###### 需求：

1. 重排界面，设置必输项、只读、隐藏；
2. 支持表单二开、脚本注入、扩展表单逻辑；
3. 支持界面按组织分配、实现千人千面；
4. UI配置信息可以平滑升级；
5. 代码规范；

###### 实现思路：

发请求获取配置好的数据，将配置好的js字符串转成对应的函数利用Vuex存储，在控件调用对应的方法时，同时触发Vuex内对应的函数对数据进行操作，触发双向绑定更新

###### 简介：

1. 界面UI（元数据）描述文件：

数据格式和NG3相似，这里只截取部分

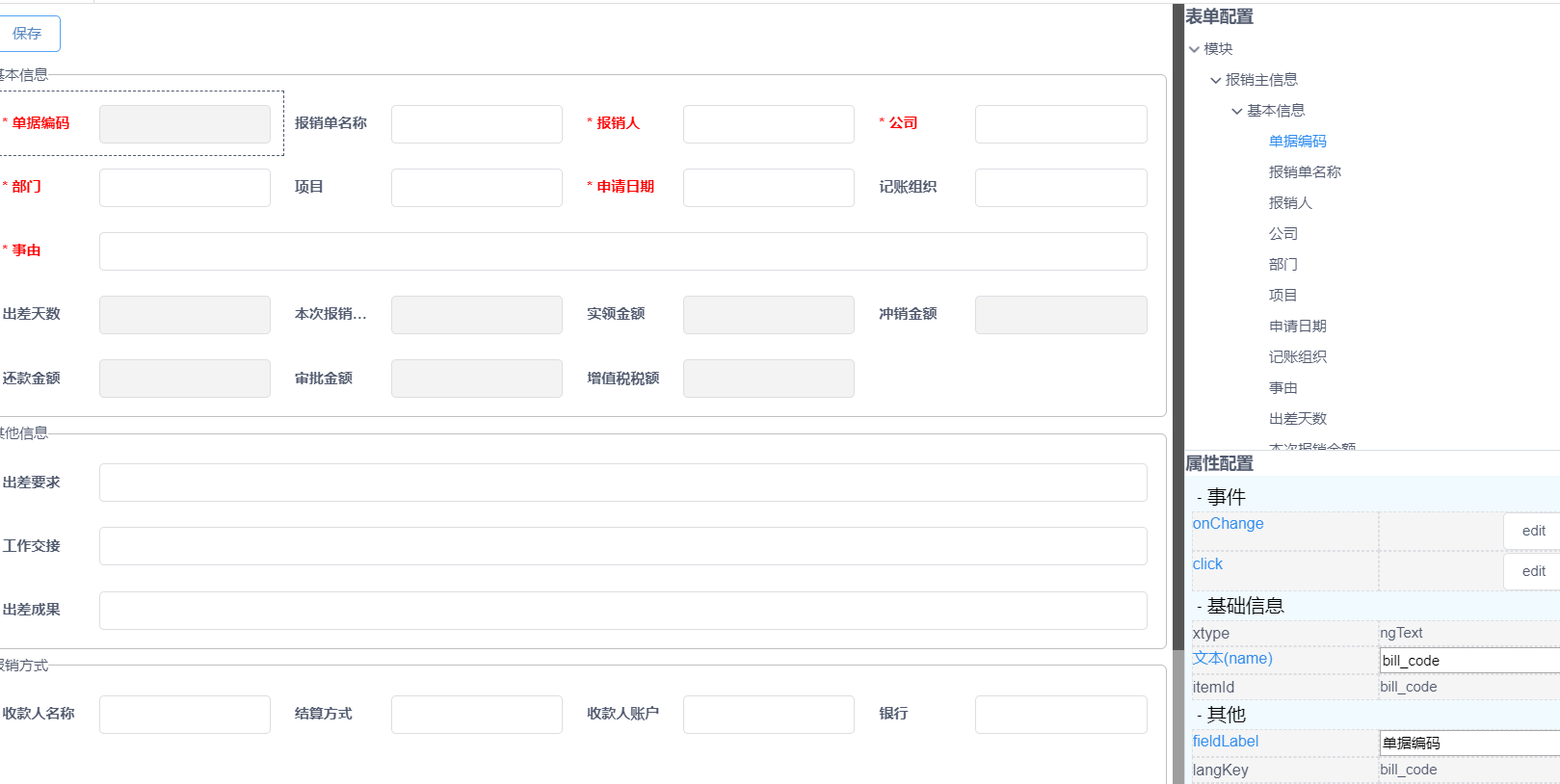


1. 自定义界面设计器

根据返回的界面UI数据，渲染出一个配置页面

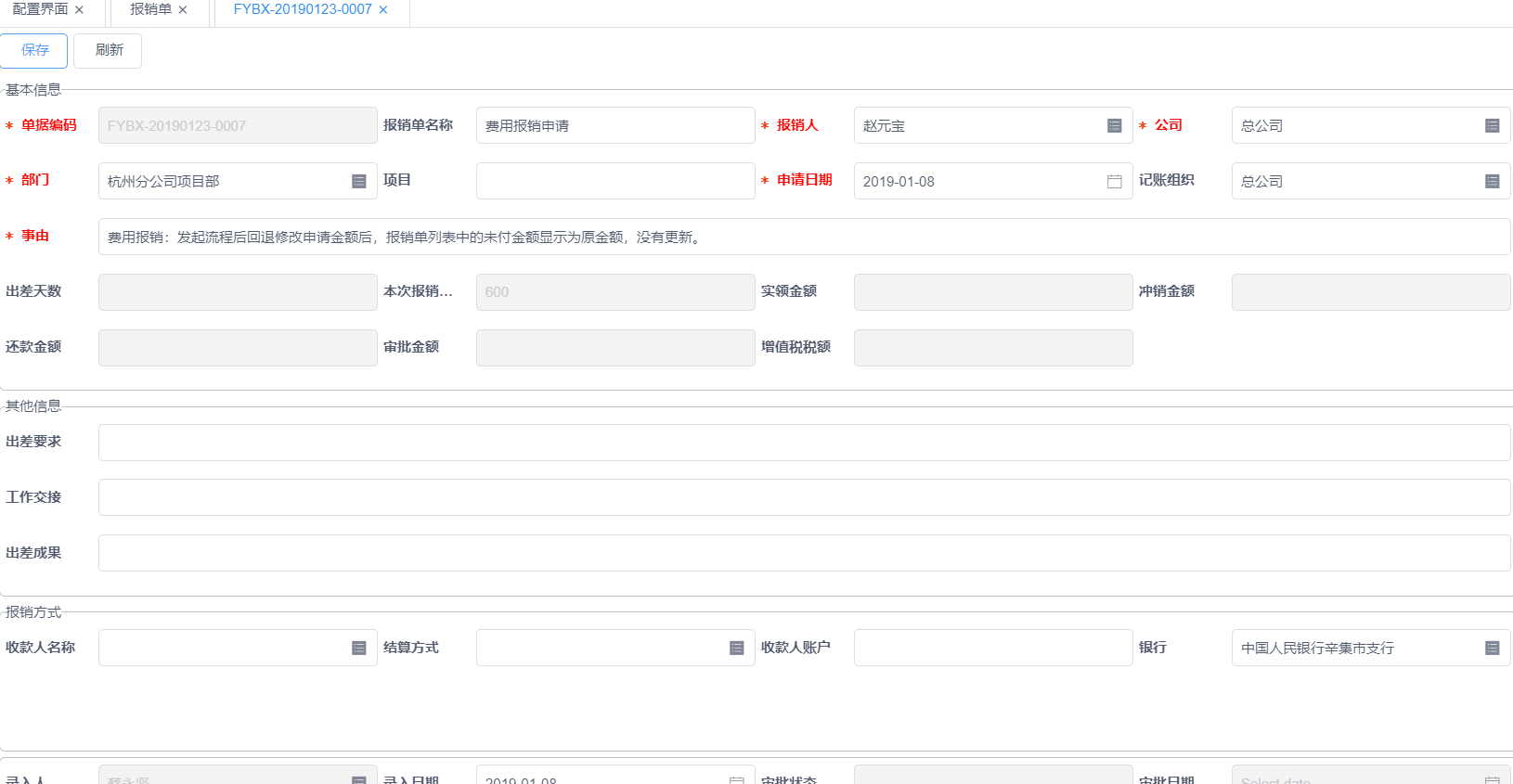
1、在自定义界面设计器中调整参数，拖拽等等操作，保存，重新打开页面获取最新配置

2、二开人员注入js代码；



1. 业务界面

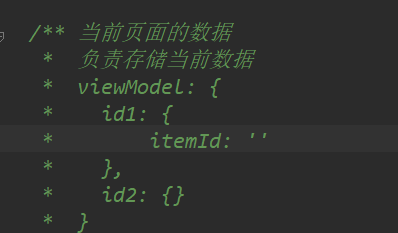
这是根据后端返回的自定义过的UI来渲染出来的页面



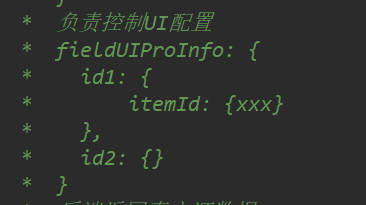
###### 研发人员使用方法

1. 业务点根组件的data添加数据项

