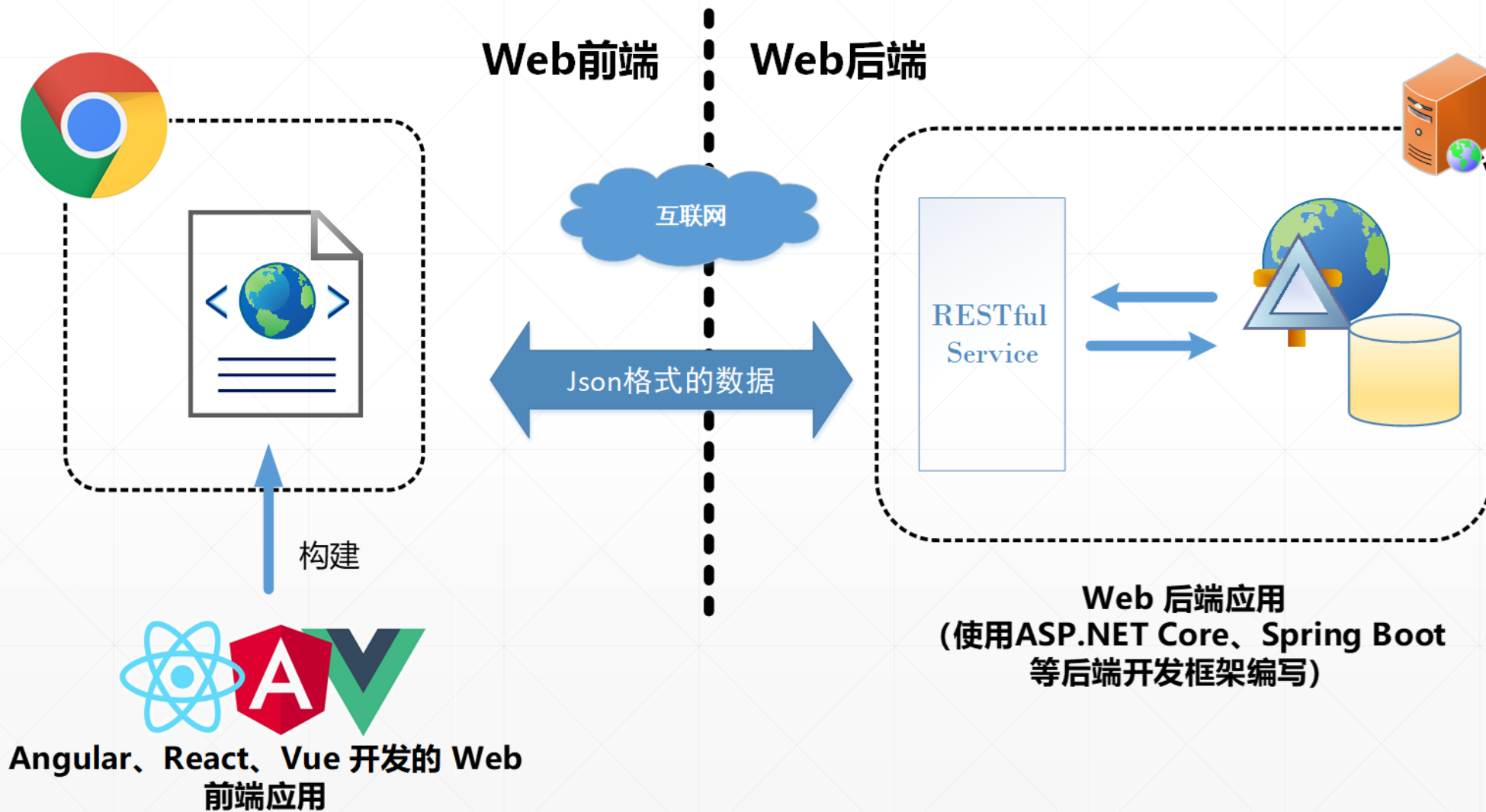




前后端分离的Web应用

北京理工大学计算机学院
金旭亮

“前后端分离”的移动互联应用



原生态

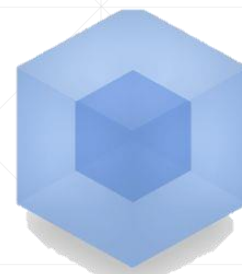
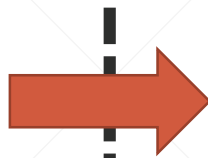
工程化



不要怂就是干!



早期网页设计，上来就直接写CSS/JS代码到网页中，立即就可以在浏览器中跑起来.....



webpack



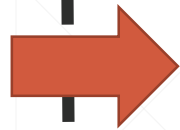
现在的Web前端应用，需要一个“**构建**”过程——对原材料进行编译和转换之后，浏览器才认识它们.....

