

第四节课 – Web编程框架Vue

写在前面

Vue是目前比较主流的Web编程框架之一，可以显著提高编程效率，同时对于多人合作编程也有一些便利之处。当然，即便Vue是一个渐进式的框架，即只学习一部分内容就可以不错的使用这个框架，再深入学习则是可以更好的使用，但学习一个新的框架总归是需要更多的学习成本，这个成本不一定能够被编程效率的提高完全抵消掉。总之，我们鼓励大家学习Vue框架及其相关的思想，但是最后的期末项目中不强制大家使用Vue或者其他框架，大家可以根据实际情况权衡一下。

总览

环境配置

Vue

- 为什么要用Vue?

- 如何引入Vue?

- 基本框架：声明式渲染

 - 数据

 - 模板语法

 - 方法

 - 生命周期与钩子函数

- 组件间的通信

 - (1) 父通过props向子组件传递参数

 - (2) 父组件监听子组件的事件，子组件使用\$emit触发事件

 - (3) 父组件可以使用\$refs直接调用子组件的方法

 - (4) 全局变量：Vuex

- 响应式

 - (1) 简单的响应

 - (2) watch：自定义响应

 - (3) 视图的切换

 - (4) 响应式的局限性：对于Object的响应

- 和d3.js的组合使用

环境配置

【Step1】安装NodeJS

在官网<https://nodejs.org/zh-cn/download/>下载对应版本，在本地进行安装

安装好后，在控制台输入命令node -v查看版本号，如果安装好了则会输出安装的版本号。

```
C:\Users\dell>node -v
v14.17.3
```

【Step2】安装live-server（可选）

```
1 # Mac系统可能需要在前面加sudo，并根据提示输入用户密码
2 npm install -g live-server
```

如果运行npm install时遇到一些报错，则根据报错上提示的方法尝试进行修复。如果仍然解决不了，可以上网查阅相关解决方法，着重关注Node的版本以及各个依赖包的版本问题。问助教的话助教大概率也只能靠CSDN帮你解决。

【Step3】Vue

(1) 安装Vue

```
1 # Mac系统可能需要在前面加sudo，并根据提示输入用户密码
2 npm install -g @vue/cli
```

(2) 创建项目

```
1 # 路径替换成需要的项目路径
2 # 对于Windows来说，如果路径不是C盘，则需要先切换磁盘（例如输入指令E:可以切换到E盘）
3 # 例如Windows 10桌面路径C:\Users\dell\Desktop，dell是用户名
4 cd 路径
5 # 项目名称替换成替换成自己的命名，下同
6 vue create 项目名称
```

选择Manually select features（第一个vue3选项是之前保存的配置，大家那里应该没有）

```
Vue CLI v5.0.1
New version available 5.0.1 → 5.0.8
Run npm i -g @vue/cli to update!

? Please pick a preset:
  vue3 ([Vue 3] dart-sass, babel, router, vuex)
  Default ([Vue 3] babel, eslint)
  Default ([Vue 2] babel, eslint)
> Manually select features
```

按照图中进行配置（空格选中或反选，选中Babel、Router、Vuex）

- 网上很多说要勾选Linter的，但是这个会严格规范代码风格，对初学者造成很多困扰

```
Vue CLI v5.0.1
New version available 5.0.1 → 5.0.8
Run npm i -g @vue/cli to update!

? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to invert selection, and
<enter> to proceed)
> (*) Babel
  ( ) TypeScript
  ( ) Progressive Web App (PWA) Support
  (*) Router
  (*) Vuex
  ( ) CSS Pre-processors
  ( ) Linter / Formatter
  ( ) Unit Testing
  ( ) E2E Testing
```

选择3.x版本

```
Vue CLI v5.0.1
New version available 5.0.1 → 5.0.8
Run npm i -g @vue/cli to update!

? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex
? Choose a version of Vue.js that you want to start the project with (Use arrow keys)
> 3.x
  2.x
```

输入n

```
Vue CLI v5.0.1

New version available 5.0.1 → 5.0.8
Run npm i -g @vue/cli to update!

? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n) n
```

选In dedicated config files（选另一个也没关系，不影响）

```
Vue CLI v5.0.1

New version available 5.0.1 → 5.0.8
Run npm i -g @vue/cli to update!

? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Where do you prefer placing config for Babel, ESLint, etc.? (Use arrow keys)
> In dedicated config files
  In package.json
```

