如何用 TypeScript 编写 Vue 项目

序: 写在最前

首先感谢 Gitbook.cn 能提供这个平台和大家分享。

在写这篇文的时候， TypeScript 已经发展到了 2.6 版本, Vue 也升级到了 2.5 版本。

本文将会以 3个例子项目 ，让大家真实的感受 TypeScript 带来的好处，希望有兴趣的同学，能动手跟着代码例子练下，有的时候，你看明白了，不代表你掌握了。

本文为了篇幅考虑，只贴关键代码，具体的还要大家自己翻下例子项目。

本文需要的知识

为了能顺利阅读本文，最好有用vue开发项目的经历，并非零起点。

GitChat

知识点	程度
vue	掌握
html5	掌握
css	掌握
ts	熟悉
node	熟悉
webpack	了解

本文代码环境

名称	说明
系统	macos
IDE	vscode
node	v8
npm	v5

一、改造一个 Vue 项目适合 TypeScript 开发

官方说是会提供 vue-cli 的 TypeScript 的项目模板，但是目前还没有，但是也不能阻止我们的脚步，希望早一天来到吧~

- 代码

按如下步骤操作

第一步：创建**vue**模板

```
sudo npm i -g vue-cli
vue init webpack helloword-vue-typescript
cd vue-typescript
cnpm i
```

> 如果你是 macos linux , 请输入 sudo 安装 vue-cli。

- 国内可以用cnpm
- cnpm安装

GitChat

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

第二步：加载需要的**js**包

```
cnpm i vue-class-component vue-property-decorator --save
cnpm i ts-loader typescript tslint tslint-loader tslint-config-standard --save-dev
```

- 包说明

名称	说明
vue-class-component	官网出的ts支持包
vue-property-decorator	对ts支持包更好封装的装饰器
ts-loader	webpack ts 解释器
typescript	ts 核心
tslint	ts 语法检查器

名称	说明
tslint-loader	webpack ts 语法检查
tslint-config-standard	ts 语法规则包

第三步：修改**webpack**配置

- 改名 ./src/main.js 改成 ./src/main.ts
- 修改 main.ts

```
// The Vue build version to load with the `import` command
// (runtime-only or standalone) has been set in webpack.base.conf with
an alias.
import Vue from 'vue'
import App from './App'
import router from './router'

Vue.config.productionTip = false

/* tslint:disable */
new Vue({
  el: '#app',
  router,
  template: '<App/>',
  components: { App }
})
/* tslint:enable */
```

> 如果要让 tslint 忽略编译检查，需要包括在 /* tslint:disable */ ... /* tslint:enable */ 。

- 编辑 ./build/webpack.base.conf.js

```
// 入口
entry: {
    app: './src/main.ts'
}

// resolve.extensions

resolve: {
  extensions: ['.js', '.vue', '.json', '.ts'],
  alias: {
    '@': resolve('src')
```

GitChat

```
  }
}

// module.rules

module: {
  rules: [
    ...
    {
      test: /\.ts$/,
      exclude: /node_modules/,
      enforce: 'pre',
      loader: 'tslint-loader'
    },
    {
      test: /\.tsx?$/,
      loader: 'ts-loader',
      exclude: /node_modules/,
      options: {
        appendTsSuffixTo: [/\.vue$/],
      }
    },
    ...
  }
}
```

GitChat

第四步：添加 **tsconfig.json**

- 这是 ts 的项目配置文件
- 文件放在根目录 ./tsconfig.json

```
{
    "include": [
        "src/**/*"
    ],
    "exclude": [
        "node_modules"
    ],
    "compilerOptions": {
        "allowSyntheticDefaultImports": true,
        "experimentalDecorators": true,
        "allowJs": true,
        "module": "esnext",
        "target": "es5",
        "moduleResolution": "node",
        "isolatedModules": true,
        "lib": [
            "dom",
            "es5",
```

```
            "es6",
            "es7",
            "es2015.promise"
        ],
        "sourceMap": true,
        "pretty": true,
        "strict": true // 这可以对 `this` 上的数据属性进行更严格的推断 ，这也是vue官网推荐加上的
    }
}
```

> "strict": true 推荐大家打开。

第五步:添加 **tslint.json**

- 这是语法检查的配置文件
- 文件位置 ./tslint.json

```
{
    "extends": "tslint-config-standard",
    "globals": {
        "require": true
    }
}
```

