硅谷课堂第三天-前端基础知识

一、前端开发和前端开发工具

1、前端开发介绍

前端工程师“Front-End-Developer”源自于美国。大约从 2005 年开始正式的前端工程师角色被行业所认可，到了 2010 年，互联网开始全面进入移动时代，前端开发的工作越来越重要。

最初所有的开发工作都是由后端工程师完成的，随着业务越来越繁杂，工作量变大，于是我们将项目中的可视化部分和一部分交互功能的开发工作剥离出来，形成了前端开发。

由于互联网行业的急速发展，导致了在不同的国家，有着截然不同的分工体制。

在日本和一些人口比较稀疏的国家，例如加拿大、澳洲等，流行“Full-Stack Engineer”，也就是我们通常所说的全栈工程师。通俗点说就是一个人除了完成前端开发和后端开发工作以外，有的公司从产品设计到项目开发再到后期运维可能都是同一个人，甚至可能还要负责 UI、配动画，也可以是扫地、擦窗、写文档、维修桌椅等等。

而在美国等互联网环境比较发达的国家项目开发的分工协作更为明确，整个项目开发分为前端、中间层和后端三个开发阶段，这三个阶段分别由三个或者更多的人来协同完成。

国内的大部分互联网公司只有前端工程师和后端工程师，中间层的工作有的由前端来完成，有的由后端来完成。

PRD（产品原型-产品经理）、PSD（视觉设计-UI 工程师）、HTML/CSS/JavaScript（PC/移动端网页，实现网页端的视觉展示和交互-前端工程师）

