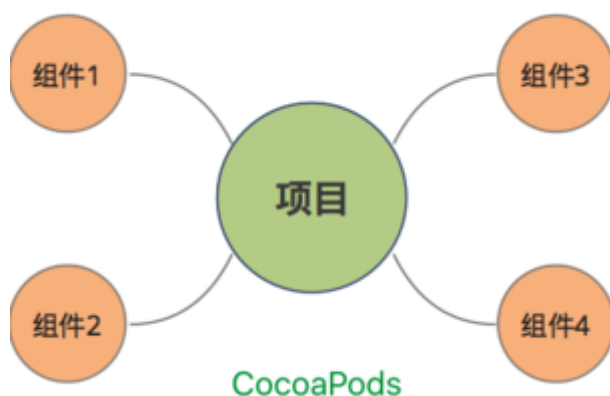


一、什么是组件化

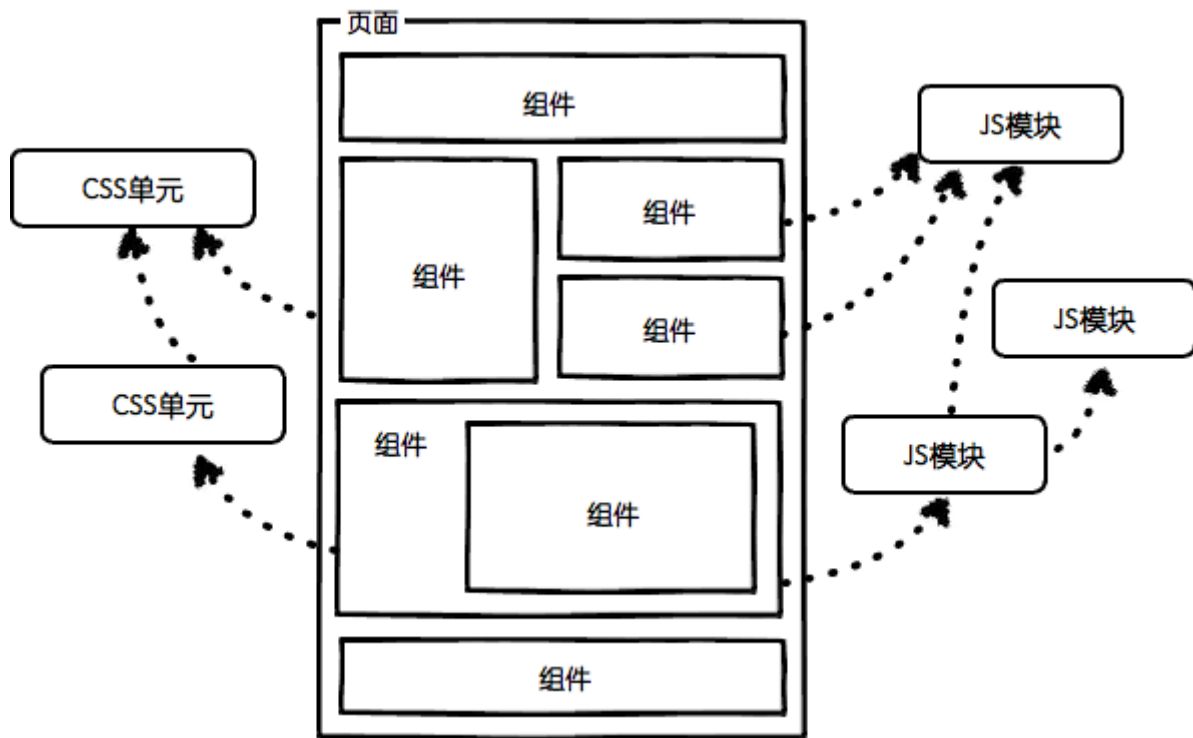
1.人们面对复杂问题的处理方式：

- 任何一个人处理信息的逻辑能力都是有限的
- 所以，当面对一个非常复杂的问题时，我们不太可能一次性搞定一大堆内容。
- 但是，我们人有一种天生的能力，就是将问题进行拆解。
- 如果将一个复杂问题，拆分成很多个可以处理的小问题，再将其放在整体当中，你会发现大问题也会迎刃而解。



