软件工程实验

Vue框架讲解

2021

小提示：

本PPT针对的是从未接触过Vue框架的同学，力图帮助同学们确定要学习的内容大纲。因此针对各项技术，本PPT均以提及为主，无法深入介绍该技术的各项应用。

实际开发中会用到各种各样的知识，这里建议有针对的，带着需求去学习。但总的来说，在Vue部分你需要使用的工具应该都会在本视频中提及。

另外，针对没讲到的CSS部分，推荐几个大家一定要了解的知识点：盒子模型，display属性以及flex布局。我在CSS资料中放入了一些之前做培训时用到的md文档，可以简单参考下。


WHY VUE.JS?

目录

01. Vue基础

02. Vue项目的构建

03. Element-UI

04. Homework

01.

Vue基础

Vue实例注册

```
<div id="app">
  <h2> {{product}} are in stock.</h2>
</div>
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
  const app = new Vue({
    el: '#app',
    data: {
      product: 'Boots'
    }
  })
</script>
```

1. 引入Vue库

2. 创建Vue对象，注册名为app的元素

3. 这样就可以引用这里的data了

条件渲染

```
<div id="app">
  <!-- 条件渲染（v-else、v-else-if自行扩充） -->
  <div v-if="seen">因为seen等于{{seen}}所以显示</div>
  <div v-else>展示else部分</div>
</div>
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></
script>
<script>
  const app = new Vue({
    el: '#app',
    data: {
      product: 'Boots',
      seen: true,
    }
  })
</script>
```

循环渲染

```
<div id="app">
  <ul>
    <li v-for="item in productList">
      {{item.name}} 剩余 {{item.num}}
    </li>
  </ul>
</div>
```

```
<script>
  const app = new Vue({
    el: '#app',
    data: {
      productList: [{
          name: "Boots",
          num: 1
        },
        {
          name: "Jacket",
          num: 2
        },
        {
          name: "Hiking Socks",
          num: 3
        }
      ]
    }
  })
</script>
```

双向绑定

```
<div id="app">
  <h2> {{product}} are in stock.</h2>
  <!-- 双向绑定 -->
  <input type="text" v-model="product" />
</div>
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
  const app = new Vue({
    el: '#app',
    data: {
      product: 'Boots'
    }
  })
</script>
```

监听事件——内联js

```
<div id="app">
  <ul>
    <li v-for="item in productList">
      {{item.name}} 剩余 {{item.num}}
       <!-- 监听事件 -->
      <button v-on:click="item.num++">Add</button>
    </li>
  </ul>
</div>
```

```
<script>
  const app = new Vue({
    el: '#app',
    data: {
      productList: [{
          name: "Boots",
          num: 1
        },
        {
          name: "Jacket",
          num: 2
        },
        {
          name: "Hiking Socks",
          num: 3
        }
      ]
    }
  })
</script>
```

监听事件
——绑定方法

```
<div id="app">
  <!-- Vue实例 -->
  <h2>{{product}} are in stock.</h2>
  <!-- 双向绑定（input或textarea或单选多选...） -->
  <input type="text" v-model="product" />
  <!-- 监听事件 -->
  <button v-on:click="reset">清空</button>
</div>
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script>
  const app = new Vue({
    el: '#app',
    data: {
      product: 'Boots',
    },
    methods: {
      reset() {
        this.product = ""
      }
    }
  })
</script>
```

绑定事件可以传参。
如果不写传参的话默认传入events

02.

