最近在做新项目，个人网站已经暂停了，新项目目前为止一切还算正常，不过昨天遇到了个技术性的小问题，最后靠着个人对代码的理解能力解决了，毕竟是一件相当令人激动的故事，还是打算复盘出来给大伙瞅瞅。

前排提醒：由于本人完全没有Vue的项目经验，第一次遇到这种问题，网上也没查到成熟的解决方案，如果有dalao认为我这样的处理方案不专业，您有更专业的处理方案，欢迎在评论区留下您的宝贵意见。如果对本篇的内容有任何疑问，也欢迎提问，在我能力范围内的我会努力为大家解答疑惑。

故事的起因是一个组件，先放出组件的设计图

图片包含 徽标

描述已自动生成

这个组件其实没有什么难度，我用的技术还是Vue3。问题出在图标上，最开始我用的img标签来导入图标，然后三个Entry按钮用的当然是v-for做layout，三个按钮的图标和文字都不一样，文字可以动态绑定没什么可说，图标用img标签的话，把src做一个v-bind动态绑定也没什么问题，下面是源代码。

我们全篇要用到3个文件:

Resource/index.js – 资源仓库

NavAvatorPanel.vue – 当前组件源文件

Icons.vue – 通用图标组件

>Resource/index.js – 资源仓库（部分）

import panel\_space from './Icons/PanelSpace.svg';

import panel\_setting from './Icons/PanelSetting.svg ';

import panel\_outlog from './Icons/PanelOutlog.svg';

import right\_arrow from './Icons/RightArrow.svg';

//仓库本体

const resources = {

avator\_panel\_options:[

{

icon : panel\_space,

text:"个人中心"

},

{

icon : panel\_setting,

text:"系统设置"

},

{

icon : panel\_outlog,

text:"退出登录"

}

],

right\_arrow

}

export default resources;

>NavAvatorPanel.vue – 组件源文件（部分）

<script setup>

const {

right\_arrow,

avator\_panel\_options

} = inject('resource');

//这里先要在APP.vue里import进来然后provide出去，详见我资源管理那篇文章

</script>

<template>

<!--此处省略上半部分的代码-->

<!-- 下半部分 -->

<div v-for="item in avator\_panel\_options" class="options">

<img :src="item.icon" alt="" class=”icon”>

<div class="text">{{ item.text }}</div>

<img :src="right\_arrow" alt="" class=”arrow”>

</div>

</template>

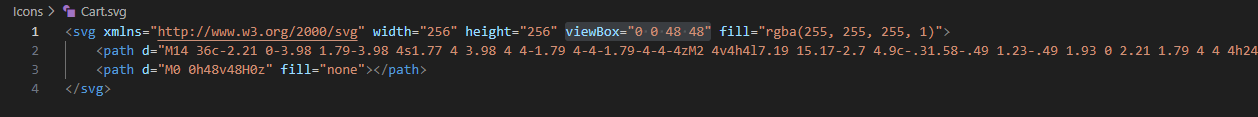
看到这里大家也应该发现了，虽然我们用的img标签，但我们使用的是svg图像。什么是svg图像呢？svg是一种矢量图像文件，内部结构就是html标签，对svg使用img标签加载，实在是太不专业了。对于svg标签的使用，Vue.js有一篇官方文档供我们参考：<https://v2.cn.vuejs.org/v2/cookbook/editable-svg-icons.html。虽然感觉把svg文件全变成.vue>文件可能也不是最优解，但我也想不出有什么更好的方案了，那我们就按照文档里的操作一下。

首先把我们的图标把svg标签拆掉全变成.vue文件：

原文件

电脑萤幕画面

描述已自动生成



修改后

图片包含 文本

描述已自动生成

屏幕上有字

描述已自动生成

然后我们新建一个Icon的模板>Icons.vue

>Icons.vue – 通用图标组件

<script setup>

const props = defineProps({

width: {

type: [Number, String],

default: 18

},

height: {

type: [Number, String],

default: 18

},

color: {

type: String,

default: '#ffffff'

}

})

</script>

<template>

<svg :width="width" :height="height" viewBox="0 0 48 48" :fill="color">

<slot>

<!--这里的内容是一个默认的svg图标内容-->

<g>

<path d="M0 0h48v48H0z" fill="none"></path>

<path d="M24 4C12.95 4 4 12.95 4 24s8.95 20 20 20 20-8.95 20-20S35.05 4 24 4zm0 36c-8.84 0-16-7.16-16-16 0-3.7 1.27-7.09 3.37-9.8L33.8 36.63C31.09 38.73 27.7 40 24 40zm12.63-6.2L14.2 11.37C16.91 9.27 20.3 8 24 8c8.84 0 16 7.16 16 16 0 3.7-1.27 7.09-3.37 9.8z"></path>

