Day1

一、邂逅Vuejs

1.1 认识Vuejs

- 为什么学习Vuejs
- Vue 的读音
- Vue 的渐进式
- Vue的特点

1.2 安装Vue

- CDN引入
- 下载引入
- npm安装

1.3 Vue的初体验

- Hello Vuejs

  * mustache--->体验vue响应式
    - Vue 列表展示
  * v-for
  * 后面给数组追加元素的时候，新的元素也可以再页面中渲染出来
    - Vue计数器小案例
  * 事件监听：v-on:click--->methods

1.4 Vue中的MVVM

- View层：视图层
- Model层：数据层
- VueModel层：视图模型层(View和Model沟通的桥梁,一方面实现了数据绑定、另一方面也实现了DOM监听)

1.5 创建Vue时，options可以放哪些东西

- el
- data
- methods
- 生命周期函数(钩子函数)

二、插值语法

- **mustache语法**：{{}}
- **v-once**:元素只渲染一次，不会随着数据的改变而改变
- **v-html** ：解析html语法
- **v-text**：使用不灵活
- **v-pre**：用于显示原本的Mustache语法
- **v-cloak**：斗篷

三、V-bind

3.1 v-bind绑定基本属性

- v-bind:src
- :href

3.2 v-bind动态绑定class

- 对象语法：作业 :class='{类名:boolean}'
- 数组语法：:class="[active,line]"

3.3 v-bind动态绑定style

- 对象语法：:style="{color:currentColor,fontSize:fontSize + 'px'}"
- 数组语法：v-bind:style="[basestyles，overridingStyles]"

四、计算属性：computed

- 案例一：firstName+lastName
- 案例二：books--->price

Day 2

一、计算属性

1.1 计算属性的本质

- fullname:{set(),get()}

2. 计算属性和methods的对比

- 计算属性在多次使用时，只会调用一次
- 它是有缓存的

二、ES6 语法补充

1.1 let/var

- ES5中的var是没有块级作用域的(if/for)

- ES6中的let是有块级作用的(if/for)
- ES5之前因为if和for都没有块级作用域的概念,所以在很多时候,我们都必须借助function的作用域来
- 我们可以将let看成更完美的var

1.2 const的使用

- 当我们修饰的标识符不会被再次赋值时，就可以使用const来保证数据的安全性
- 建议：在开发中，优先使用const,只有需要改变某一个标识符的时候才使用let
- 一旦给const修饰的标识符被赋值之后，不能修改（**const只记住一个内存地址**）
- 在使用const定义标识符，必须赋值
- 常量的含义是指向的对象不能修改，但是可以修改对象内部的属性**(改变对象属性其内存地址不变)**

1.3 对象增强写法

- 属性的增强写法：const obj={name,age,height}
- 函数的增强写法：const obj=run(){}

三、事件监听

2.1 事件监听的基本使用

- **作用**：绑定事件监听器
- **缩写**：@
- **预期**：Function | Inline Statement | Object
- **参数**：event
- 监听按钮的点击事件:v-on:click="counter++"
- 另外，我们可以将事件指向一个methods中定义的函数：v-on:click="increment"
    - v-on:click可以写成@click

2.2 参数问题

- btnClick:(满足条件：1.在事件监听的时候 2.方法不需要额外参数)。
- btnClick(event)
- btnClick(abc,event)->$event

2.3 修饰符

- .stop---调用event.stopPropagation(),停止冒泡（在div还有其他内容时，使用按钮）
- .prevent---调用event.preventDefault()，阻止默认行为

- .{keyCode | keyAlias}---只当事件是从特定键触发时才触发回调，监听某个键盘键帽
- .native---监听组件根元素的原生事件
- .once---只触发一次回调

