Vue card组件开发

在界面开发中，我们会看见很多的card，也就是图文效果，这些效果有很多种，有楼层的card有商品列表的card还有日历上使用的card等等，这些card种类繁多，那我们将如何使用呢？这里我们来研究一下接下来该如何实现这些card。

首先我们现在看一下这些card最主要的特点是图文混合的特效比较多，而且显示的内容上基本上是相近的，因此我们可以把这些card统一进行管理，那该如何进行分离和使用呢。这就需要我们有一个很好的分配机制的公共处理程序，这个公共处理程序是方便对组件进行分发处理的，那我们新建一个dmCard文件夹，在这个文件夹中我们做这样的处理，新建一个index.vue文件和一个over.vue文件，index.vue文件中的代码如下：

**HTML代码：**

<template>

<div class="dm-card">

<overCard :param="param" :data="data" v-if="param && param.type =='over' " />

</div>

</template>

**JavaScript脚本代码：**

<script>

import overCard from "./overCard";

export default {

data() {

return {

};

},

props: ["param","data"],

mounted() {},

components: {

overCard

}

};

</script>

**CSS样式代码：**

<style lang="less">

.dm-card {

width: 100%;

height: 100%;

display: inline-block;

}

</style>

在这段代码中是涉及到分发机制的，比如我在调研dmCard时，给其参数param 传递了一个类型，组件会根据这个类型去选取合适的card组件进行显示。我们在开发组件时，针对于现在不同的card我们是分文件进行开发，通过分发机制调取，这样组件之间相互影响比较小而且易于维护后面的代码。其中我们新建的over.vue就是一个子组件，具体的over.vue文件中的代码如下：

<template>

<div ref='overDom' :class="'over-card '+extendType">

<div class="card-box">

<a :href="'/#/ticketDesc?id='+id" target="\_blank">

<div class="box-img over-img">

<div class="wrap">

<span class="ribbon6">TOP1</span>

</div>

<img :src="imgUrl" alt="" />

</div>

<div class="box-content over-content">

<div ref='overDomDesc' name='boxDesc' class="box-desc">

<p>{{name}}</p>

<p>{{startDate}}</p>

</div>

<a class="button" target="\_blank" :href="'/#/ticketDesc?id='+id" v-if="extendType!='hot'" href="/#/multiplePage?page=4">立即查看</a>

</div>

</a>

</div>

</div>

</template>

<script>

export default {

data() {

return {

extendType: (() => {

return this.param.extendType != undefined && this.param.extendType

? this.param.extendType.join(" ")

: " ";

})(),

imgUrl: this.getByKey("imgUrl"),

name: this.getByKey("itemName"),

startDate: this.getByKey("startDate"),

endDate: this.getByKey("endDate"),

id: this.getByKey("id"),

minPrice: this.getByKey("minPrice"),

address: this.getByKey("address")

};

},

mounted() {

this.init();

},

methods: {

init() {

// 根据节点的高度动态设置界面显示效果，

let node = this.$refs.overDom;

let cardHeight = node.offsetHeight;

if (this.$refs.overDom.offsetHeight > 500) {

this.$refs.overDomDesc.style.marginTop = "200px";

} else if (this.extendType && this.extendType == "calendar") {

this.$refs.overDomDesc.style.marginTop = "50px";

} else if (this.extendType && this.extendType != "hot") {

this.$refs.overDomDesc.style.marginTop = "100px";

}

},// 获取data对象中对应的数据值

getByKey(key) {

return this.data != undefined && this.data[key] ? this.data[key] : "";

}

},

props: ["param", "data"]

};

</script>

<style lang="less" scoped>

.over-card {

overflow: hidden;

width: 100%;

height: 100%;

.card-box {

width: 100%;

height: 100%;

background: #eee;

position: relative;

perspective: 1000px;

.box-img,

.box-img img,

.box-content {

width: 100%;

height: 100%;

position: absolute;

left: 0;

bottom: 0;

// background-color: aqua;

transition: all 0.3s ease-in-out;

}

.box-img {

img {

position: absolute;

top: 0;

}

}

.box-content {

width: 100%;

height: 100%;

text-align: center;

background-color: rgba(255, 60, 27, 0.64);

color: #fff;

font-size: 14px;

.box-desc {

width: 130px;

height: 90px;

border: 1px solid #eee;

margin: 0 auto;

vertical-align: middle;

padding: 5px;

}

a {

color: #fff;

width: 130px;

height: 25px;

display: block;

line-height: 25px;

margin: 3px auto;

border: 1px solid #eee;

}

}

.over-img {

opacity: 1;

height: 100%;

}

.over-content {

opacity: 0;

height: 0;

overflow: hidden;

}

}

.card-box:hover .over-content {

opacity: 1;

height: 100%;

}

}

.hot {

.box-img {

position: relative;

.wrap {

width: 52px;

height: 52px;

position: absolute;

overflow: hidden;

display: inline-block;

}

.ribbon6 {

display: inline-block;

text-align: center;

width: 75px;

height: 20px;

// line-height: 40px;

color: #ffffff;

position: absolute;

top: 10px;

left: -20px;

z-index: 2;

overflow: hidden;

transform: rotate(-45deg);

-ms-transform: rotate(-45deg);

-moz-transform: rotate(-45deg);

-webkit-transform: rotate(-45deg);

-o-transform: rotate(-45deg);

border: 1px dashed;

// box-shadow: 0 0 0 3px #57dd43, 0px 21px 5px -18px rgba(0, 0, 0, 0.6);

background: #e20c29;

}

}

.card-box {

.box-content {

opacity: 1;

height: 50px;

.box-desc {

font-size: 12px;

width: 100%;

height: 90px;

border: 0;

margin: 0 auto;

vertical-align: middle;

padding: 5px;

}

}

}

.card-box:hover .over-content {

height: 50px;

}

}

.calendar {

padding: 5px;

.card-box {

.box-content {

.box-desc {

font-size: 12px;

width: 100%;

height: 90px;

border: 0;

margin: 0 auto;

vertical-align: middle;

padding: 5px;

}

.button {

display: none;

}

}

}

}

</style>

这个card默认显示效果是一张图，当鼠标选中之后，显示效果是对应的文字具体效果如下：



正常显示 鼠标悬浮效果

这个效果中有些属性是自定义的，需要通过父节点将param传递至当前组件over.vue这个文件中，通过参数改变这个组件的具体样式效果。

页面展示的图片和文字是通过传递的Data这个属性进行数据传递，子页面需要针对性的去获取出对应的数据信息。这样页面显示的效果就正常了。在JavaScript中getByKey函数是用来获取到对应的参数数据，并将其展示到组件中的。