# 会议议程

* 1.整合目前移动端和pc的基本开发架构，基于nuxt.js的前后端同构应用
* 2.git工作流模式转变
* 3.编码规范,vue编程风格统一化
* 4.强调组件复用(基础组件、操作反馈、业务组件)
* 5.更换node包管理器，使用yarn

## nuxt.js框架总结

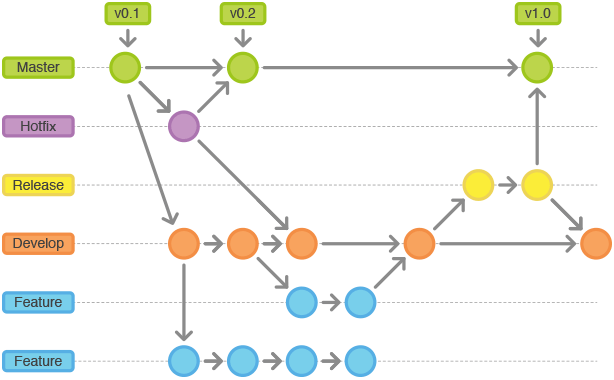
### 1.使用目的

为了解决目前web应用前后端分离带来的体验及seo优化问题

### 2.nuxt.js核心

* asyncData
* 路由
* 配置

## git工作流变更



git working flow

### 1.分支说明

* Master(用于正式线上环境,在每次合并到master分支后应把commit log合并,只保留每次版本迭代历史,并打上tag)
* Hotfix(用于线上bug修复使用的分支)
* Release(目前暂时作为测试环境分支使用,注意：开发过程如果需要频繁给到项目经理或者测试人员时,不要频率迭代release分支,在develop分支更新然后使用局域网地址)
* Develop(开发环境使用的分支,在完成每一个特性功能分支开发后需要合并到该分支)
* Featrue(功能性分支,实际上命名不要以Featrue1,Featrue2这样去命名，应该用Featrue-login,Featrue-personal去命名)

### 2.使用commit合并保持分支提交历史的简洁度

步骤:

1.git reset 指针

2.git add .

3.git commit

## vue编程风格统一

<script>  
 export default{  
 data(){  
 return {  
 number: Math.random()  
 }  
 },  
 created(){  
 this.network().getListA()  
 },  
 mounted(){},  
 methods: {  
 /\*\*  
 \*触摸，响应事件放在event下，如onclick函数,onchange函数等  
 \*\*/  
 event() {  
 let self = this  
 return {  
 onClick(){  
 alert(self.number)  
 }  
 }  
 },  
 /\*\*  
 \*网络访问相关函数  
 \*\*/  
 network(){  
 let self = this  
 return {  
 getListA(){  
  
 }  
 }  
 },  
 /\*\*  
 \* 一般放处理数组相关函数  
 \*\*/  
 handler(){  
 let self = this  
 return {  
 handleListA(list){  
 for(var i=0;i<list.length;i++){  
 //handle service  
 }  
 }  
 }  
 }  
 },  
 watch: {},  
 computed: {}  
 }  
</script>

## 异步函数尽量使用async,await特性

//before   
axios.get('/api').then(response=>{  
  
}).catch(error=>{  
  
})  
  
//now  
var response= await axios.get('/api')  
//如果需要异常处理,使用try catch块  
try{  
 var response=axios.get('/api')  
}  
catch (error){  
 console.log(error.message)  
}