1、viewModel:为当前业务单据的所有数据（为多层，与UI层次对应），id为（form或grid）容器id，itemId为表单项对应的itemId；

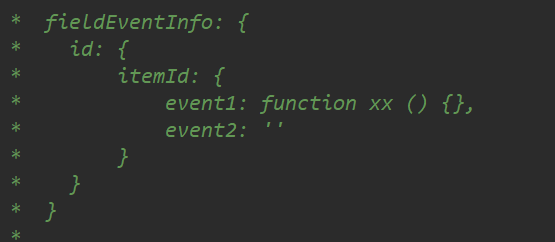


2、fieldUIProInfo:表单项UI对应的配置描述项,用于快速找到对应节点并修改UI的展示，结构和viewModel一样；



3、UIConfig:初始化后获取到的配置数据，用于存储渲染整个业务的配置数据，是viewModel和fieldUIProInfo的数据来源；

4、fieldEventInfo:研发人员编写的表单项上对应的事件集合，存放在单独的一个js内，结构需和viewModel一样，详情在编写业务代码模块会讲；

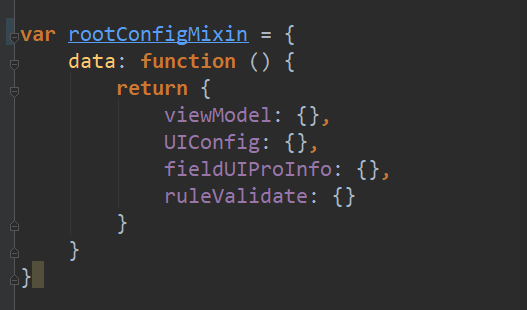


5、以上几个为关键项，研发人员只需要在data中添加fieldEventInfo即可



1. 初始化Data

将后续需要$set操作的数据用mixins插入data，下面部分是util封装的数据，把上面需要的几个关键点都放在里面。

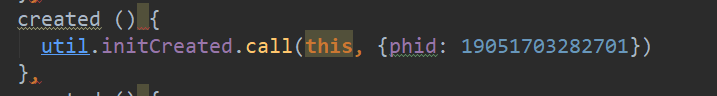


研发人员需要在data平级部分，加入mixins