Web前端开发进入“组件化开发”时代



以网页为中心

过去



以组件为中心

现在



开发前后端分离的Web应用

示例的总体结构

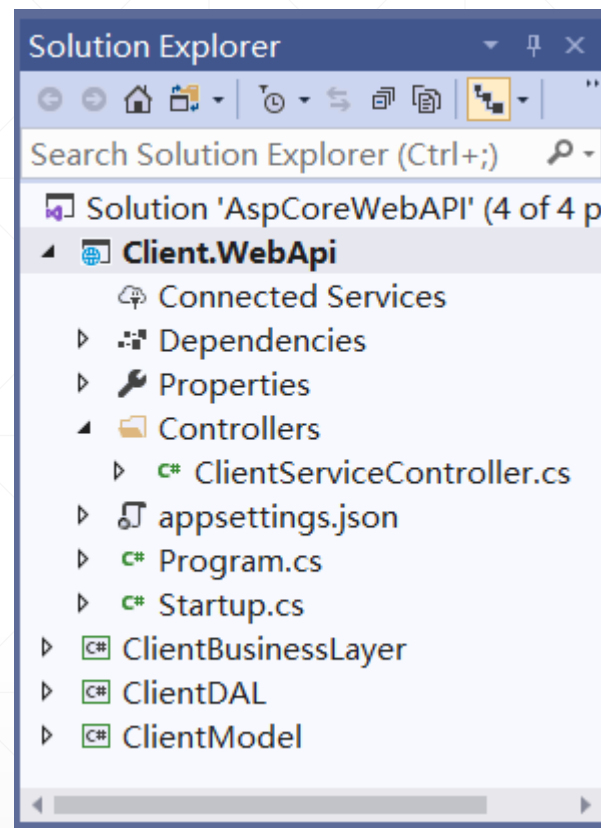


使用Vue CLI创建的
独立Web前端项目：
client_app

RESTful
Service



客户信息
{Json}



在前一讲AspCoreSPA项目基
础上改造而成的Web后端项目

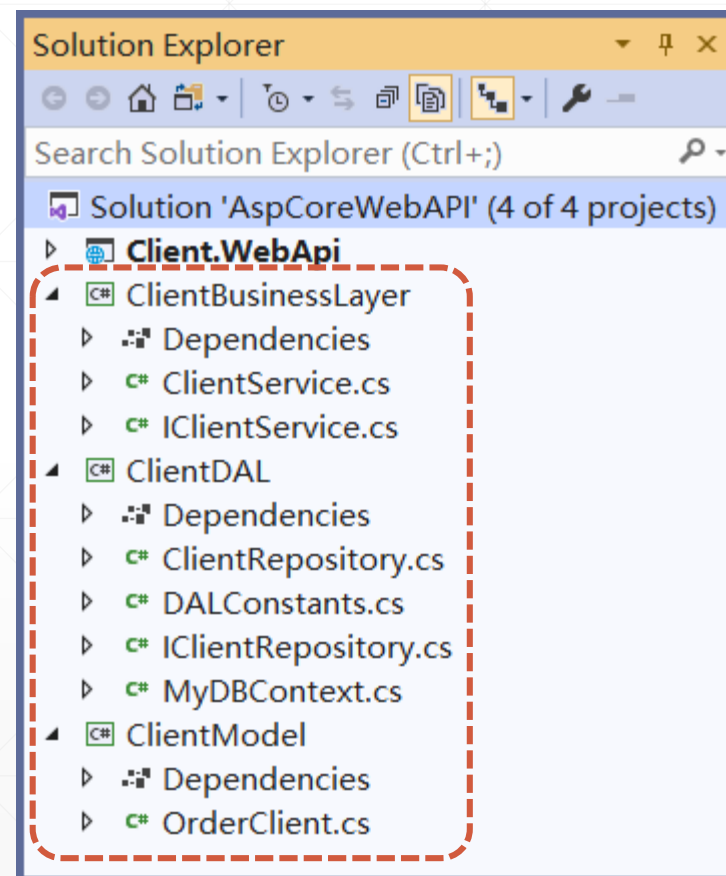
准备工作

1

示例数据库MyDB已经附加到了本机SQL Server之上。

2

从前一讲示例AspCoreSPA中复制三个类库项目到本讲示例解决方案中。
再创建一个Web API项目作为后端服务。



开发Web API项目

1

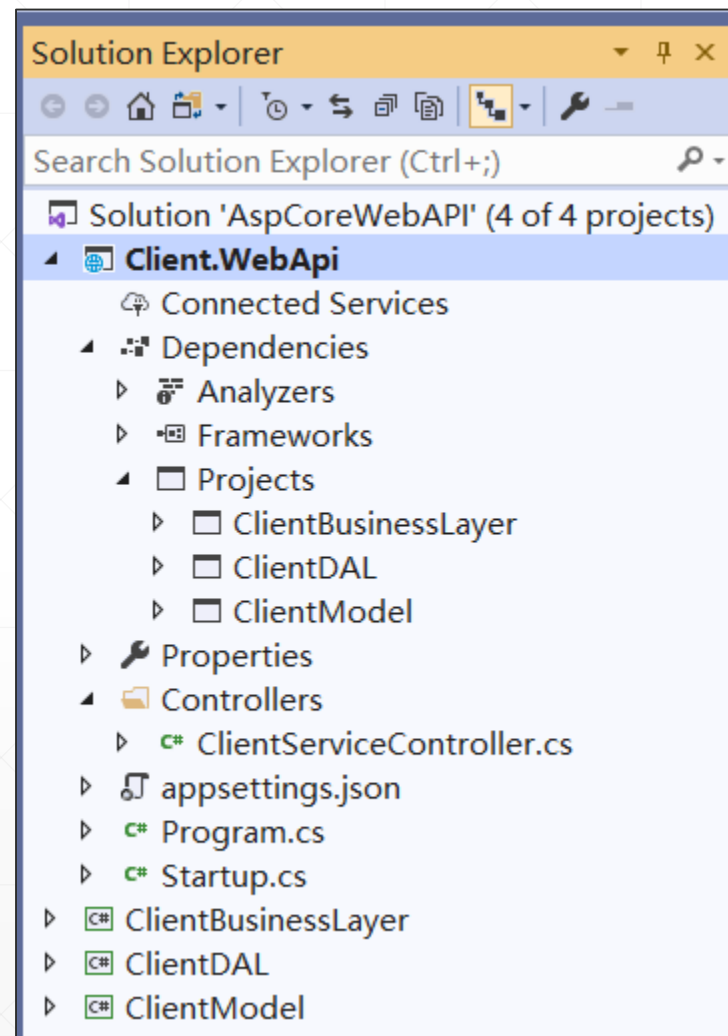
给Web API项目添加对三个类库项目的引用，修改Startup.cs，向IoC容器注册三个类库项目中的类型。

2

在appsettings.json中添加MyDB示例数据库的连接字符串。

3

删除WeatherForecastController，添加ClientServiceController。



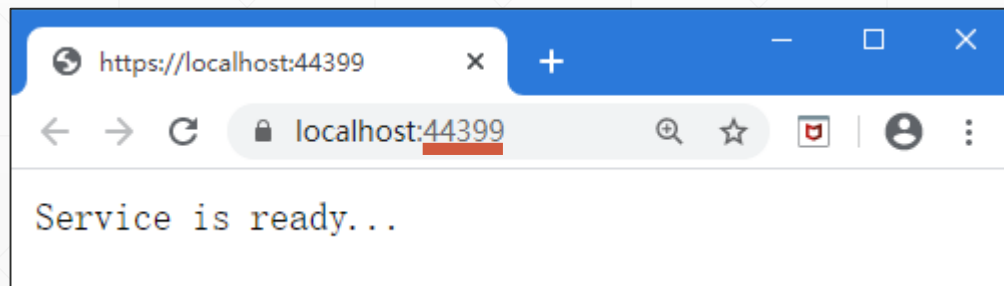
实现Web API控制器

```
[Route("api/[controller]")]
[ApiController]
public class ClientServiceController : ControllerBase
{
    private IClientService clientService = null;
    //将IClientService注入到控制器中
    public ClientServiceController(IClientService clientService)
    {
        this.clientService = clientService;
    }
    //提取所有的客户
    [HttpGet("clients")]
    public async Task<IActionResult> GetClients()
    {
        var clients = await clientService.GetAllClients();
        return Ok(clients);
    }
}
```

试运行.....