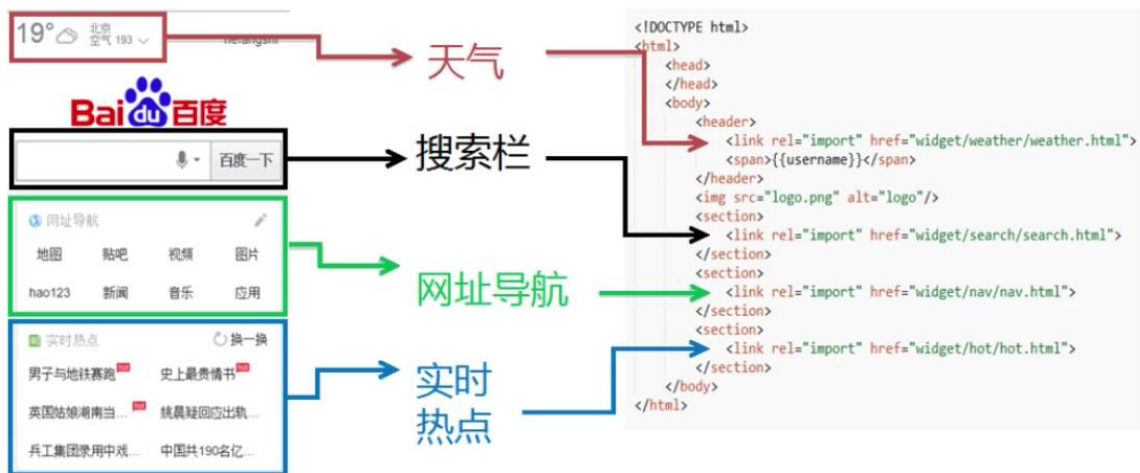
2.组件化也是类似的思想：

- 如果我们将一个页面中所有的处理逻辑全部放在一起，处理起来就会变得非常复杂，而且不利于后续的管理以及扩展。
- 但如果，我们将一个页面拆分成一个个小的功能块，每个功能块完成属于自己这部分独立的功能，那么之后整个页面管理和维护就变得非常容易了。