1. 初始化UI和注入二开函数

1、在created（vue生命周期函数）里面调用一下封装好的初始化方法，传入请求UI数据需要的参数（根据bustype,获取ui，这里暂时先用phid），这里会把二开函数注入vuex和对viewModel基础数据进行双向绑定，后续再详细说明。



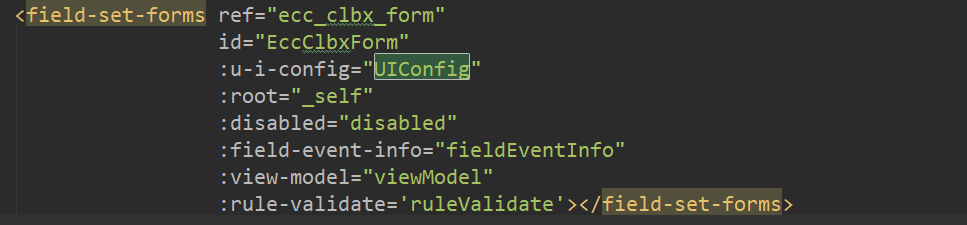
1. 绑定后台的值

1、在mounted钩子函数里面通过ajax获取数据并赋值，所有的数据都绑在viewModel里面，找到id对应的值进行赋值操作



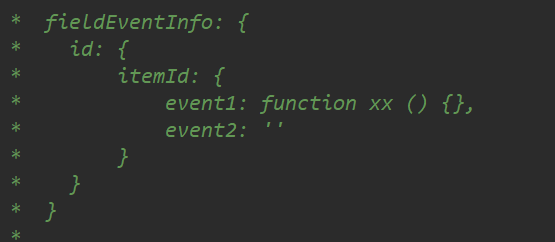
1. 编写页面模板

1、在模板里面加入对应需要使用的组件使用自己需要的组件,传入里面对应的id(模块里对应的id)、UIConfig、fieldEventInfo、viewModel、root(用于里面绑定fieldEventInfo事件的this指向最外层vm)。



1. 编写业务代码

1、编写fieldEventInfo，可以单独用一个js文件存放业务对应的函数，下面是fieldEventInfo的模型，函数执行时，this指向最外层vm，直接操作this.viewModel和this.fieldUIProInfo数据即可

