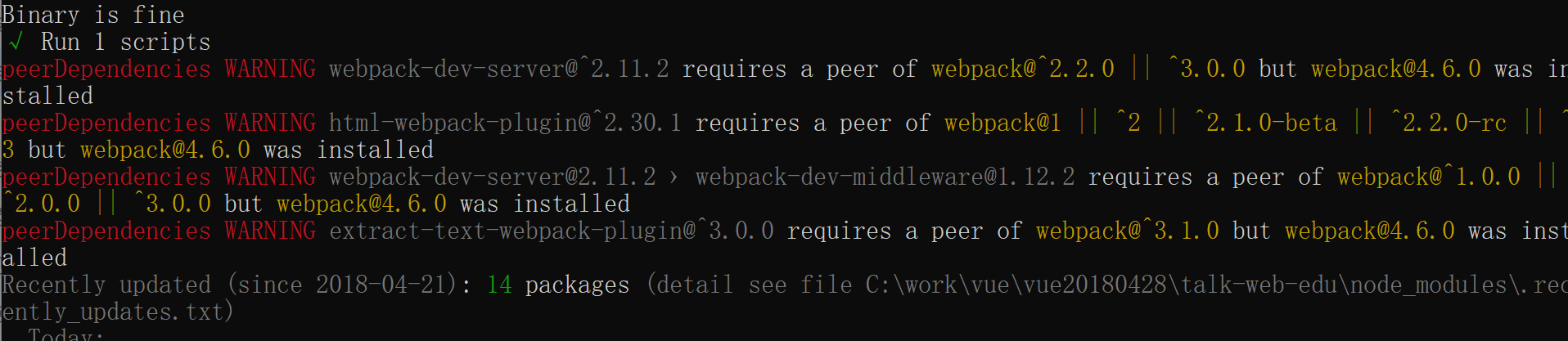
json-loader：用来加载本地的json文件

【百度编辑器，echarts（按需集成）】

1. 下载依赖包：cnpm install

添加cnpm add xxx

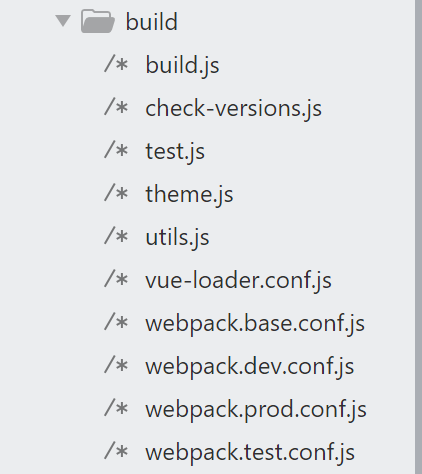
移除命令：cnpm uninstall xxx

1. npm install -g cnpm --registry=https://registry.npm.taobao.org
2. cnpm install
3. 注意webpack的版本。Webpack-dev-server@2.11.2等插件只支持webpack@ ^1.0.0、^2.0.0、^3.0.0，如果提示安装webpack相关插件失败（如图）请查看项目package.json文件中的wepack版本号。
4. 
5. 运行开发环境：cnpm run dev
6. 编译测试环境：cnpm run test
7. 编译生产环境：cnpm run build
8. 修改port=>修改config文件夹下的index.js



