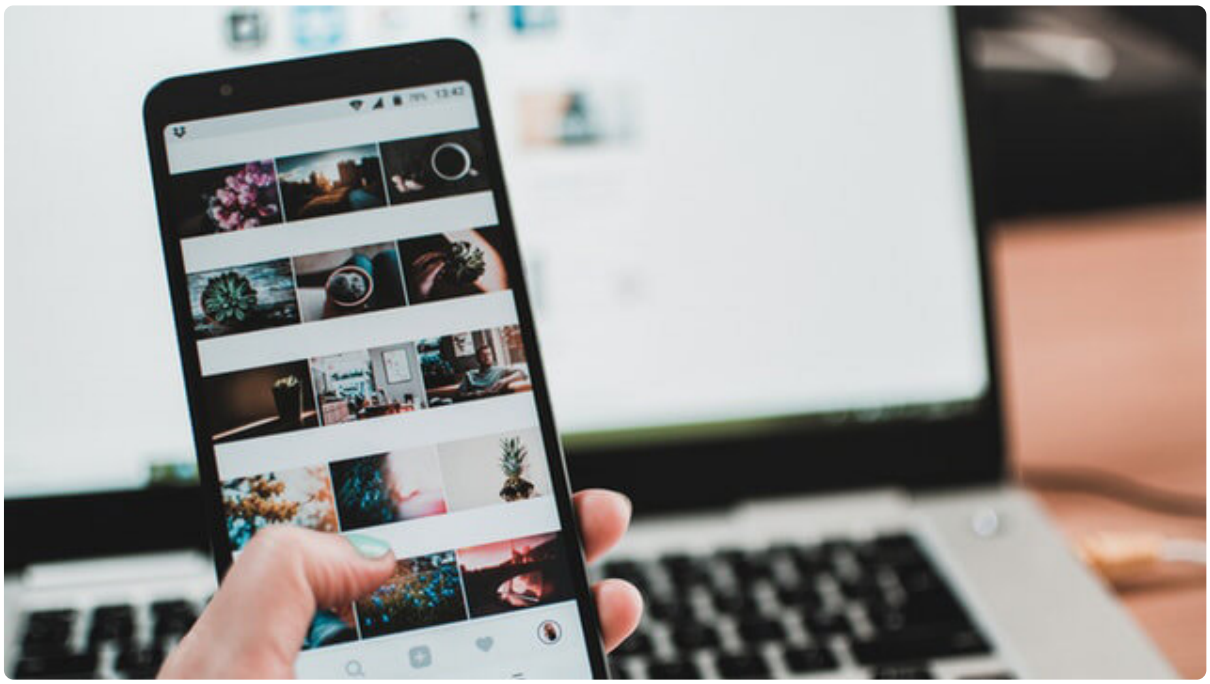
使用TypeScript开发Vue组件和使用Vue组件

更新时间：2019-08-01 19:57:52



> 能够生存下来的物种,并不是那些最强壮的,也不是那些最聪明的,而是那些对变化作出快速反应的。
>
> ——达尔文

本小节我们将学习使用Vue+TypeScript进行组件的开发和使用。我们在本章前面的小节中，虽然讲了很多内容，但是还没有封装一个可以复用的组件，没有涉及到属性和自定义事件。本小节我们将通过封装一个**CountTo**组件，展示如何封装一个组件。

首先我们先将创建项目时初始添加的*src/views/Home.vue*文件改为如下简单结构，把没用的先删掉：

```
<template>
  <div class="home">
  </div>
</template>

<script lang="ts">
import { Component, Vue } from 'vue-property-decorator'

@Component({
  name: 'HomePage'
})
export default class Home extends Vue {}
</script>
```

然后我们来封装CountTo组件，这里需要安装一个第三方插件：`npm install countup`，这个插件可以实现一个数字的动画过渡效果，它虽然有声明文件，但是有问题，所以我们自己写一份声明文件，使用我们自己定义的声明文件。

首先在根目录下创建一个*types*文件夹，然后在*types*文件夹下创建一个*countup*文件夹，也就是和我们要引入的模块同名的文件夹，然后在这个文件夹下创建一个*index.d.ts*声明文件。创建好之后，我们要用这个声明文件覆盖*node_modules/countup*里的声明文件，注意，不是去修改*node_modules/countup/index.d.ts*文件的内容，而是通过配置*tsconfig.json*里的*paths*字段来指定countup模块的声明文件位置，配置如下：

```
{
  "comilerOptions": {
    "paths": {
      "countup": [
        "./types/countup/index.d.ts"
      ],
    }
  }
}
```