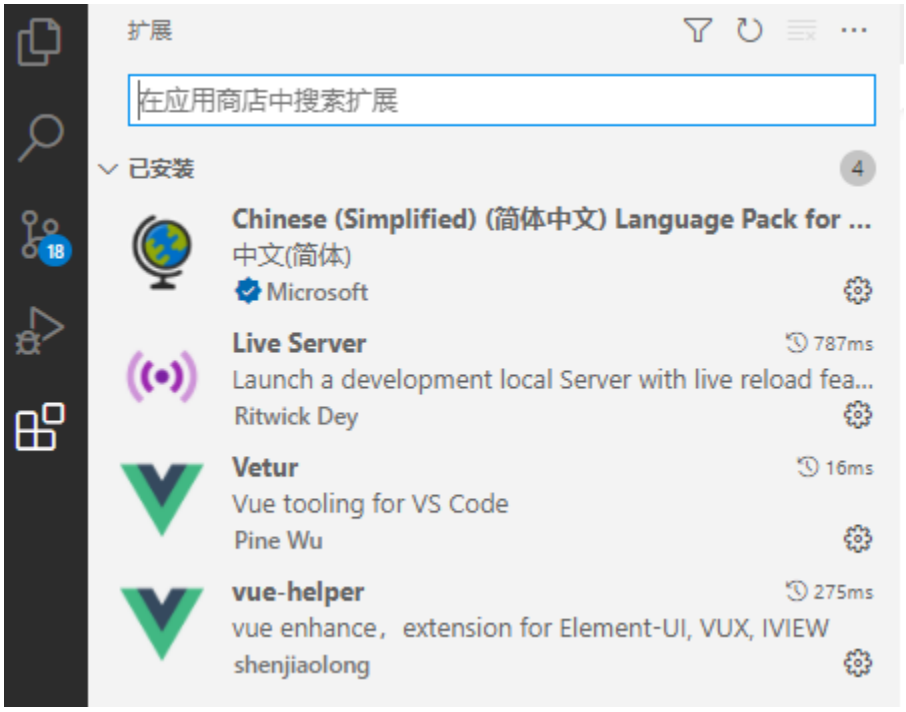
2、下载和安装 VS Code

2.1、下载地址

https://code.visualstudio.com/

2.2、插件安装

为方便后续开发，建议安装如下插件



2.3、创建项目

vscode 本身没有新建项目的选项，所以要先创建一个空的文件夹，如 project_xxxx。

然后打开 vscode，再在 vscode 里面选择 File -> Open Folder 打开文件夹，这样才可以创建项目。

2.4、保存工作区

打开文件夹后，选择“文件 -> 将工作区另存为…”，为工作区文件起一个名字，存储在刚才的文件夹下即可

2.5、新建文件夹和网页

2.6、预览网页

以文件路径方式打开网页预览

需要安装“open in browser”插件：

文件右键 -> Open In Default Browser

以服务器方式打开网页预览

需要安装“Live Server”插件：

文件右键 -> Open with Live Server

2.7、设置字体大小

左边栏 Manage -> settings -> 搜索 “font” -> Font size

二、ECMAScript 6 简介

ECMAScript 6.0（以下简称 ES6）是 JavaScript 语言的下一代标准，已经在 2015 年 6 月正式发布了。它的目标是使得 JavaScript 语言可以用来编写复杂的大型应用程序，成为企业级开发语言。

1、与 JavaScript 的关系

一个常见的问题是，ECMAScript 和 JavaScript 到底是什么关系？

要讲清楚这个问题，需要回顾历史。1996 年 11 月，JavaScript 的创造者 Netscape 公司，决定

将 JavaScript 提交给标准化组织 ECMA，希望这种语言能够成为国际标准。次年，ECMA 发布 262 号标准文件（ECMA-262）的第一版，规定了浏览器脚本语言的标准，并将这种语言称为 ECMAScript，这个版本就是 1.0 版。

因此，ECMAScript 和 JavaScript 的关系是，前者是后者的规格，后者是前者的一种实现（另外的 ECMAScript 方言还有 Jscript 和 ActionScript）

2、与 ECMAScript 2015 的关系

ECMAScript 2015（简称 ES2015）这个词，也是经常可以看到的。它与 ES6 是什么关系呢？

2011 年，ECMAScript 5.1 版发布后，就开始制定 6.0 版了。因此，ES6 这个词的原意，就是指 JavaScript 语言的下一个版本。

ES6 的第一个版本，在 2015 年 6 月发布，正式名称是《ECMAScript 2015 标准》（简称 ES2015）。

2016 年 6 月，小幅修订的《ECMAScript 2016 标准》（简称 ES2016）如期发布，这个版本可以看作是 ES6.1 版，因为两者的差异非常小，基本上是同一个标准。根据计划，2017 年 6 月发布 ES2017 标准。

因此，ES6 既是一个历史名词，也是一个泛指，含义是 5.1 版以后的 JavaScript 的下一代标准，涵盖了 ES2015、ES2016、ES2017 等等，而 ES2015 则是正式名称，特指该年发布的正式版本的语言标准。

三、ES6 基本语法

ES 标准中不包含 DOM 和 BOM 的定义，只涵盖基本数据类型、关键字、语句、运算符、内建对象、内建函数等通用语法。

本部分只学习前端开发中 ES6 的最少必要知识，方便后面项目开发中对代码的理解。

1、let 声明变量

创建 let.html

```
// var 声明的变量没有局部作用域
// let 声明的变量  有局部作用域
{
  var a = 0;
  let b = 1;
}
console.log(a); // 0
console.log(b); // ReferenceError: b is not defined
```

```
// var 可以声明多次
// let 只能声明一次
var m = 1;
var m = 2;
let n = 3;
let n = 4;
console.log(m); // 2
console.log(n); // Identifier 'n' has already been declared
```

2、const 声明常量（只读变量）

创建 const.html

```
// 1、声明之后不允许改变
const PI = "3.1415926";
PI = 3; // TypeError: Assignment to constant variable.
```

```
// 2、一但声明必须初始化，否则会报错
const MY_AGE  // SyntaxError: Missing initializer in const declaration
```