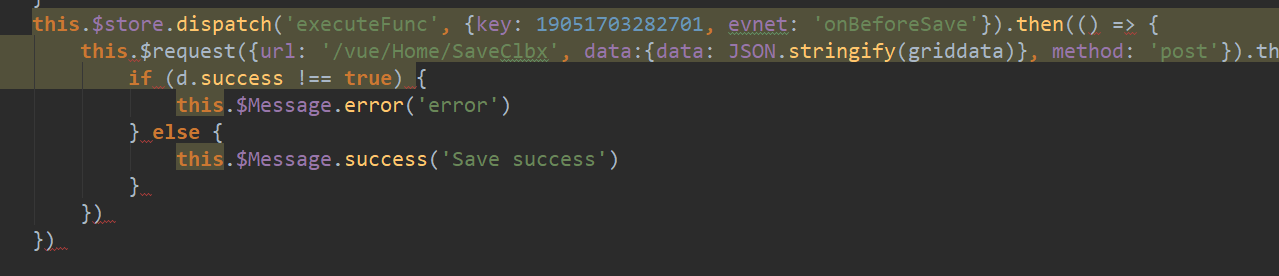


2、这里是报销主信息的一个小例子fieldEventInfo.EccClbxForm.phid\_emp代表报销人这个节点，.onHelpSelect代表该节点通用帮助回调事件。



1. 二开保存前校验

1、保存onSave之前，$store.dispatch(‘executeFunc’, {key: ‘’, event: ‘onBeforeSave’})通过vuex调用二开注册的校验函数在回调后进行保存，key为对应业务点的唯一标识（根据bustype,获取ui，这里暂时先用phid），



1. 注意事项

1、必须用mixin引入rootConfigMixin，否则在初始化UI配置内部$set时会报错。

2、必须在对应业务的最外层的组件内调用初始化initCreated()，这样才能保证二开函数的this指向最外层。

3、组件必须传入root，用来保证研发人员编写的函数在组件中执行时this指向最顶层的vm。

4、fieldEventInfo结构请按照要求编写，在组件内对应的函数是根据[id][itemId][event]来查找的。

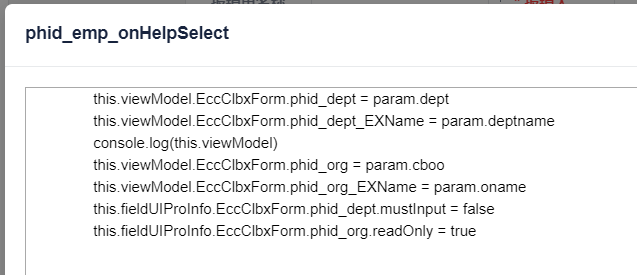
###### 二开人员使用方法

1. UI调整

1、在自定义界面设计器调整参数，拖拽等等操作，保存，重新打开页面获取最新配置

1. 编写二开脚本

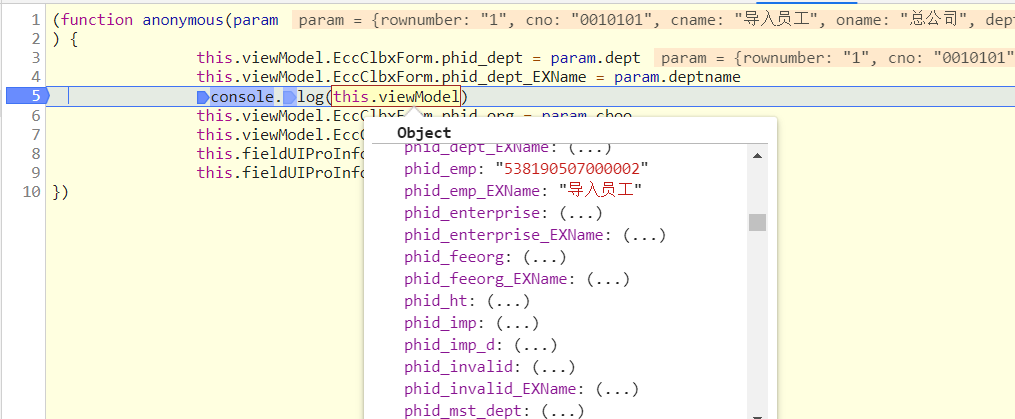
1、在配置页，找到对应节点，再右下里面找到对应事件点击edit，在弹窗内编辑，关于代码的编写在实际例子中再讨论。



1. 调试js代码

1、如上图，在代码中加入一个console.log,执行后在控制台可以找到一行输出，点最后那个找到对应代码，打断点进行调试。





1. 注意事项

1、接收到的参数固定命名为param

2、保存完后需重新打开对应页面获得新UI配置才有效果

3、注意校验前事件（onBeforeSave）需要return true或者false判断校验结果

###### 实例

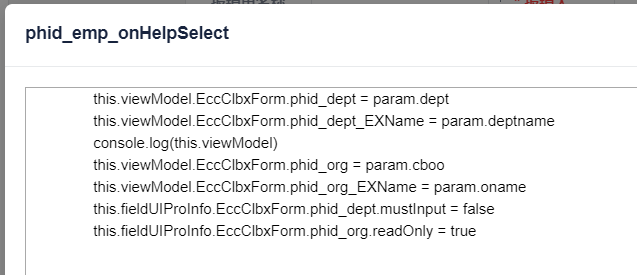
1. 简单说明

1、fieldEventInfo和二开执行的函数this我们会保证都指向顶层vm，只需要this.viewModel.id.itemId = xxx即可修改对应的数据，通过双向绑定触发更新看，this.fieldUIProInfo.id.itemId.xxx = xxx即可控制UI配置；把代码写js文件里和写二开里面差不多，仅仅是获取参数不同（二开获取参数为param），下面就把代码均放在二开里面保存进行演示。