1. 脚手架相关文件夹说明：（build、config两个文件夹在项目开发过程中尽可能的不要修改）

项目下Build：用于存放项目运行相关文件，包括webpack、主题异步加载配置、vue-loader基本配置、版本检查等文件

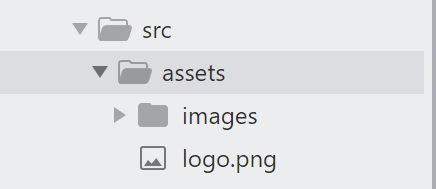


项目下Config：运行环境配置文件（开发环境：dev.env.js,测试环境test.env.js,生产环境prod.env.js）

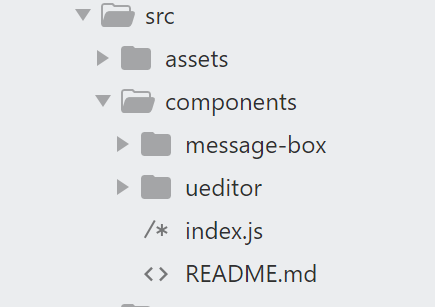


项目-Src下：

Assets：项目静态资源如图片、字体等



Componets：公共组件相关文件



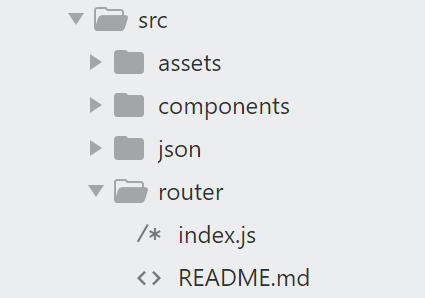
private-component：存放业务类组件



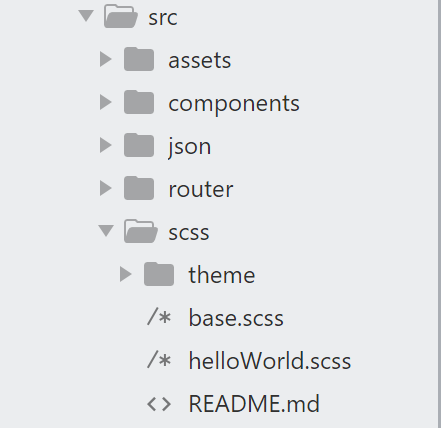
Json：存放本地json文件,可以处理一些枚举字段



Router：业务路由相关配置

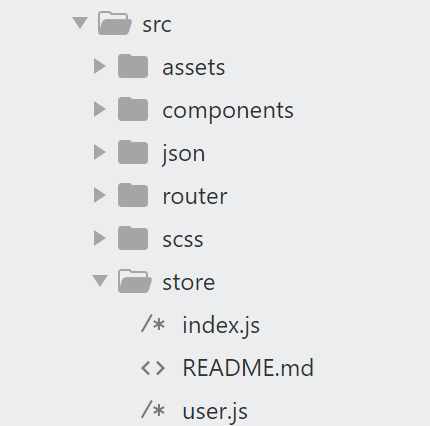


Scss：存放主题相关文件

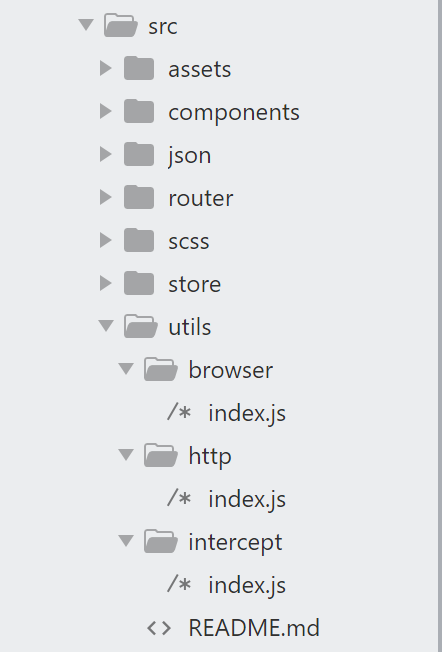


1. Scss/base.scss:用于处理全局的默认样式
2. Scss/theme文件夹下用于处理皮肤样式，主要定义按钮颜色，字体颜色，该主题下的通用背景图片，定位信息等。
3. Scss/mixin.scss:用于处理全局的函数定义
4. Scss/animation.scss:用于处理全局动画
5. 页面相关样式应写入对应vue文件的<style scoped lang="scss"></style>中。其中scoped属性表示此样式只应用在当前页面

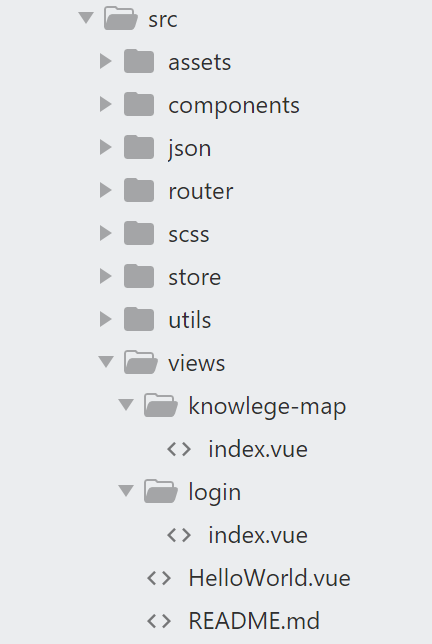
Store：用于存放全局变量、全局枚举类文件



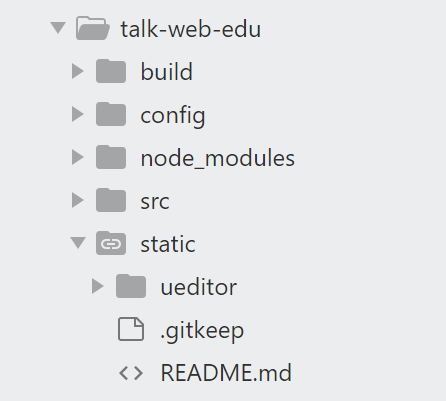
Utils：用于存放工具类如时间格式处理、字符串格式处理、请求拦截处理之类的脚本文件