Vue项目的构建

安装与项目创建

依赖的环境：　Vue CLI 4.x 需要 Node.js v8.9 或更高版本 (推荐 v10 以上)。

安装指令：　npm install -g @vue/cli

检查指令：　vue --version　　（出现@vue/cli 4.x就可以了）

创建项目：　vue create 项目名

【创建项目时需要配置，可以等视频也可以直接自行配置一个有babel、vuex、router的项目，不建议选择Linter】

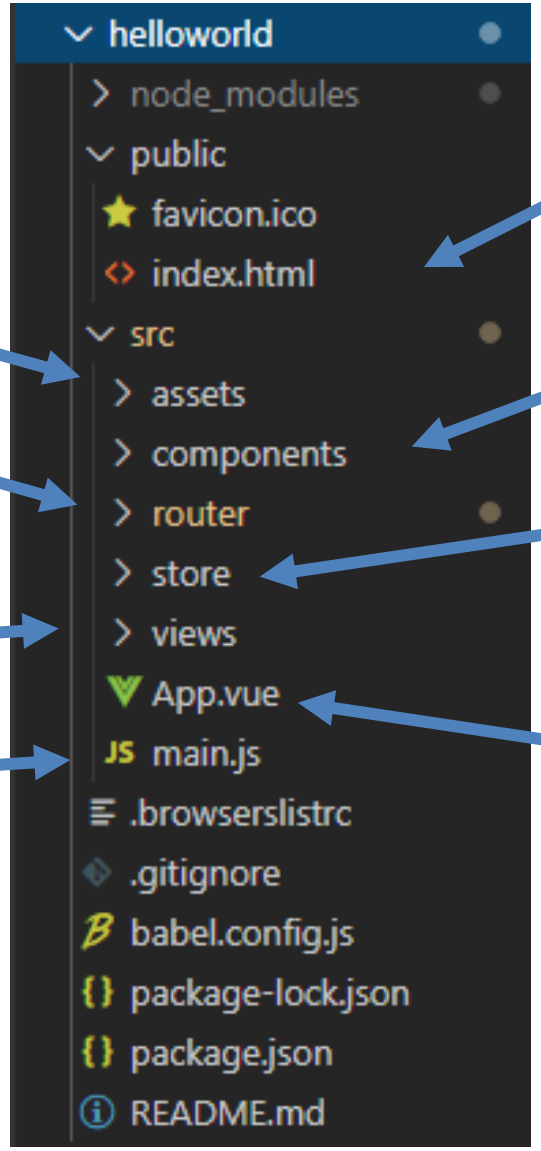
关于Vue-cli

运行项目：　　npm run serve

安装依赖：　　npm install xxx　　（对应依赖的文档中一般有注明具体指令）

打包项目：　　npm run build　　（用于部署项目，详情可自行百度：vue项目部署）

关于Vue-cli



唯一的html文件，
head相关设置在这里修改

静态资源存放处