</g>

</slot>

</svg>

</template>

这里要用到一个知识叫插槽，就是代码里的slot标签，这部分知识我也直接贴出官方文档：<https://cn.vuejs.org/guide/components/slots.html>。简单来说插槽就是用来服务于组件式开发的，在一个通用模板中先放置几个插槽，然后使用者在使用这个通用组件时可以根据不同的需求在插槽中放入不同的内容。插槽一共分两步，设置插槽和使用插槽。

>item.vue – 设置插槽的组件

<script setup>

//这里设置了可以通过<item 参数>传进来的值

const props = defineProps({

//这里的text就可以作为一个参数传进来：<item text=”HelloWorld”>

text: {

//type指定了text的类型

type: String,

//default指定了不传参时text的默认值

default: “Hello”

}

})

</script>

<template>

<div>

{{text}}

<!--这里是插槽本体-->

<slot>

<!—插槽里面设置的是不使用插槽时的默认内容-->

<div />

</slot>

</div>

</template>

>main.vue – 使用插槽的组件

<item text=”HelloWorld”>

<!--在组件内部放入标签就是在使用插槽-->

<span />

<item />

当然插槽作为Vue中最重要的功能之一还有很多高级应用，这里就不展开了。在创建好这个组件后我们就可以开始使用了。先把资源仓库导入的.svg文件换成新转换的.vue文件。然后用import导入这个图标组件Icon.vue。最后把img标签换成Icon：

>Resource/index.js – 资源仓库（部分）

import right\_arrow from './Icons/RightArrow.vue';

>NavAvatorPanel.vue – 组件源文件（部分）

const {

right\_arrow,

} = inject('resource');

<Icons class="arrow"><right\_arrow /></Icons>

目前为止一切顺利，right\_arrow.vue里的内容可以随资源仓库导入然后inject传入作为template使用，在此就能体现出Vue的方便之处了。

图形用户界面, 文本

描述已自动生成

但还记得我们下半部分还有一个要从item里提取的图标吗

<div v-for="item in avator\_panel\_options" class="options">

<img :src="item.icon" alt="" class=”icon”>

</div>

template无法通过这种方式使用，我们可以试一试

<Icons class="icon">< item.icon /></Icons>

结果可想而知

图形用户界面

描述已自动生成

作为一个template，提示的颜色却是变量的颜色，就已经可以发现不对了，这样操作的结果如下

文本, 信件

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

这种错误的使用方式往往会产生十分滑稽的效果。在此之后我把我能想到的所有方法的试了一遍，但一无所获，网上查也查不到合适的教程，整个项目一度陷入僵局。这里顺便再给大家分享一个同样滑稽的v-html和{{}}动态绑定使用例:

<Icons class="icon" v-html= “item.icon”></Icons>

图表

中度可信度描述已自动生成

<Icons class="icon">{{ item.icon }}</Icons>

文本

描述已自动生成

在这之后我就去打游戏了，打了一会游戏又开始研究这个问题，此时一篇文章给了我灵感，虽然具体的文章内容已经找不到了。我们把思路逆转过来，之所以上面的方法都无法使用，涉及到Vue中template的原理，这里不详述，因为太底层的我也不知道，但我们可以浅浅的介绍一下，就是只有再script里定义的内容，才可以作为template使用，无论经过了多少次传递也没关系，只要是在script里定义且满足以下两个条件：1.不在数组或对象内2.Object类型变量，就可以作为template使用，只有作为template使用才能解析里面的内容并嵌入代码中。

那我们就要抛弃这个item.icon，创造出这样的环境，那么就只剩一个问题了，在哪声明这样的变量，首先肯定不能在当前组件的script里声明，因为我们必须要用到的item的声明周期仅限于当前的v-for内部，那么就只剩下一个地方了，就是Icon.vue的script里。答案出来了，我们就可以开始操作了。首先要将Icon.vue大改

>Icons.vue – 通用图标组件

<script setup>

//此处将原来slot标签的默认内容写进另一个文件里导入，模拟实际使用时的环境

import None from './Icons/None.vue';

const props = defineProps({

width: {

type: [Number, String],

default: 18

},

height: {

type: [Number, String],

default: 18

},

color: {

type: String,

default: '#ffffff'

},

//这个新加的IsInList便是用来处理要用的template在数组或对象内部需要多层调用的情况

IsInList: {

type: Object,

//默认指向的是最顶上导入进来的图标

default:None

}

})

</script>

<template>

<svg :width="width" :height="height" viewBox="0 0 48 48" :fill="color">

<!--此处将slot标签里的内容替换为template，模拟实际使用时的环境-->

<slot><IsInList /></slot>

</svg>

</template>

这样在遇到要用的template在数组或对象内部需要多层调用的情况我们就可以这样使用这个组件了

<Icons class="icon" :IsInList = "item.icon"></Icons>

注：由于我的配色都存在CSS变量里，所以组件的属性我就全统一在CSS里设置了。

最后放出实现结果（整体设计就是纯纯抄的B站）