Views：用于存放业务页面相关逻辑代码文件



项目下static：用于存放外部插件文件



### 四、数据存储

1、永久数据存储优先使用localstorage、sessionStorage，用于缓存数据。

Localstorage:用于永久存储数据，解决跨窗口数据存储读取问题。

SessionStorage:用于处理当前窗口的数据存储读取问题。

2、其次使用cookie，用于标识用户身份，可携带在http请求当中

3、全局变量及父子组件通信可使用vueX[由于vuex的存储是在内存当中，强刷时js会重新加载导 致数据丢失，此时我们可以根据业务需求结合localstorage和sessionStorage使用]

### 五、包、文件命名、ReadMe规范

包、项目结构：

1. 好的项目结构的准则就是快速定位，快速识别，尽可能保持扁平，不要写冗余／重复代码
2. 把所有项目代码放入src文件夹，为每个特性创建文件夹
3. 第三方库放在src外的一个文件夹
4. 每个组件、服务、管道、指令都独成一个文件
5. 当组件有多个附属文件时（htm，css，js , vue），建议创建一个独立的文件夹
6. 尽可能的保证文件的单一性，一个文件定义一样东西，比如一个组件，一个服务，一个指令

文件命名：

1、首页：index（为每个板块的入口页面）

2、新增/修改：set\_

3、报表：report\_

4、其他业务意义相关名

ReadMe：一个板块为一个文件夹，每个文件夹中需要新增readme文件用于描述板块的内容、各子板块对应的开发文件、涉及的组件及插件、以及访问路径，方便问题定位及修改

css主题： 将不同的样式写入theme的scss文件当中，相同的文件则写在当前开发页面处，指定scoped降低样式污染，组件样式写在当前文件夹中或者文件夹内用引入的方式处理。

### 六、注释编写规范

1、Js,css多行注释：

/\*\*

标题：功能概述

参数：变量对应的业务属性，参数类型

变量名：与业务属性相关的变量名需要对应描述

例如userName：用户名；age：年龄

业务逻辑：根据原型提供和分析之后的业务规则及算法

作者：XXX

邮箱：XXX@talkweb.com.cn

\*/

/\*

用于注释废弃代码，注明注释原因，例如业务需求变更从A变更至B,逻辑算法变更。Bug修复等等……

\*/

单行注释（css样式中请不要使用这种单行注释，容易导致环境运行编译错误）：

//描述注释下方的变量作用及代码逻辑

2、Html代码注释及注释废弃代码：

<!-- body之下对该页面进行业务描述 -->

3、配置文件注释:

