Vue.js 基础

vue.js 简介

vue 是一个响应式的前端视图层框架。

响应式：我们在编写模版代码时，仅需要关注数据的变化即可，数据变化 UI 即可随之变化，有联动的感觉。

视图层：类似于我们之前接触的其他前端模版一样，它仅仅只是 UI 层面的内容。

框架：库提供要由其父代码调用的功能，而框架则定义整个应用程序设计。开发人员不调用框架；而是调用框架。相反，框架以某种特定方式调用和使用代码。

vue.js 的 hello world

我们可以通过这种形式，写两个简单 demo。

```
<div id="app">
    <p>Message is: {{ message }}</p>
</div>
<script>
    var app = new Vue({
        el: '#app',
        data: {
            message: 'hello world'
        }
    });
</script>
```

另一个例子，将 input 中的数据又能随时随地渲染到其他绑定到 DOM 元素上。

```
<div id="app">
    <input v-on:input="onInput" v-bind:value="message">
    <p>Message is: {{ message }}</p>
</div>
<script>
      var app = new Vue({
        el: '#app',
        data: {
            message: 'hello world'
        },
        methods: {
            onInput(e) {
                this.message = e.target.value;
            }
        }
```

```
    });
</script>
```

vue.js 引入

我们可以通过多种方式引入 vue ， 这里来详细的进行对比：

1. 直接引用 vue.js ， 适合小型项目或部分使用 vue
    1. 引用全部 vue.js，运行时编译及渲染
    2. 引用部分 vue.js，仅引入渲染部分
2. 使用 vue-cli 工程化启动整体 vue 项目

vue.js 细节

模版中的 {{}}

`{{}}` 内部接受一个表达式，这里我们要区分一下表达式和语句间的区别。

标签中的新属性

- v-bind：我们能将 data 中的值绑定到当前属性中，可简写为 :
- v-on：能够绑定实例中配置的事件，可简写为 @

- v-for：列表级别渲染，迭代渲染所有子元素
- v-if/v-else/v-show：控制子元素视图显隐
- v-model：应用于表单，创建与元素的双向绑定
- v-html：将最终值的结果渲染为 html
- v-text：等同于直接在文本处使用 `{{xx}}`

计算属性

大部分时候，我们在模版也就是 html 中写表达式会让模版变得复杂，所以我们可以通过计算属性来简化我们的模版。

但大多数时候，我们也可以通过定义方法的形式来直接在表达式内调用函数。不过计算属性也可以模拟出使用参数的形式。

vue.js 组件

上面示例中的 `new Vue` 实例化之后，在具体的元素上实例化了根节点。对于框架而言，一般都会给用户提供一种复用的方式，而在 vue 中，我们可以通过定义组件的方式，来实现模版的复用来减少代码。

```
Vue.component('hello-world', {
  data: function () {
    return {
      message: 'hello world'
    }
  },
  template: '<p>{{message}}</p>'
});

new Vue({ el: '#app' })
```

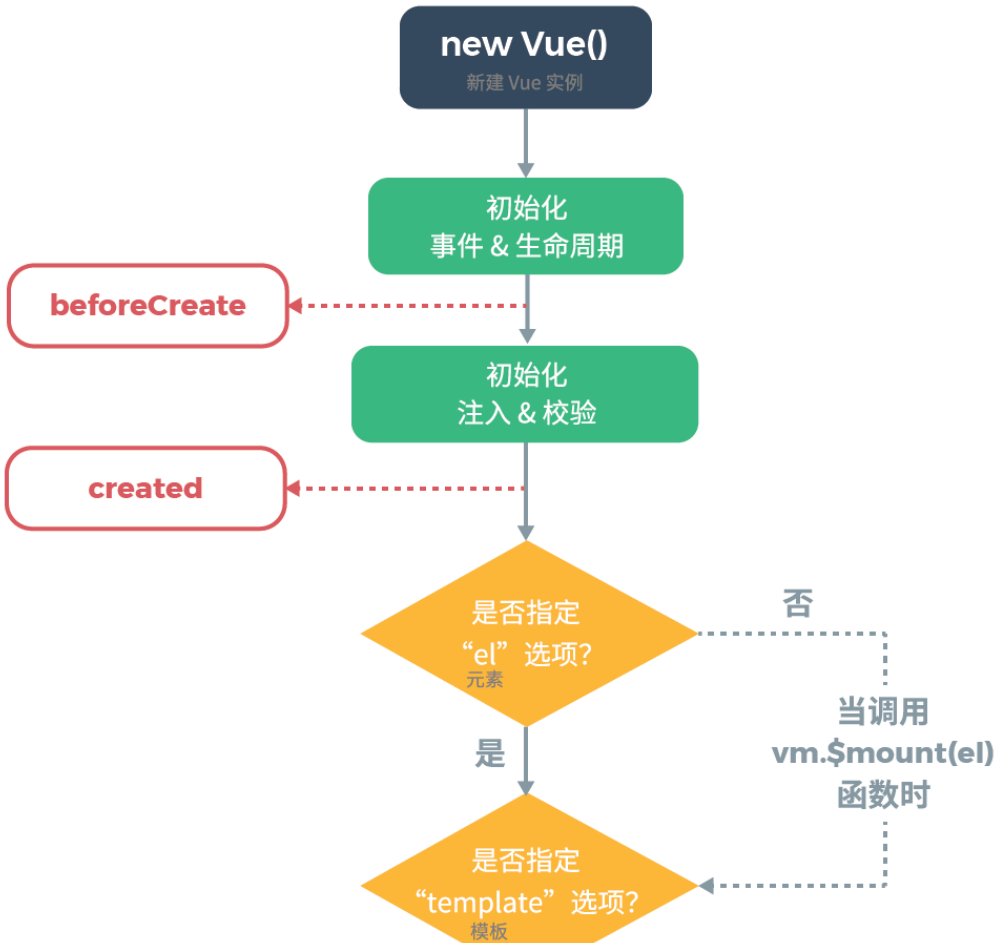
```
<div>
  <hello-world></hello-world>
</div>
```