GitChat

第六步:创建 **vue-shim.d.ts**

- 这是定义文件，为了让ts代码识别 .vue 文件

> 在ts代码中导入vue代码包，需要写成 import xxx form xxx.vue ， 需要带上后缀 vue 。

- 文件位置 ./src/vue-shim.d.ts

```
declare module "*.vue" {
    import Vue from "vue";
    export default Vue;
}
```

第七步:修改 **src/app.vue**

```
<template>
  <div>
```

```
      <input v-model="msg">
      <p>msg: {{ msg }}</p>
      <button @click="handelHello">hello</button>
    </div>
  </template>

  <script lang="ts">
    import Vue from 'vue'
    import Component from 'vue-class-component'

    @Component
    export default class App extends Vue {
      // 初始化数据
      msg: string = 'hello word'

      mounted () {
      }

      // 方法
      handelHello () {
        this.msg = 'haha~~~ hello word!'
      }
    }
  </script>
```

GitChat

> <script ...> 这里 script 需要 设置成 lang="ts" 。

运行看下效果

```
npm start
```

二、如何用 TypeScript 编写 Vue 项目

我们来写一个图表展示程序来感受下：

- 代码

知识点

- 载入 xxx.d.ts 定义文件
- 编写 interface 接口类
- axios 异步请求数据并接受

- 装饰器 的使用
- mixin 在 TypeScript 中使用
- Vue 全局对象

开始动手,动手才是硬道理

第一步:安装依赖包 **axios echarts vue-echarts-v3 jquery** 包

```
cnpm install axios --save
cnpm install echarts --save
cnpm install vue-echarts-v3 --save
cnpm install jquery --save
```

> vue-echarts-v3 是百度 echarts 的vue版本封装,封装的不错，有时间大家可以阅读下源码

第二步:编写 **interface** 数据接口类

为了实现真实 http 请求，我把数据放在了 easy-mock 上，一个不错的 mock-server 。

GitChat

- get数据
- 样式

```
[
  {
    "year": "1991",
    "value": 3
  },
  {
    "year": "1992",
    "value": 4
  },
  ...
}
```

- 创建接口文件 src/interface/IDataValue.ts

```
export interface IDataValue {
    year: string
    value: number
}
```

> 接口名字一般建议 Ixxxx ，方便别人阅读。
>
> year value 设置了静态类型。

第三步:请求并接受数据

```
ajaxData () {
  let that = this
  Axios.get('https://www.easy-
mock.com/mock/5a1bb2639144e669fc6e43fa/example/g2-data')
    .then(function (response) {
      let dataList: Array<IDataValue> = response.data // 接收并转换的类
型的关键
      ...
    })
    .catch(function (error) {
      console.log(error)
    })
}
```

> response.data 到 dataList 是一个自动转换，很简单啊，如果你格式不匹配 interface 接口，编译器会提示错误，这点很棒，大家可以尝试了改下看看效果。

GitChat

第四步:获取 **echarts** 的 **instance** 对象

- 安装定义

```
cnpm install --save @types/echarts
```

- 代码

```
<template>
    <IEcharts ... @ready="handelReady" ></IEcharts>
</template>

<script lang="ts">
import { ECharts } from 'echarts'

export default class App extends Vue {
  instance: ECharts
...
  handelReady (instance: ECharts) {
    this.instance = instance
```

```
  }
</script>
```

> 组件安装完成后 @ready=... 回调返回 instance 。
>
> 这一步主要是让大家细细的体会下 xxx.d.ts 定义的作用，只有导入了定义文件，编译器、编辑器才会自动识别。

第五步:使用 **jquery** 监控窗口大小

- 安装定义

```
cnpm install --save @types/jquery
```

- 代码

```
import $ from 'jquery'

export default class App extends Vue {

  mounted () {
    $(window).resize(() => {
      let width = $(window).width()
      let height = $(window).height()
      this.msg1 = `[illegible]`
    })
  }
```

GitChat

> 不要应为在 Vue 中用 Jquery 操作 Dom 而感觉不适，毕竟 Vue 操作的重点是在数据展示的 Dom 对象上，比如操作 window 窗体对象，用 Jquery 可以节省很多代码。