要不要保存这个配置（根据实际情况选择即可）

```
Vue CLI v5.0.1

New version available 5.0.1 → 5.0.8
Run npm i -g @vue/cli to update!

? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? (y/N) N
```

(3) 安装依赖（主要是d3.js）

```
1 # 进入项目目录
2 cd 项目名称
3 # 安装d3.js库
4 npm install -s d3
```

在js代码中引用：

```
1 import * as d3 from 'd3'
```

(4) 启动项目

```
1 npm run serve
```

看到下面的界面说明启动成功：

```
DONE Compiled successfully in 3350ms 下午5:52:19

App running at:
- Local: http://localhost:8080/
- Network: http://10.6.50.37:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

在浏览器进入上面界面显示的网址（这里是<http://localhost:8080/>），应该会看到下面的页面：



Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project,
check out the [vue-cli documentation](#).

Installed CLI Plugins

[babel](#) [router](#) [vuex](#)

Essential Links

[Core Docs](#) [Forum](#) [Community Chat](#) [Twitter](#) [News](#)

Ecosystem

[vue-router](#) [vuex](#) [vue-devtools](#) [vue-loader](#) [awesome-vue](#)

Vue有内置的热更新机制，即修改代码并保存之后会同步更新视图效果，无需刷新网页。但是热更新结果经常会出错，建议还是刷新一下。

(5) 打包项目

Vue文件不能直接放到服务器上，要打包成原始的HTML、CSS、JavaScript、图片、数据资源等文件。如果还在原先运行项目的控制台页面，记得先使用Ctrl+C终止项目。输入

```
1 npm run build
```

成功后界面大致如下：

```
DONE Compiled successfully in 9231ms 下午5:56:30

File                               Size           Gzipped
dist\js\chunk-vendors.49fb514e.js  110.36 KiB     39.68 KiB
dist\js\app.6cfa85cc.js           15.45 KiB      9.48 KiB
dist\js\about.de3bebb7.js          0.39 KiB       0.29 KiB
dist\css\app.ae69e941.css          0.42 KiB       0.26 KiB

Images and other types of assets omitted.
Build at: 2022-10-24T09:56:30.731Z - Hash: 4ef93310aa81bb6a - Time: 9231ms

DONE Build complete. The dist directory is ready to be deployed.
INFO Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html
```

(6) 查看打包结果

打包结果默认放在项目里的dist文件夹下，可以进入dist文件使用live-server查看最终结果：

```
1 cd dist
2 # 如果前面没有使用npm安装live-server，也可以在vscode里打开dist文件夹，
3 # 再使用vscode插件live-server，其效果是相同的
4 live-server
```

Vue

官方文档：<https://cn.vuejs.org/>

为什么要用Vue?

原生JavaScript：

修改变量，将修改手动应用到HTML上

// 标签种类固定，需要自行组合成实际需要的功能组件

Vue:

修改变量，修改会响应式地自动应用到HTML上

// 可以自定义组件拓展现有的标签，实现代码的复用（单文件组件）

Count is: 0

```
<template>
  <div>
    <button @click="add">Count is: {{count}}</button>
  </div>
</template>
<script>
export default {
  data(){
    return {
      count: 0,
    },
  },
  methods: {
    add(){
      this.count++
    }
  }
}
</script>
<style>
</style>
<body>
  <button onclick="add()"></button>
  <script>
    let count = 0
    let button = document.getElementsByTagName('button')[0]
    function add(){
      count++
      button.innerText = `Count is: ${count}`
    }
    button.innerText = `Count is: ${count}`
  </script>
</body>
```

例子给的是单文件组件的情况，这个是更常用的一种写法，还有一种直接嵌入HTML的写法这里不再提及。

如何引入Vue?

将前面的代码写入ButtonAdd.vue文件里，并放在src/components/目录下
src/App.vue里写法如下：

```

<template>
  <ButtonAdd></ButtonAdd>
</template>
<script>
import ButtonAdd from '@components/ButtonAdd.vue'
export default {
  components: {
    ButtonAdd,
  }
}
</script>
<style>
</style>

```

当然可以更多层嵌套

assets: 静态资源 (图片、静态数据等)

components: 单文件组件目录 (前面的ButtonAdd.vue)

router: 路由的定义, 用于在不同views之间进行切换

store: Vuex全局变量

views: 由各种组件拼接成的完整视图

App.vue: 创建HTML的根内容, 由main.js定义

main.js: 程序的入口

基本框架：声明式渲染

数据

data:

模板所拥有的数据, 可以修改

props: 外部参数传递进来的属性

只读属性, 只有外部才能修改

computed: 计算属性

在相关数据被更新之后会自动更新, 不会重复计算

computed内部可以使用其他computed变量 (只要不循环引用即可)