2、记得二开里面参数用param，记得保存后要重新打开详情页获取新的UI配置

3、由于目前配置里面只用到了一个组件渲染，所以这边手动添加了一个组件作为例子，但是值用let tabs = this.$refs.tabs进行操作

1. 给兄弟组件赋值  
    这里我们给部门和公司赋值，事件在onHelpSelect的回调里





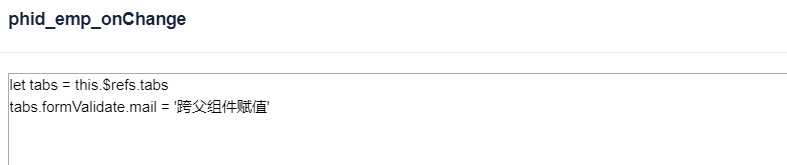
选择一条点确定



赋值成功是这样的，公司部门的值也被修改了



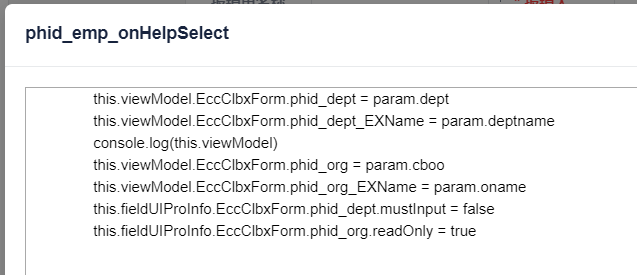
1. 跨父组件给其他组件赋值  
    这里我们就给那个写死的组件赋值，如果实际有的话只需要this.viewModel[另一个id]就好了，这里先用let tabs = this.$refs.tabs代替



这里把事件写在了通用帮助的onChange里面,触发事件后



1. 控制必填或者禁用  
    直接在之前那个通用帮助里面多加2行，fieldUIProInfo和viewModel一样，用id.itemId找到对应的位置，然后我们把部门的必填取消，把公司变成只读



