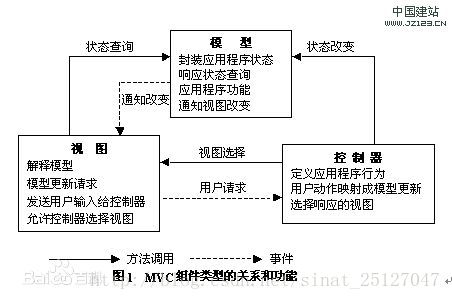
## Vue

**vue是一套用于构建用户界面的渐进式框架，采用的是以MVVM为基础 自底向上增量开发的设计模式，并且vue的核心库只针对视图层，在和其他库或者项目整合使用时有着极大的优势。**

**MVC和MVVM（前端架构模式）**

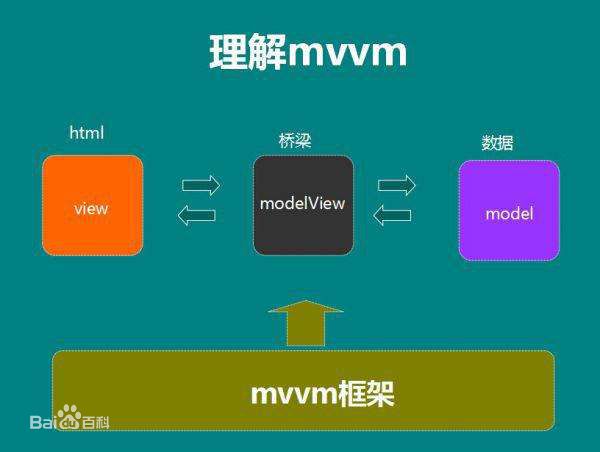
**MVC：**

**M：Model模型，拥有数据以及操作数据具体的业务逻辑；**

**V：view视图，具体的用户页面，HTML+CSS**

**C：controller控制器，负责将 View 中用户的动作传达给 Model，将 Model 的数据通过 View 展现出来。**

**用户操作> View (负责接受用户的输入操作)>Controller（业务逻辑处理）>Model（数据持久化）>View（将结果通过View反馈给用户）**

**MVVM：**

**VM：ViewModel视图模型，主要包括界面逻辑和模型数据封装**

**优势：**

**1. 低耦合。视图（View）可以独立于Model变化和修改，一个ViewModel可以绑定到不同的"View"上，当View变化的时候Model可以不变，当Model变化的时候View也可以不变。**

**2. 可重用性。你可以把一些视图逻辑放在一个ViewModel里面，让很多view重用这段视图逻辑。**

**3. 独立开发。开发人员可以专注于业务逻辑和数据的开发（ViewModel），设计人员可以专注于页面设计，使用Expression Blend可以很容易设计界面并生成xaml代码。**

**4. 可测试。界面元素向来是比较难于测试的，而现在测试可以针对ViewModel来写。**

**渐进式框架：**

**对于渐进式框架的概念并没有明确的定义，个人的理解是主张最少的框架，以Angular为例，在使用它的时候必须遵守着它的一些规则比如模块机制，依赖注入等等。而Vue则不同，因为它是渐进式框架所以它没有那么多的硬性要求，你可以在一个大项目中用上几个适合使用Vue的组件使用，也可以用vue+vue-router+vuex+axios全家桶完成整体的项目开发，没有做多余的工作即为渐进式。**

**自底向上增量开发：**

**在我们具体的项目开发所运用的框架中，很难确保框架以及代码的复杂程度的契合度，假如仅仅需要Jquery中的某个很小的功能却要引入整个JS文件是不是显得有些冗余，又或者框架本身不足以解决项目中遇见的所有问题，这时Vue自底向上增量开发的优势便显露无遗。Vue的核心虽然很小，但它有着完善的配套工具如MINT-UI、ELEMENT等，当把他们一个一个加进来的时候，就可以组合成非常强大的栈，就可以涵盖其他的这些更完整的框架所涵盖的问题。**