Startup.cs 中的代码:

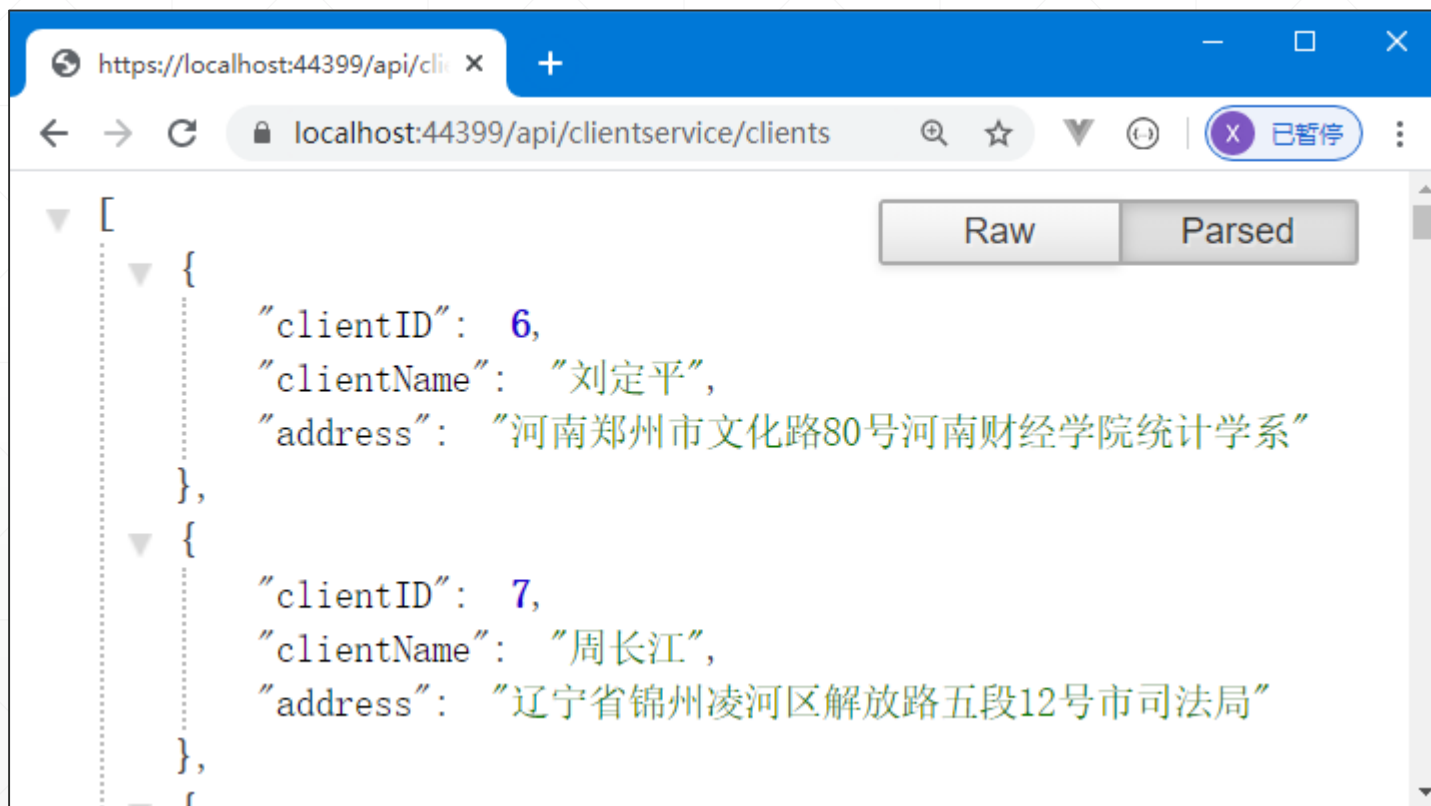
```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllers();
    endpoints.MapGet("/", async context =>
    {
        await context.Response.WriteAsync("Service is ready...");
    });
});
```



注意端口号

测试ClientService控制器工作是否正常

`https://localhost:44399/api/clientservice/clients`





开发Web前端应用

安装前端工具



<https://nodejs.org>



<https://code.visualstudio.com/>

安装yarn



<https://yarn.bootcss.com/>

- 1 打开“命令提示符窗口（或PowerShell）”，安装yarn：

```
npm install -g yarn
```

- 2 验证yarn的安装是否成功：

```
yarn --version
```

```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\Users\JinXuLiang> node --version
v8.11.4
PS C:\Users\JinXuLiang> yarn --version
1.21.1
PS C:\Users\JinXuLiang>
```

安装Vue CLI

1

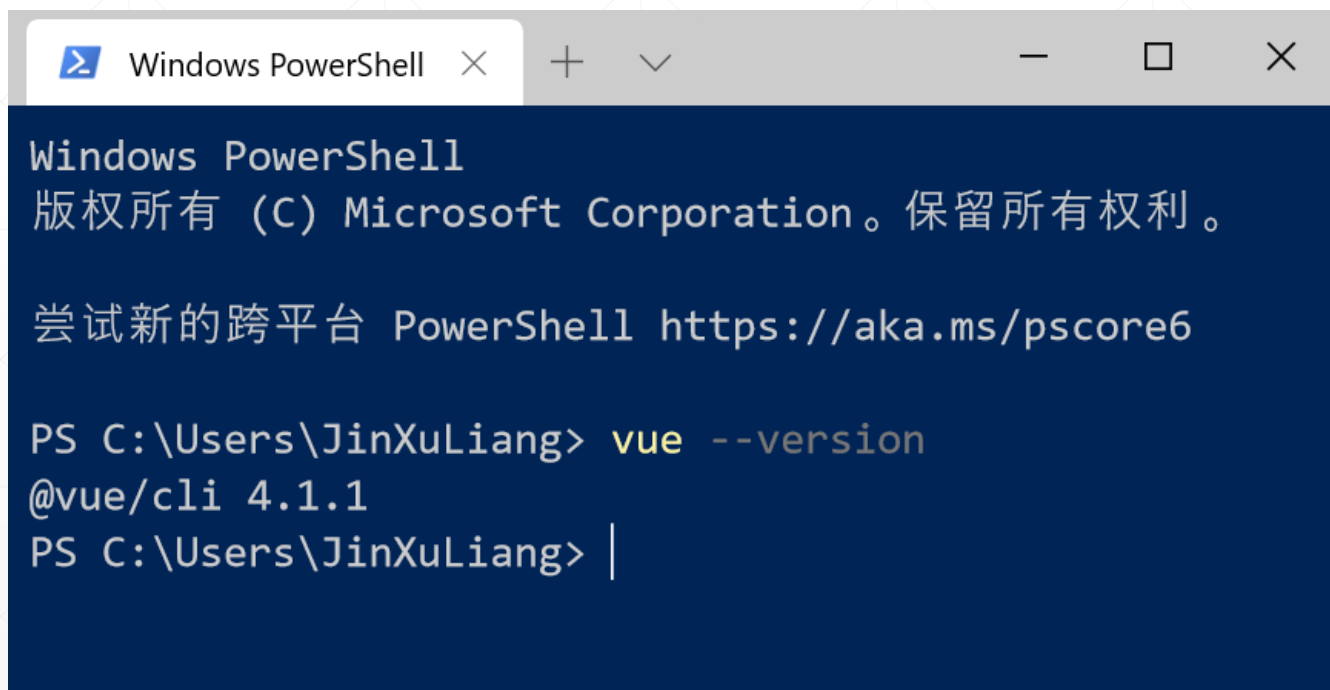
安装Vue-CLI:

```
npm install -g @vue/cli
```

