**Vue.js-Day03**

# 课程介绍

* **1. 组件通信(掌握)。**
* 2. Slot插槽（理解）。
* **3. Element-ui（掌握）。**

# 组件通信【掌握】

在实际的项目开发过程中，很多业务逻辑里面，vue组件之间需要相互**传递数据**，传递数据就是组件通信。

## 父组件传给子组件【掌握】

### 父传子语法：

**>> 1. 父组件使用v-bind绑定动态数据到子组件的标签上。**

<子组件标签 **:属性名=“动态数据”**></子组件标签>

**>> 2. 子组件通过props接收。**

**export default {**

**props: [“属性名”]**

**}**

**>> 3. props接收的对象写法/类型校验/默认值。**

**export default {**

**props: {**

**属性名：{**

**type: String， // 类型可以是Number/Object/Boolean/Array/Date/Funcion**

**default: ‘默认值’ // 对象或数组需要写函数。**

**}**

**}**

**}**

### 父传子示例代码:

**父组件代码：**

|  |
| --- |
| // Father.vue  <template>    <div class="father">      父组件      <Son :msg1="msg1" :user="user" />    </div>  </template>  <script>  // 引入子组件  import Son from "@/components/Son.vue";  export default {    components: {      Son    },    data() {      return {        msg1: "来自爸爸的数据",        user: { name: "李寻欢", age: 32, sex: "男" }      };    }  };  </script>  <style lang="less" scoped>  .father {    color: red;    border: 1px solid red;    height: 400px;    padding: 30px;  }  </style> |

**子组件代码：**

|  |
| --- |
| // Son.vue  <template>    <div class="son">      子组件      <br />      <!-- 父传入的：{{msg1}} -->      <br />      自己的: {{ sonMsg }}      <br />      父传入的: {{ user }}    </div>  </template>  <script>  export default {    //   props: ["msg1"], // 接收父组件传入的数据 (数组写法)    // 对象写法    props: {      user: {        type: Object, // 类型验证        default: () => {} // 默认值      }    },    data() {      return {        sonMsg: "子自己的数据"      };    }  };  </script>  <style lang="less" scoped>  .son {    color: green;    border: 1px solid green;    height: 100px;  }  </style> |

## 子组件传递给父组件【掌握】

### 子传父语法

**>> 子组件通过 this.$emit() 把数据传递出去**

**this.$emit(“自定义事件”, 要传递出去的数据)**

**>> 父组件通过在子组件标签上v-on绑定自定义事件接收数据**

**<子组件标签 @自定义事件=”函数名”></子组件标签>**

methods: {

**函数名( sonData ) {**

**// sonData就是子组件传递出来的数据。**

**}**

}

### 1.2.2 子传父示例代码

**>>子组件代码：**

|  |
| --- |
| // Son.vue  <template>    <div class="son">      子组件      <br />      <button @click="sendToFather">发送给爸爸</button>    </div>  </template>  <script>  export default {    data() {      return {        msg2: "来自儿子的数据"      };    },    methods: {      sendToFather() {        // 发送数据给父组件        this.$emit("sendToFa", this.msg2);      }    }  };  </script>  <style lang="less" scoped>  .son {    color: green;    border: 1px solid green;    height: 100px;  }  </style> |

**>> 父组件代码**

|  |
| --- |
| // Father.vue  <template>    <div class="father">      父组件      <br />      自己的数据: {{ faMsg }}      <br />      子传过来的: {{ sonData }}      <!-- 绑定自定义事件，接收子传递过来的数据 -->      <Son @sendToFa="receiveMsg" />    </div>  </template>  <script>  // 引入子组件  import Son from "@/components/Son.vue";  export default {    components: {      Son    },    data() {      return {        faMsg: "父自己的数据",        sonData: ""      };    },    methods: {      receiveMsg(sonData) {        this.sonData = sonData;      }    }  };  </script>  <style lang="less" scoped>  .father {    color: red;    border: 1px solid red;    height: 400px;    padding: 30px;  }  </style> |