第六步：在 **TypeScript** 中使用 **mixin** 特性

- 编写 mixin\pageMixin.ts

```
import { Vue, Component } from 'vue-property-decorator'

/**
 * Mixin Page
 *
 * @export
 * @class PageMixin
 * @extends {Vue}
```

```
 */
@Component({})
export default class PageMixin extends Vue {

  mounted () {
    alert(`hello echarts example`)
  }

}
```

> mixin 那么重要的特性，请在 TypeScript 中用起来吧~

- 修改 app.vue

```
<script lang="ts">
import PageMixin from './mixin/pageMixin'

@Component({
  components: {
    ...
  },
  mixins: [PageMixin]
})
export default class App extends Vue {
```

GitChat

> 你现在运行，将会看到弹出框，这是 PageMixin 模块起作用了。

第七步:设置 **Vue** 全局对象

- 修改 vue-shim.d.ts

```
import Vue from 'vue'

declare module "*.vue" {
    export default Vue;
}

declare module 'vue/types/vue' {
    interface Vue {
      $textInfo: string
    }
}
```

> $textInfo: string 是我们加的扩张定义。在定义文件中，采用 interface 方式来扩展，加入自己的对象、方法、类。

- 代码调用

```
mounted () {
  this.msg2 = this.$textInfo
```

运行看下效果

```
npm start
```

三、转 TypeScript 后对开发 Vue 到底带来哪些好处？

存在即合理 --- 黑格尔

我用 TypeScript 开发了几个 Vue 项目，谈几点吧：

GitChat

- TypeScript github上 star 28208 issues closed 12153 ，已经很成熟了
- 最终编译成 es5 代码，和 Vue 目标代码不冲突
- 作为 JavaScript 的超集，不强求使用类型定义，可写可不写，灵活度高
- 你可以直接把一个 .js 文件直接改成 .ts 让编译器编译，兼容性好
- 静态类型 在编码阶段 就能错误提示，降低了测试的成本，所以能不用 any 就不用
- 重构 风险小非常适合构建大型项目，特别是服务接口API定义发生了更新，在 编码 编译 阶段就能处理
- IDE 开发工具都有了插件 WebStorm VSCode Sublime Text Atom
- 上一章的代码中，我们可以发现用了装饰器后代码精简多了，越是精简的代码，我们越容易书写调试
- 有了 interface 接口定义，再也不用担心运行时出现数据错误了，编译阶段就提示了
- 非常适合编写大型项目，因为 重构 后，项目风险被控了,如果小项目一共没几个页面、也没几行代码用啥写都没关系

- 如果你手上有个 JavaScript 的 Vue 项目，你中途都可以直接用 TypeScript 写新的功能代码，和原来的 JavaScript 代码融合的非常好、无缝衔接，我已经在这样用啦

第四章： xxx.d.ts 文件到底是什么鬼??

类型定义技术随着 TypeScript 2.0 的发展，提出了 DefinitelyTyped

我的体会主要作用有两点:

- 让IDE编辑器能之别，提供 代码智能提示 代码错误提示
- 在代码编译阶段能提示错误、警告、相关提示

我们动手写一个程序体会下

我们在 helloword 的基础上编写例子项目 definitelyTyped-example-vue-typescript

- 源码

编写 src/typtes/global.d.ts。

GitChat

下面我把常见的定义，都罗列了，大家可以在 /test.ts 中编写定义的对象， IDE 会有相应的提示：

> 如果你修改了定义文件，编辑器没有提示，请重启编辑器。

type 定义

```
// type 定义类型
type myString = string
type myNumber = number

// type 定义函数
type hello = (msg: string) => void

// type 对象
type WebSite = {
    url: string;
    title: number;
}
```

interface 定义

```
// interface 对象
export interface IDataValue {
    year: string
    value: number
}
```

type && interface 的区别

- type 是给内部使用的，当默认的类型不够用时
- type 不能被继承， interface 可以被继承
- interface 是对外输出的接口

declare 定义

```
// 变量定义
declare var val1:myString

// 变量类型可以是 字符串 数字
declare var val2:myString | myNumber

// 函数定义
declare function sayHello(msg: string):void

// 重载
declare function sayHello(num: number, logNum:number):void

// 类定义
declare class HtmlMetaTag {
    keywords: string
    description: string
    constructor(keywords: string, description: string)
}
```

namespace 全局命名空间

```
declare namespace HtmlTag{
    var name: string
    function getTagName(): string

    class Div {
        name: string
        constructor(name: string)
        getName(): string
    }
}
```

