使用keep-alive组件 进入页面经历的过程是 created -》mounted -》activated
当离开时会触发deactivated 再次进入时只会触发activated函数

1. 所以如果每次进入需要更新对应代码的话 需要将函数接口移至activated内部
2. 如果是通过$route跳转的话， 需要在activated函数中重新对其参数重新复制data

属性 prop
include： 字符串或者正则表达式 只有匹配的组件才会被缓存
exclude： 字符换或者正则表达式 匹配到的组件都不会被缓存

keep-alive的几种使用形式：

1. 父组件中keep-alive匹配组件

```
// 组件
export default {
  name: 'test-keep-alive',
  data () {
    return {
        includedComponents: "test-keep-alive"
    }
  }
}
```

```
<keep-alive include="test-keep-alive">
  <!-- 将缓存name为test-keep-alive的组件 -->
  <component></component>
</keep-alive>

<keep-alive include="a,b">
  <!-- 将缓存name为a或者b的组件，结合动态组件使用 -->
  <component :is="view"></component>
</keep-alive>

<!-- 使用正则表达式，需使用v-bind -->
<keep-alive :include="/a|b/">
  <component :is="view"></component>
</keep-alive>

<!-- 动态判断 -->
<keep-alive :include="includedComponents">
  <router-view></router-view>
</keep-alive>

<keep-alive exclude="test-keep-alive">
  <!-- 将不缓存name为test-keep-alive的组件 -->
  <component></component>
</keep-alive>
```

2. 结合$route缓存部分页面
使用$route.meta 中的keepAlive属性

```
<keep-alive>
    <router-view v-if="$route.meta.keepAlive"></router-view>
</keep-alive>
<router-view v-if="!$route.meta.keepAlive"></router-view>
```

需要在 `router` 中设置router的元信息meta：

```
//...router.js
export default new Router({
  routes: [
    {
      path: '/',
      name: 'Hello',
      component: Hello,
      meta: {
        keepAlive: false // 不需要缓存
      }
    },
    {
      path: '/page1',
      name: 'Page1',
      component: Page1,
      meta: {
        keepAlive: true // 需要被缓存
      }
    }
  ]
})
```

也有可能会出现我们路由间相互跳转时分不同情况来决定是否缓存页面 这个时候我们就可以使用第二种与$route结合的方式 在beforeRouteLeave钩子函数中设置to.meta.keepAlive的值