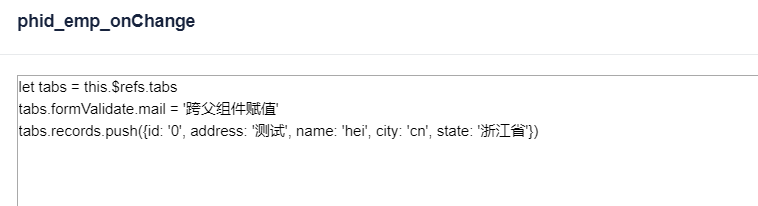
修改成功



1. 给列表插入新数据

配置grid的组件还在研发中，先用之前之前插入的那个列表代替一下。最后操作实际是一样的。

直接在报销人onChange里面触发了，给tabs里面对应的record数组push了一个值

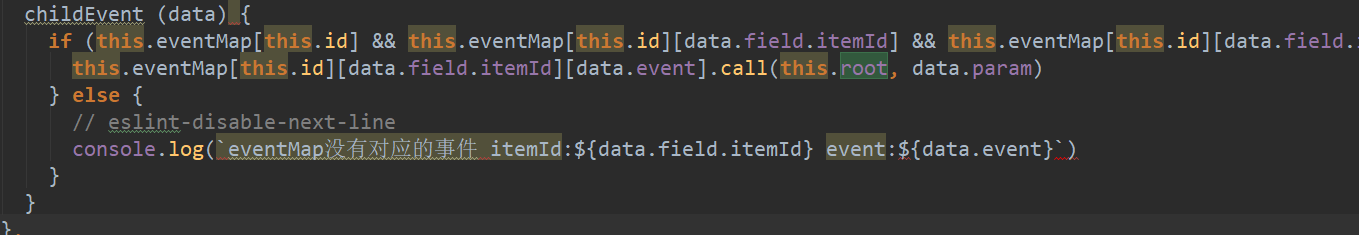


添加成功



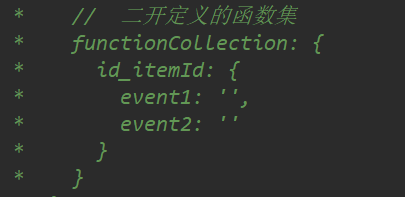
###### 实现原理

1. 简介
2. 利用mixin把我们需要的viewModel,fieldUIProInfo,UIConfig加入data内，
3. 使用$set设置双向绑定。
4. 利用new Function()将字符串转变为函数存入vuex，通过触发对应事件（后续$on会讲到），通知vuex，在vuex中找到唯一标识key，执行对应的函数。
5. 在上一步new Function()时，bind改变函数this指向，使this指向最外层vm，这样二开编写的函数内部可以用this.viewModel.id.itemId直接操作数据
6. 在组件内部监听child-event事件，根据事件参数，找到fieldEventInfo对应业务代码，并执行，使用.call使this指向最外层vm。



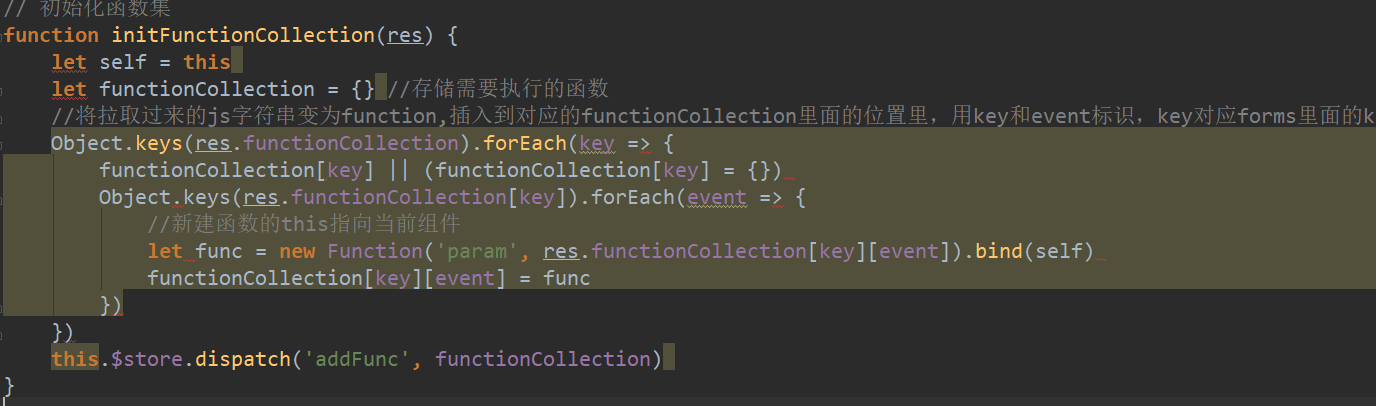
1. 在表单项使用$on绑定对应event类型（根据fieldEventInfo和functionCollection所包含的事件类型进行绑定），并在相应的时机使用$emit触发事件，被$emit触发的事件会做2件事，先通知vuex去触发二开执行的函数（promise），再抛出child-event事件通知组件执行fieldEventInfo内对应的业务代码。
2. functionCollection

functionCollection用于存放函数集合，在返回的数据中结构是这样



initCreated()的时候，遍历functionCollection，找到对应节点的事件，用new Function将存在event的js字符串转成函数，用bind使this指向该业务节点最外层vm，存储在functionCollection[key][event]内，最后将functionCollection存入vuex内。