- 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。

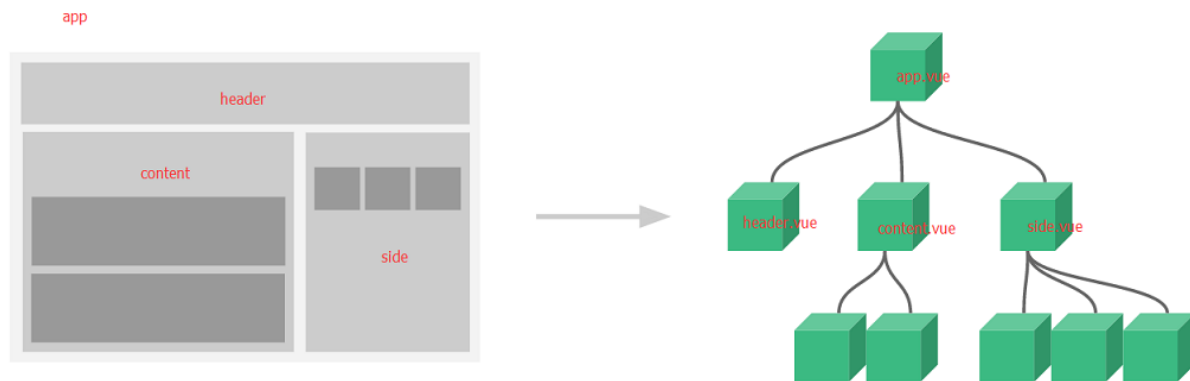
什么是组件化



二、Vue组件化思想

1. 组件化是Vue.js中的重要思想

- 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。



2.组件化思想的应用：

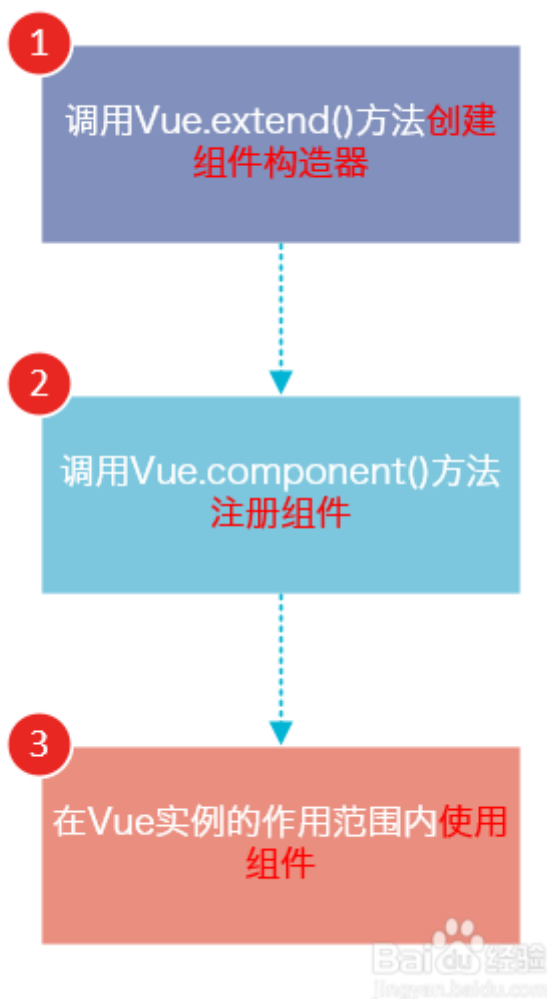
- 有了组件化的思想，我们在之后的开发中就要充分的利用它。
- 尽可能的将页面拆分成一个个小的、可复用的组件。
- 这样让我们的代码更加方便组织和管理，并且扩展性也更强。

3.所以，组件是Vue开发中，非常重要的一个篇章，要认真学习。

三、注册组件的基本步骤

1.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



2.我们来看看通过代码如何注册组件

组件标题

我是组件中的一个段落内容

组件标题

我是组件中的一个段落内容

组件标题

我是组件中的一个段落内容

```
<div id="app">
  <!-- 3.使用组件 -->
  <my-cpn></my-cpn>
  <my-cpn></my-cpn>
  <my-cpn></my-cpn>
</div>

<script>
  //1.创建组件构造器
  const myComponent = Vue.extend({
    template: `
      <div>
        <h2> 组件标题 </h2>
        <p> 我是组件中的一个段落内容 </p>
      </div>`
  })

  //2.注册组件, 并且定义组件标签的名称
  Vue.component('my-cpn', myComponent)

  let app = new Vue({
    el: '#app'
  })
</script>
```

步骤三

步骤一

步骤二

3.查看运行结果:

- 和直接使用一个div看起来并没有什么区别
- 但是我们可以设想, 如果很多地方都显示这样的信息, 我们是不是就可以直接使用<my-cpn> </my-cpn>来完成呢?

四、注册组件步骤解析

这里的步骤都代表什么含义呢？

1. Vue.extend():

- 调用Vue.extend()创建的是一个组件构造器
- 通常在创建组件构造器时，传入`template`代表我们自定义组件的模板。
- 该模板就是在使用到组件的地方，要显示的HTML代码
- 事实上，这种写法在Vue2.x的文档中几乎看不到了，他会直接使用下面我们会讲到的语法糖，但是在很多资料还是会提到这种方式，而且这种方式是学习后面方式的基础。

2. Vue.component():

- 调用Vue.component()是将刚才的组件构造器注册为一个组件，并且给它起一个组件的标签名称。
- 所以需要传递两个参数：(1)注册组件的标签名 (2)组件构造器

3. 组件必须挂载在某个Vue实例下，否则它不会生效。

- 我们来看下面我使用了三次`<my-cpn>`
- 而第三次其实并没有生效：



五、全局组件与局部组件

1. 当我们通过调用Vue.component()注册组件时，组件的注册是全局的

- 这意味着该组件可以在任意Vue实例下使用

组件标题

我是组件中的一个段落内容

组件标题

我是组件中的一个段落内容

组件标题

我是组件中的一个段落内容

组件标题

我是组件中的一个段落内容

```
<div id="app">
  <!-- 3.使用组件 -->
  <cpn></cpn>
  <cpn></cpn>
  <cpn></cpn>
</div>
<hr>
<div id="app2">
  <!-- 3.使用组件 -->
  <cpn></cpn>
</div>
<script>
  //1.创建组件构造器
  const cpnC = Vue.extend({
    template: `
      <div>
        <h2> 组件标题 </h2>
        <p> 我是组件中的一个段落内容 </p>
      </div>`
  })
  //2.注册组件(全局组件,意味着可以在多个Vue的实例下面使用)
  Vue.component('cpn', cpnC)

  //创建Vue实例1
  let app = new Vue({
    el: '#app',
  })
  //创建Vue实例2
  let app2 = new Vue({
    el: '#app2',
  })
</script>
```

