除了使用 <router-link> 创建 a 标签来定义导航链接，我们还可以借助 router 的实例方法，通过编写代码来实现。

导航到不同的位置#

注意：在 Vue 实例中，你可以通过 $router **访问路由实例。因此你可以调用** this.$router.push。
想要导航到不同的 URL，可以使用 router.push 方法。这个方法会向 history 栈添加一个新的记录，所以，当用户点击浏览器后退按钮时，会回到之前的 URL。
当你点击 <router-link> 时，内部会调用这个方法，所以点击 <router-link :to="..."> 相当于调用 router.push(...)：

声明式	编程式
<router-link :to="...">	router.push(...)

该方法的参数可以是一个字符串路径，或者一个描述地址的对象。例如：

```
// 字符串路径
router.push('/users/eduardo')

// 带有路径的对象
router.push({ path: '/users/eduardo' })

///下面的这几种用法都非常重要
// 命名的路由，并加上参数，让路由建立 url
router.push({ name: 'user', params: { username: 'eduardo' } })
//query的会变成？后面的参数
// 带查询参数，结果是 /register?plan=private
router.push({ path: '/register', query: { plan: 'private' } })
//hash里面的会转换成#后面的参数
// 带 hash，结果是 /about#team
router.push({ path: '/about', hash: '#team' })

```

注意：如果提供了 path，params 会被忽略，上述例子中的 query 并不属于这种情况。取而代之的是下面例子的做法，你需要提供路由的 name 或手写完整的带有参数的 path：

```
const username = 'eduardo'
// 我们可以手动建立 url，但我们必须自己处理编码
```

```
router.push(`/user/${username}`) // -> /user/eduardo
// 同样
router.push({ path: `/user/${username}` }) // -> /user/eduardo
// 如果可能的话，使用 `name` 和 `params` 从自动 URL 编码中获益
router.push({ name: 'user', params: { username } }) // -> /user/eduardo
// `params` 不能与 `path` 一起使用
router.push({ path: '/user', params: { username } }) // -> /user
```

由于属性 to 与 router.push 接受的对象种类相同，所以两者的规则完全相同。

router.push 和所有其他导航方法都会返回一个 **Promise**，让我们可以等到导航完成后才知道是成功还是失败。我们将在 Navigation Handling 中详细介绍。

替换当前位置#

它的作用类似于 router.push，唯一不同的是，它在导航时不会向 history 添加新记录，正如它的名字所暗示的那样——它取代了当前的条目。

声明式	编程式
<router-link :to="..." replace>	router.replace(...)

也可以直接在传递给 router.push 的 routeLocation 中增加一个属性 replace: true：

```
router.push({ path: '/home', replace: true })
// 相当于
router.replace({ path: '/home' })
```

横跨历史#

该方法采用一个整数作为参数，表示在历史堆栈中前进或后退多少步，类似于 window.history.go(n)。

例子

```
// 向前移动一条记录，与 router.forward() 相同
router.go(1)

// 返回一条记录，与router.back() 相同
router.go(-1)
```

```
// 前进 3 条记录
router.go(3)

// 如果没有那么多记录，静默失败
router.go(-100)
router.go(100)

```

篡改历史#

你可能已经注意到，router.push、router.replace 和 router.go 是 window.history.pushState、window.history.replaceState 和 window.history.go 的翻版，它们确实模仿了 window.history 的 API。

因此，如果你已经熟悉 Browser History APIs，在使用 Vue Router 时，操作历史记录就会觉得很熟悉。

值得一提的是，无论在创建路由器实例时传递什么样的history 配置，Vue Router 的导航方法(push、replace、go)都能始终如一地工作。