比较关键的点就是要想一套规则，保证key是唯一的，这个key是在配置页面初始化的时候插入的funcId，只要保证配置页面的funcId唯一这边是不需要处理的。

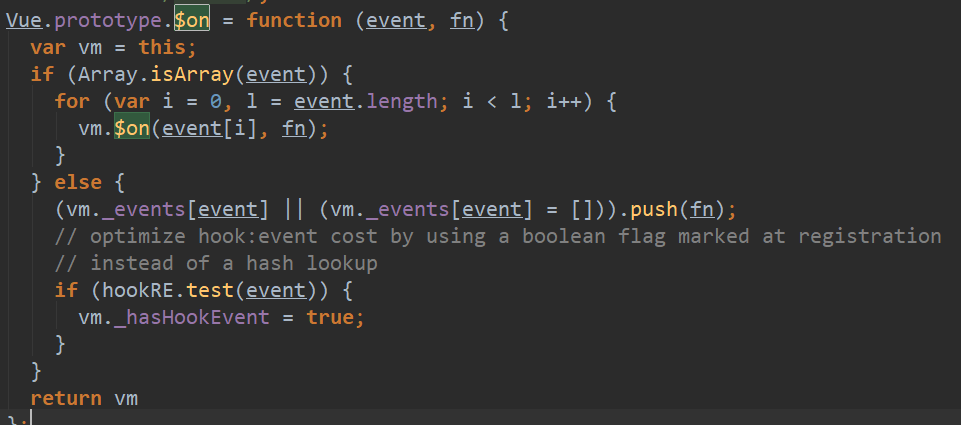


1. 关于this为什么要指向最外层节点

比如我们在子节点调用这个函数，此时如果要和同级节点进行联动的话，需要$parent.xxx.xxx；假如外层由tab包着，那$parent要一层层往上找，碰到一些复杂的组件可能需要$parent.$parent.xxx.xxx，用起来就很麻烦。如果我们在该业务节点的最外层进行初始化，this指向该业务最外层的vm，在最外层定义有2个特殊值，fieldUIProInfo控制UI渲染，viewModel控制数据渲染，在编写业务代码和二开代码时，可以通过这2个值找到任意一个你需要的修改的地方，就像之前实例里面用到的this.viewModel.EccClbxForm.phid\_dept直接找到了对应的部门那个数据，可以让代码看起来更直观。