2.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

组件标题

我是组件中的一个段落内容

组件标题

我是组件中的一个段落内容

组件标题

我是组件中的一个段落内容

没有被渲染出来

```
<div id="app">
  <!-- 3.使用组件 -->
  <cpn></cpn>
  <cpn></cpn>
  <cpn></cpn>
</div>
<hr>
<div id="app2">
  <!-- 3.使用组件 -->
  <cpn></cpn>
</div>
<script>
  //1.创建组件构造器
  const cpnC = Vue.extend({
    template: `
      <div>
        <h2> 组件标题 </h2>
        <p> 我是组件中的一个段落内容 </p>
      </div>`
  })
  //2.注册组件(全局组件,意味着可以在多个Vue的实例下面使用)
  //Vue.component('cpn', cpnC)
  //疑问: 怎么注册的组件才是局部组件?
  //创建Vue实例1
  let app = new Vue({
    el: '#app',
    components: {
      //cpn使用组件时的标签名
      cpn: cpnC
    }
  })
  //创建Vue实例2
  let app2 = new Vue({
    el: '#app2',
  })
</script>
```

将组件构造器放在了实例当中

六、父组件和子组件

在前面我们看到了组件数:

- 组件和组件之间存在层级关系
- 而其中一种非常重要的关系就是父子组件关系

我们来看通过代码如何组成这种层级关系：

我是父组件内容，我是parent哦

我是父组件内容，我是parent哦

我是子组件内容，我是child哦

```
<div id="app">
  <!-- 3.使用组件 -->
  <parent_cpn></parent_cpn>
</div>
<script>
  //1.创建第一个组件构造器(子组件)
  const childComponent = Vue.extend({
    template: `
      <div>
        <p> 我是子组件内容，我是child哦</p>
      </div>`
  })

  //1.创建第二个组件构造器(父组件)
  const parentComponent = Vue.extend({
    template: `
      <div>
        <p> 我是父组件内容，我是parent哦 </p>
        <h2> 我是父组件内容，我是parent哦 </h2>
        <child_cpn></child_cpn>
      </div>`,
    components: {
      child_cpn: childComponent,
    }
  })

  //root组件
  const app = new Vue({
    el: '#app',
    //2.注册组件，并且定义组件标签的名称
    components: {
      parent_cpn: parentComponent
    }
  })
</script>
```

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。

七、注册组件语法糖

在上面注册组件的方式，可能会有些繁琐。

- Vue为了简化这个过程，提供了注册的语法糖。
- 主要是省去了调用Vue.extend()的步骤，而是可以直接使用一个对象来代替。

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

八、组件模板的分离写法

刚才，我们通过语法糖简化了Vue组件的注册过程，另外还有一个地方的写法比较麻烦，就是template模板写法。

如果我们能将其中的HTML分离出来写，然后挂载到对应的组件上，必然结果会变得非常清晰。

Vue提供了两种方案来定义HTML模板内容:

- 使用<script>标签
- 使用<template>标签

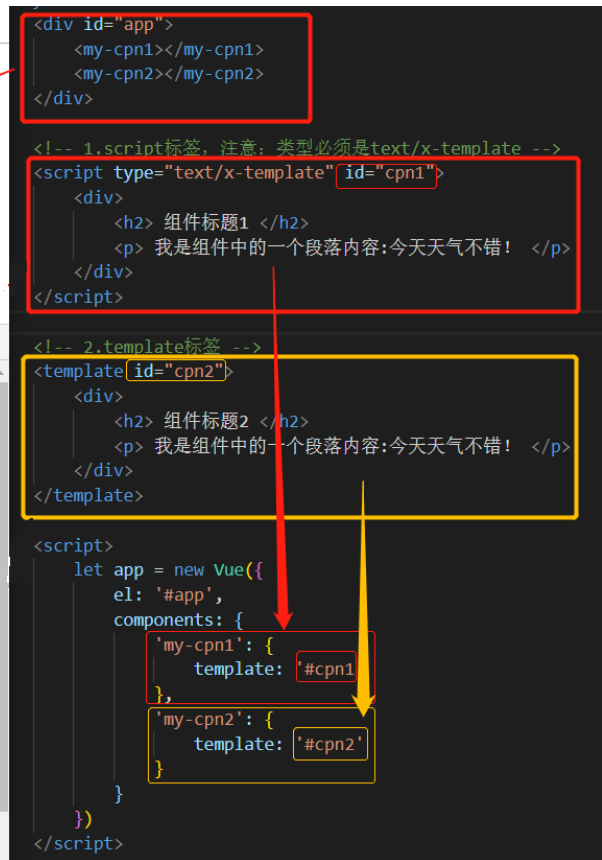
应用 GitHub Google 翻译 其他书签

组件标题1

我是组件中的一个段落内容:今天天气不错!

