一、模版指令

通过模版指令(写在html中的)，即是html和vue对象的粘合剂。

- 数据渲染 v-text、v-html、{{}}

```
  <div id="app">
    {{ message }}
  </div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

> `v-text`更新元素的 `textContent`，`v-html`更新元素的 `innerHTML`，内容按普通 HTML 插入，不会作为 Vue 模板进行编译。如果想要`title: 'TOMVC <sub>2.0</sub>'`不以字符串形式显示，就可以使用v-html，`<h2 v-html='title'></h2>`

- 控制模块显示隐藏 v-if、v-show

```
  <div id="view">
    <p v-if='isShow'></p>
    <p v-show='isShow'></p>
</div>
```

```
new Vue({
    el: '#view',
    data: {
      isShow: true
    }
 });
```

> v-if是直接不渲染该元素； v-show是通过display: none进行隐藏;

- 渲染循环列表 v-for 模板引擎都会提供循环的支持。Vue也不例外，Vue是提供了一个v-for指令。基本的用法类似于foreach的用法。

```
    <ul class="view">
        // 遍历数组，取出每个元素
        <li v-for='item in list'>
            {{item.text}}
        </li>
    </ul>

    <ul>
        // 取出元素和对应下标
      <li v-for="(item, index) in list">
          第{{index+1}}个-{{item.text}}
      </li>
  </ul>
```

```
<script>
    var app = new Vue({
        el: '.view',
        data: {
            list: [  // 数组
                {text: '01、HTML'},
                {text: '02、CSS'},
                {text: '03、JavaScript'}
            ]
        }
    });
</script>
```

- 事件绑定 v-on

```
<div id="view4">
    <input type="button" value="按钮" v-on:click='doThis' />
    // 简写
    <input type="button" value="按钮" @click='doThis' />
</div>
```

```
<script>
  new Vue({
    el: '#view4',
    methods: {
      doThis: function(){
        alert('hello');
      }
    }
});
</script>
```

- 属性绑定 v-bind Vue中不能直接使用`{{ expression}}`语法进行绑定html的标签，而是用它特有的v-bind指令，语法`<标签 v-bind:属性名="要绑定的Vue对象的data里的属性名"></标签>`。 由于v-bind 使用非常频繁，所以Vue提供了简单的写法，可以去掉v-bind直接使用:即可。

```
<div id="view">
    <img :src='imgSrc'>
    <img :src='imgSrc'>
    <p :class='InfoClass'></p>
</div>
```

```
<script>
new Vue({
  el: '#view',
  data: {
    imgSrc: 'm_3_100.png',
    InfoClass: 'infoRed'
  }
});
</script>
```

- 样式绑定 v-bind 对于普通的属性的绑定，只能用上面的讲的绑定属性的方式。而Vue专门加强了class和style的属性的绑定。可以有复杂的对象绑定、数组绑定样式和类。 经常我们需要对样式进行切换，比如：div的显示和隐藏，某些标签active等。Vue提供的对象绑定样式的方式就很容做这些事情。

```
  <style>
      .view{
          width: 300px;
          height: 30px;
          border: 1px solid gray;
          line-height: 30px;
          text-align: center;
        }
       .active{  // 使能样式
          color: red;
        }
</style>
```

```
  // 当isActive为true时，这个div就会添加类名active；当isActive为false时，这个div就会移出类名active；
  <div class="view" :class='{active:isActive}'>
      {{message}}
  </div>
```

```
    var app = new Vue({
       el: '.view',
       data: {
            message: 'hello Vue.',
            isActive: false
        }
    });
```

- 双向数据绑定 v-model 上面的例子基本都是***单向***的js对象向 HTML数据进行绑定，那HTML怎样向js进行反馈数据呢？ Vue提供了一个新的指令：***v-model进行双向数据的绑定***，注意不是v-bind。

```
<div id="app">
    <p>{{ message }}</p>
     // 双向数据绑定
    <input v-model="message">
</div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

二、Vue组件中重要选项

Vue的实例是Vue框架的入口，其实也就是前端的ViewModel，它包含了页面中的业务逻辑处理、数据模型等，当然它也有自己的一系列的生命周期的事件钩子，辅助我们进行对整个Vue实例生成、编译、挂载、销毁等过程进行js控制。