**Vue特点：**

1. **声明式渲染**
2. **组件化开发**
3. **Virtual DOM (虚拟DOM)**

**vue双向数据绑定原理：**

**数据双向绑定即数据变化更新视图以及视图变化更新数据，视图变化更新数据只需对某个标签添加事件监听即可，而对于数据变化更新视图，vue采用的是数据劫持结合发布者-订阅者模式的方式，vue通过**Object.defineProperty() **来实现对数据的劫持，从而实现监听数据变化的目的，然后再通过**Object.defineProperty()**对属性设置set函数，当数据发生变化时会触发set函数执行相关的方法达到更新视图的效果。这就是Vue双向绑定数据的原理。**

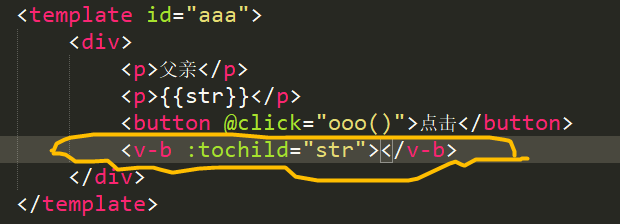
**附：发布者-订阅者模式**

1. **监听器Observer，用来劫持并监听所有属性，并通知订阅者。**
2. **订阅者Watcher，每一个Watcher都绑定一个更新函数，watcher可以收到属性的变化通知并执行相应的函数，从而更新视图。**
3. **解析器Compile，可以扫描和解析每个节点的相关指令（v-model，v-on等指令），如果节点存在v-model，v-on等指令，则解析器Compile初始化这类节点的模板数据，使之可以显示在视图上，然后初始化相应的订阅者（Watcher）。**

**父子间传值：**

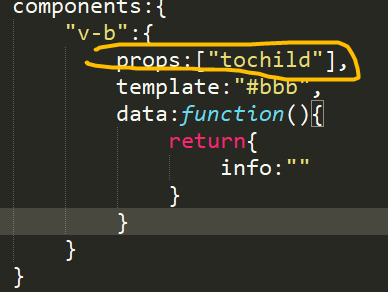
**父到子：**

**在父组件中的子组件标签上绑定自定义属性，如下图所示**

****

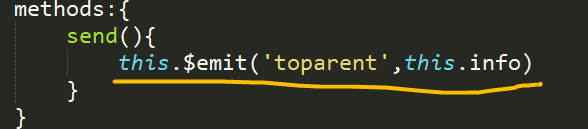
**然后在子组件中通过props接收，并且可以直接当作data内部的属性直接{{tochild}}使用**

**注：可以绑定多个属性，同时接收，如props：[“tochild”,”aaa”]**

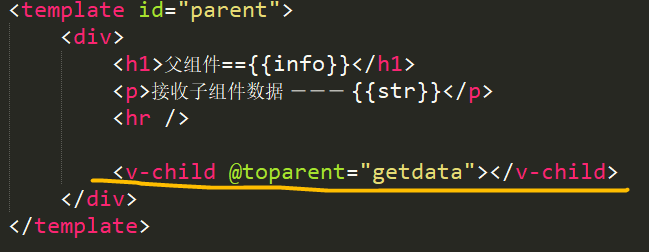
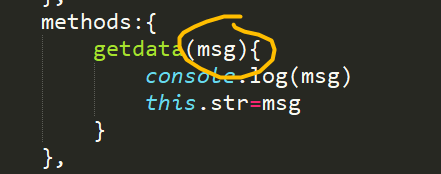
****

**子到父：**

**子组件通过$emit通过自定义方法向父组件发送数据，如：**

****

**在父组件中的子组件标签上绑定自定义方法接收子组件信息，如下图所示**

** **

**Vuex**

**Vuex官方的概念是专为Vue.js应用程序开发的状态管理模式。特点是采用集中式的存储模式管理应用的所有组件的状态。在我看来，vuex就是把需要共享的变量全部存储在一个对象里，然后把这个对象放在顶层组件中供其他组件使用。**

**为什么要用以及什么时候要用：在开发项目中经常会遇到当前组件的状态需要传递给其他组件，根据已知的父子间传值方式传递存在着很多弊端，例如通信双方的关系复杂甚至没有关系，又或者一个状态需要传递给其他多个组件时都会很麻烦，难以维护。这是vuex的存在就完美的解决了这个问题。**

