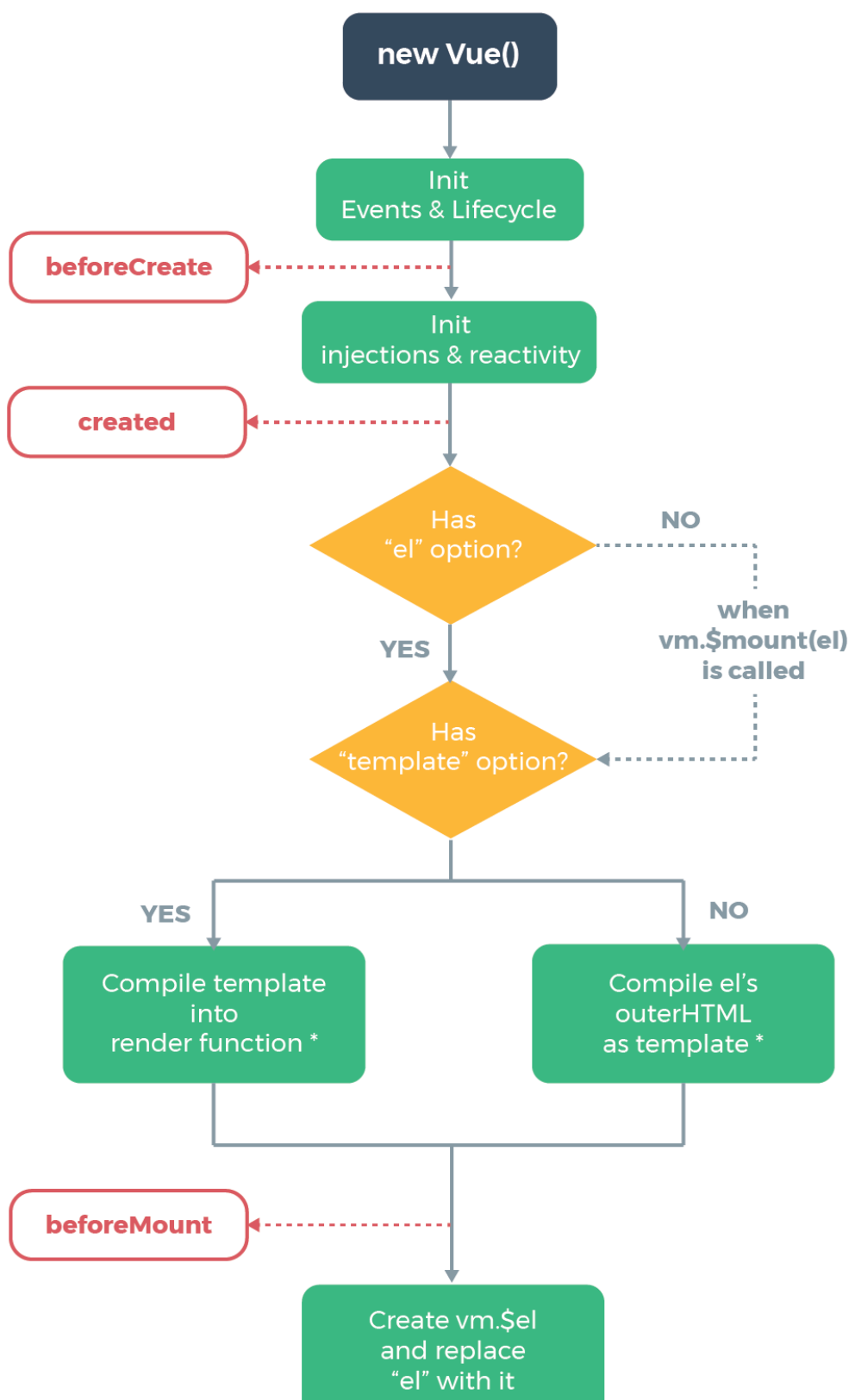


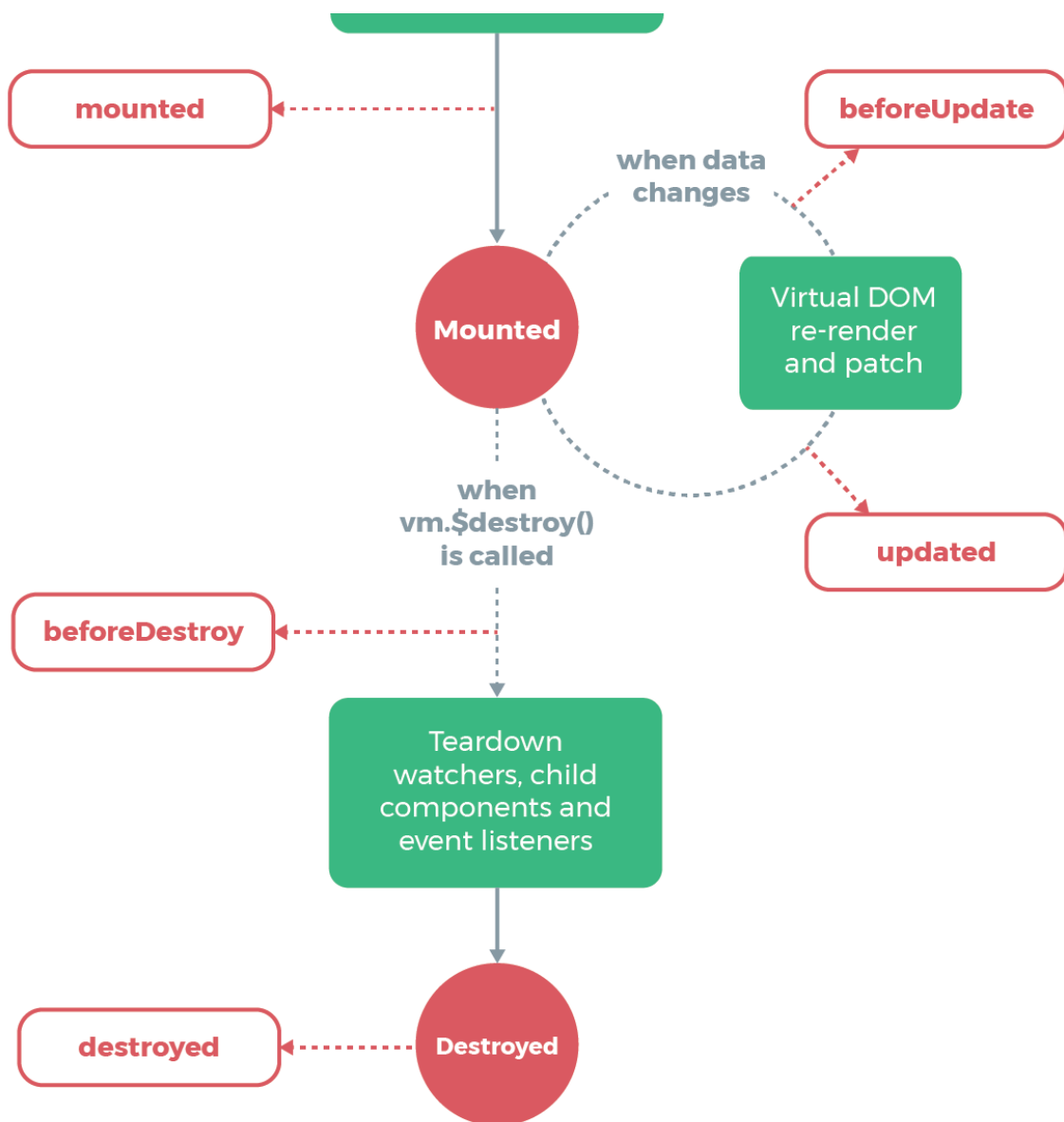
1.每个 Vue 应用都是通过用 `Vue` 函数创建一个新的 Vue 实例开始的:

```
var vm = new Vue({ // 选项 })
```

2.实例生命周期钩子

每个 Vue 实例在被创建时都要经过一系列的初始化过程——例如，需要设置数据监听、编译模板、将实例挂载到 DOM 并在数据变化时更新 DOM 等。同时在这个过程中也会运行一些叫做生命周期钩子的函数，这给了用户在不同阶段添加自己的代码的机会





* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

3.计算属性和侦听器

1.你可以像绑定普通 property 一样在模板中绑定计算属性。Vue 知道 `vm.reversedMessage` 依赖于 `vm.message`，因此当 `vm.message` 发生改变时，所有依赖 `vm.reversedMessage` 的绑定也会更新。而且最妙的是我们已经以声明的方式创建了这种依赖关系：计算属性的 getter 函数是没有副作用 (side effect) 的，这使它更易于测试和理解

2.我们可以将同一函数定义为一个方法而不是一个计算属性。两种方式的结果确实是完全相同的。然而，不同的是计算属性是基于它们的响应式依赖进行缓存的。只在相关响应式依赖发生改变时它们才会重新求值。这就意味着只要 `message` 还没有发生改变，多次访问 `reversedMessage` 计算属性会立即返回之前的计算结果，而不必再次执行函数。

这也同样意味着下面的计算属性将不再更新，因为 `Date.now()` 不是响应式依赖：

```
computed: { now: function () { return Date.now() } }
```

相比之下，每当触发重新渲染时，调用方法将总会再次执行函数。

我们为什么需要缓存？假设我们有一个性能开销比较大的计算属性 A，它需要遍历一个巨大的数组并做大量的计算。然后我们可能有其他的计算属性依赖于 A。如果没有缓存，我们将不可避免的多次执行 A 的 getter！如果你不希望有缓存，请用方法来替代。

2.computed和watch的区别