- data数据选项，代表vue对象的数据 创建的Vue对象中的***data属性就是用来绑定数据到HTML的***，Vue框架会自动监视data里面的数据变化，自动更新数据到HTML标签上去。本质原理是：Vue会自动将data里面的数据进行递归抓换成getter和setter，然后就可以自动更新HTML标签了，当然用getter和setter所以老的浏览器Vue支持的不够好。

```
<div id="app">
    {{ message }}
</div>
```

```
var app = new Vue({
    el: '#app',
    data: {  // data数据选项
      message: 'Hello Vue!'
    }
 })
```

- methods方法选项，代表vue对象的方法 方法中的 this 自动绑定为 Vue 实例。

```
<div id="view">
  <input type="button" value="按钮" @click='doThis' />
</div>
```

```
<script>
new Vue({
  el: '#view',
  methods: {  // methods方法选项
    doThis: function(){
      alert('hello');
    }
  }
});
</script>
```

- computed计算属性 在做数据的绑定的时候,数据要进行处理之后才能展示到html页面上，虽然vue提供了非常好的表达式绑定的方法，但是只能应对低强度的需求，另外放入太多的逻辑会让模板过重且难以维护。而***计算属性，即属性可以是一个方法。*** 所有 getter 和 setter 的 this 上下文自动地绑定为 Vue 实例。这就很强大了，在计算属性中定义的函数里面可以直接使用指向了vue实例的this。

```
<div class="view">
   // 计算属性，好处在于会自动根据totalPrice显示不同内容
   {{showStr}}
</div>
```

```
var app = new Vue({
  el: '.view',
  data: {
    totalPrice: 18
  },
  computed: {  // Vue对象的computed属性
    // 计算属性，但该属性是一个方法
    showStr: function(){
       // this指向vue实例
      if(this.totalPrice < 20){
        return '金额小于20，没有优惠喔!'
```

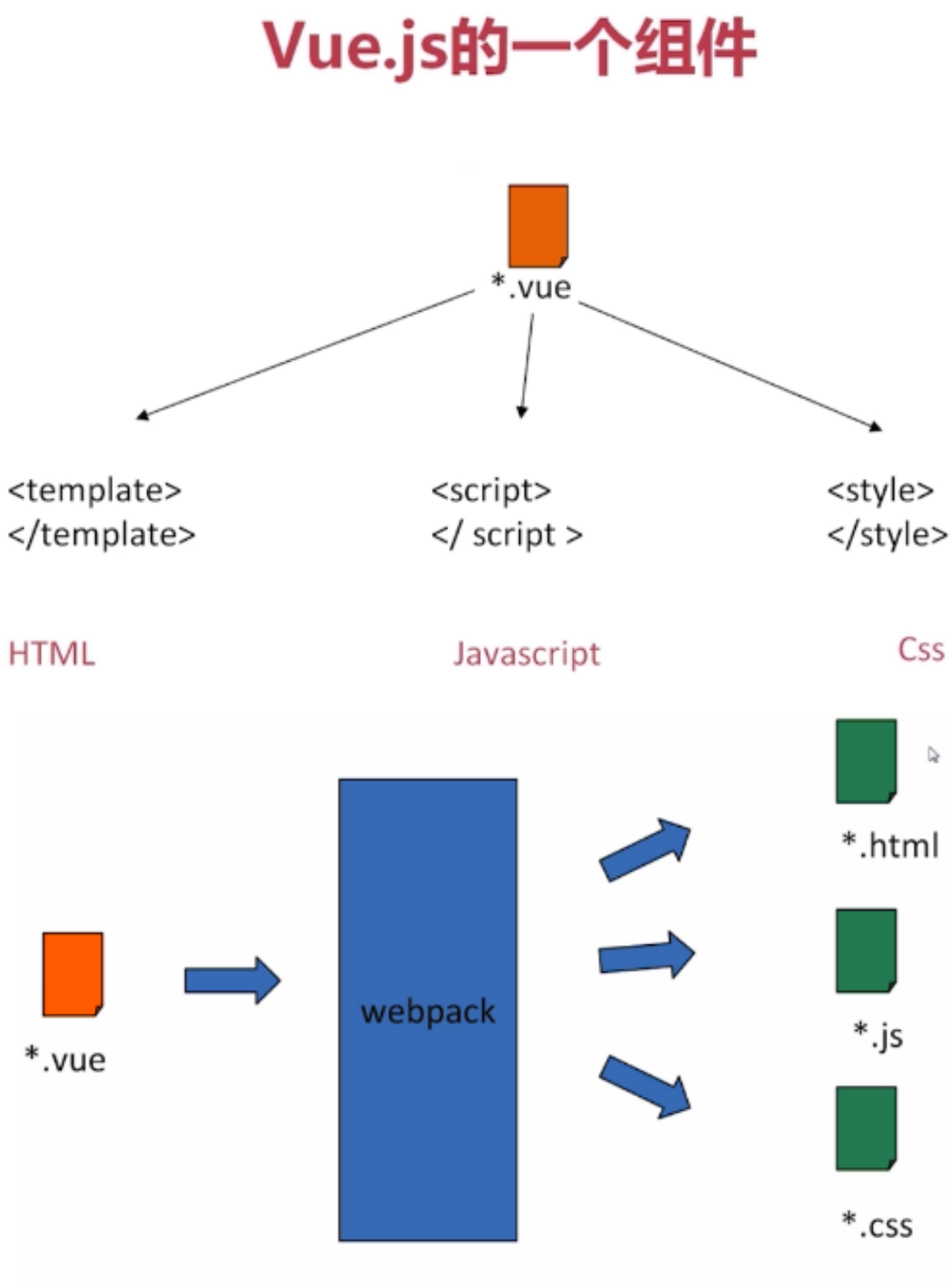
```
        } else {
          return '金额大于20，免配送费!'
        }
      }
    }
  });
```