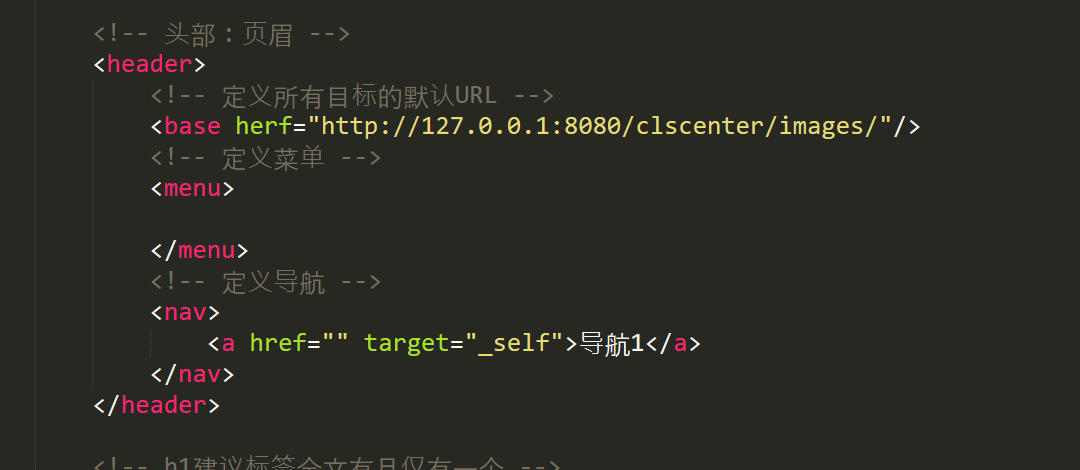
#port=8080

### 七、HTML编写规范

使用H5语义化标签进行开发。

1. H5标准化声明：<!DOCTYPE html>。
2. 标签内属性值使用双引号””。
3. 自定义属性使用data-为前缀。
4. 标签使用规范如下：

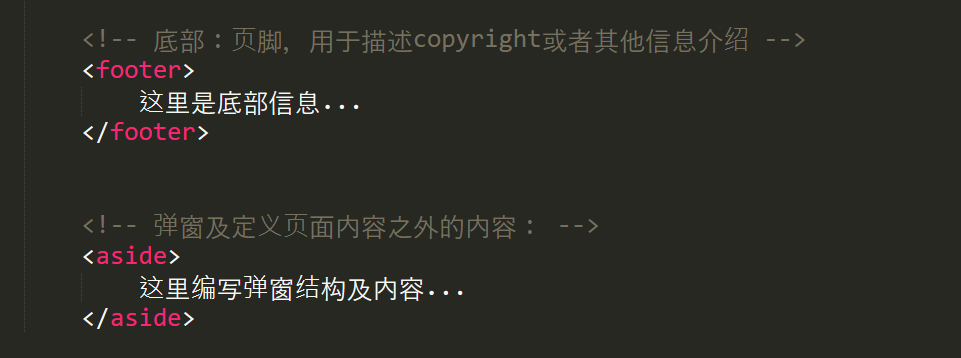
头部：



主体部分：（以下只列举了部分标签，其他标签请参照html5语义描述）



尾部：



### 八、CSS编写规范

CSS文件使用utf-8编码格式

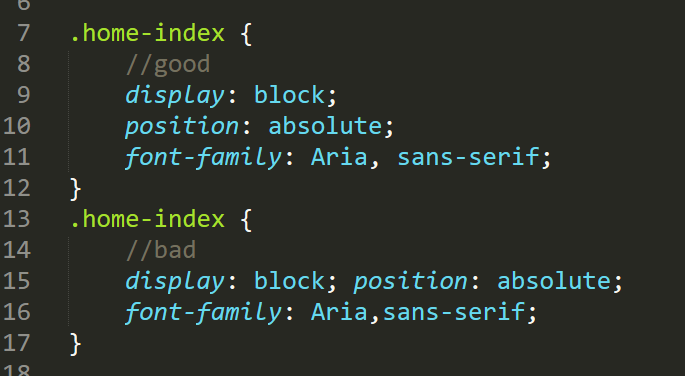
1. 样式命名规则：使用-号分隔两个单词之间。尽可能的描述清楚该样式的功能作用。