3、解构赋值

创建解构赋值.html

解构赋值是对赋值运算符的扩展。

他是一种针对数组或者对象进行模式匹配，然后对其中的变量进行赋值。

在代码书写上简洁且易读，语义更加清晰明了；也方便了复杂对象中数据字段获取。

```
// 对象解构
let user = { name: "Helen", age: 18 };
// 传统
let name1 = user.name;
let age1 = user.age;
console.log(name1, age1);
// ES6
let { name, age } = user; // 注意：结构的变量必须是user中的属性
console.log(name, age);
```

4、模板字符串

创建模板字符串.html

模板字符串相当于加强版的字符串，用反引号 `,除了作为普通字符串，还可以在字符串中加入变量和表达式。

```
// 字符串插入变量和表达式。变量名写在 ${} 中，${} 中可以放入 JavaScript 表达式。
let name = "Mike";
let age = 27;
let info = `My Name is ${name},I am ${age + 1} years old next year.`;
console.log(info);
// My Name is Mike,I am 28 years old next year.
```

```
// 字符串中调用函数
function f() {
  return "have fun!";
}
let string2 = `Game start,${f()}`;
console.log(string2); // Game start,have fun!
```

5、定义方法简写

创建定义方法简写.html

```
// 传统
const person1 = {
  sayHi: function () {
    console.log("Hi");
  },
};
person1.sayHi(); // "Hi"
// ES6
const person2 = {
  sayHi() {
    console.log("Hi");
  },
};
person2.sayHi(); // "Hi"
```

6、对象拓展运算符

创建对象拓展运算符.html

拓展运算符（...）用于取出参数对象所有可遍历属性然后拷贝到当前对象。

```
// 1、拷贝对象
let person1 = { name: "Amy", age: 15 };
let someone = { ...person1 };
console.log(someone); // {name: "Amy", age: 15}
```

7、箭头函数

创建箭头函数.html

箭头函数提供了一种更加简洁的函数书写方式。基本语法是：`参数 => 函数体`

```
// 传统
var f1 = function (a) {
  return a;
};
console.log(f1(1));
// ES6
var f2 = (a) => a;
console.log(f2(1));
```

```
// 当箭头函数没有参数或者有多个参数，要用（）括起来。
// 当箭头函数函数体有多行语句，用 {} 包裹起来，表示代码块，
// 当只有一行语句，并且需要返回结果时，可以省略 {} ，结果会自动返回。
var f3 = (a, b) => {
  let result = a + b;
  return result;
};
console.log(f3(6, 2)); // 8
// 前面代码相当于：
var f4 = (a, b) => a + b;
```

箭头函数多用于匿名函数的定义

四、Vue

1、Vue.js 是什么

Vue (读音 /vju:/，类似于 view) 是一套用于构建用户界面的渐进式框架。

Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。

官方网站：https://cn.vuejs.org

2、初始 Vue.js

创建 demo.html

```
<!-- id标识vue作用的范围 -->
<div id="app">
  <!-- {{}} 插值表达式，绑定vue中的data数据 -->
  {{ message }}
</div>
<script src="vue.min.js"></script>
<script>
  // 创建一个vue对象
  new Vue({
    el: "#app", // 绑定vue作用的范围
    data: {
      // 定义页面中显示的模型数据
      message: "Hello Vue!",
    },
  });
</script>
```