2

验证安装:

```
vue --version
```



A screenshot of a Windows PowerShell terminal window. The title bar shows 'Windows PowerShell' with standard window controls. The terminal text includes the Windows PowerShell logo, copyright information for Microsoft Corporation, a link to the cross-platform PowerShell, and the execution of the 'vue --version' command which returns '@vue/cli 4.1.1'.

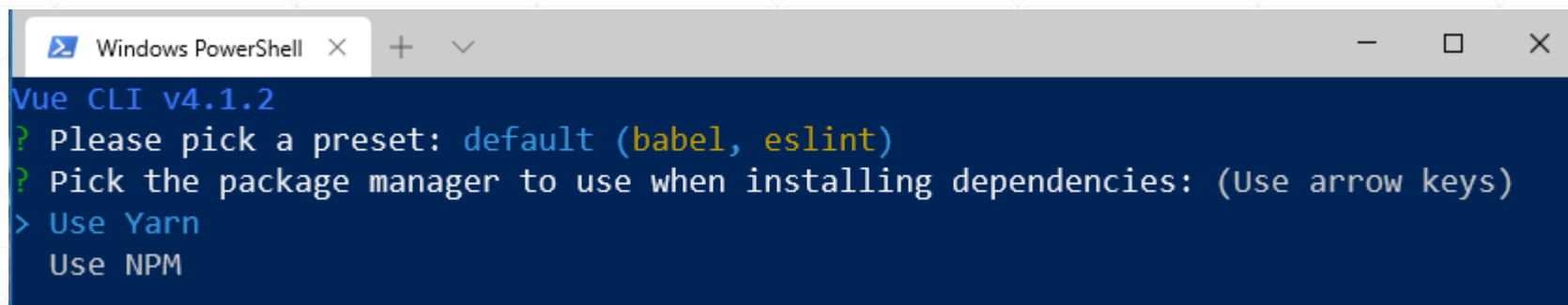
```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\Users\JinXuLiang> vue --version
@vue/cli 4.1.1
PS C:\Users\JinXuLiang> |
```

使用Vue命令行工具创建单独的Web前端项目

```
vue create client_app
```



```
Windows PowerShell  x  +  v  -  □  x
Vue CLI v4.1.2
? Please pick a preset: default (babel, eslint)
? Pick the package manager to use when installing dependencies: (Use arrow keys)
> Use Yarn
  Use NPM
```


启动Web前端应用

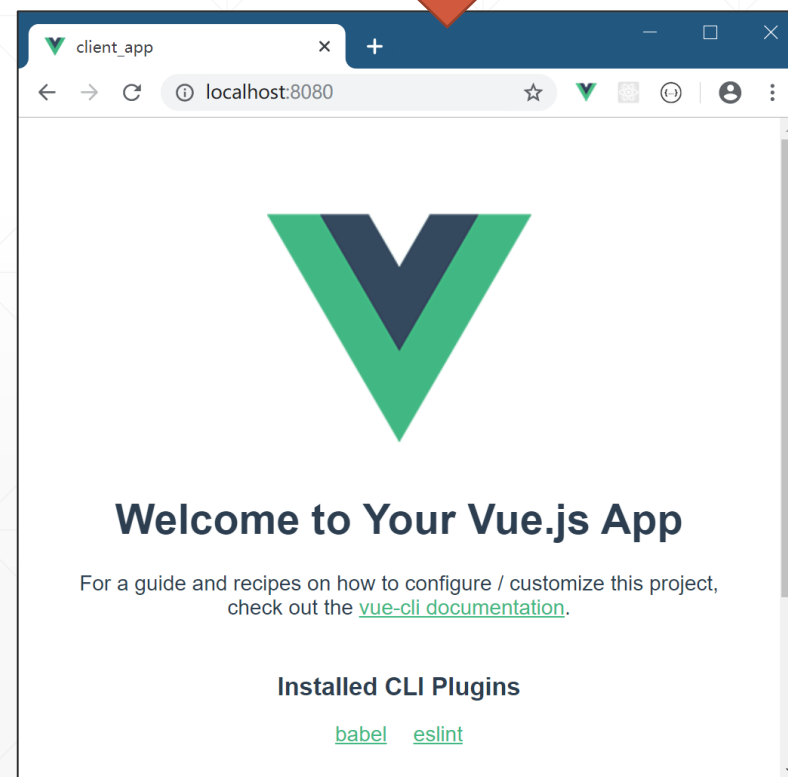
```
Windows PowerShell x + - □ x
info "fsevents@1.2.11" is an optional dependency and failed
compatibility check. Excluding it from installation.
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Saved lockfile.
Done in 9.99s.
Running completion hooks...
Generating README.md...
Successfully created project client_app.
Get started with the following commands:
$ cd client_app
$ yarn serve
PS D:\>
```

启动运行:

yarn serve

```
Windows PowerShell x + - □ x
App running at:
- Local: http://localhost:8080/
- Network: http://192.168.0.107:8080/

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



安装bootstrap

1

先安装jQuery和popper.js:

```
yarn add jquery --save
```

```
yarn add popper.js --save
```

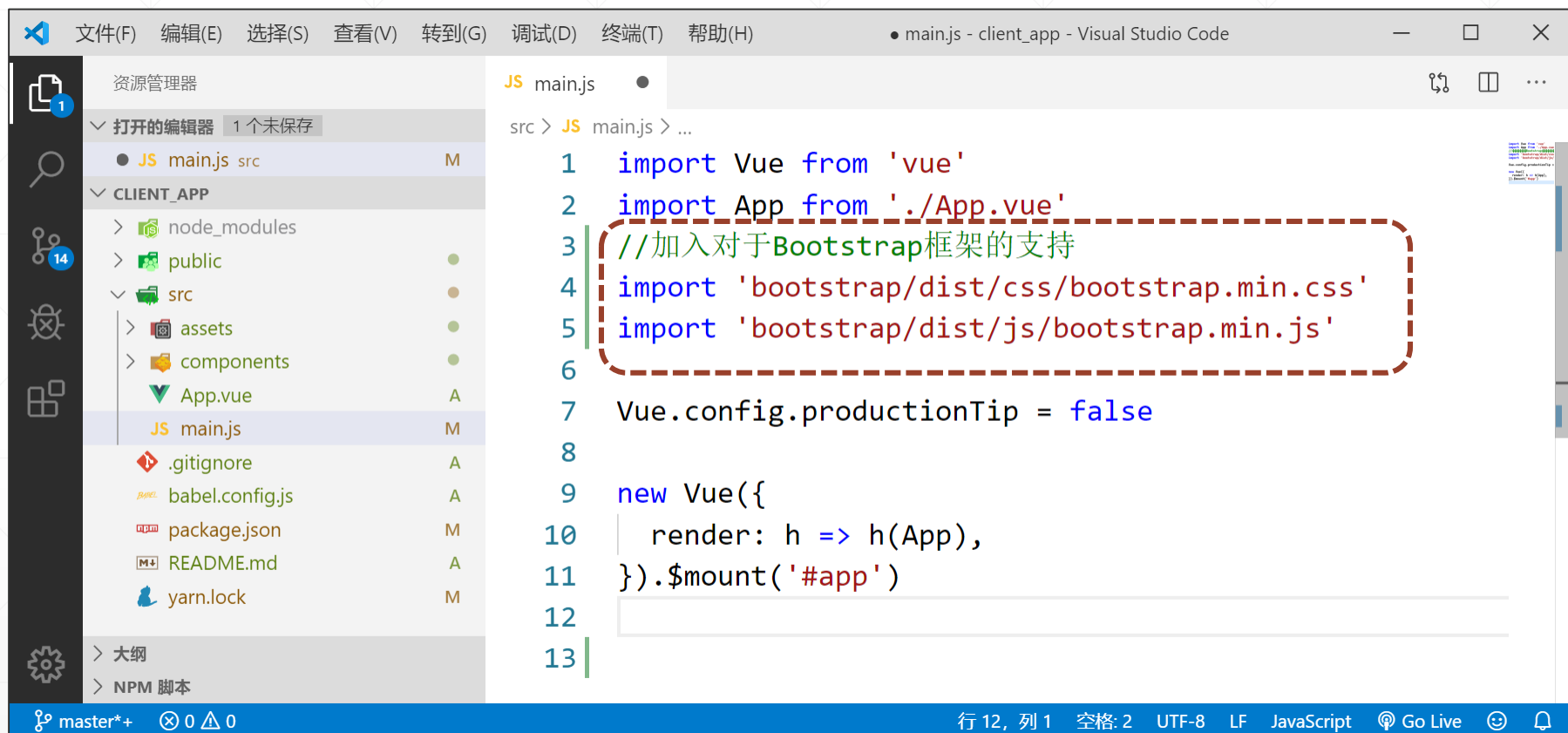
2

再安装bootstrap:

```
yarn add bootstrap --save
```

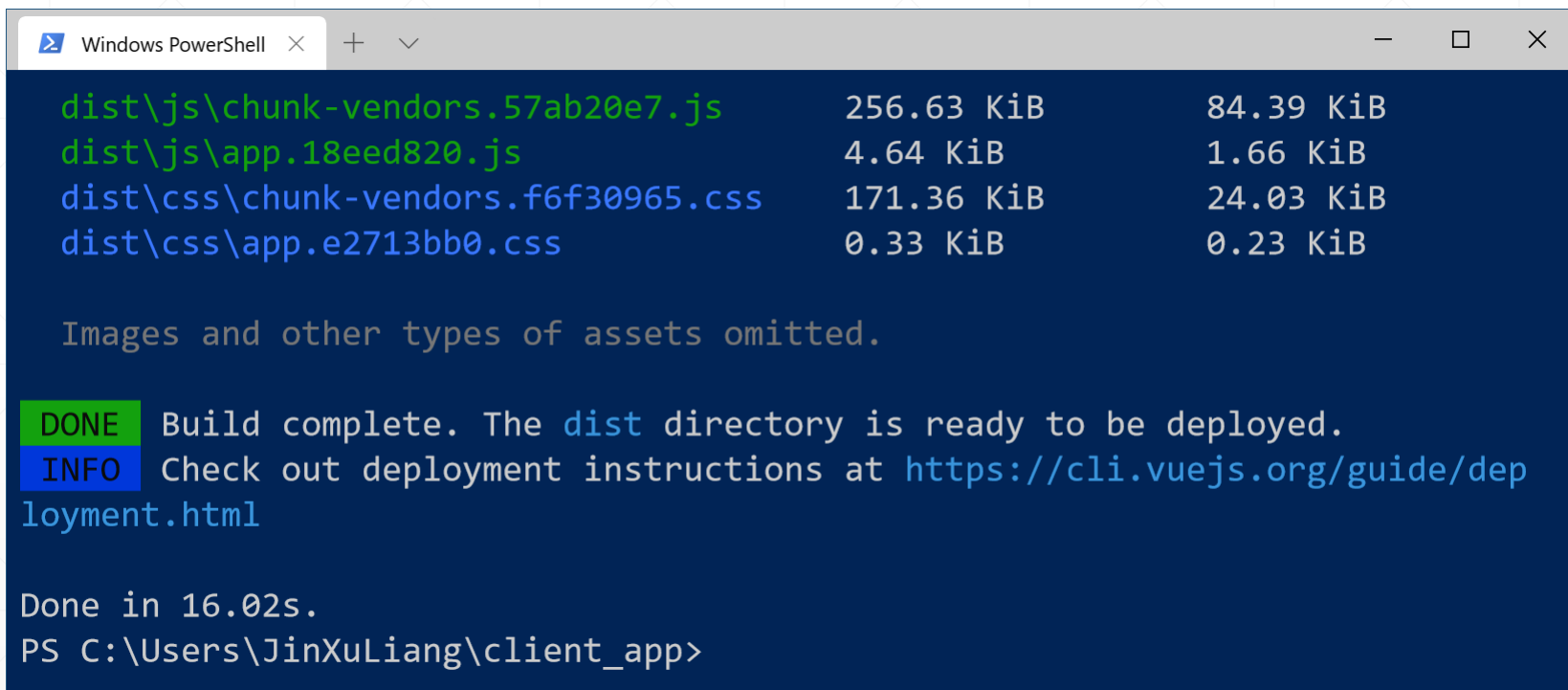


使用VS Code打开项目



尝试编译，看看有无错误.....

yarn build



```
Windows PowerShell  X + v - □ X

dist\js\chunk-vendors.57ab20e7.js      256.63 KiB      84.39 KiB
dist\js\app.18eed820.js                4.64 KiB        1.66 KiB
dist\css\chunk-vendors.f6f30965.css    171.36 KiB      24.03 KiB
dist\css\app.e2713bb0.css              0.33 KiB        0.23 KiB

Images and other types of assets omitted.