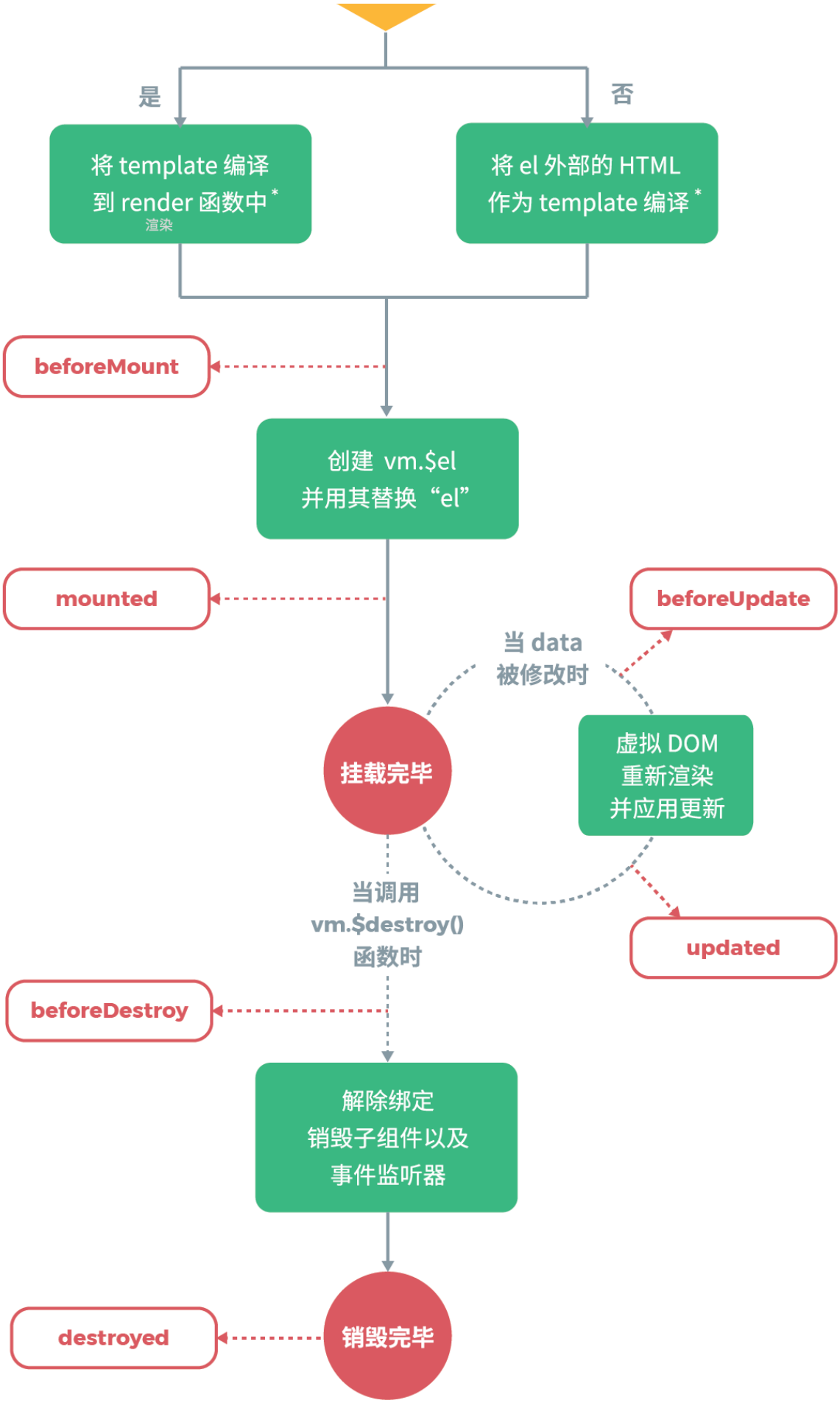
区分有状态组件与无状态组件

大部分时候，我们需要区分一些具有副作用的组件，例如某些组件我们需要发送 ajax 请求之后渲染一些数据，这时候我们就需要将这部分数据内容进行一个区分，推荐做法将 UI 部分渲染于子组件中，做一个只通过传入数据渲染的无状态组件，而在有副作用组件中维护所有的数据。

生命周期

实例化了 Vue 之后，随着内部逻辑的推进，慢慢的 Vue 会对应执行配置的部分方法，这部分方法名是提前约定好的，按照官网的图为：




是
否
将 template 编译
到 render 函数中*
渲染
将 el 外部的 HTML
作为 template 编译*
beforeMount
创建 vm.$el
并用其替换 “el”
mounted
beforeUpdate
当 data
被修改时
挂载完毕
虚拟 DOM
重新渲染
并应用更新
当调用
vm.$destroy()
函数时
updated
beforeDestroy
解除绑定
销毁子组件以及
事件监听器
destroyed
销毁完毕

* 如果使用构造生成文件（例如构造单文件组件），
模板编译将提前执行