这就是声明式渲染：Vue.js 的核心是一个允许采用简洁的模板语法来声明式地将数据渲染进 DOM 的系统

这里的核心思想就是没有繁琐的 DOM 操作，例如 jQuery 中，我们需要先找到 div 节点，获取到 DOM 对象，然后进行一系列的节点操作

在 vs code 中创建代码片段：

文件 -> 首选项 -> 用户代码片段 -> 新建全局代码片段或文件夹代码片段：vue-html.code-snippets

注意：制作代码片段的时候，字符串中如果包含文件中复制过来的“Tab”键的空格，要换成“空格键”的空格

```
{
  "vue htm": {
    "scope": "html",
    "prefix": "vuehtml",
    "body": [
      "<!DOCTYPE html>",
      "<html lang=\"en\">",
      "",
      "<head>",
      "    <meta charset=\"UTF-8\">",
      "    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">",
      "    <meta http-equiv=\"X-UA-Compatible\" content=\"ie=edge\">",
      "    <title>Document</title>",
      "</head>",
      "",
      "<body>",
      "    <div id=\"app\">",
      "",
      "    </div>",
      "    <script src=\"vue.min.js\"></script>",
      "    <script>",
      "        new Vue({",
      "            el: '#app',",
      "            data: {",
      "                $1",
      "            }",
      "        })",
      "    </script>",
      "</body>",
      "",
      "</html>"
    ],
    "description": "my vue template in html"
  }
}
```

3、Vue 指令和差值表达式

3.1、基本数据渲染和指令

创建 01-基本数据渲染和指令.html

你看到的 v-bind 特性被称为指令。指令带有前缀 v-

除了使用插值表达式{{}}进行数据渲染，也可以使用 v-bind 指令，它的简写的形式就是一个冒号:

```
data: {
    content: '我是标题',
    message: '页面加载于 ' + new Date().toLocaleString()
}
```

```
<!-- 如果要将模型数据绑定在html属性中，则使用 v-bind 指令
     此时title中显示的是模型数据
-->
<h1 v-bind:title="message">{{content}}</h1>
<!-- v-bind 指令的简写形式：  冒号 (:)  -->
<h1 :title="message">{{content}}</h1>
```

3.2、双向数据绑定

创建 02-双向数据绑定.html

双向数据绑定和单向数据绑定：使用 v-model 进行双向数据绑定

```
data: {
  searchMap: {
    keyWord: "尚硅谷";
  }
}
```

```
<!-- v-bind:value只能进行单向的数据渲染 -->
<input type="text" v-bind:value="searchMap.keyWord" />
<!-- v-model 可以进行双向的数据绑定  -->
<input type="text" v-model="searchMap.keyWord" />
<p>您要查询的是：{{searchMap.keyWord}}</p>
```

3.3、事件

创建 03-事件.html

需求： 点击查询按钮，调用方法

增加 methods 节点 并定义 search 方法

```
data: {

},
methods:{
    search(){
        console.log('search')
    }
}
```

html 中增加 button 和 p

使用 v-on 进行数件处理，v-on:click 表示处理鼠标点击事件，事件调用的方法定义在 vue 对象声明的 methods 节点中

```
<!-- v-on 指令绑定事件，click指定绑定的事件类型，事件发生时调用vue中methods节点中定义的方法 -->
<button v-on:click="search()">查询</button>
```

简写

```
<!-- v-on 指令的简写形式 @ -->
<button @click="search()">查询</button>
```

3.4、条件渲染

创建 04-条件渲染.html

v-if：条件指令

```
data: {
  ok: false;
}
```

注意：单个复选框绑定到布尔值

```
<input type="checkbox" v-model="ok" />同意许可协议
<!-- v:if 条件指令：还有v-else、v-else-if ，要注意性能开销 -->
<h1 v-if="ok">这是一个通用的中文占位文本短语，用于表示占位文本的内容。</h1>
<h1 v-else>no</h1>
```