一般包括图片、外部CSS等

全局组件位置
（详见后续组件化）

Vue Router
（Vue项目中的路由管理）

状态管理模式
（多界面数据同步）

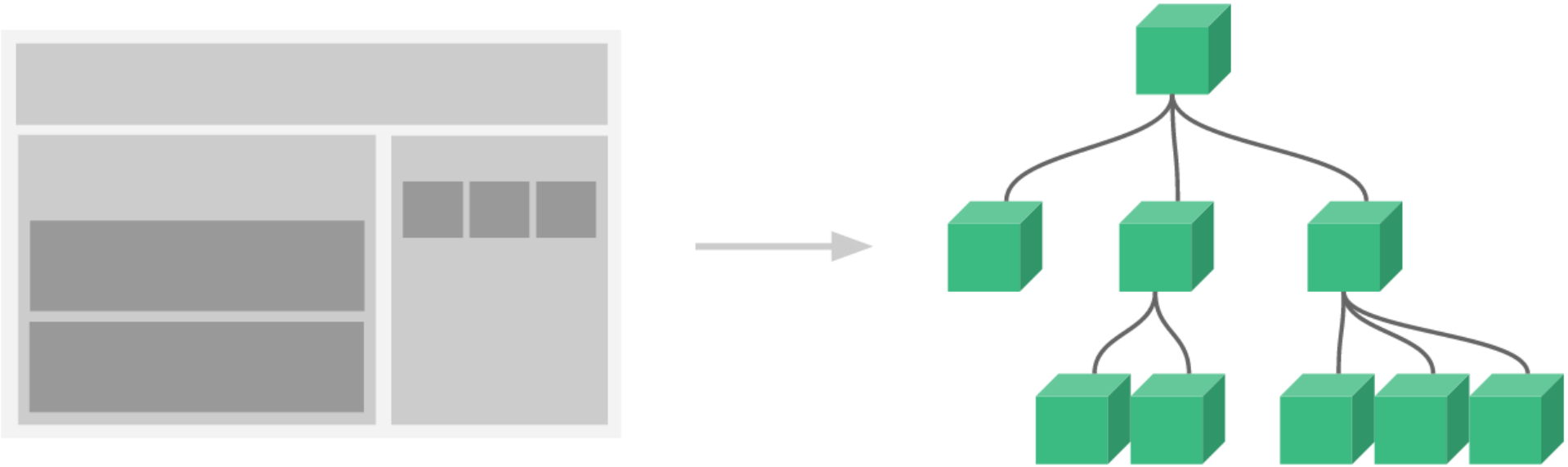
页面组件存放位置

Vue示例创建位置
有后续新的外部库安装也需要在这里注册

全局的App.vue
Vue单页面应用的基础
所有其他（view）中的组件
都挂载在它下面

组件与单页面组件

组件是可复用的 Vue 实例，通常页面由多个组件构成，我们以树的形式组织他们。



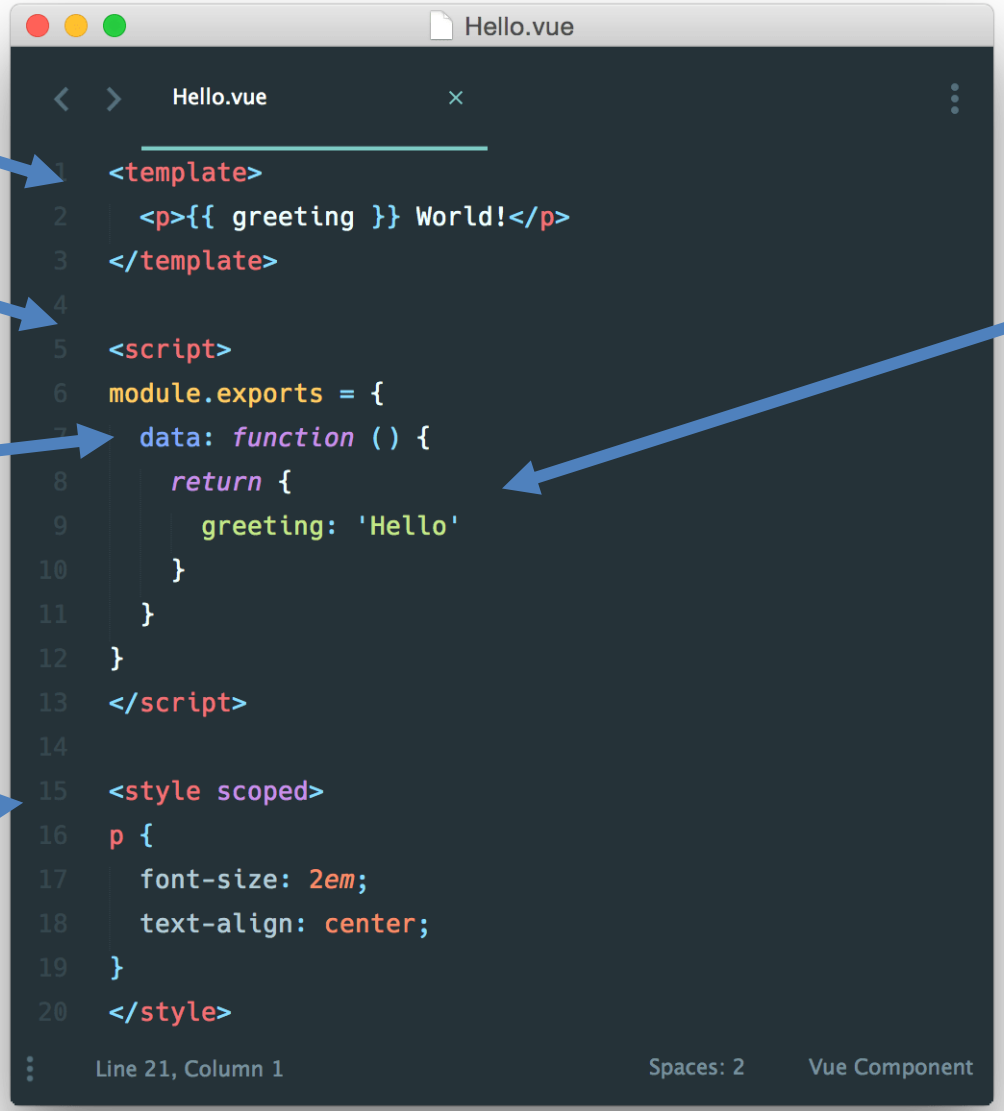
组件与单页面组件

将html部分包裹在template内
注意只能暴露一个根element

JS部分

这里data一定要
用函数的形式
为了复用后每个组件的data独立

CSS部分



组件内部可以使用Vue的一切：
data，methods，生命周期，
component...

