# 一、Vue生命周期

1、在beforeCreate之前，先初始化一些实例上的参数参数（$refs、$children、$parent、\_events、$slots等），然后调用beforeCreate钩子函数

2、执行完beforeCreate后，会将data、props、methods、inject、computed属性中的字段挂载到实例上，使之可以直接this.xxx进行访问。还有就是对data和props、inject实现响应式。并对watch的字段新建watcher。其中还包括格式化props为统一的格式，即对象，有type，value，default，required等字段。然后调用created钩子函数。

3、然后将el选项，赋值给$el，执行beforeMount钩子函数。

4、调用完beforeMount之后，会调用\_render函数渲染VNode，然后是\_update方法将VNode patch到真实DOM，执行mounted钩子。

5、beforeUpdate和updated是相对数据更新的。

6、beforeDestroy 钩子函数的执行时机是在 $destroy 函数执行最开始的地方，接着执行了一系列的销毁动作，包括从 parent 的 $children 中删掉自身，删除 watcher，当前渲染的 VNode 执行销毁钩子函数等，执行完毕后再调用 destroy 钩子函数。

| **生命周期钩子** | **详细** |
| --- | --- |
| beforeCreate | 在实例初始化之后，数据观测(data observer) 和 event/watcher 事件配置之前被调用。 |
| created | 实例已经创建完成之后被调用。在这一步，实例已完成以下的配置：数据观测(data observer)，属性和方法的运算， watch/event 事件回调。  在执行data()方法前props属性有数据已经可以访问，watch和computed监听函数此时为null，此时this.computed里的计算属性值为undefined。data函数执行完后，watch和computed监听函数才可用，因为data函数执行完后，data函数return的属性这时才可用。然而，挂载阶段还没开始，$el 属性目前不可见。 |
| beforeMount | 在挂载开始之前被调用：相关的 render 函数首次被调用。 |
| mounted | el 被新创建的 vm.$el 替换，并挂载到实例上去之后调用该钩子。如果 root 实例挂载了一个文档内元素，当 mounted 被调用时 vm.$el 也在文档内。 |
| beforeUpdate | 数据更新时调用，发生在虚拟 DOM 重新渲染和打补丁之前。你可以在这个钩子中进一步地更改状态，这不会触发附加的重渲染过程。 |
| updated | 由于数据更改导致的虚拟 DOM 重新渲染和打补丁，在这之后会调用该钩子。当这个钩子被调用时，组件 DOM 已经更新，所以你现在可以执行依赖于 DOM 的操作。 |
| activated | keep-alive 组件激活时调用。 |
| deactivated | keep-alive 组件停用时调用。 |
| beforeDestroy | 实例销毁之前调用。在这一步，实例仍然完全可用。 |
| destroyed | Vue 实例销毁后调用。调用后，Vue 实例指示的所有东西都会解绑定，所有的事件监听器会被移除，所有的子实例也会被销毁。 |

# 二、生命周期执行顺序

加载渲染过程：

父beforeCreate->父created->父beforeMount->子beforeCreate->子created->子beforeMount->子mounted->父mounted

子组件更新过程：

父beforeUpdate->子beforeUpdate->子updated->父updated

父组件更新过程：

父beforeUpdate->父updated

销毁过程：

父beforeDestroy->子beforeDestroy->子destroyed->父destroyed

# 三、js

1、原型

function A() {

this.x = 'x';

}

A.prototype.x = 'xx';

var a = new A();

a.x //x

A.x //undefined

2、call、apply、bind