**Vue的生命周期**

beforeCreate：组件实例初始化时，这时组件的属性还未被计算（如data和methods）

created:组件实例创建完成，属性已计算完成，但还未挂载到dom中，不能访问$el

beforeMount：组件挂载前，此时是首次挂载前最后一次修改data的机会

mounted:组件完成首次挂载，此时可以ajax获取数据

beforeUpdate:组件即将更新

updated:组件更新完毕

beforeDestory:组件销毁前

destoryed:组件销毁后

**Vue组件的通信**

1: 父传子：在父组件中设置属性 v-bind:a=‘data’**;**子组件通过this.props获取属性

2：子传父：父组件自定义事件v-on:recive=’dosthing’，子组件触发this.$emit("recive ",a,..)

3：父子双向绑定 ，结合父传子和子传父

**Vue的组件包含的属性**

**1：**data(数据)--对象/方法 在模块化中它是一个方法函数 返回一个数据对象

2：模板(component) --对象 其中每一个属性的value也是一个对象，它有自己的template data methods等属性

3：挂载元素(el) --字符串属性

4：方法(methods) --对象 和计算属性一样 其中的每一个方法都是函数 在模板和视图调用方法时执行函数

5：生命周期(mounted...)

6：计算属性(computed)--对象 每一个计算属性是一个方法 返回根据data中数据计算和演绎之后的数据 它自动根据data数据的改变而改变

7：监听( watch)--对象 键为监听的数据名 值为函数 当数据发生变更时函数将会执行

8：指令(directives)--对象 每一个指令又是一个对象 对象里是钩子函数 key是钩子 value是函数

**Vue自定义组件：**

除了核心功能默认内置的指令 (v-model 和 v-show)，Vue 也允许注册自定义指令。注意，在 Vue2.0 中，代码复用和抽象的主要形式是组件。然而，有的情况下，你仍然需要对普通 DOM 元素进行底层操作，这时候就会用到自定义指令。

钩子： bind:第一次绑定时调用。只会调用一次。

inserted:在绑定元素插入父节点时调用，

updata；在包含指令的组件更新后调用

directive:{

"A":{

bind:function(el,binding){

el--指令绑定的元素 可用于直接操作dom

binding--一个对象包括value(向指令传入的值)

}

}

}

**Vue响应式的原理：**

Vue的响应式使开发中在修改组件data里的数据时自动更新视图和触发生命周期。当我们定于一个Vue实例的data属性的时候，Vue将遍历此对象所有的属性，并使用Object.defineProperty()把这些属性转化为含有set和get。这使得属性被修改和访问时实例可以追踪到，并通知变化。每个组件实例都有相应的 **watcher** 实例对象，它会在组件渲染的过程中把属性记录为依赖，之后当依赖项的 setter 被调用时，会通知 watcher 重新计算，从而致使它关联的组件得以更新。

**算法题：**

1. length为n-1的数组，包换值为1~n的不重复的元素，求不在数组中的元素是多少。

解法1：累加数组元素，和1~n的和比较，差为所求元素

解法2：异或位运算符，数组所有元素的^结果 与1~n元素的^结果 相异或

二：树的遍历：

BFS广度优先：从顶点出发访，横向访问其所有相邻节点，在依次横向访问其相邻节点的相邻节点，直到所有节点被访问。

DFS深度优先：从任选顶点v起 若当前顶点问的相邻顶点，访问与其中一个相邻的顶点，依次类推向下访问，若其相邻订点都访问过或没有则返回上一次访问的顶点，直到所有能与v顶点想通的顶点都被访问到。若图中还存在为访问的顶点则已该顶点起重复上述步骤。

用代码来变现的: 广度优先：每次获取顶点下的所有的可访问节点，顺序插入数组，循环访问（while）,先进先出，并循环上述操作，直到数组为空，

深度优先：每次获取顶点下的所有的可访问节点，倒序插入数组，后进先出，每次取最后一次插入的新元素