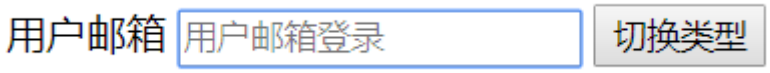
四、条件判断

3.1 v-if/v-else-if/v-else

- v-if后面的条件为false时，对应的元素以及其子元素不会渲染
- 也就是根本不会有对应的标签出现在DOM中

3.2 登录小案例

- 用户在登录时，可以切换使用用户账号登录还是邮箱地址登录
- 类型如下情景：



- **案例存在的小问题**：如果我们在有输入内容的情况下，切换了类型，我们会发现文字依然显示之前输入的内容
- 这是因为Vue在进行DOM渲染时，出于性能考虑，会尽可能的复用已经存在的元素，而不是重新创建新的元素。
- **解决方案**：如果我们不希望Vue出现类似重复利用的问题，可以给对应的input添加key，并且我们需要保证key的不同。

3.3 v-show

- v-if 当条件为false时，压根不会有对应的元素在DOM中。
- v-show当条件为false时，仅仅将元素的display属性设置为none而已。
- 当需要在显示和隐藏之间切换很频繁时，使用v-show
- 当只有一次切换时，通过使用v-if

五、循环遍历

4.1 遍历数组

- 以item in items的格式：v-for="movie in movies"
- 如果需要拿到元素在数组中的索引值，语法格式：v-for=(item,index) in items，其中的index就代表了取出的item在原数组的索引值。

4.2 遍历对象

- value
- value,key
- value,key,index

4.3 组件的key属性

- key的作用主要是为了高效的更新虚拟DOM

4.3 数组哪些方法是响应式的

- push():数组末尾追加元素
- pop():删除数组中的最后一个元素
- shift():删除数组中的第一个元素
- unshift():在数组最前面添加元素
- splice():删除元素/插入元素/替换元素
- sort():正向排序
- reverse()：翻转数组

4.4 高阶函数:filter/map/reduce

- filter中的回调函数有一个要求：必须返回一个boolean值
- map函数：对所有的值进行相同的操作
- reduce函数作用：对数组中所有内容进行汇总：reduce(prevalue,n)

4.4 作业完成

- v-for与v-bind的结合：

```
<div id="app">
    <!-- 作业需求：点击列表中的哪一项，那么该项的文字变成红色 -->
    <ul>
        <li v-for="(item,index) in movies" :class="{active:currentIndex===index}" @click="liClick(index)">{{index}}.{{item}}</li>
    </ul>
</div>
<script>
    const app = new Vue({
        el: '#app',
        data: {
            message: '你好哈',
            movies: ['海王', '火影', '柯南', '小樱'],
            currentIndex: 0
        },
        methods: {
            liClick(index) {
                this.currentIndex = index
            }
        },
    })
</script>
```

六、书籍案例

序号	书籍名称	出版日期	价格	购买数量	操作
1	《算法导论》	2006-9	¥85.00	- 1 +	移除
2	《UNIX编程艺术》	2006-2	¥59.00	- 1 +	移除
3	《编程珠玑》	2008-10	¥39.00	- 1 +	移除
4	《代码大全》	2006-3	¥128.00	- 1 +	移除

总价格:￥311.00

七、v-model的使用

6.1 v-model的基本使用

- **原理**：v-model=>v-bind:value v-on:input
- **作用**：v-model实现了数据的双向绑定

```
<input type="text" v-model="message">
<!-- 等同于下面 -->
<input type="text" :value="message" @input="message=$event.target.value">
```