computed可以直接对被改变进一步触发的改变进行监控，而watch只能监控那些可能发生改变的值

```
var vm = new Vue({
  el: '#demo',
  data: {
    firstName: 'Foo',
    lastName: 'Bar',
    fullName: 'Foo Bar'
  },
  watch: {
    firstName: function (val) {
      this.fullName = val + ' ' + this.lastName
    },
    lastName: function (val) {
      this.fullName = this.firstName + ' ' + val
    }
  }
})
```

上面代码是命令式且重复的。将它与计算属性的版本进行比较：

```
var vm = new Vue({
  el: '#demo',
  data: {
    firstName: 'Foo',
    lastName: 'Bar'
  },
  computed: {
    fullName: function () {
      return this.firstName + ' ' + this.lastName
    }
  }
})
```

说白了就是计算属性是可以直接根据对应的属性得到相关的值。而watch需要值改变后再触发相关的动作才行。

计算属性的 setter

计算属性默认只有 getter，不过在需要时你也可以提供一个 setter：

```
// ... computed: { fullName: { // getter get: function () { return this.firstName + ' ' + this.lastName }, // setter set: function (newValue) { var names = newValue.split(' ')
this.firstName = names[0] this.lastName = names[names.length - 1] } } } // ...
```

现在再运行 `vm.fullName = 'John Doe'` 时，setter 会被调用，`vm.firstName` 和 `vm.lastName` 也会相应地被更新。