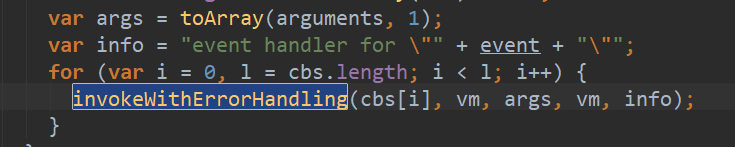
1. 事件绑定$on和如何通知vuex触发二开函数

Vue有一个用于父子组件通信的机制$emit和$on，$on可以将事件绑定在当前组件\_events里面，



通过$emit触发，一个事件可能绑定多个函数，所以他这里有个循环，我们后续只绑定一个

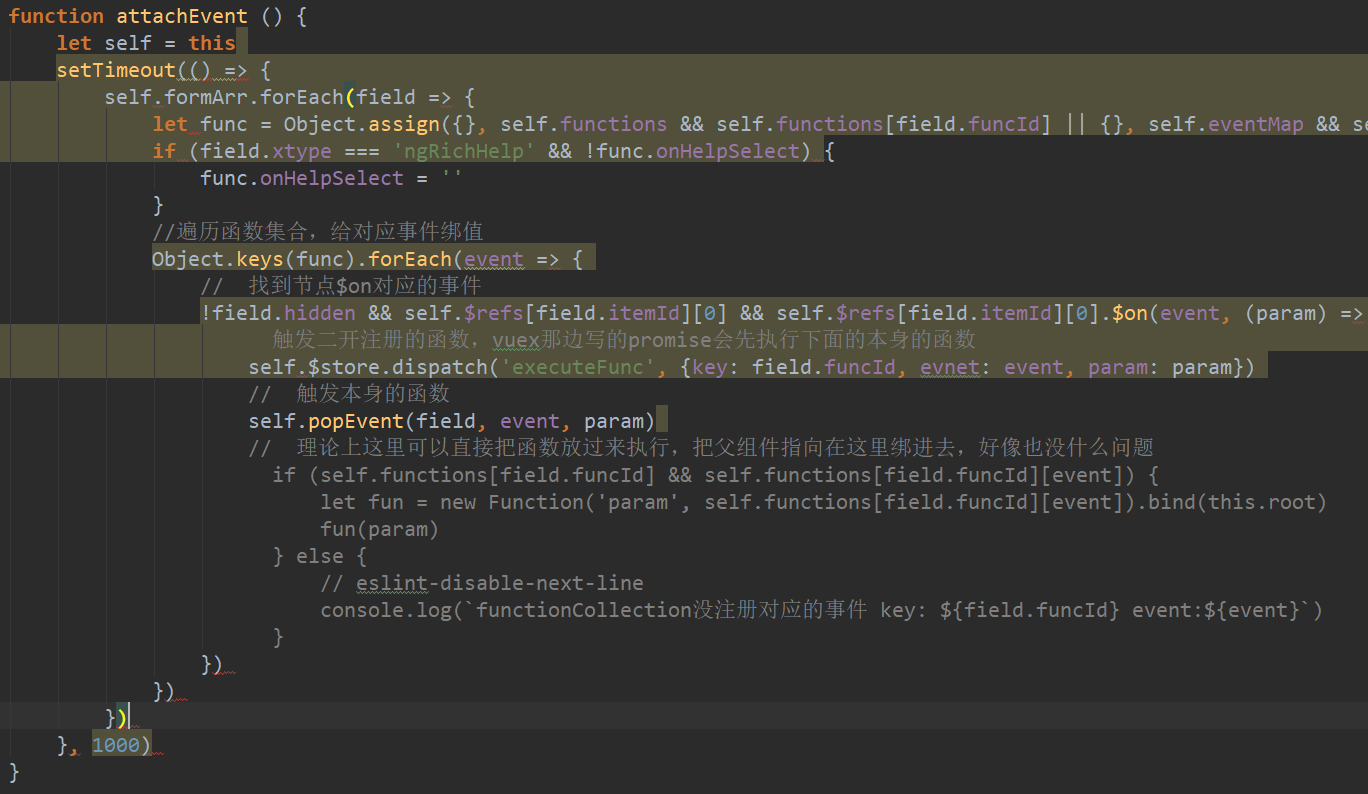


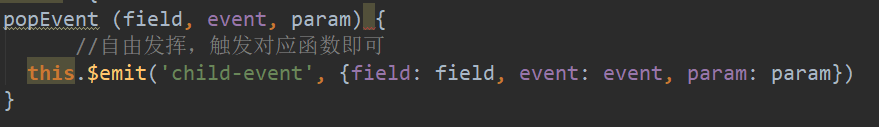


handler为对应cbs[i],context为vm，args为我们emit时传入的参数，详情可以自己研究一波源码

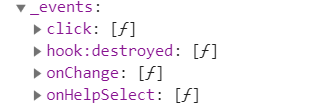


在组件内找到对应的节点，把prop传入的event类型进行绑定，然后在$emit触发对应事件时通知vuex执行二开函数并抛出child-event事件去触发fieldEventInfo对应事件，这样我们就不需要在各个节点手动去绑定那些事件。这里之前也有测试过把二开事件直接在这里执行，运行起来也没什么问题，后续可以考虑深入研究一波。



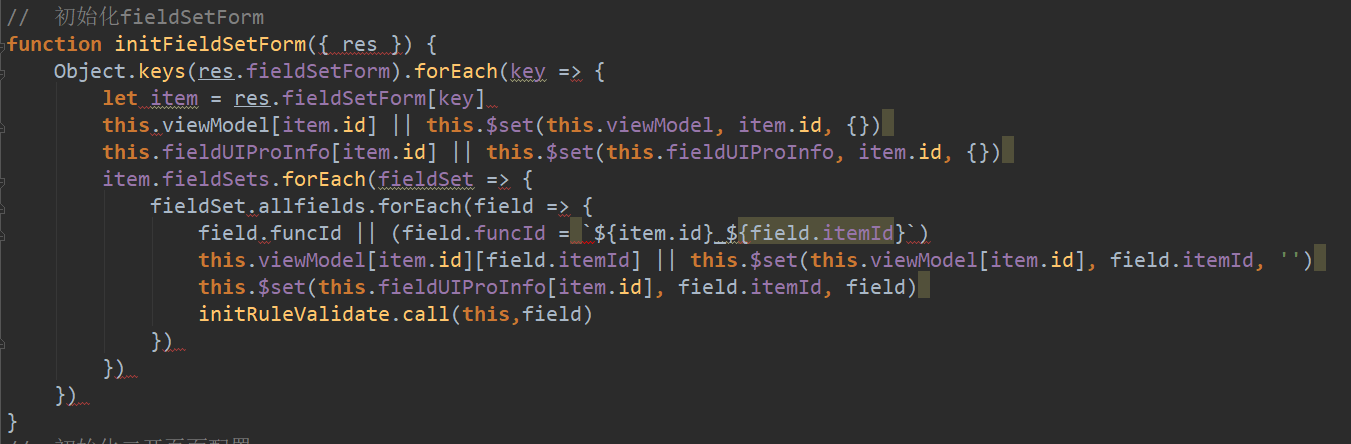


对应节点可以看到我们$on绑定上去的事件



这里获取节点的时候，部分节点还在渲染，导致可能在mounted下获取不到节点，其实update钩子后也存在部分拿不到的情况，所以在1s的延时之后再进行事件绑定，这1s为事件的挂载，不影响UI渲染，后续需要进一步研究。

1. ViewModel、fieldUIProInfo初始化
2. 根据UIConfig的数据，生成对应viewModel和fieldUIProInfo。
3. 表单项显示的值，我们通过props将viewModel的值分别传给相应的子组件，将该值绑定到显示的部分，通过改变viewModel的值即可达到更新页面值的效果
4. 表单项均为UIConfig内对应的配置渲染出来，在初始化时，我们将UIConfig对应单项的值存入fieldUIProInfo.id.itemId，即通过fieldUIProInfo.id.itemId就可以找到并直接修改在UIConfig内的那个值，从而触发页面UI显示的变化。



将该数据定义在最外层，使用this.viewModel.id.itemId = xx即可修改对应节点的值，this.fieldUIProInfo.id.itemId.readOnly = true即可修改页面显示配置，fieldUIProInfo还可以和viewModel配合起来最后做校验用；上面讲到二开的this指向最外层，此时在编写二开代码的时候也只需要this.viewModel、this.fieldUIProInfo即可对想要操作的值进行操作，一层一层往下点。