3.5、列表渲染

创建 05-列表渲染.html

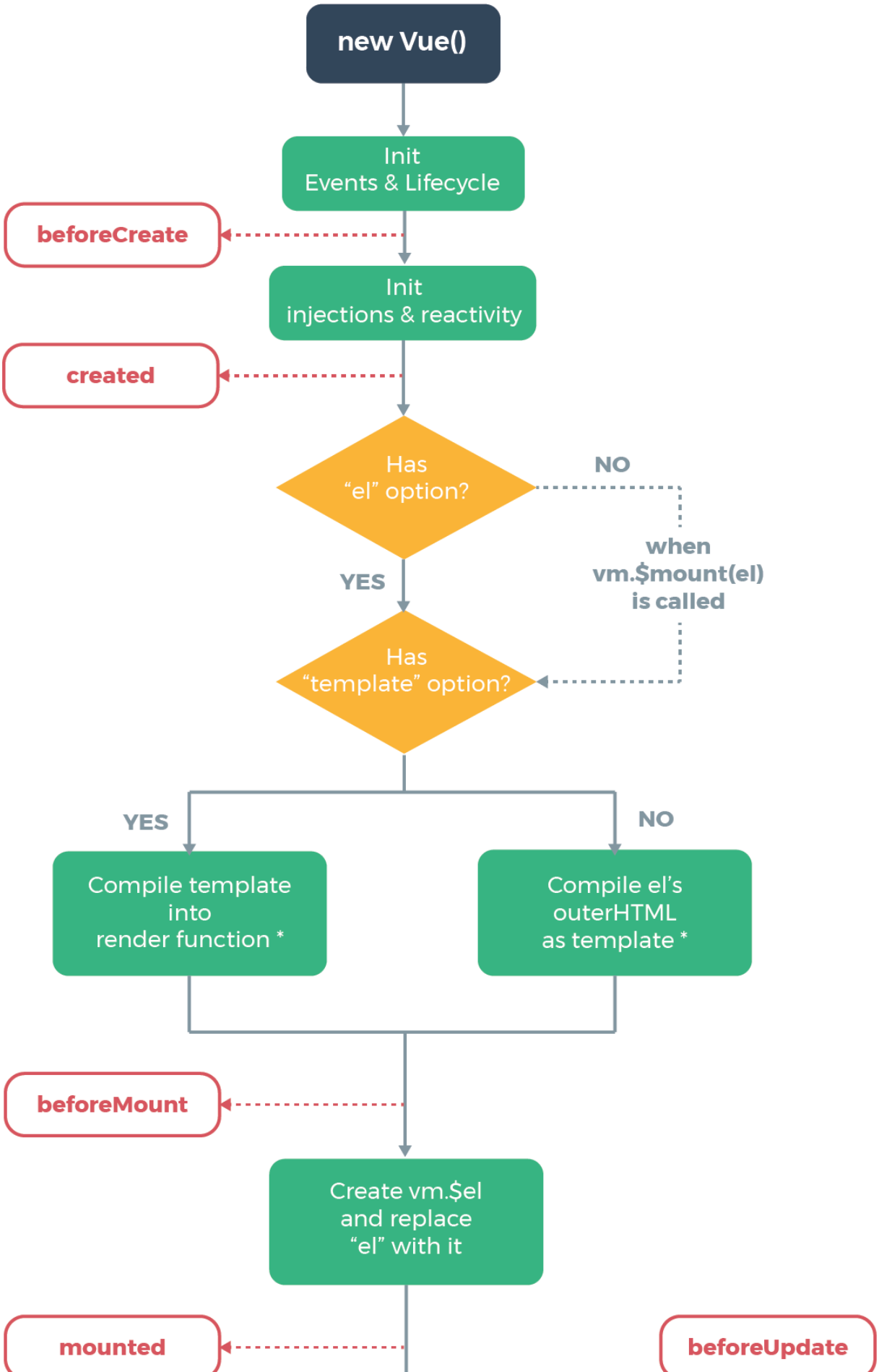
v-for：列表循环指令

例 1：遍历数据列表

```
data: {
  userList: [
    { id: 1, username: "helen" },
    { id: 2, username: "peter" },
    { id: 3, username: "andy" },
  ];
}
```

```
<!-- 遍历数据列表 -->
<table border="1">
  <!-- <tr v-for="item in userList"></tr> -->
  <tr v-for="(item, index) in userList">
    <td>{{index}}</td>
    <td>{{item.id}}</td>
    <td>{{item.username}}</td>
  </tr>
</table>
```

