问题一：什么是vue.js中的自定义指令?

自定义一些指令对底层DOM进行操作

更多参考

Vue里面有许多内置的指令，比如`v-if`和`v-show`，这些丰富的指令能满足我们的绝大部分业务需求，不过在需要一些特殊功能时，我们仍然希望对`DOM`进行底层的操作，这时就要用到自定义指令。

问题二：自定义指令的几个钩子函数

- `bind`：只调用一次，指令第一次绑定到元素时调用。在这里可以进行一次性的初始化设置。
- `inserted`：被绑定元素插入父节点时调用 (仅保证父节点存在，但不一定已被插入文档中)。
- `update`：所在组件的 VNode 更新时调用，但是可能发生在其子 VNode 更新之前。指令的值可能发生了改变，也可能没有。但是你可以通过比较更新前后的值来忽略不必要的模板更新 。
- `componentUpdated`：指令所在组件的 VNode 及其子 VNode 全部更新后调用。
- `unbind`：只调用一次，指令与元素解绑时调用。

问题三：钩子函数参数

除了 el 之外，其它参数都应该是只读的，切勿进行修改。如果需要在钩子之间共享数据，建议通过元素的 dataset 来进行。

指令钩子函数会被传入以下参数：

- `el`：指令所绑定的元素，可以用来直接操作 DOM 。
- `binding`：一个对象，包含以下属性：
    - name：指令名，不包括 v- 前缀。
    - value：指令的绑定值，例如：`v-my-directive="1 + 1"` 中，绑定值为`2`。
    - oldValue：指令绑定的前一个值，仅在`update`和 `componentUpdated`钩子中可用。无论值是否改变都可用。

- expression：字符串形式的指令表达式。例如 `v-my-directive="1 + 1"` 中，表达式为 `"1 + 1"`。
- arg：传给指令的参数，可选。例如 `v-my-directive:foo` 中，参数为 "foo"。
- modifiers：一个包含修饰符的对象。例如：`v-my-directive.foo.bar` 中，修饰符对象为 `{ foo: true, bar: true }`。

- `vnode`：Vue 编译生成的虚拟节点。
- `oldVnode`：上一个虚拟节点，仅在 `update` 和 `componentUpdated` 钩子中可用。

问题四：如何在vue-cli中使用自定义指令？

文件结构：

```
.
├── src
│   ├── directives
│   │   ├── index.js
│   │   ├── modules
│   └── main.js
└── ...
```

在`modules`下新建`foucs.js`下

```
// 聚焦指令
export default {
  bind (el, binding, vnode) {},
  inserted (el, binding, vnode) {
    el.focus()
  },
  update (el, binding, vnode) {},
  componentUpdated (el, binding, vnode) {},
  unbind (el, binding, vnode) {}
}
```

在`src/directives/index.js`下

```
import focus from './modules/focus'
export {focus}
```

在src/main.js下，使用directives自定义指令

```
//引入自定义指令
import * as directives from './directives'
//注册指令
Object.keys(directives).forEach(k => Vue.directive(k, directives[k]));
```

在.vue组件中使用

```
<input v-focus type="text" />
```