GitChat

> 对象的形式对外暴露，内部成员不需要 declare 。

module 模块化

```
declare module "FormInputTag" {
    export let classTag: string
    export let idTag: string
    export let nameTag: string
    export let valueTag: string

    export function getFullTag(): string
}
```

> 模块形式导出，内部需要 export 指定导出对象。

第五章：Vue 装饰器是如何让 TS 变得那么优雅的？

- 如果要使用官方的 装饰器 需要先安装 vue-property-decorator 。

GitChat

```
npm i vue-property-decorator --save
```

这个包需要 vue-class-component 的依赖。

- 官方提供了 7 个装饰器

```
@Emit
@Inject
@Model
@Prop
@Provide
@Watch
@Component (exported from vue-class-component)
```

- 到底代码发生了什么改变，我们直接看 <script>...</script>

采用装饰器后，代码如下：

```
import { Component, Emit, Inject, Model, Prop, Provide, Vue, Watch }
from 'vue-property-decorator'

const s = Symbol('baz')
```

```
@Component
export class MyComponent extends Vue {

  @Emit()
  addToCount(n: number){ this.count += n }

  @Emit('reset')
  resetCount(){ this.count = 0 }

  @Inject() foo: string
  @Inject('bar') bar: string
  @Inject(s) baz: string

  @Model('change') checked: boolean

  @Prop()
  propA: number

  @Prop({ default: 'default value' })
  propB: string

  @Prop([String, Boolean])
  propC: string | boolean

  @Provide() foo = 'foo'
  @Provide('bar') baz = 'bar'

  @Watch('child')
  onChildChanged(val: string, oldVal: string) { }

  @Watch('person', { immediate: true, deep: true })
  onPersonChanged(val: Person, oldVal: Person) { }
}
```

GitChat

换成之前的传统代码：

```
const s = Symbol('baz')

export const MyComponent = Vue.extend({
  name: 'MyComponent',
  inject: {
    foo: 'foo',
    bar: 'bar',
    [s]: s
  },
  model: {
    prop: 'checked',
    event: 'change'
  },
```

```
  props: {
    checked: Boolean,
    propA: Number,
    propB: {
      type: String,
      default: 'default value'
    },
    propC: [String, Boolean],
  },
  data () {
    return {
      foo: 'foo',
      baz: 'bar'
    }
  },
  provide () {
    return {
      foo: this.foo,
      bar: this.baz
    }
  },
  methods: {
    addToCount(n){
      this.count += n
      this.$emit("add-to-count", n)
    },
    resetCount(){
      this.count = 0
      this.$emit("reset")
    },
    onChildChanged(val, oldVal) { },
    onPersonChanged(val, oldVal) { }
  },
  watch: {
    'child': {
      handler: 'onChildChanged',
      immediate: false,
      deep: false
    },
    'person': {
      handler: 'onPersonChanged',
      immediate: true,
      deep: true
    }
  }
})
```

GitChat

很明显了吧，代码更清晰了，一个定义直接使用，无需多层嵌套，如果对代码行多的项目，真的很有帮助。

如果大家对 ES6 装饰器 这个特性不熟悉，请移步w3cschool。

第六章：TS 在整合 Vue 生态圈中处于什么状态？

其实这个话题也是本文的总结，从两个方面来看：

官方支持

能得到官方的支持才是最好的：

- `vue-class-component` 官方的类包支持
- `vue-property-decorator` 官方提供的 装饰器
- `vuex@>= 3.0` 已经自带 `types` 定义,对应 `Vue2.5`

> https://github.com/vuejs/vuex/tree/dev/types

社区支持

- 你要找的包都在这里 `@types organization`

> @types organization
>
> 你可以直接 `install i @types/xxx -S` , 常见的第三方包都能顺利安装

- 很多第三方包，都已经自带 `xxx.d.ts` 文件，无需另外找寻

总之放心的在自己项目中尝试吧~

GitChat

本文代码

- helloword-vue-typescript
- echarts-example-vue-typescript
- definitelyTyped-example-vue-typescript

本文用到的组件

- @types organization
- axios

- echarts
- vue-echarts-v3
- vue-class-component
- vue-property-decorator

参考文章

- Microsoft/TypeScript
- vue官网-TypeScript支持说明
- EvanYou博文说ts
- ts中文手册
- tslint
- DefinitelyTyped
- Vuex

GitChat