## 任意组件之间传递数据（跨级组件）【掌握】

### 语法：

任意组件之间传递数据，需要通过一个空的vue实例，步骤如下：

1. 在入口文件main.js中，创建一个空的vue实例，挂载在Vue的原型上

**Vue.prototype.$bus = new Vue();**

1. 通过emit方法把数据传递出去。

**this.$bus.$emit(‘自定义事件’, 要传递出去的数据)**

1. 在生命周期created中，通过$on监听自定义事件的方式，接收数据。

created ( ) {

**this.$bus.$on(‘自定义事件’, function ( data ) {**

**// data就是从传递过来的数据**

**})**

}

### 示例代码：

**>> 把数据传递出去的组件代码**

|  |
| --- |
| // Son.vue  <template>    <div class="son">      子组件      <br />      <button @click="sendToOther">发送给其他儿子</button>    </div>  </template>  <script>  export default {    data() {      return {        msg2: "来自儿子的数据",        msg: "任意组件间传递数据"      };    },    methods: {      sendToOther() {        // 把数据传递出去        this.$bus.$emit("atWill", this.msg);      }    }  };  </script>  <style lang="less" scoped>  .son {    color: green;    border: 1px solid green;    height: 100px;  }  </style> |

**>> 接收数据的组件代码**

|  |
| --- |
| <template>    <div class="other-son">      其他儿子组件      <br />      自己的：{{ msg }}      <br />      别的组件传过来的: {{ otherMsg }}    </div>  </template>  <script>  export default {    data() {      return {        msg: "其他儿子自己的数据",        otherMsg: ""      };    },    created() {      // 接收数据      this.$bus.$on("atWill", msg => {        this.otherMsg = msg;      });    }  };  </script>  <style lang="less" scoped>  .other-son {    margin-top: 20px;    border: 1px solid orange;    height: 150px;  }  </style> |

# slot插槽(了解)

插槽就是在子组件中预留位置（插槽），父组件在使用子组件标签的时候，可以把内容插入进去。

## 默认slot

一个没有名字的插槽，就是默认插槽

**子组件(** Layout.vue **):**

|  |
| --- |
| <!-- 默认插槽: name=default不写 -->        <slot>默认header</slot> |

**父组件:**

|  |
| --- |
| <Layout>        <!-- 这里写的所有内容 都会被插入默认插槽中 -->   </Layout> |

## 具名slot

一个具有名字的插槽，就是具名插槽

**子组件中:** 插槽具有了名字

|  |
| --- |
| <slot name="header">默认header</slot>  <slot name="main">默认main</slot>    <slot name="footer">默认footer</slot> |

**父组件：** 内容会找到对应名字的插槽插入

|  |
| --- |
| <!-- 具名插槽： 找到slot的值对应的这个名字的插槽，把内容插入 -->  <h1 slot="header">Header</h1>  <h1 slot="main">main</h1>  <h1 slot="footer">footer</h1> |

## 作用域slot

子组件中的数据，通过v-bind绑定到插槽上：

|  |
| --- |
| <slot :num="num" :msg="msg" name="main">默认main</slot> |

父组件中，通过作用域插槽使用数据：

|  |
| --- |
| <h1 slot="main" slot-scope="scope">main main mian: {{ scope.msg }} === {{ scope.num }}</h1> |

## v-slot

**注意：**

1. v-slot的缩写是#。
2. 需要使用template包裹。

v-slot是2.6.x新增的指令，用于取代以上三种写法。

|  |
| --- |
| <Layout>        <template #header>          <h1>Header</h1>        </template>        <template #main="scope">          <h1>Main</h1>          {{ scope }}          {{ scope.num }}          {{ scope.msg }}        </template>        <template #footer>          <h1>Footer</h1>        </template>      </Layout> |

# ElementUI【掌握】

## ElementUI简介

ElementUI是来自饿了么前端团队基于 Vue 2.0 的桌面端组件库，目的是实现PC管理系统界面的快速搭建。

## ElementUI安装

1、本地安装ElementUI

>> **yarn add element-ui**  // 直接安装到dependence

1. 在main.js中引入配置  
    1） 引入模块

**import ElementUI from 'element-ui';**

1. 引入全局样式

**import 'element-ui/lib/theme-chalk/index.css';**

1. 全局注册Element

**Vue.use(ElementUI);**

## ElementUI测试使用（组件使用）

# 课程总结

## 难点

* + - 1. ElementUI的企业级应用

## 如何掌握？

* + - 1. 通过代码验证原理。
      2. 绘制思维导图表。

# 课后练习

**完成时间: 90分钟**

**简答题（手抄笔记）**

1. 简述Vue组件的3种通信方式。 4种 父传子 子传父 乱传
2. 简述UI框架集成的基本步骤?

**操作题（电脑操作）**

1. 完成课堂代码（三种通信方式） 2遍
2. 参照element-ui官方案例，简单使用UI组件(表格 表单 按钮 导航 日期时间)
3. 使用element-ui实现一个登陆页面，和管理系统的首页。
4. 配置路由，登陆成功后跳转到首页。

# 面试题

# 扩展资料

<https://editorconfig.org/>

<https://cn.eslint.org/docs/user-guide/configuring>