6.2 v-model和radio/checkbox/select

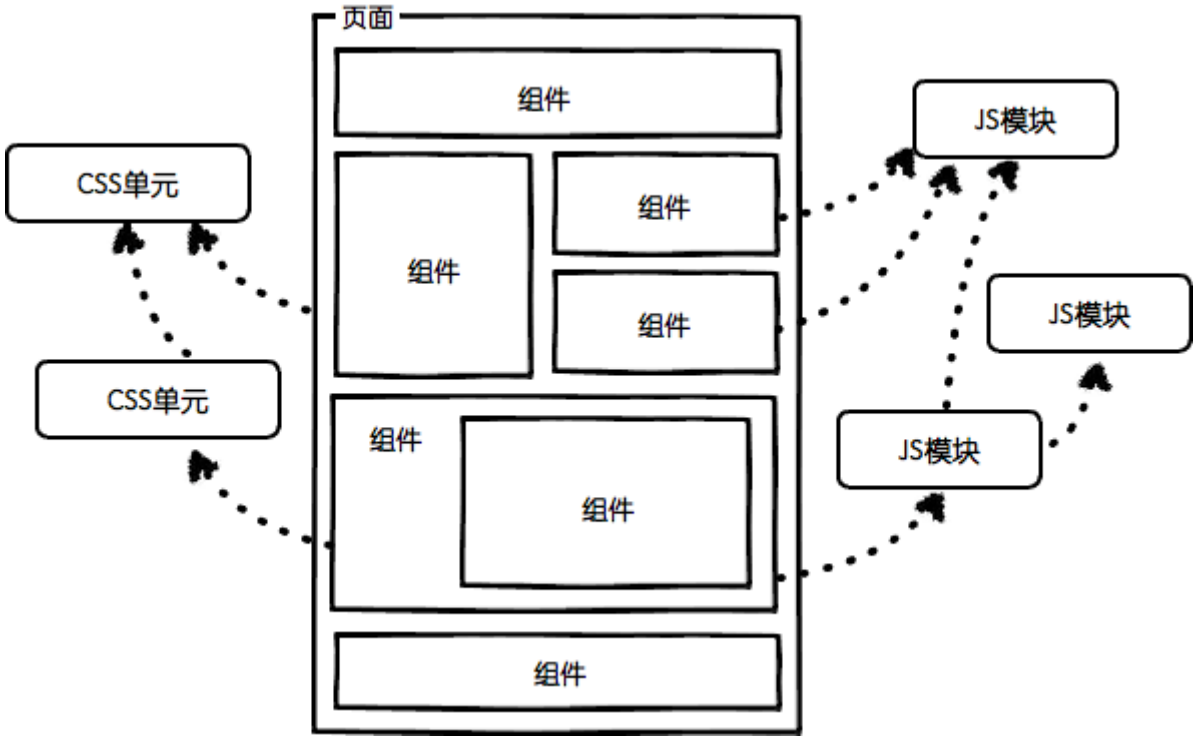
6.3 修饰符

- **lazy**：让数据在失去焦点或者回车时才会更新。
- **number**：让在输入框输入的内容自动转成数字类型。
- **trim**：过滤内容左右两边的空格。