组件标题2

我是组件中的一个段落内容:今天天气不错!



九、组件可以访问Vue实例数据吗?

1.组件是一个单独功能模块的封装:

- 这个模板有属于自己的HTML模板, 也应该有属于自己数据data.

2.组件中的数据是保存在哪里呢? 顶层的Vue实例中吗?

- 我们先来测试一下, 组件中能不能直接访问Vue实例中的data

应用 GITHUD Google 翻译 其他书签

消息:

解析:

组件去访问message

message定义在Vue实例

我们发现并没有显示结果。

结论:

组件不能直接访问Vue实例中的data数据



- 我们发现不能访问, 而且即使可以访问, 如果将所有的数据都放在Vue实例中, Vue实例就会变得非常臃肿。

- 结论：Vue组件应该有自己的保存数据的地方。

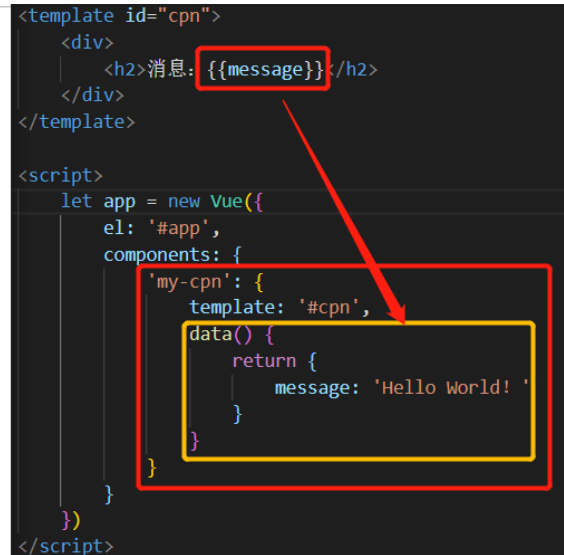
十、组件数据的存放

组件自己的数据存放在哪里呢？

- 组件对象也有一个data属性(也可以有methods等属性，下面我们有用到)
- 只是这个data属性必须是一个函数
- 而且这个函数返回一个对象，对象内部保存着数据

消息：Hello World!

解析：
Hello World 可以显示
原因：
这是因为目前组件访问的是自己当中的data



```
<template id="cpn">
  <div>
    <h2>消息: {{message}}</h2>
  </div>
</template>

<script>
  let app = new Vue({
    el: '#app',
    components: {
      'my-cpn': {
        template: '#cpn',
        data() {
          return {
            message: 'Hello World! '
          }
        }
      }
    }
  })
</script>
```

