组件

以对接新许可证平台为例:

父组件为一个dialog, 当新选择的许可证与原来的许可证有模块减少时,显示子组件,来展示模块差异;

1.组件写法

原来的写法为:

```
<template>
  <!-- 导入许可证文件 -->
  <q-dialog>
    <q-form label-position="top">
      <q-form-item :label="$t('license.labels.importLicense')">
        <q-upload">
          <q-input v-model="file.name" :title="file.name" :readonly="true">
            <q-button>
              {{ $t('license.btns.selectFile') }}
            </q-button>
          </q-input>

          <p slot="tip" class="q-upload__tip">
            {{ $t('license.contents.fileAcceptTip') }}
            <span v-if="errorMsg" class="text-error">
              <br>
              {{ errorMsg }}
            </span>
          </p>
        </q-upload>
      </q-form-item>
    </q-form>
```

```
    <!-- 确认卸载 -->
    <q-dialog>
      <div class="license-tip-content">
        <i class="qax-icon-Circle-warning" />
        <div class="text-content">
          {{ $t('license.contents.tipContent') }}
        </div>
      </div>
      <div class="q-button_control">
        <q-button @click="foldAllCollapse">
          {{ $t('license.changedStatus.fold') }}
        </q-button>
        <q-button @click="unfoldAllCollapse">
          {{ $t('license.changedStatus.unfold') }}
        </q-button>
      </div>
      <!-- 展示模块变化 -->
      <license-modules-diff ref="diff" :client-modules-diff="productDiff"
class="license-modules-diff" />
      <span slot="footer" class="license-tip-footer">
        <q-button type="primary" @click.once="handelTipEnter">{{
$t('common.btns.confirm') }}</q-button>
        <q-button @click="tipDialogVisible = false">{{ $t('common.btns.cancel')
}}</q-button>
      </span>
    </q-dialog>
  </q-dialog>
</template>

import LicenseModulesDiff from './comparison.vue';
```

dialog为父组件,包裹了这个子组件, 而这个DIFF又作为了子组件的子组件; 形成三层嵌套;

> 1. 只把孙组件抽离出来, 逻辑很不对; 因为子组件和孙组件共同形成的一个dialogue; 如果要抽离, 应该从子组件开始抽离;
> 2. 其次, `import LicenseModulesDiff from './comparison.vue';`这一句, 用同义词表示同一个组件的情况,也要避免, 包括代码中, 不要又用diff又用comparison;

现在把子和孙组件进行抽离, 相当于新写一个组件了. 新写一个组件之前, 先要想好这个组件是一个`视图组件`还是一个`业务逻辑组件`, 如果是个视图组件,那么不应该在组件内处理数据,如果是个业务逻辑组件,那么就要考虑组件要做的处理. 这里的情况下,组件包含的信息还挺多的, 有提示语, 有展示栏目, 所以这里做成一个业务逻辑组件.

决定了写成什么类型组件之后, 考虑组件的职责--[如果模块出现减少,本组件将出现,请用户确认是否卸载哪些模块,并对模块的变化进行展示]; 所以本组件会做3件事:[1. 判断模块是否减少,决定自己的visible; 2. 获取模块变化数据,展示对比详情; 3. 点击确认后再uploadFIle, 点击取消时改变自己的visible]

接下来根据组件的职责, 决定大概的组件向内/向外暴露的方法和属性;

```
<!--
## props //从父组件的upload触发后获取到以下两个数据, 可通过refs传
- oldLicenseModel
- newLicenseModel

## data
- differences
```

```
- isShrinked: 是否发生收缩
- visible: 弹窗是否可见

## computed

## methods: {
- _togggleGlobalCollapse,//或者 foldAllCollapse, unfoldAllCollapse
- comparate(prevLicenseModel, nextLicenseModel) => modulesDifferences [{name,
changedStatus, modules:[name, changedStatus]}]
},

## events:
- close

-->
```

> 为什么已经有了isShrinked,还有用visible决定弹窗是否可见: isShrinked只表示新旧模块之间是否有收缩, 弹窗关闭时, 如果把isShrinked改为false, 与其含义对不上

从父子组件的功能, 决定父组件需要做哪些操作来控制子组件:

> 1. 获取oldLicenseModel和newLicenseModel
> 2. 父组件作为一个共用组件, 再不弹出提示框时也有uploadFile的功能, 所以通过大致过程如下:
> if(需要弹子组件){调用子组件方法}else{uploadFile}

```
async handleSubmit({file}){ //handleSubmit为upload组件的上传方法
    try{
        oldLicenseModel = await getOldLicenseModel();
        newLicenseModel = await getNewLicenseModel();
        //判断模块收缩
        await this.$refs.子组件.compareLicense(oldLicenseModel, newLicenseModel)
        //没收缩则直接上传
        if(isShrinked){
            await 关闭子组件
        }else {
            上传
        }
    } catch (err) {
        弹tip;
        throw err;
    }
}
```

> 跟写组件一样,写函数之前也要想好该函数实现什么功能,有无副作用(改变了某个变量);
>
> 子组件内的compareLicense函数,是一个有"副作用"的函数, 在里面做对比时候, 能够决定isShrinked, 并且同时通过这次对比获得对比的结果, 正好给后续展示;

```
//子组件内:
compareLicense(oldLModel, newLModel) {
    return { productDifference, isShrinked };
}
showComparator() {
    this.tipDialogVisible = this.isShrinked;
}
```

组件开发流程

奥卡姆剃刀原理：**如无必要，勿增实体**

如何实现一个翻页器？

明确需求

- 不清楚需求要不要实现，那就不实现

- 改变每页的 limit
- 跳转功能
- 总页数

设计

- 模型设计：UML
- 接口设计：TypeScript

Preview

目的：确保设计合理且符合需求，避免大方向错误