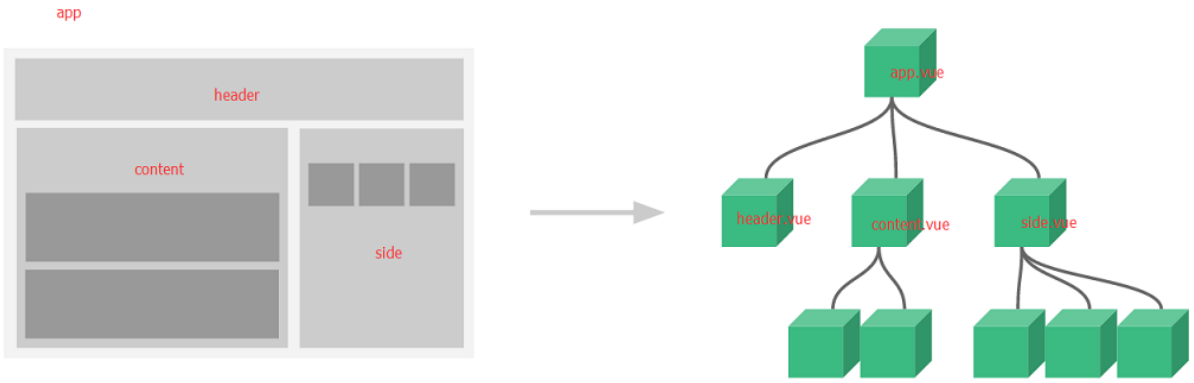
八、组件化开发

7.1 认识组件化

- 我们将一个完整的页面分成很多个组件，每个组件都用于实现页面的一个功能块，而每一个组件又可以进行细分。



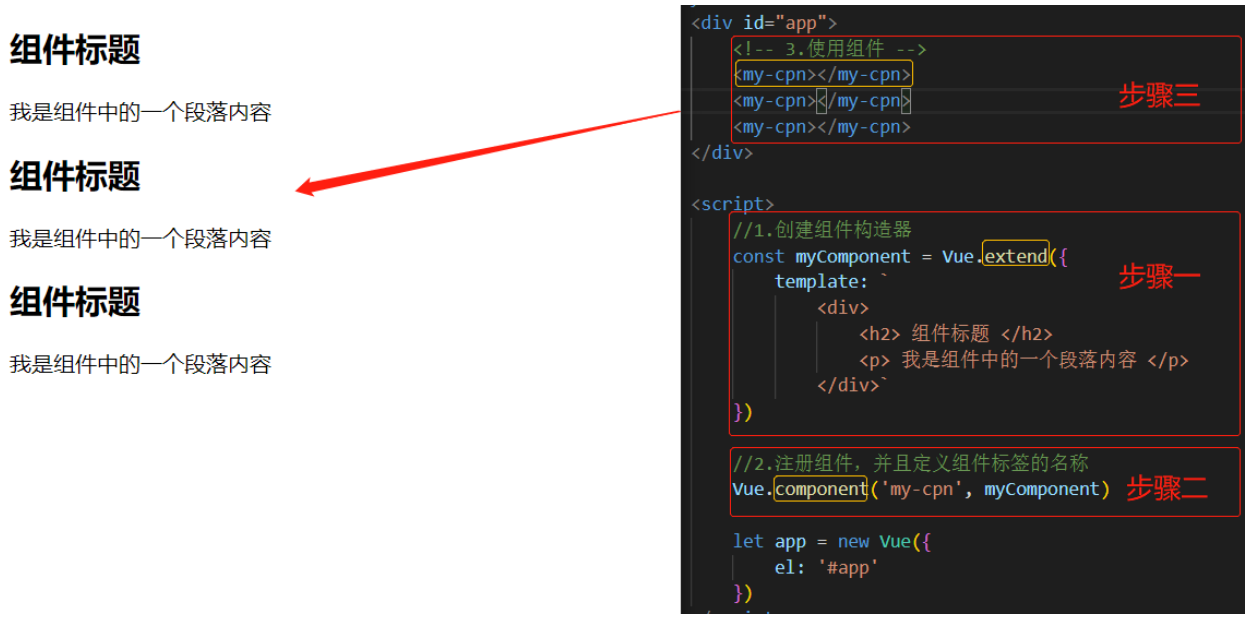
- **Vue 组件化思想：**它提供了一种抽象，让我们可以开发出一个个独立可复用的小组件来构造我们的应用，任何的应用都会被抽象成一颗组件树。



7.2 组件的基本使用

组件的使用分成三个步骤：

- 创建组件构造器：Vue.extend()
- 注册组件:Vue.component()
- 使用组件:在Vue实例的作用范围内使用



7.3 全局组件和局部组件

- 当我们通过**调用Vue.component()注册组件时，组件的注册是全局的**
- 如果我们注册的组件是**挂载在某个实例中**，那么就是一个**局部组件**

7.4 父组件和子组件

- 通过代码如何组成这种层级关系：

父子组件**错误用法**：以子标签的形式在Vue实例中使用

- 因为当子组件注册到父组件的components时，Vue会编译好父组件的模板
- 该模板的内容已经决定父组件将要渲染的HTML(相当于父组件中已经有了子组件中的内容了)

- <child_cpn></child_cpn>是只能在父组件中被识别的。
- 类似这种用法，<child_cpn></child_cpn>是会被浏览器忽略的。