十一、组件中的data为什么是函数

因为，使用函数，每次调用都会产生新的对象，而不会相互影响。

如果不是函数，组件之间的通信会受到干扰

当前计数:1



当前计数:2

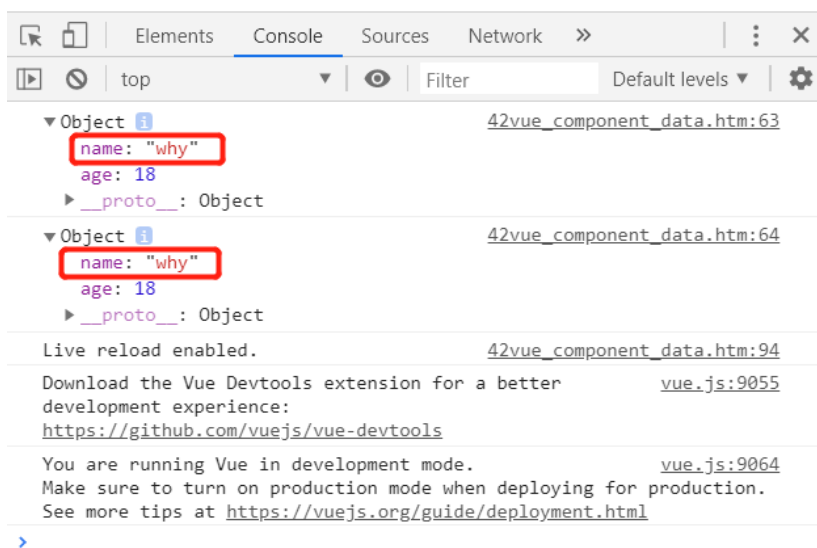


当前计数:3



正因为data是函数，
各组件的调用才不会相互影响，

每次创建新的组件实例
都会是新的内存地址



```
Vue.component('cpn', {
  template: '#cpn',
  data() {
    return {
      counter: 0
    }
  },
  methods: {
    increment() {
      this.counter++
    },
    decrement() {
      this.counter--
    }
  }
})

const app = new Vue({
  el: '#app'
})

</script>

<script>
function abc() {
  return {
    name: 'why',
    age: 18
  }
}

let obj1 = abc()
let obj2 = abc()
let obj3 = abc()

obj1.name = "kobe"
console.log(obj2)
console.log(obj3)
</script>
```

十二、父子组件的通信

在上一个小节中，我们提到了子组件是不能引用父组件或者Vue实例的数据的。

但是，在开发中，往往一些数据确实需要从上层传递到下层：

- 比如在一个页面中，我们从服务器请求到了很多的数据。
- 其中一部分数据，并非是我们整个页面的大组件来展示的，而是需要下面的子组件进行展示。
- 这个时候，并不会让子组件再次发送一个网络请求，而是直接让**大组件(父组件)**将数据传递给**小组件(子组件)**

如何进行父子组件间的通信呢？Vue官方提到：

- 通过props向子组件传递数据
- 通过事件向父组件发生消息



下面的代码中，我们直接将Vue实例当作父组件，并且其中包含子组件来简化代码
在真实的开发中，**Vue实例和子组件的通信**和**父组件和子组件的通信**过程是一样的。

十三、props基本用法

在组件中，使用选项props来声明需要从父级接收到的数据。

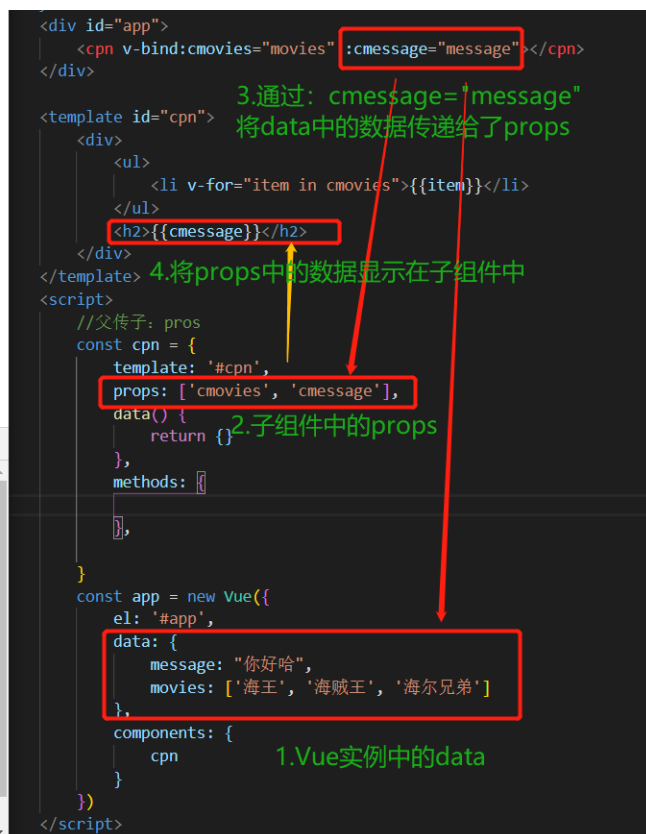
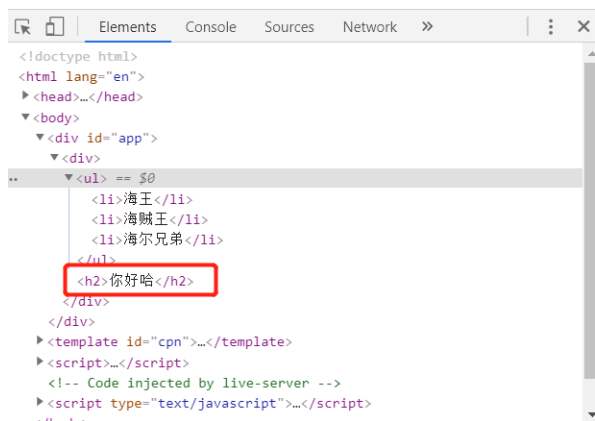
props的值有两种方式：

