vue阶段总结

> 暗号：笔记

1. 指令与自定义指令

1）常用指令

1. **在vue中，你可以在HTML标签上使用 if，else来进行条件判断：**

```
<div id="app">
  <p v-if="seen">现在你看到我了seen</p>
  <p v-else-if="seen1">现在你看到我了seen</p>
  <p v-else>现在你看到我了</p>
</div>
```

2. **还可以使用 for 循环：**

```
<div id="app">
  <ol>
    <li v-for="todo in todos">
      {{ todo.text }}
    </li>
  </ol>
</div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    todos: [
      { text: '学习 JavaScript' },
      { text: '学习 Vue' },
      { text: '整个牛项目' }
    ]
  }
})
```

这对于需要渲染多个重复内容的时候非常有用！

3. **v-if 指令可直接控制该元素是否渲染，而 v-show 指令则是将元素先渲染出来，在样式上对其进行隐藏，满足条件后再进行展示**

```
<div id="app">
  <p v-show="seen">我存在，但你看不见</p>
</div>
```

4. **可以通过 v-on 指令给标签绑定一个事件**

```
<div id="app-5">
  <button v-on:click="handleClick">反转消息</button>
</div>
```

这个指令可以简写为 @

```
<div id="app-5">
  <button @click="handleClick">反转消息</button>
</div>
```

他们的效果是一样的

5. **使用 v-bind 指令，可以实现数据的动态绑定：**

```
<div v-bind:id="dynamicId"></div>
```

这个 dynamicId 是一个变量，div标签的id可以随着这个变量的改变而改变；它也有一个简写，就是一个冒号 `:`

```
<div :id="dynamicId"></div>
```

6. **使用 v-model 指令，可以实现数据的双向绑定：**

```
<input v-model="message" placeholder="edit me">
<p>Message is: {{ message }}</p>
```

使用 v-model 指令绑定变量 message，那么当输入框输入一些值的时候，这个变量的值也会跟着发生变化；同样，当使用代码修改 message 的值的时候，输入框中的内容也会跟着变化；因此叫做双向绑定。

这个指令实际上是两个指令的简写：

```
<input v-bind:value="message"
       v-on:input="message = $event.target.value"
       placeholder="edit me">
<p>Message is: {{ message }}</p>
```

将input标签的 value 属性和message通过 v-bind 进行绑定，再使用v-on指令给标签绑定input事件，当事件触发时，将输入值赋值给 message，从而实现了双向绑定

以上这些基本就是最常用的vue指令了，除了这些，我们还可以自己定义指令！

2）自定义指令

使用 Vue.directive 方法注册一个全局指令：

```
// 注册一个全局自定义指令 `v-focus`
Vue.directive('focus', {
  // 当被绑定的元素插入到 DOM 中时……
  inserted: function (el) {
    // 聚焦元素
    el.focus()
  }
})
```

然后就可以在一个标签上使用该指令：

```
<input v-focus>
```

指令也有生命周期：

```
Vue.directive('my-directive', {
  bind: function () {},
  inserted: function () {},
  update: function () {},
  componentUpdated: function () {},
  unbind: function () {}
})
```

2. vue中的组件

在vue中，一个项目就是一个组件，这个组件被称为根组件，根组件里面又有很多组件，这些组件组成了整个项目。我们看到的一个页面，就是由许多个组件组合而成的。

使用 Vue.component 方法定义一个全局组件：

```
// 定义一个名为 button-counter 的新组件
Vue.component('button-counter', {
  data: function () {
    return {
      count: 0
    }
  },
  template: `<button v-on:click="count++">
               You clicked me {{ count }} times.
             </button>`
})
```

一个组件就像一个对象，它有自己的属性和方法，还有自己的模板，生命周期等等，这些通过一些选项或函数来定义

选项：

- data：用来定义组件的本地变量
- methods：用来定义组件中使用到的方法
- props：定义该组件可接收的参数，就像给构造函数传参一样
- component：定义组件中的局部组件，在此定义的组件只能在这个组件中使用
- computed：计算属性，定义一个变量，可以设置该变量的get和set方法，这个变量可能是通过其他数据加工之后的结果，当这个变量依赖的数据发生改变时，就会重新计算这个变量的值
- watch：监听器，可以监听一个变量的变化。当某个变量变化后需要做一些事情的时候，可以使用watch

生命周期函数：

1. beforeCreate
2. created
3. beforeMount
4. mounted
5. beforeUpdate
6. updated

7. beforeDestroy
8. destroyed

生命周期就是一个组件从出生到死亡的过程，生命周期函数就是这个过程中的某几个关键点，例如人的一生：出生，说话，上学，工作，结婚，生孩子，退休，死亡。

组件的渲染依赖很多数据，而有很多数据是需要从别处获取来的，那么我们就可以在组件的 created 方法中来获取数据，然后组件使用这些数据来渲染页面