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

Done in 16.02s.
PS C:\Users\JinXuLiang\client_app>
```

测试使用bootstrap样式

HelloWorld.vue

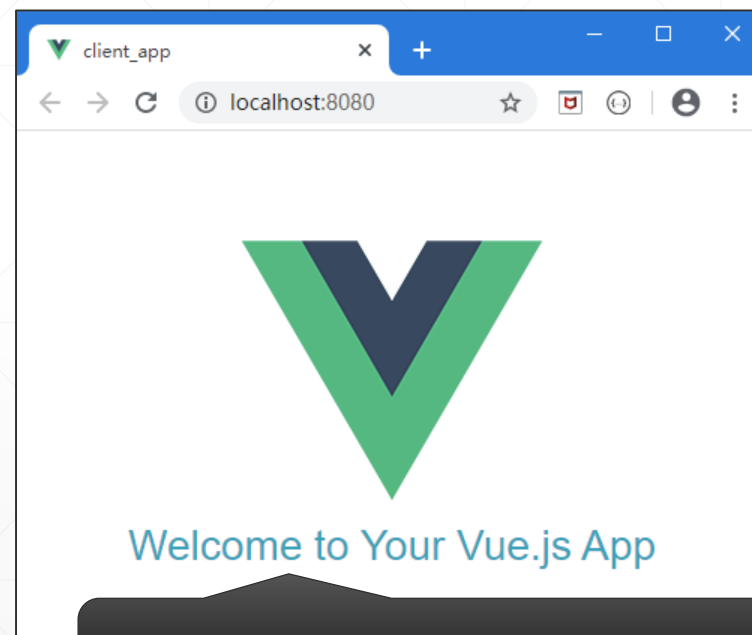
src > components > HelloWorld.vue > {} "HelloWorld.vue" > tem

```
1 <template>
2   <div class="text-center">
3     <h3 class="text-info">{{ msg }}</h3>
4   </div>
5 </template>
6
7 <script>
8 export default {
9   name: 'HelloWorld',
10  props: {
11    msg: String
12  }
13 }
14 </script>
15
16 <style scoped>
17 </style>
18
```

这里使用Bootstrap提供的
的样式

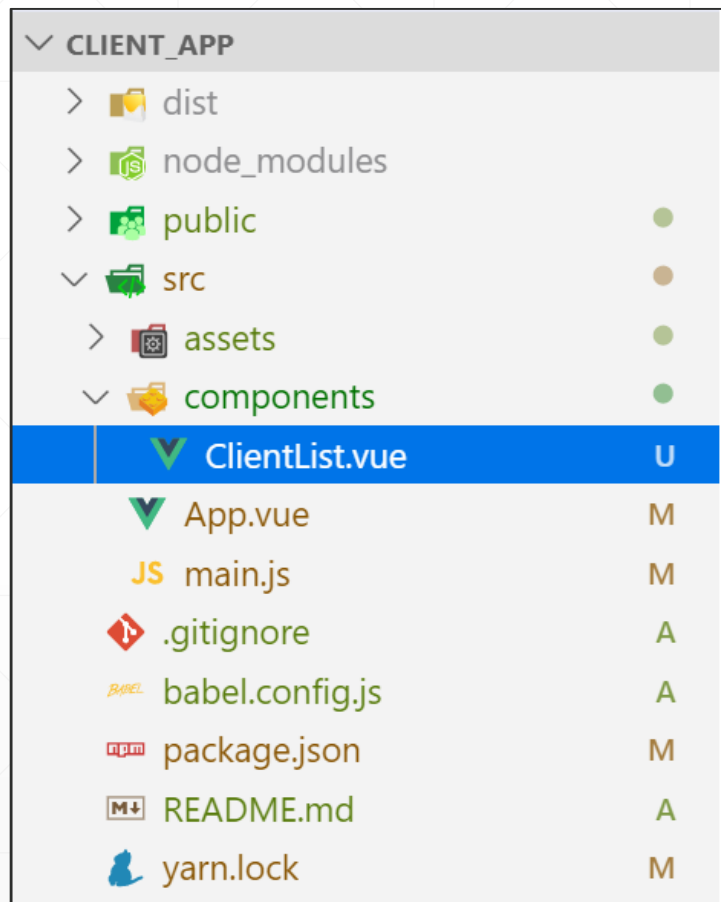
HelloWorld.vue

Ok, 一切正常。



Bootstrap样式起作用了!

创建自己的Vue组件



ClientList.vue X

src > components > ClientList.vue > {} "ClientList.vue"

```
1 <template>
2   <div class="text-center">
3     <h3 class="text-info">{{ message }}</h3>
4   </div>
5 </template>
6
7 <script>
8   export default {
9     name: 'ClientList',
10    data: function(){
11      return {
12        message: '客户清单'
13      }
14    }
15  }
16 </script>
17
18 <style scoped>
19 </style>
```

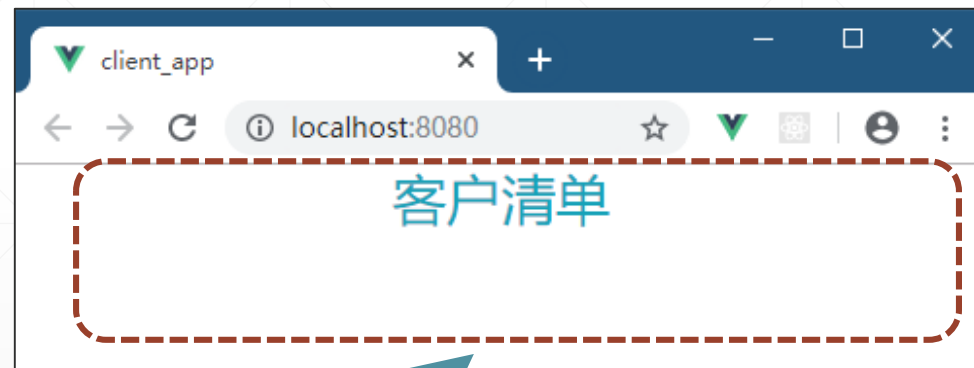
ClientList.vue

修改App组件