这样，当我们使用import去引入countup插件的时候，编译器就会去我们这里指定的路径加载声明文件。接下来我们修改下声明文件，内容如下：

```
// types/countup/index.d.ts
declare function CountUp(target: string, startVal: number, endVal: number, decimals: number, duration: number, options: any): void;

declare module CountUp {
  var options: CountUpOptions;
  function version(): string;
  function printValue(value: any): void;
  function count(timestamp: any): void;
  function start(callback: Function): boolean;
  function pauseResume(): void;
  function reset(): void;
  function update(newEndVal: number): void;
}

interface CountUp {
  new(target: string, startVal: number, endVal: number, decimals: number, duration: number, options: any): CountUp;
  options: CountUpOptions;
  version(): string;
  printValue(value: any): void;
  count(timestamp: any): void;
  start(callback?: Function): boolean;
  pauseResume(): void;
  reset(): void;
  update(newEndVal: number): void;
}

interface CountUpOptions {
  useEasing: boolean; // toggle easing
  useGrouping: boolean; // 1,000,000 vs 1000000
  separator: string; // character to use as a separator
  decimal: string; // character to use as a decimal
  easingFn: Function; // optional custom easing closure function, default is Robert Penner's easeOutExpo
  formattingFn: Function; // optional custom formatting function, default is self.formatNumber below
}

export = CountUp;
```

你可能会奇怪，这些内容怎么写出来的。首先你肯定要去看countup插件的文档，先了解它有哪些api，然后再去看它的根目录下的index.d.ts文件。这里我们修改了一下CountUp接口的定义，添加了options和方法的定义，这样我们在创建实例后，在实例上调用方法就不会报错了。这个声明文件里涉及到的语法知识都比较基础，涉及到基本类型、接口、函数和模块插件声明文件的书写等，如果忘记了这些知识，可以看下前面的小节复习下。

接下来我们就可以封装CountTo组件了，在*src/components*文件夹创建*CountTo*文件夹，然后创建*index.tsx*文件，在这个文件里，我们先引入需要的装饰器，写下基本的内容：

```
import { Component, Emit, Prop, Vue, Watch } from 'vue-property-decorator'
import CountUp from 'countup'

@Component({
  name: 'CountTo'
})
export default class CountTo extends Vue {}
```

CountUp是一个构造函数，它有6个参数。第一个是要传入一个html元素的id字符串，用来在这个html标签中显示数值。但我们知道，这个组件肯定是要在多个地方复用的，如果写死一个id，那么在一个页面多次使用这个组件就会有问题，你会发现多个组件只显示了一个。所以这个id就需要是每个组件都独立的，那么就可以利用vue实例上没有提供给外界使用的属性 _uid ，它是每个组件都唯一的，所以可以利用它拼接一个id。我们来看些如何写：

```
// ...
export default class CountTo extends Vue {
  public get eleId() { // 计算属性的写法
    return `count_to_${(this as any)._uid}`
  }
}
```

这里顺便看一下vue的计算属性在这种类形式组件中的写法，这里只写get读取器函数，如果你学过vue应该知道还可以设置set存值器函数。这里如果直接通过 this._uid 来获取_uid的值，你会发现会提示 类型“CountTo”上不存在属性“_uid” ，这是因为vue的声明文件本身是没有提供这个字段的，所以我们可以通过类型断言指定this的类型为any来解决。

现在id有了，我们就可以来写一下render函数和mounted生命周期钩子了：

```
// ...
export default class CountTo extends Vue {
    public counter: CountUp = null
    public get eleId() {
        return `count_to_${(this as any)._uid}`
    }
    protected render() {
        return (
            <div class='count-up-wrap'>
                <span id={this.eleId}></span>
            </div>
        )
    }
    protected mounted() {
        this.counter = new CountUp(this.eleId, 0, 1000, 0, 1, {}) // 创建CountUp实例，并保存在counter上
        this.counter.start() // 调用此方法让动画效果开始
    }
}
```

我们看到，这里使用 new CountUp 来创建实例，第一个参数是html标签id，第二个是起始值，第三个是结束值，第四个是小数点后显示几位，第五个是以秒为单位的动画过渡时长，最后一个是额外的配置项。创建实例后保存在this.counter上，然后调用this.counter的start方法，让动画效果启动。

现在你可以在home页引入这个组件，看下效果：

```
<template>
  <div class="home">
    <count-to></count-to>
    <count-to></count-to>
  </div>
</template>

<script lang="ts">
import { Component, Vue } from 'vue-property-decorator'
import CountTo from '@/components/CountTo'

@Component({
  name: 'HomePage',
  components: {
    CountTo
  }
})
export default class Home extends Vue {}
</script>
```