7.5 注册的语法糖

语法糖注册全局组件和局部组件：

```
<div id="app">
    <!-- 3.使用组件(全局) -->
    <my-cpn1></my-cpn1>
    <!-- 3.使用组件(局部) -->
    <my-cpn2></my-cpn2>
</div>
<my-cpn></my-cpn>
<script>
    //1.全局组件注册的语法糖
    //1.创建组件构造器
    //const myComponent = Vue.extend()

    //2.注册组件，并且定义组件标签的名称
    Vue.component('my-cpn1', {
        template: `
            <div>
                <h2> 全局组件标题1 </h2>
                <p> 我是全局组件中的一个段落内容1 </p>
            </div>`
    })

    //注册局部组件的语法糖
    const app = new Vue({
        el: '#app',
        components: {
            'my-cpn2': {
                template: `
                    <div>
                        <h2> 局部组件标题2 </h2>
                        <p> 我是局部组件中的一个段落内容2 </p>
                    </div>`
            }
        }
    })
</script>
```

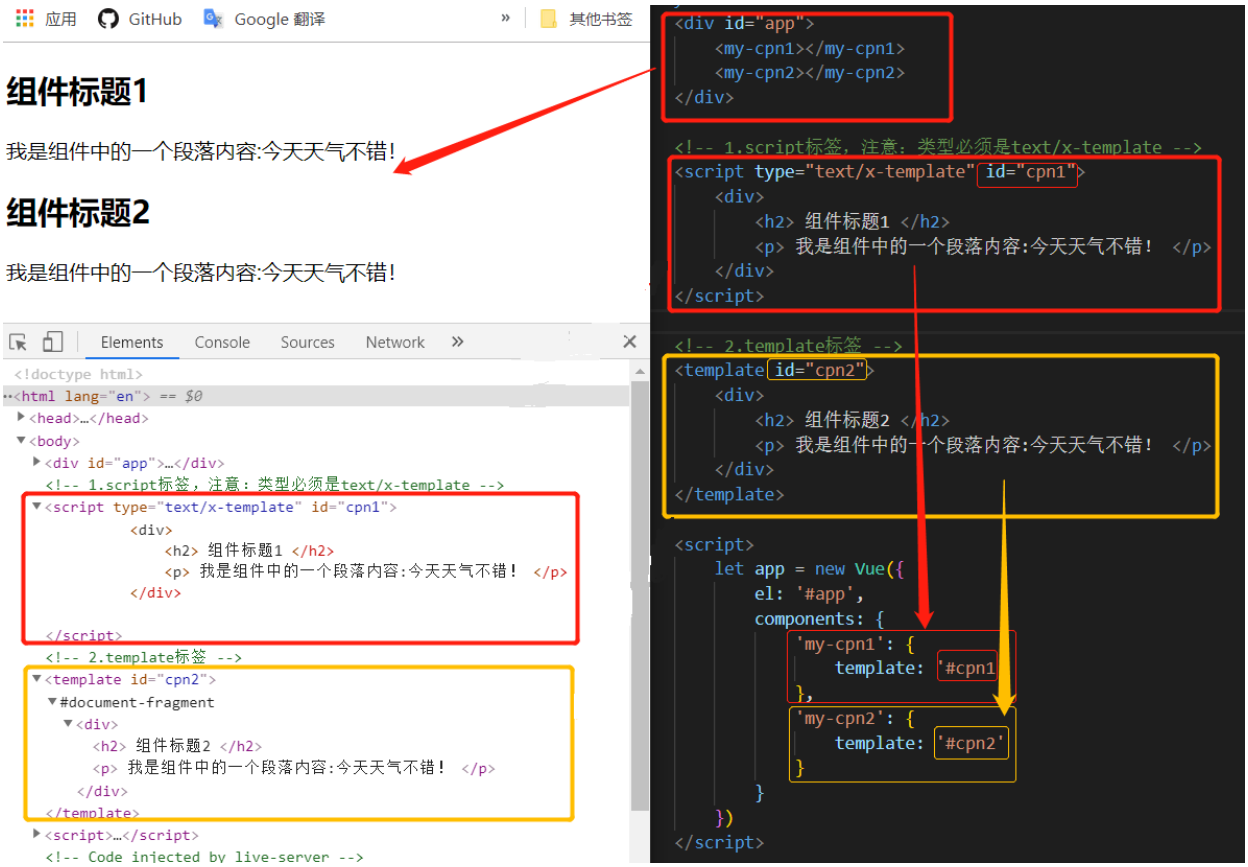
语法糖全局组件

语法糖局部组件

7.6 模板的分离写法

- script

- template



7.7 数据的存放

- 子组件不能直接访问父组件
- 子组件中有自己的data,而且必须是一个函数
- 为什么是一个函数:使用函数，每次调用都会产生新的对象，而不会相互影响

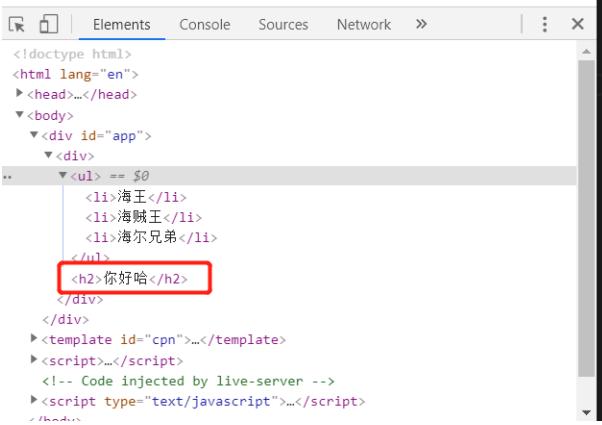
7.8 父子组件通信

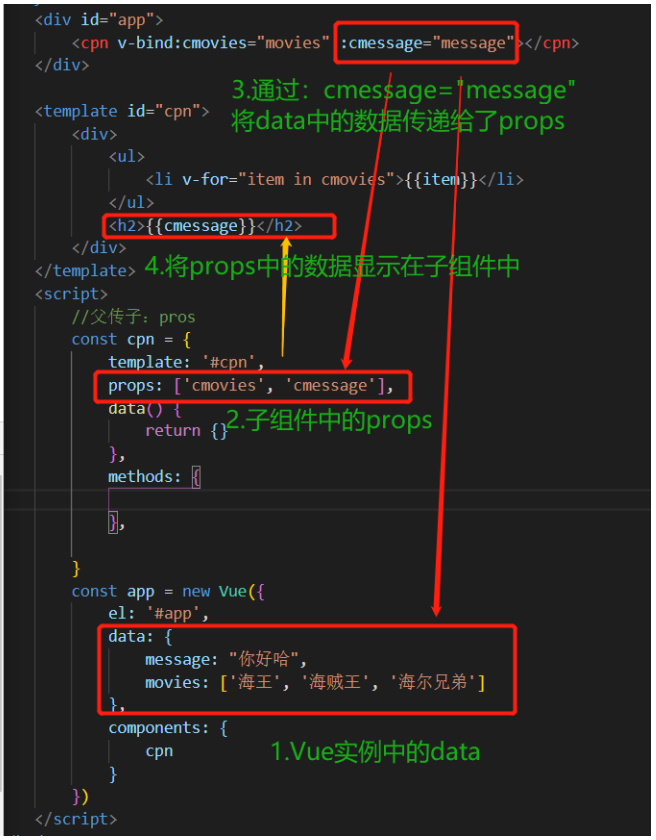
- 父传子：props
- 子传父：$emit

父传子：

- 海王
- 海贼王
- 海尔兄弟

你好哈





子传父：

7.9 项目

github --->用户名:coderwhy 项目名:**HYMall**

Day 3

一、组件化开发

二、前端模块化

三、webpack

四、Vue CLI