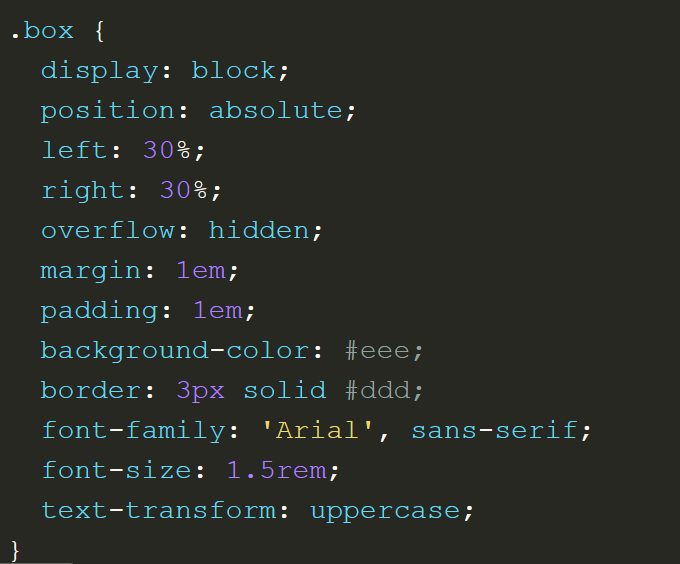
1. 缩进、空格与换行：

缩进使用TAB键，样式名与{之间保留一个空格，}结束标签应该换行。

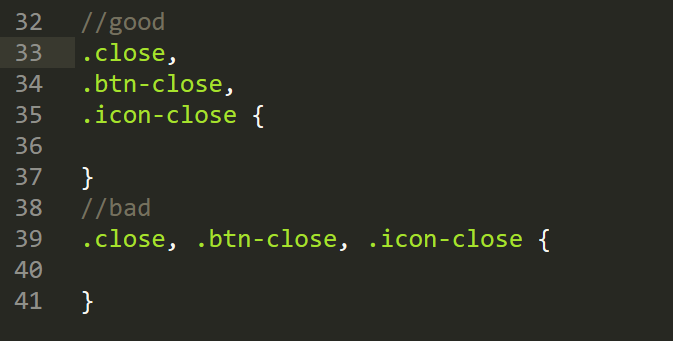
一行只书写单一属性及属性值，以;号结尾。属性名：（冒号）与属性值之间应保留一个空格，列表属性值有多个时，需要保留一个空格。>、+、~符号 两边各保留一个空格



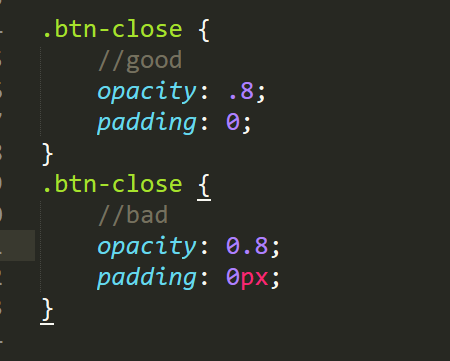
1. 优先遵循结构顺序：
2. display
3. position,left,top,right…
4. width,height
5. overflow,float,clear
6. margin,padding
7. background,border
8. font,text



1. 当使用了float属性值时应尽量避免声明display属性。
2. 当一个规则共享多个选择器时，声明的选择器应换行。



1. 数值，小于1的数值，省略整数部分0。长度为0时应省略单位。



1. 颜色值采用英文小写字母标识。例如：#f0ff00;可采用缩写数值：#fff；
2. background-position属性2d位置应给出2个方向的值。如果只给一个方向的值，另一个值会被解释为center;
3. 字体大小应不小于12px。再window以及苹果手机上小于12px会按照12px解析。
4. 在布局上尽可能使用样式来处理相关的信息，不要一整块都用背景图片
5. 不要使用行内标签包裹块级标签
6. 小图标应使用拼凑整图，使用background-postion和background-size处理单个小图标
7. font-family:应使用中文对应的英文值,不要使用中文名字。多个字体的顺序应该由西方字体到中文字体或者由效果最佳字体到效果不佳字体的顺序。

例如：微软雅黑=>Microsoft YaHei

### 九、图片处理规范

1、静态图片文件需要压缩，在不失真的情况下尽可能的压缩其大小。

压缩图片网址：<http://tinypng.com>（仅供参考）

2、减少图片的数量及使用频率，尽可能使用css处理相关的样式

### 十、前端JS规范

1. 变量声明

使用const或let来声明所有局部变量。如果变量不需要被重新赋值，默认应该使用const。应该拒绝使用关键字var。

|  |
| --- |
| // bad  var x =3.14;  var y = 100;  // good  const X = 3.14;  Let y = 100; |

1. 常量的命名

常量命名应该使用全大写格式，如果这个常量是一个函数，那么应该使用驼峰式命名法。

|  |
| --- |
| // bad  const number = 123;  const getnumber = () => { return number };  // good  const NUMBER = 123;  const getnumber = () => { return NUMBER }; |

1. 每次只声明一个变量

每一个变量声明都应该只对应一个变量。

|  |
| --- |
| // bad  Let a = 1, b =2, c = 3;  // good  Let a = 1;  Let b = 2;  Let c = 3; |

1. 使用单引号

只允许使用单引号包裹字符串，禁止使用双引号。如果字符串中包含单引号字符，应该使用模版字符串。

|  |
| --- |
| // bad  Let message = "This is a message. ";  Let saying = 'Say it ain\u0027t so. ';  // good  Let message = 'This is a message. ' ;  Let saying = ` Say it ain’t so.`; |

1. 不能省略分号

每个语句必须以分号结尾。不允许依赖于JS自动添加分号的功能。

|  |
| --- |
| // bad  Let message = "This is a message. "  Let saying = 'Say it ain\u0027t so. '  // good  Let message = 'This is a message. ' ;  Let saying = ` Say it ain’t so.`; |

