keep-alive：主要用于保留组件状态或避免重新渲染。

比如： 有一个列表页面和一个 详情页面，那么用户就会经常执行打开详情=>返回列表=>打开详情这样的话 列表 和 详情 都是一个频率很高的页面，那么就可以对列表组件使用<keep-alive></keep-alive>进行缓存，这样用户每次返回列表的时候，都能从缓存中快速渲染，而不是重新渲染。

1、属性：

- include:字符串或正则表达式。只有匹配的组件会被缓存。
- exclude：字符串或正则表达式。任何匹配的组件都不会被缓存。

2、用法：

包裹动态组件时，会缓存不活动的组件实例，而不是销毁它们。和 <transition>相似，<keep-alive>是一个抽象组件：它自身不会渲染一个 DOM 元素，也不会出现在父组件链中。

当组件在<keep-alive> 内被切换，在 2.2.0 及其更高版本中，

activated 和 deactivated生命周期 将会在 树内的所有嵌套组件中触发。

```
<!-- 基本 -->
<keep-alive>
  <component :is="view"></component>
</keep-alive>

<!-- 多个条件判断的子组件 -->
<keep-alive>
  <comp-a v-if="a > 1"></comp-a>
  <comp-b v-else></comp-b>
</keep-alive>

<!-- 和 `<transition>` 一起使用 -->
<transition>
  <keep-alive>
    <component :is="view"></component>
  </keep-alive>
</transition>
```

注意：<keep-alive>是用在其一个直属的子组件被开关的情形。如果你在其中有 v-for 则不会工作。如果有上述的多个条件性的子元素，<keep-alive> 要求同时只有一

个子元素被渲染。

3、include 和 exclude 属性的使用：

2.1.0 新增

include 和 exclude 属性允许组件有条件地缓存。二者都可以用逗号分隔字符串、正则表达式或一个数组来表示：

```
<!-- 逗号分隔字符串 -->
<keep-alive include="a,b">
  <component :is="view"></component>
</keep-alive>

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

<!-- 数组 (使用 `v-bind`) -->
<keep-alive :include="['a', 'b']">
  <component :is="view"></component>
</keep-alive>
```

匹配首先检查组件自身的 name 选项，如果 name 选项不可用，则匹配它的局部注册名称 (父组件 components 选项的键值)。匿名组件不能被匹配。

不会在函数式组件中正常工作，因为它们没有缓存实例。

1. 使用 router.meta 属性，预先定义需要缓存的组件

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

路由部分：

```
routes: [
    {
      path: '/test1',
      component: test1,
```

```
    meta: { keepAlive: true }     // 需要缓存
  },
  {
    path: '/test2',
    component: test2,
    meta: { keepAlive: false}    // 不需要缓存
  },
```

- test1 组件会被缓存，而 test2 组件不会被缓存。

2. 动态缓存 router-view 里面的部分组件页面

如果只想 router-view 里面某个、或某些页面组件被缓存，通常有如下两种办法：

- 使用 include/exclude 来实现
- 配合 router.meta 属性来实现

1）. 使用 include/exclude 来实现，每个组件中需要加 name 来匹配

- include：只有匹配的组件会被缓存（支持字符串或正则表达）
- exclude：任何匹配的组件都不会被缓存（支持字符串或正则表达）

```
// 只缓存 name 为 index 的组件
<keep-alive include="index">
  <router-view/>
</keep-alive>

// 不缓存 name 为 index 的组件
<keep-alive exclude="index">
  <router-view/>
</keep-alive>

// 只缓存 name 为 index 或 hello 的组件
<keep-alive include="index,hello">
  <router-view/>
</keep-alive>

// 只缓存以 in 开头的组件（使用正则表达式，需使用 v-bind）
<keep-alive :include="/^in.*/">
  <router-view/>
</keep-alive>
```

// 也可以动态绑定需要缓存的组件（tagsList：存储组件name值的数组，数组是js动态控制的）

```
<keep-alive :include="tagsList">
  <router-view/>
</keep-alive>
```

2）. 配合 router.meta 属性来实现

主要依赖 beforeRouteLeave函数动态设置 meta.keepAlive，示例代码如下：

```
export default {
  name: 'hello',
  //keep-alive钩子函数：组件被激活时调用
  activated() {
    console.log('首页被激活');
  },
  //keep-alive钩子函数：组件消失，被缓存时调用
  deactivated() {
    console.log('首页被缓存');
  },
  beforeRouteLeave(to, from, next) {
    //设置下一个路由的meta（即首页）
    to.meta.keepAlive = true;  // 让首页缓存，即不刷新
    next();
  }
}
</script>
```