在你的组件中使用 $route 会与路由紧密耦合，这限制了组件的灵活性，因为它只能用于特定的 URL。虽然这不一定是件坏事，但我们可以通过 props 配置来解除这种行为：
我们可以将下面的代码

```
const User = {
  template: '<div>User {{ $route.params.id }}</div>'
}
const routes = [{ path: '/user/:id', component: User }]

```

替换成

```
const User = {
  props: ['id'],
  template: '<div>User {{ id }}</div>'
}
const routes = [{ path: '/user/:id', component: User, props: true }]

```

这允许你在任何地方使用该组件，使得该组件更容易重用和测试。

布尔模式#

当 props 设置为 true 时，route.params 将被设置为组件的 props。

命名视图#

对于有命名视图的路由，你必须为每个命名视图定义 props 配置：

```
const routes = [
  {
    path: '/user/:id',
    components: { default: User, sidebar: Sidebar },
    props: { default: true, sidebar: false }
  }
]

```

对象模式#

当 props 是一个对象时，它将原样设置为组件 props。当 props 是静态的时候很有用。

```
const routes = [
  {
    path: '/promotion/from-newsletter',
    component: Promotion,
    props: { newsletterPopup: false }
  }
]

```

函数模式#

你可以创建一个返回 props 的函数。这允许你将参数转换为其他类型，将静态值与基于路由的值相结合等等。

```
const routes = [
  {
    path: '/search',
    component: SearchUser,
    props: route => ({ query: route.query.q })
  }
]

```

URL /search?q=vue 将传递 {query: 'vue'} 作为 props 传给 SearchUser 组件。

请尽可能保持 props 函数为无状态的，因为它只会在路由发生变化时起作用。如果你需要状态来定义 props，请使用包装组件，这样 vue 才可以对状态变化做出反应。