4、Vue 生命周期


new Vue()
Init
Events & Lifecycle
beforeCreate
Init
injections & reactivity
created
Has
“el” option?
NO
when
vm.$mount(el)
is called
YES
Has
“template” option?
YES
NO
Compile template
into
render function *
Compile el’s
outerHTML
as template *
beforeMount
Create vm.$el
and replace
“el” with it
mounted
beforeUpdate

创建 vue 实例的生命周期.html

```
<h3 id="h3">{{ message }}</h3>
```

分析生命周期相关方法的执行时机

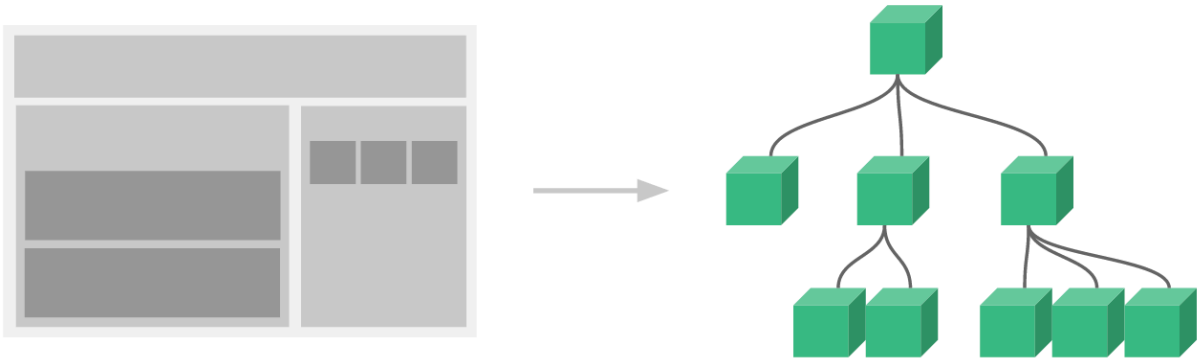
```
data: {
    message: '床前明月光'
},
created() { // 在渲染之前执行
    console.log("created....")
},
mounted() { // 在渲染之后执行
    console.log("mounted....")
},
methods: {

}
```

5、Vue 组件

组件（Component）是 Vue.js 最强大的功能之一。

组件可以扩展 HTML 元素，封装可重用的代码。组件系统让我们可以用独立可复用的小组件来构建大型应用，几乎任意类型的应用的界面都可以抽象为一个组件树



5.1、定义组件

```
var app = new Vue({
  el: "#app",
  // 定义局部组件，这里可以定义多个局部组件
  components: {
    // 组件的名字
    Navbar: {
      // 组件的内容
      template: "<ul><li>首页</li><li>学员管理</li></ul>",
    },
  },
});
```

5.2、使用组件

```
<div id="app">
  <Navbar></Navbar>
</div>
```

6、路由

Vue.js 路由允许我们通过不同的 URL 访问不同的内容。

通过 Vue.js 可以实现多视图的单页 Web 应用（single page web application，SPA）。

Vue.js 路由需要载入 vue-router 库

创建 路由.html

6.1、引入 js

```
<script src="vue.min.js"></script>
<script src="vue-router.min.js"></script>
```

6.2、编写 html

```
<div id="app">
  <h1>Hello App!</h1>
  <p>
    <!-- 使用 router-link 组件来导航 -->
    <!-- 通过传入 to 属性指定链接 -->
    <!-- <router-link> 默认会被渲染成一个 <a> 标签 -->
    <router-link to="/">首页</router-link>
    <router-link to="/student">会员管理</router-link>
    <router-link to="/teacher">讲师管理</router-link>
  </p>
  <!-- 路由出口 -->
  <!-- 路由匹配到的组件将渲染在这里 -->
  <router-view></router-view>
</div>
```

