重定向也是通过 routes 配置来完成，下面例子是从 /home 重定向到 /：

```
const routes = [{ path: '/home', redirect: '/' }]

```

重定向的目标也可以是一个命名的路由：

```
const routes = [{ path: '/home', redirect: { name: 'homepage' } }]

```

甚至是一个方法，动态返回重定向目标：

```
const routes = [
  {
    // /search/screens -> /search?q=screens
    path: '/search/:searchText',
    redirect: to => {
      // 方法接收目标路由作为参数
      // return 重定向的字符串路径/路径对象
      return { path: '/search', query: { q: to.params.searchText } }
    },
  },
  {
    path: '/search',
    // ...
  },
]

```

请注意，**导航守卫并没有应用在跳转路由上，而仅仅应用在其目标上。**在上面的例子中，在 /home 路由中添加 beforeEnter 守卫不会有任何效果。

在写 redirect 的时候，可以省略 component 配置，因为它从来没有被直接访问过，所以没有组件要渲染。唯一的例外是嵌套路由：如果一个路由记录有 children 和 redirect 属性，它也应该有 component 属性。

相对重定向#

也可以重定向到相对位置：

```
const routes = [
  {
    path: '/users/:id/posts',
    redirect: to => {
      // 方法接收目标路由作为参数
      // return 重定向的字符串路径/路径对象
    },
  },
]

```

别名#

重定向是指当用户访问 /home 时，URL 会被▌替换，然后匹配成▌。那么什么是别名呢？
将 / 别名为 /home，意味着当用户访问 /home 时，URL 仍然是 /home，但会被匹配为用户正在访问 /。
上面对应的路由配置为：

```
const routes = [{ path: '/', component: Homepage, alias: '/home' }]

```

通过别名，你可以自由地将 UI 结构映射到一个任意的 URL，而不受配置的嵌套结构的限制。使别名以▌开头，以使嵌套路径中的路径成为绝对路径。你甚至可以将两者结合起来，用一个数组提供多个别名：

```
const routes = [
  {
    path: '/users',
    component: UsersLayout,
    children: [
      // 为这 3 个 URL 呈现 UserList
      // - /users
```

```
      // - /users/list
      // - /people
      { path: '', component: UserList, alias: ['/people', 'list'] },
    ],
  },
]

```

如果你的路由有参数，请确保在任何绝对别名中包含它们：

```
const routes = [
  {
    path: '/users/:id',
    component: UsersByIdLayout,
    children: [
      // 为这 3 个 URL 呈现 UserDetails
      // - /users/24
      // - /users/24/profile
      // - /24
      { path: 'profile', component: UserDetails, alias: ['/:id', ''] },
    ],
  },
]
```