```

<template>
  <div>
    <button @click="add">{{percent}}%</button>
  </div>
</template>
<script>
export default {
  data(){
    return {
      count: 0
    },
    props: {
      maxCount: {
        type: Number, // 类型是Number
        default: 10, // 没有传参时的缺省值
        required: false // 是否必须传该参数
      },
      computed: {
        percent(){
          return this.count / this.maxCount * 100
        },
        methods: {
          add(){
            this.count++
          }
        }
      }
    }
  }
}
</script>
<ButtonAdd :maxCount="100"></ButtonAdd>

```

0%

模板语法

文本插值

v-bind: 绑定属性

v-if、v-show

v-on

```

<button
  v-if="showButton"
  v-bind:style="{height: buttonHeight+'px'}"
  :disabled="false"
  :value="buttonValue"
  type="button"
  v-on:click="add"
  @mouseover="focus"
  @mouseout="unfocus"
>
Count is: {{count}}
</button>

```

v-for: 需要有一个唯一的标记属性key

v-html: 直接定义内部的html

```

<template>
  <div>
    <button v-for="(count, i) in counts"
      :key="i"
      v-html="html(count)"
      @click="add(i)"
    >
  </button>
</div>
</template>
<script>
export default {
  data(){
    return {
      counts: [0, 1, 2],
    },
    methods: {
      add(i){
        this.counts[i]++
      },
      html(count){
        return `Count is: ${count}`
      }
    },
  }
}
</script>

```

Count is: 0 Count is: 1 Count is: 2

方法

methods部分，借用上面的例子即可

(1) 可以在模板、计算变量、以及created以及后续的钩子函数（后面会讲）中使用

- (2) 可以调用data、props、computed变量以及其他方法，调用时前面要加上this
- (3) 方法和变量不能重名（实际上不同变量之间也不能重名），否则会覆盖

生命周期与钩子函数

编程人员可以在组件不同的构建阶段自行定义一些处理函数

最常用的是对组件进行初始化

created：数据和method已经存在，模板还没有编译

mounted：模板已经完成编译，可以访问相关的html

```
<script>
export default {
  data(){
    return {
      .
      .
    }
  },
  created(){

  },
  mounted(){

  }
}
</script>
```

组件间的通信

- (1) 父通过props向子组件传递参数

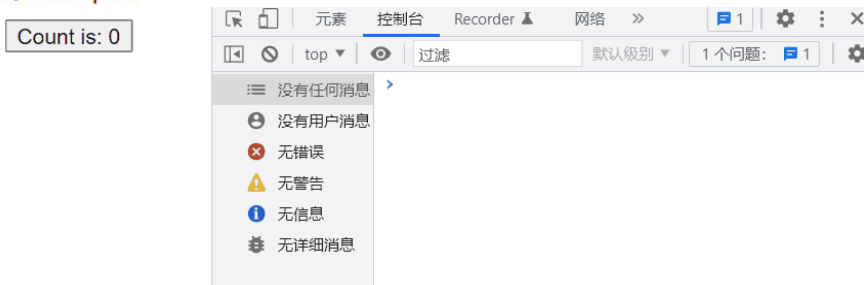
这个参数对于子组件来说是只读的

- (2) 父组件监听子组件的事件，子组件使用\$emit触发事件

```

<template>
  <div>
    <button @click="add">Count is: {{count}}</button>
  </div>
</template>
<script>
export default {
  data(){
    return {
      count: 0
    }
  },
  methods: {
    add(){
      this.count++
      this.$emit('add', this.count)
    }
  },
}
</script>
<template>
  <ButtonAdd
    :maxCount="100"
    @add="addEvent"
  ></ButtonAdd>
</template>
<script>
import ButtonAdd from '@components/ButtonAdd.vue'
export default {
  components: {
    ButtonAdd,
  },
  methods: {
    addEvent(count){
      console.log(`Now the count is ${count}`)
    }
  }
}
</script>

```



(3) 父组件可以使用\$refs直接调用子组件的方法

注意：要确保子组件已经加载完成

```

<template>
  <ButtonAdd
    ref="button"
    :maxCount="100"
  ></ButtonAdd>
</template>
<script>
import ButtonAdd from '@components/ButtonAdd.vue'
export default {
  components: {
    ButtonAdd,
  },
  mounted(){
    this.$refs.button.add()
  }
}
</script>

```

(4) 全局变量：Vuex

优点：变量是全局共享的