现在你打开Home页，会发现页面上有两个数值，带着动效地从0变为1000，那么说明你封装的组件没问题，如果报错或者没效果，要检查你的代码咯。

效果有了，现在我们要把里面的一些参数通过组件属性的方式传进去，而不是写死了。我们这里就简单实现两个功能：第一个是传入起始值和结束值，第二个是提供一个update方法供动态设置结束值。

属性要使用Prop装饰器：

```
export default class CountTo extends Vue {
    @Prop({ type: Number, default: 0 }) public readonly start!: number
    @Prop(Number) public readonly end!: number
    // ...
    protected mounted() {
        this.counter = new CountUp(this.eleId, this.start, this.end, 0, 1, {})
        this.counter.start()
    }
}
```

Prop装饰器工厂函数接受的参数和以往开发时给Vue组件属性指定的值一样，可以指定一个类似*Number*这种类型，也可以是一个数组，也可以是一个对象指定类型和默认值等。然后它修饰的是实例属性，用readonly修饰，因为它是只读的，不可以直接修改，然后我们用`!`显示赋值断言来指定这个肯定不为null，这样就不会报错了。我们就可以把钩子函数中创建CountUp实例时传的第二、三个参数替换成属性传进来的值了。

现在我们再使用CountTo组件，就可以指定起始值：

```
<count-to :start="10" :end="99999"></count-to>
```

然后我们再在CountTo类里添加一个方法，用于更新数值：

```
    public update(endVal: number): void {
        this.counter.update(endVal)
    }
```

这样我们在使用CountTo组件的时候如何调用这个方法呢？可以通过ref获取到组件实例，然后调用，我们看下在Home.vue里怎么写：

```
<template>
  <div class="home">
    <count-to ref="counter" :end="10000"></count-to>
  </div>
</template>

<script lang="ts">
import { Component, Vue } from 'vue-property-decorator'
import CountTo from '@/components/CountTo'

@Component({
  name: 'HomePage',
  components: {
    CountTo
  }
})
export default class Home extends Vue {
  protected mounted() {
    setInterval(() => {
      (this.$refs.counter as CountTo).update(Math.random() * 10000) // 这样每2秒钟这里会更新一次，更新的值是随机的
    }, 2000)
  }
}
</script>
```

其实这里我们完全没必要在CountTo组件里写一个update方法，直接在Home页里通过 this.$refs.counter.counter.update() 去调用CountTo里保存在counter上的实例方法即可，这里只是为了演示你在使用第三方组件，需要调用它的方法时的方式。

最后我们学习如何自定义事件。这里我们给CountTo组件添加一个自定义的"on-click"事件，绑定在CountTo组件最外层的html标签上。虽然要在一个组件最外层绑定点击事件，不需要组件提供，直接在组件上使用"@click.native"绑定事件即可，我们这里为了学习使用自定义事件，所以在组件内添加一个事件。

添加自定义事件可以使用以往 this.$emit 来抛出事件，也可以使用*vue-property-decorator*提供的Emit装饰器：

```
// src/components/CountTo/index.tsx
import { Component, Emit, Prop, Vue, Watch } from 'vue-property-decorator'
import CountUp from 'countup'

@Component({
  name: 'CountTo'
})
export default class CountTo extends Vue {
  // ...

  @Emit('on-click') // 这里的on-click即为自定义事件名
  public click(event) { // 这个方法名用于组件内调用
    return event // return的值即为事件回调函数的参数
  }
  // 这里写完后效果等同于：public click(event) { this.$emit('on-click', event) }

  protected render() {
    return (
      <div class='count-up-wrap' on-click={ this.click }>
        <span id={this.eleId}></span>
      </div>
    )
  }

  // ...
}
```

这样我们就可以给CountTo组件绑定on-click事件了：

```
<!-- 省略部分代码 -->
   <count-to @on-click="handleClick" ref="counter" :end="10000"></count-to>
<!-- 省略部分代码 -->
<script lang="ts">
// ... 省略部分代码
export default class Home extends Vue {
  public handleClick(event) {
    console.log(event)
  }
  // ...
}
```

CountTo组件到这里就简单封装完了，虽然简单，但是涉及到了很多的知识，你可以再复习下。本章实战部分到这里就结束了，相信你已经学会了如何在Vue中使用TypeScript进行开发了，如果你对Vue还不太了解，可以去查阅下Vue的官方文档，还可以参考Vue官方的TypeScript支持文章。

> Tips：你可以去 Github 获取项目实战源代码！！！

搭建后台界面布局和结合Vuex实现完整登录流程

结束语

精选留言 0

欢迎在这里发表留言，作者筛选后可公开显示

目前暂无任何讨论