- watch监听选项，设置了对象的监听方法 一个对象，键是需要观察的表达式，值是对应回调函数。值也可以是方法名，或者包含选项的对象。

```
  var view2 = new Vue({
    el: '#view2',
    data: {
      a: 1
    },
    methods: {
      doSomething: function(){
        this.a++;
      }
    },
    watch: {
      a: function(newvalue, oldvalue){
        console.log(newvalue, oldvalue);
      }
    }
  });
```

三、Vue组件

Vue组件`*.vue`由三部分组成分别是`<template></template>`HTML代码、`<script></script>`javascript脚本、`<style></style>`css样式。***【官方推荐，将三个部分都写到一个`.vue`文件中】***



四、vue的安装

- 直接通过`script`引入

```
<script src="https://unpkg.com/vue/dist/vue.js"></script>
```

> 直接下载并用 script 标签引入，Vue 会被注册为一个全局变量。重要提示：在开发时请用开发版本，遇到常见错误它会给出友好的警告。

- 通过npm

```
# 最新稳定版
$ npm install vue
```

> 需要先安装Node环境，npm其实是Node.js的包管理工具！

> 安装参考: https://www.liaoxuefeng.com/wiki/001434446689867b27157e896e74d51a89c25cc8b43bdb3000/001434501418434 88beddae2a1044cab5acb5125baf0882000

> 更换npm的镜像: npm config set registry http://registry.npm.taobao.org/

- 通过Vue-cli脚手架 [备注: Vue-CLI 3.3.0版本]

  Vue-cli是Vue的脚手架工具，是官方命令行工具 (CLI)，脚手架即编写好基础的代码，包括目录结构、本地调试、代码部署、热加载、单元测试。

```
    - $ npm install -g @vue/cli   // 安装操作
    - $ vue -V  // 查看版本，检查是否安装成功
    - $ vue create <project-name>
      例如 $ vue create hello-vue  // 官方模版，也可以使用自定义等模版

    // 基本项目设置
    $ vue create hello-vue
      👉  Get started with the following commands:

          $ cd hello-vue
          $ npm run serve

    // 之后再按照上面提示执行对应命令即可
    $ cd hello-vue
    $ npm run serve // 开启服务器，之后提示例如打开http://localhost:8080页面，如果可以打开说明配置完成
```