Vue-router的引入

引用Vue对象

/router/index.js

```
import Vue from "vue";
import VueRouter from "vue-router";
import Home from "../views/Home.vue";

const About = () => import("../views/About.vue");

Vue.use(VueRouter);
```

挂载到Vue对象

/main.js

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'

Vue.config.productionTip = false

new Vue({
  router,
  store,
  render: h => h(App)
}).$mount('#app')
```

Vue-router的使用

/router/index.js

```
import Vue from "vue";
import VueRouter from "vue-router";
import Home from "../views/Home.vue";

const About = () => import("../views/About.vue");

const routes = [
  {
    path: "/",
    name: "Home",
    component: Home,
  },
  {
    path: "/about",
    name: "About",
    component: About,
  },
];

const router = new VueRouter({
  mode: "history",
  base: process.env.BASE_URL,
  routes,
});
```

直接引用

引用页面组件

懒加载
（推荐）

路由编辑

路径

名字

注册组件路由

history跳转模式

Vue-router的使用

```
<div id="app">
  <div id="nav">
    <!-- 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="/">Home</router-link> |
    <router-link to="/about">About</router-link>
  </div>
  <!-- 路由出口(必备)  -->
  <!-- 路由匹配到的组件将渲染在这里 -->
  <router-view />
</div>
```

Vue-router的使用

```vue
<template>
  <div id="app">
    <div @click="toAbout">About</div>
    <router-view />
  </div>
</template>

<script>
export default {
  name: "App",
  methods: {
    toAbout() {
      this.$router.push({ path: "/about" });
    },
  },
};
</script>
```

Vue-x的使用

/store/index.js

```
import Vue from "vue";
import Vuex from "vuex";

Vue.use(Vuex);

export default new Vuex.Store({
  state: {
    count: 0,
  },
  mutations: {
    increment(state) {
      state.count++;
    },
  },
  actions: {},
  modules: {},
});
```

1. 在全局状态中注册

2. 生成一个mutation来改变它

3. store.state.count可以引用该状态
4. store.commit('increment')可以调用mutation

Vue-x的引入

引用Vue对象

/store/index.js

```
import Vue from "vue";
import Vuex from "vuex";

Vue.use(Vuex);
```

/main.js

为 Vue 实例提供创建好的 store

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'

Vue.config.productionTip = false

new Vue({
  router,
  store,
  render: h => h(App)
}).$mount('#app')
```

现在在所有vue的组件中：

3. this.$store.state.count可以引用该状态
4. this.$store.commit('increment')可以调用mutation

03.