```

import { createStore } from 'vuex'

export default createStore({
  state: { // 维护的全局变量
    count: 0,
    maxCount: 100,
  },
  getters: { // 类似计算属性
    percent(state){
      return state.count / state.maxCount * 100
    }
  },
  mutations: { // 更新变量的方法，必须是同步的
    updateCount(state, payload){
      state.count = payload
    },
    updateMaxCount(state, payload){
      state.maxCount = payload
    }
  },
  actions: { // 更新变量的方法，可以是异步的

  },
  modules: { // 子模块
  }
})

```

```

<template>
  <div>
    <button @click="add">{{percent}}%</button>
  </div>
</template>
<script>
import { mapState, mapGetters, mapMutations } from "vuex"
export default {
  data(){
    return {
    },
    computed: {
      ...mapState([
        'count',
      ]),
      ...mapGetters([
        'percent',
      ])
    },
    methods: {
      ...mapMutations([
        'updateCount',
        'updateMaxCount'
      ]),
      add(){
        this.updateCount(this.count+1)
      }
    },
  }
}
</script>

```

响应式

(1) 简单的响应

修改变量后会立即触发HTML的变化

注意：这个变量必须是在vue里面定义的数据、props、computed及其相关的表达式

```

<template>
  <div>
    <button @click="add">Count is: {{count}}</button>
  </div>
</template>
<script>
export default {
  data(){
    return {
      count: 0,
    }
  },
  methods: {
    add(){
      this.count++
    }
  }
}
</script>
<style>
</style>

```

(2) watch：自定义响应

有时候在数据变更的时候需要更复杂的处理逻辑

```

<template>
  <div>
    <button @click="add">{{percent.toFixed(2)}}%</button>
  </div>
</template>
<script>
import * as d3 from 'd3'
export default {
  data(){
    return {
      count: 0,
      maxCount: 100,
      percent: 0,
    }
  },
  methods: {
    add(){
      this.count++
    }
  },
  watch: {
    count: {
      handler(newValue, oldValue){
        d3.transition()
          .duration(500)
          .tween('text', ()=>{
            let i = d3.interpolateNumber(oldValue, newValue)
            return (t)=>{
              this.percent = i(t) / this.maxCount * 100
            }
          })
      }
    }
  },
  mounted(){
    this.percent = this.count / this.maxCount * 100
  }
}
</script>

```

0.00%

(3) 视图的切换

src/router/index.js

```

const routes = [
  {
    path: '/',
    name: 'home',
    component: HomeView
  },
  {
    path: '/about',
    name: 'about',
    // route level code-splitting
    // this generates a separate chunk (about.[hash].js) for this route
    // which is lazy-loaded when the route is visited.
    component: () => import(/* webpackChunkName: "about" */ '../views/AboutView.vue')
  }
]

```

模板里可以使用<router-link>元素实现切换

```

<nav>
  <router-link to="/">Home</router-link> |
  <router-link to="/about">About</router-link>
</nav>

```

代码里面可以使用this.\$router.push(new_path)或者this.\$router.replace(new_path)

(4) 响应式的局限性：对于Object的响应

前面讲过判断Object相等使用的是地址是否相等

解决方法:

- (1) watch设置deep属性
- (2) 每次更新变量时新建一个Object

```
<template>
  <div>
    <button @click="add">{{percent.toFixed(2)}}%</button>
  </div>
</template>
<script>
import * as d3 from 'd3'
export default {
  data(){
    return {
      count: { value: 0 },
      maxCount: 100,
      percent: 0,
    }
  },
  methods: {
    add(){
      this.count.value++
      // watch不会监听到，要换成this.count = { value: this.count + 1 }
    }
  },
  watch: {
    count: {
      handler(newValue, oldValue){
        d3.transition()
          .duration(500)
          .tween('text', ()=>{
            let i = d3.interpolateNumber(oldValue.value, newValue.value)
            return (t)=>{
              this.percent = i(t) / this.maxCount * 100
            }
          })
      },
      // deep: true,
    }
  },
  mounted(){
    this.percent = this.count.value / this.maxCount * 100
  }
}
</script>
```

和d3.js的组合使用

d3.select(this.\$el)选中当前组件

定义不同部分的渲染函数，使用watch监听外部状态控制渲染

在watch中定义d3.transition