**怎么用：vuex是vue全家桶的一部分，主要包含有以下五个核心对象**

1. **state：存储状态，即存储公共的变量值。**
2. **mutation：提交状态修改。这是vuex中唯一修改state的方式，不支持异步，第一个参数默认是state。外部通过this.$store.commit(‘aaa’,18)的方式使用。（aaa在mutations中需要定义）。**
3. **getters：派生状态。也就是set、get中的get，有两个可选参数：state、getters分别可以获取state中的变量和其他的getters。外部调用方式：store.getters.personInfo()。就和vue的computed差不多。**

**4，actions：和mutations类似。不过actions支持异步操作。第一个参数默认是和store具有相同参数属性的对象。外部调用方式：store.dispatch('nameAsyn')。**

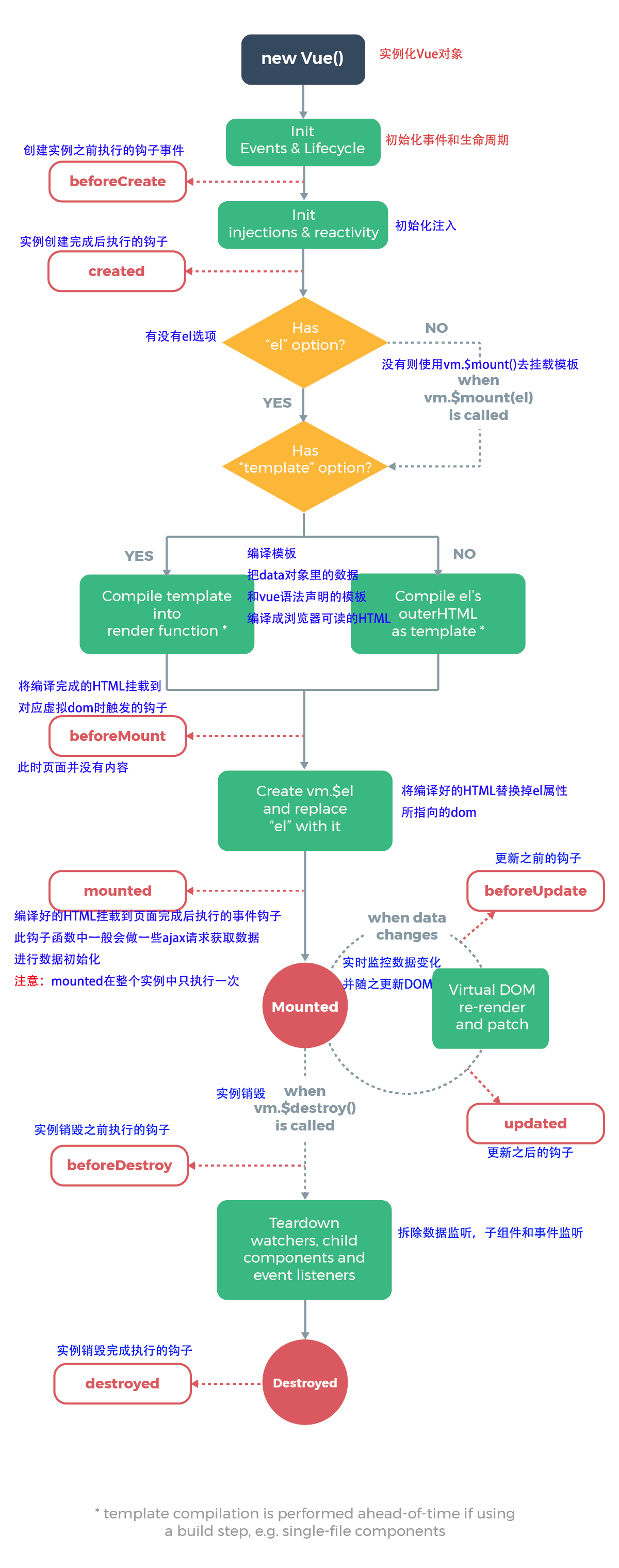
**5，modules：store的子模块，内容就相当于是store的一个实例。调用方式和前面介绍的相似，只是要加上当前子模块名，如：store.a.getters.xxx()。**