```
App.vue x
src > App.vue > {} "App.vue" > style
1 <template>
2   <div id="app">
3     <client-list/>
4   </div>
5 </template>
6
7 <script>
8   import ClientList from './components/ClientList.vue'
9
10  export default {
11    name: 'app',
12    components: {
13      ClientList
14    }
15  }
16 </script>
17
18 <style>
19 </style>
20
```

将ClientList组件插入到这里.....

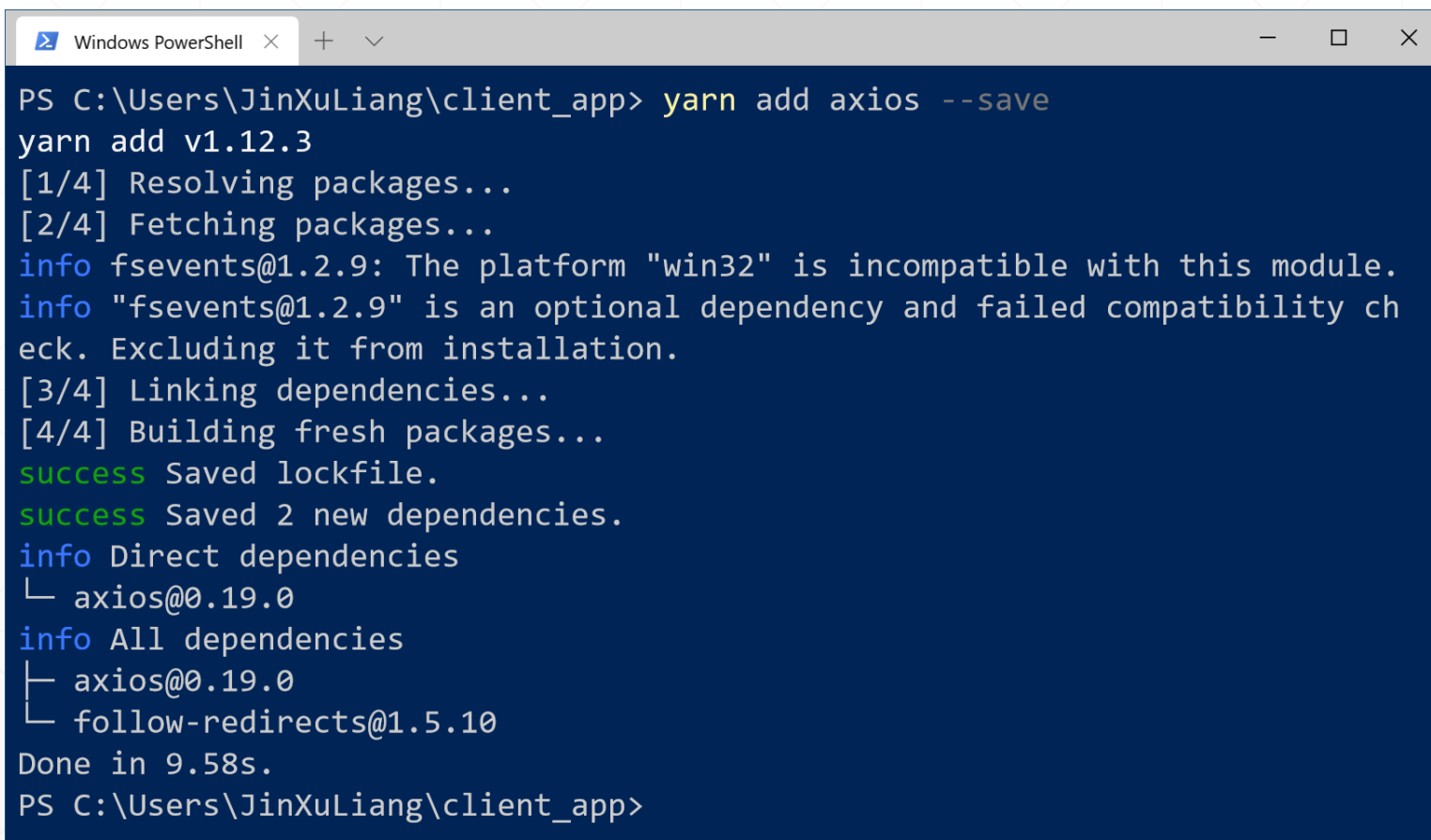
App.vue



ClientList组件所包含的内容

安装axios

```
yarn add axios --save
```



```
Windows PowerShell
PS C:\Users\JinXuLiang\client_app> yarn add axios --save
yarn add v1.12.3
[1/4] Resolving packages...
[2/4] Fetching packages...
info fsevents@1.2.9: The platform "win32" is incompatible with this module.
info "fsevents@1.2.9" is an optional dependency and failed compatibility check. Excluding it from installation.
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Saved lockfile.
success Saved 2 new dependencies.
info Direct dependencies
└─ axios@0.19.0
info All dependencies
├─ axios@0.19.0
└─ follow-redirects@1.5.10
Done in 9.58s.
PS C:\Users\JinXuLiang\client_app>
```


导入axios

```
JS main.js  ×
src > JS main.js > ...
1  import Vue from 'vue'
2  import App from './App.vue'
3  //加入对于Bootstrap框架的支持
4  import 'bootstrap/dist/css/bootstrap.min.css'
5  import 'bootstrap/dist/js/bootstrap.min.js'
6  //导入axios
7  import axios from 'axios'
8
9  Vue.prototype.axios = axios
10 Vue.config.productionTip = false
11
12 new Vue({
13   render: h => h(App),
14 }).$mount('#app')
```

main.js

设计页面

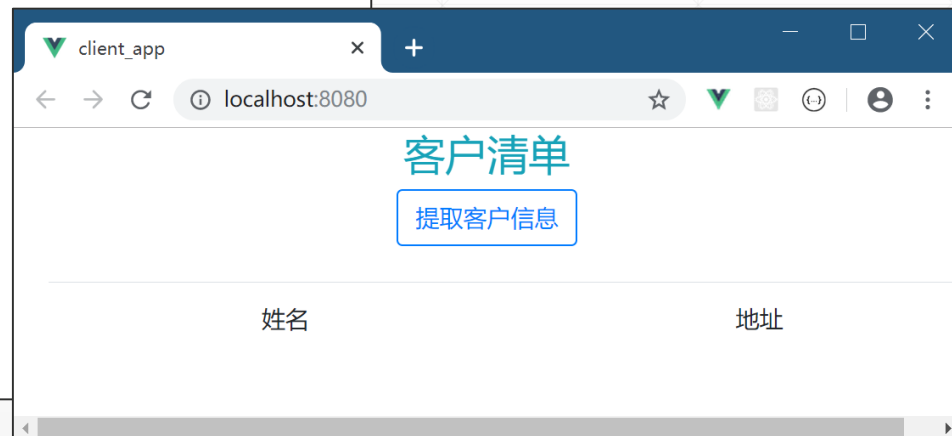
用户点击按钮之后，使用表格显示客户清单。

ClientList.vue

ClientList.vue

src > components > ClientList.vue > {} "ClientList.vue" > template > div.text-center