Element-UI

关于Element-UI

安装与引入

安装指令：　　npm i element-ui -S

引入方法：

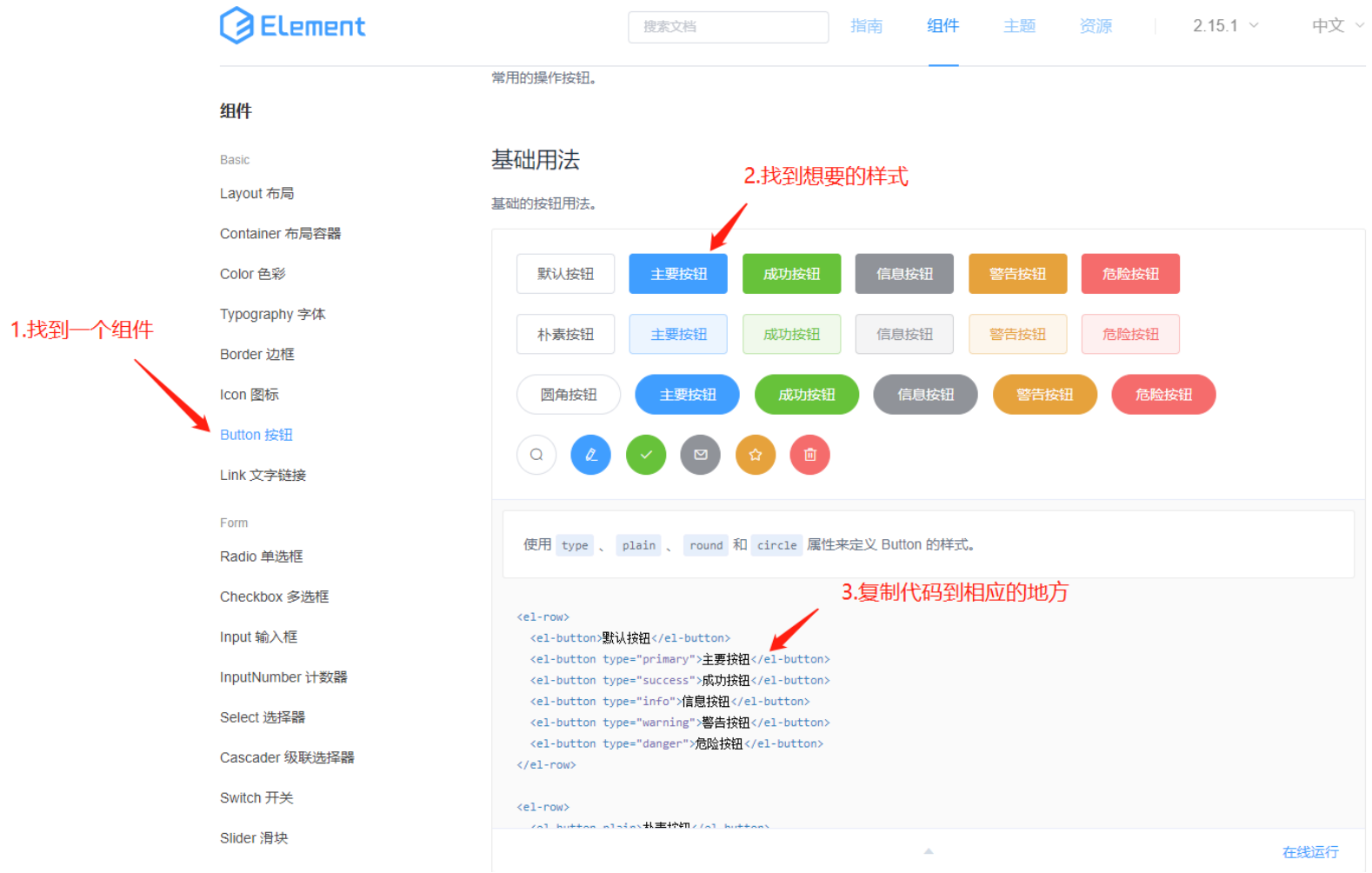
```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'

import ElementUI from 'element-ui';
import 'element-ui/lib/theme-chalk/index.css';
Vue.use(ElementUI);

Vue.config.productionTip = false

new Vue({
  router,
  store,
  render: h => h(App)
}).$mount('#app')
```

使用


Element
搜索文档
指南
组件
主题
资源
2.15.1
中文
组件
Basic
Layout 布局
Container 布局容器
Color 色彩
Typography 字体
Border 边框
Icon 图标
Button 按钮
Link 文字链接
Form
Radio 单选框
Checkbox 多选框
Input 输入框
InputNumber 计数器
Select 选择器
Cascader 级联选择器
Switch 开关
Slider 滑块
常用的操作按钮。
基础用法
基础的按钮用法。
2.找到想要的样式
默认按钮
主要按钮
成功按钮
信息按钮
警告按钮
危险按钮
朴素按钮
圆角按钮
使用 type 、 plain 、 round 和 circle 属性来定义 Button 的样式。
3.复制代码到相应的地方
<el-row>
<el-button>默认按钮</el-button>
<el-button type="primary">主要按钮</el-button>
<el-button type="success">成功按钮</el-button>
<el-button type="info">信息按钮</el-button>
<el-button type="warning">警告按钮</el-button>
<el-button type="danger">危险按钮</el-button>
</el-row>
<el-row>
在线运行
1.找到一个组件

SP.

Http请求

前后端交互

主要行为是后端书写接口文档，与前端约定数据格式，前端发送http请求，获取数据加载到页面

发送请求的方式可以有：ajax，jQuery-ajax，axios.....

学习参考

- 前后端交互axios的使用和封装（P143-147）：https://www.bilibili.com/video/BV15741177Eh（其他内容可能有些老，但是想看的也可以看）
- vue官方文档https://cn.vuejs.org/
- mdn-web文档https://developer.mozilla.org/zh-CN/