1. 不推荐代码水平对齐

不推荐对代码进行水平对齐。即使之前的代码中做了水平对齐的处理，以后也应该避免这种行为。

|  |
| --- |
| // bad  {  number: 4,  longer: 435  }  // good  {  number: 4,  longer: 435  } |

1. 使用模板字符串取代连接字符串

在处理多行字符串时，模板字符串比复杂的拼接字符串要表现的更出色。

|  |
| --- |
| // bad  function sayHi(name) {   return 'How are you, ' + name + '?'; }  function sayHi(name) {   return ['How are you, ', name, '?'].join(); }  function sayHi(name) {   return `How are you, ${ name }?`; } // good  function sayHi(name) {   return `How are you, ${name}?`; } |

1. 不使用eval语句

除非是在code loader中，否则不用使用eval或是Function(...string)结构。这个功能具有潜在的危险性，并且在CSP环境中无法起作用。

|  |
| --- |
| // bad let obj = { a: 20, b: 30 }; let propName = getPropName();  // returns "a" or "b" eval( 'var result = obj.' + propName ); // good let obj = { a: 20, b: 30 }; let propName = getPropName();  // returns "a" or "b" let result = obj[ propName ];  //  obj[ "a" ] is the same as obj.a |

1. 不要使用续行符分割长字符串

在JS中，\也代表着续行符。尽管ES5中允许这么做，但如果在\后跟着某些结束空白符，这种行为会导致一些错误，而这些错误在审阅代码时很难注意到。

|  |
| --- |
| // bad const longString = 'This is a very long string that \     far exceeds the 80 column limit. It unfortunately \     contains long stretches of spaces due to how the \     continued lines are indented.'; // good const longString = 'This is a very long string that ' +      'far exceeds the 80 column limit. It does not contain ' +      'long stretches of spaces since the concatenated ' +     'strings are cleaner.'; |

1. 优先使用箭头函数

箭头函数提供了一种简洁的语法，并且避免了一些关于this指向的问题。相比较与function关键字，开发者应该优先使用箭头函数来声明函数，尤其是声明嵌套函数。

|  |
| --- |
| // bad [1, 2, 3].map(function (x) {   const y = x + 1;   return x \* y; }); // good [1, 2, 3].map((x) => {   const y = x + 1;   return x \* y; }); |

1. 尽可能用“===”取代“==”

“==“在比较的时候会自动转换数据类型，这种行为有时候会导致一些错误，而这些错误在审阅代码时很难注意到。

|  |
| --- |
| // bad if ('1' == 1) {  …  } // good if ('1' === 1) {  …  } |

1. 定义工具类脚本的规范：命名应加$符号与原生进行区分
2. 在src/utils/下新建文件夹，
3. 新建index.js编写内容，格式如下
4. /\*\*
5. 方法注释
6. \*/
7. export function fanctionName() {
8. ……
9. }
10. Src/main.js下新增
11. import { fanctionName } from './utils/xx/index'
12. Vue.prototype.$functionName = functionName;
13. 使用：this.$functionName();

### 十一、组件及组件说明规范

1. 组件命名用-分割，尽可能的让组件的名字贴近功能用途及区分样式。

例如：box-message, box-message-normal,box-message-theme1,box-toast

1. 每个组件文件当中应该备注readme文件注明该组件的作用、传入参数类型及业务内容、示例代码。
2. 组件的每个属性应注释其代表的含义，方法上方应注释该方法的作用、应用的业务场景。方法体中应注释逻辑思路及算法。
3. 组件内尽可能为单一功能，不要耦合数据，除枚举数据外，数据应根据业务需要从调用主体以参数的形式传入，降低耦合度。
4. 样式文件可直接写于文件内部，如有色值需求可声明传入参数。结构或样式差别大的条件判断复杂的，应另外封装组件。使用相近名进行同类区分。
5. 所有公共组件存放在src/components文件夹下。

所有业务组件存放在src/private-components文件夹下

前期使用复制粘贴的方法，后期可以搭建服务器，使用npm install xxx 的方式提供所有需要该组件的伙伴使用

### 十二、api文档

1. es6：<http://es6.ruanyifeng.com/#README>
2. vue2.0: <https://cn.vuejs.org/v2/api/>
3. sass: <https://www.sass.hk/docs/>
4. 字体图标font-awesome: <https://fontawesome.com/icons?d=gallery>