**在vue-cli搭建的脚手架中**

**安装依赖>>创建store.js（仓库的创建、数据的存储和操作，可以分开细化着写）>>挂载到vue实例化对象里（入口文件main.js引入）**

**将公用信息例如用户名、余额等在store.state中设置相关的参数，在其他组件中使用this.$store.state.xxx即可使用，并且可以直接赋值修改。//？？？？（官方的定义是通过提交mutation修改，方便记录每一次状态的修改，直接赋值的方法应该没有什么大问题，只是不规范）**

**Vue生命周期：**



**每个Vue实例在被创建时都要经过一系列的初始化过程，即开始创建、初始化数据、编译模板、挂载DOM、渲染-更新-渲染、销毁等一系列过程，这些被称为vue的生命周期。**

**大致分为三个阶段：初始化、运行中、销毁**

**常用的钩子函数共有八个，即**

**beforCreate,created,**

**beforMount,mounted,beforUpdate,updated,**

**beforDestory,destroyed**

1. **在组件通过new vue（）创建vue实例化对象后会初始化时间和生命周期，然后首先执行beforCreated钩子函数，此时进行数据观测，数据尚未挂载，data的值为undefined。接着执行created钩子函数，数据开始和data绑定，但此时DOM仍未被挂载，el的值仍为undefined。在此时已经可以获取到数据，并且修改数据不会触发其他钩子函数，可以在此钩子函数内做初始数据的获取。**
2. **在执行beforMount之前它首先会判断实例化对象内部是否有el选项，即有没有挂载DOM的节点，如果没有则会停止生命周期，若存在则编译模板（template）为虚拟dom放入到render函数中准备渲染，然后执行beforMount钩子函数，在这个函数中el已经被初始化，但仍是以虚拟DOM的形式存在，因此{{message}}仍未被渲染。执行mounted钩子函数的时候，真实DOM已经被渲染完成，事件数据均已挂载完毕，可以操作真实DOM。（在此处修改data数据将会触发update操作）**
3. **当data数据发生变化的时候，会立即执行beforUpdate，此时vue虚拟DOM机制会重新构建虚拟DOM与现有DOM树利用DIFF算法进行对比之后重新渲染，updated钩子函数和mounted钩子函数大同小异，在这里可以操作更新后的DOM。**
4. **当调用$destroy方法后，beforDestory钩子函数会立即执行，此时仍可以操作DOM，可以做一些善后工作，destroyed执行完毕后Vue实例已被销毁，所有事件绑定事件监听均被移除。**

**简单的使用方法：**

**beforecreate : 举个栗子：可以在这加个loading事件   
created ：在这结束loading，还做一些初始化，实现函数自执行   
mounted ： 在这发起后端请求，拿回数据，配合路由钩子做一些事情  
beforeDestroy： 你确认删除XX吗？ destroyed ：当前组件已被删除，清空相关内容**

**Vue路由传参的几种方式（vue-cli）：**

1. **router-link直接通过**

**:to={ path:’/detail’,params:{key:value,...} ,query:{key:value,...} }**

**然后通过this.$router.query/params.key获取**

1. **编程式路由需要在事件内部加相应参数**

**this.$router.push( {path:’/detail’,params:{ key:value } } )**

**获取方式同router-link一样**

**注：嵌套式路由在传参时需要注意运用watch监听，不然视图无法更新。**

**Watch:{ ‘$route’(a){ //a即为params对象 } }**