- 方式一：字符串数组，数组中的字符串就是传递时的名称。
- 方式二：对象，对象可以设置传递时的类型，也可以设置默认值等。

我们先来看一个最简单的props传递：

- 海王
- 海贼王
- 海尔兄弟

你好哈



十四、props数据验证

在前面，我们的props选项是使用一个数组

我们说过，除了数组之外，我们也可以使用对象，当需要对**props进行类型等验证时**，就需要对象写法了。

验证都支持哪些数据类型呢？

- String
- Number
- Boolean
- Array
- Object
- Date
- Function
- Symbol

当我们有自定义构造函数时，验证也支持自定义类型

```
//父传子: pros
const cpn = {
  template: '#cpn',
  // props: ['cmovies', 'cmessage'],
  props: {
    //1.类型限制
    //cmovies: Array,
    // cmessage:String

    //2.提供一些默认值，以及必传值
    cmessage: {
      type: String,
      default: "你好哈v边防官兵", //默认值
      required: true //必须传值
    },
    //类型时对象或者数组时，默认值必须是一个函数
    cmovies: {
      type: Array,
      //default: [] //vue2.5.17以下
      default () {
        return []
      }
    }
  }
},
```

```
function Person (firstName, lastName) {  
  this.firstName = firstName  
  this.lastName = lastName  
}
```

自定义类型

```
Vue.component('blog-post', {  
  props: {  
    author: Person  
  }  
})
```

Vue组件通信：父传子(props中的驼峰标识):

```
<div id="app">  
  <!-- v-bind不支持驼峰,需转化为以-连接的语法 -->  
  <cpn :cinfo="info" :child-my-message="message"></cpn>  
</div>  
  
<template id="cpn">  
  <div>  
    <h2>{{cinfo}}</h2>  
    <h2>{{childMyMessage}}</h2>  
  </div>  
</template>
```

十五、组件通信：子级向父级传递

props用于父组件向子组件传递数据，还有一种比较常见的是子组件传递数据或事件到父组件中。

我们应该如何处理呢？这个时候，我们需要使用**自定义事件**来完成。

什么时候需要自定义事件呢？

- 当子组件需要向父组件传递数据时，就要用到自定义事件了。
- 我们之前学习的v-on不仅仅可以用于监听DOM事件，也可以用于组件间的自定义事件。

自定义事件的流程：

- 在子组件中，通过\$emit()来触发事件
- 在父组件中，通过v-on来监听子组件事件。

我们来看一个简单的例子：

- 我们之前做过一个两个按钮+1和-1,点击后修改counter.
- 我们整个操作的过程还是在子组件中完成, 但是之后的展示交给父组件。
- 这样, 我们就需要将子组件中的counter,传给父组件的某个属性, 比如total。

```

<div id="app">
  <child-cpn @increment="changeTotal" @decrement="changeTotal"></child-cpn>
  <h2>点击次数: {{total}}</h2>
</div>

<template id="childCpn">
  <div>
    <button @click="increment">+1</button>
    <button @click="decrement">-1</button>
  </div>
</template>

<script>
  let app = new Vue({
    el: '#app',
    data: {
      total: 0
    },
    methods: {
      changeTotal(counter) {
        this.total = counter
      }
    },
    components: {
      'child-cpn': {
        template: '#childCpn',
        data() {
          return {
            counter: 0
          }
        },
        methods: {
          increment() {
            this.counter++;
            this.$emit('increment', this.counter)
          },
          decrement() {
            this.counter--;
            this.$emit('decrement', this.counter)
          }
        }
      }
    }
  })
</script>

```

发生两个事件时, 调用同一个函数changeTotal

Vue实例

子组件

1.子组件发出事件
2.在使用child-cpn时, 通过@increment和@decrement监听事件

十六 案例