6.3、编写 js

```
<script>
  // 1、定义（路由）组件
  // 可以从其他文件 import 进来
  const Welcome = { template: "<div>欢迎</div>" };
  const Student = { template: "<div>student list</div>" };
  const Teacher = { template: "<div>teacher list</div>" };
  // 2、定义路由
  // 每个路由应该映射一个组件
  const routes = [
    { path: "/", redirect: "/welcome" }, // 设置默认指向的路径
    { path: "/welcome", component: Welcome },
    { path: "/student", component: Student },
    { path: "/teacher", component: Teacher },
  ];
  // 3、创建 router 实例，然后传 routes 配置
  const router = new VueRouter({
    routes, // 缩写，相当于 routes: routes
  });
  // 4、创建和挂载根实例
  // 从而让整个应用都有路由功能
  const app = new Vue({
    el: "#app",
    router,
  });
  // 现在，应用已经启动了
</script>
```

7、axios

axios 是独立于 vue 的一个项目，基于 promise 用于浏览器和 node.js 的 http 客户端

- 在浏览器中可以帮助我们完成 ajax 请求的发送
- 在 node.js 中可以向远程接口发送请求

7.1、获取数据

```
<script src="vue.min.js"></script>
<script src="axios.min.js"></script>
```

```
var app = new Vue({
  el: "#app",
  data: {
    memberList: [], // 数组
  },
  created() {
    this.getList();
  },
  methods: {
    getList(id) {
      // vm = this
      axios
        .get("data.json")
        .then((response) => {
          console.log(response);
          this.memberList = response.data.data.items;
        })
        .catch((error) => {
          console.log(error);
        });
    },
  },
});
```

控制台查看输出

7.2、显示数据

```
<div id="app">
    <table border="1">
        <tr>
            <td>id</td>
            <td>姓名</td>
        </tr>
        <tr v-for="item in memberList">
            <td>{{item.memberId}}</td>
            <td>{{item.nickname}}</td>
        </td>
    </tr>
</table>
</div>
```

8、element-ui

element-ui 是饿了么前端出品的基于 Vue.js 的 后台组件库，方便程序员进行页面快速布局和构建

官网：http://element-cn.eleme.io/#/zh-CN

ui 相关组件我们在项目中学习

五、Node.js

1、Node.js 简介

1.1、什么是 Node.js

简单的说 Node.js 就是运行在服务端的 JavaScript。

Node.js 是一个事件驱动 I/O 服务端 JavaScript 环境，基于 Google 的 V8 引擎，V8 引擎执行 Javascript 的速度非常快，性能非常好。

1.2、Node.js 有什么用

如果你是一个前端程序员，你不懂得像 PHP、Python 或 Ruby 等动态编程语言，然后你想创建自己的服务，那么 Node.js 是一个非常好的选择。

Node.js 是运行在服务端的 JavaScript，如果你熟悉 Javascript，那么你将会很容易的学会 Node.js。

当然，如果你是后端程序员，想部署一些高性能的服务，那么学习 Node.js 也是一个非常好的选择。

2、Node.js 安装

2.1、下载

官网：https://nodejs.org/en/

中文网：http://nodejs.cn/

LTS：长期支持版本

Current：最新版

2.2、安装和查看版本

```
node -v
```

3、简单入门

创建 01-控制台程序.js

```
console.log("Hello Node.js");
```

进入到程序所在的目录，输入

```
node 01-控制台程序.js
```

浏览器的内核包括两部分核心：

- DOM 渲染引擎；
- js 解析器（js 引擎），
- js 运行在浏览器中的内核中的 js 引擎内部。

Node.js 是脱离浏览器环境运行的 JavaScript 程序，基于 V8 引擎（Chrome 的 JavaScript 的引擎）