```
1 <template>
2   <div class="text-center">
3     <h3 class="text-info">{{ message }}</h3>
4     <button class="btn btn-outline-primary" @click="getClient">提取客户信息</button>
5     <table class="table">
6       <thead>
7         <td>姓名</td>
8         <td>地址</td>
9       </thead>
10      <tr v-for="client in clients" :key="client.ClientID">
11        <td>{{client.clientName}}</td>
12        <td>{{client.address}}</td>
13      </tr>
14    </table>
15  </div>
16 </template>
```



如果报告以下错误.....

yarn serve

问题 输出 调试控制台 终端

1: node

98% after emitting CopyPlugin

ERROR Failed to compile with 1 errors

error in ./src/components/ClientList.vue

Module Error (from ./node_modules/eslint-loader/index.js):

error: Unexpected console statement (no-console) at src\components\ClientList.vue:35:11:

```
33 |         })
34 |         .catch(err => {
> 35 |             console.error(err);
      |             ^
36 |         });
37 |     }
38 | }
```

1 error found.



ESLint

package.json

```
{
  "eslintConfig": {
    "root": true,
    "env": {
      "node": true
    },
    "extends": [
      "plugin:vue/essential",
      "eslint:recommended"
    ],
    "rules": {
      ""no-console": "off""
    },
    "parserOptions": {
      "parser": "babel-eslint"
    }
  },
}
```

使用axios提取数据

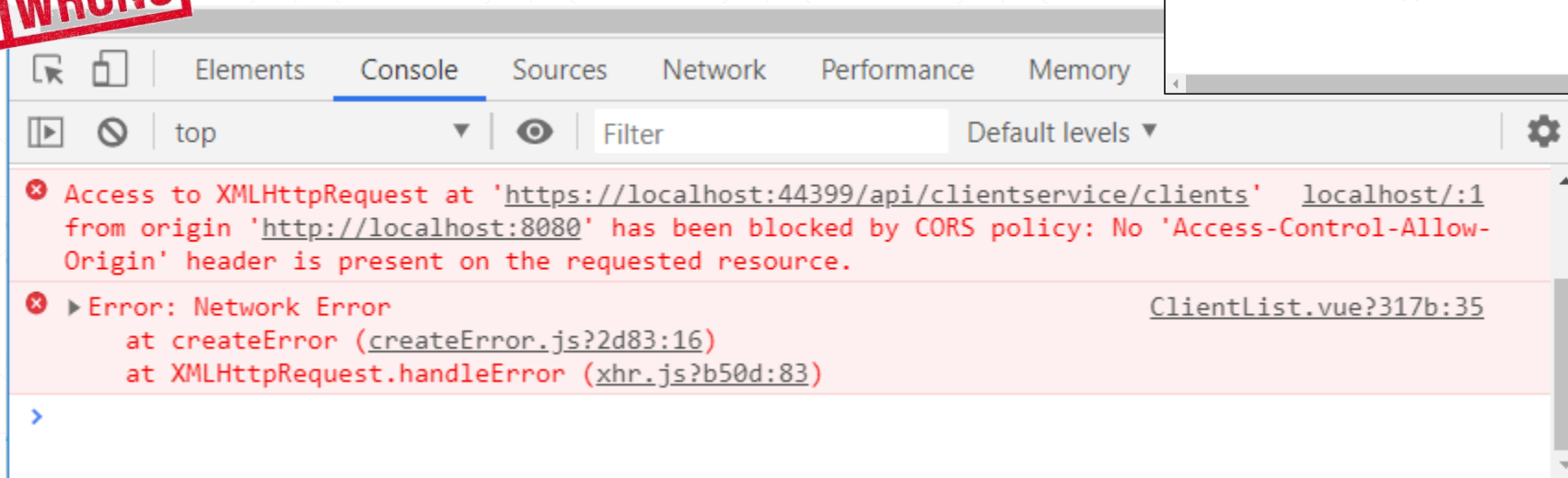
```
<script>
export default {
  name: "ClientList",
  data: function() {
    return {
      message: "客户清单",
      clients: []
    };
  },
  methods: {
    getClient() {
      this.axios
        .get("https://localhost:44399/api/clientservice/clients")
        .then(res => {
          this.clients = res.data;
        })
        .catch(err => {
          console.error(err);
        });
    }
  }
};
</script>
```

使用axios发出HTTP Get
请求，提取客户信息。

要求服务端先运行。

ClientList.vue

提取数据时发生错误.....



原因：由于此Web前端App运行于localhost:8080，却尝试访问后端localhost:44399的数据，所以浏览器阻止了这个访问，这种现象称为“**CORS（跨域请求）拦截**”。

在Server端针对前端Web App启用CORS

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
    RegisterMyTypeIoCContainer(services);
    services.AddDbContext<MyDbContext>(options =>...);
    //启用CORS
    services.AddCors(options =>
    {
        //添加一个CORS策略, 允许本机8080端口的应用访问
        options.AddPolicy("AllowLocalhost", builder =>
        {
            builder.WithOrigins("http://localhost:8080")
                .AllowAnyMethod()
                .AllowAnyMethod();
        });
    });
}
```

```
public void Configure(IApplicationBuilder app,
    IWebHostEnvironment env)
{
    if (env.IsDevelopment())...
    else...

    //针对所有终结点, 启用"AllowLocalhost"策略
    app.UseCors("AllowLocalhost");

    app.UseHttpsRedirection();
    app.UseStaticFiles();
    app.UseRouting();
    app.UseAuthorization();
    app.UseEndpoints(endpoints =>...);
}
```

ASP.NET Core的CORS特性官方文档:

<https://docs.microsoft.com/en-us/aspnet/core/security/cors?view=aspnetcore-3.1>

工作正常的前端应用



由于服务端设置允许8080端口的HTTP请求，所以现在前端应用可以访问到客户清单了。



小结



在本讲中我们展示了一个采用了前后端分离架构的Web应用的开发过程。



在开发Web前端应用时，CORS是一个需要仔细对待的特性。



Web安全是一个非常大的话题，本课程对此不作展开，将放在独立的技术专题课程中介绍。

“.NET Core软件开发技术导论与自学指南”



各种技术专题
课程

本课程主干内容结束。

.NET Core各种具体领域的相
关技术，将作为专题，开设单
独的“技术专题”课程介绍。

主干：本课程