1. 编程式 router.push 的声明式是什么样的？使用时有哪些注意的地方？
   - 声明式 <router-link :to="...">
   - 用法：router.push(location,onComplete? ,onAbort) loaction 是一个字符串（路径）或（描述地址的）对象。
     - router.push('home') //字符串路径
     - router.push({path: 'home'}) //对象
     - router.push({name:'user',params:{userId:123}})
     - router.push({path:'register',params:{userId:123}}) //对象中用了 path，params 将会失效
2. 编程式 router.replace 的声明式是什么样的？router.replace 和 router.push 区别？
   - <router-link :to="..." replace>
   - router.replace 不会向 history 中添加新纪录，而是替换
3. router.go(n)的意义是什么？参数 n 的意义。
   - router.go(n) 中 n 是一个整数。表示在 history 记录中向前或向后退几步。
   - router.go(n) 等价于 window.history.go(n) (history.go 亲测有效)
   - 当 n 为正值时，向前进 n 页；n 为负值时，向后退 n 页
   - 下面补充下 window.history 常用的 api
     - window.history.back() 和浏览器后退按钮相同
     - window.history.forward() 和浏览器中向前按钮相同
4. 标签中 to 属性填的是什么？举个例子。
   - to 后面一般接 path，如/dashboard 等
5. redirect 是什么意思？举例说明。redirect 如何取值？
   - redirect 是重定向的意思，用户访问 /a 时，URL 将会被替换成 /b，然后匹配路由为 /b 。比如，下面表示，/ 重定向到 /b
   - redirect 取值可以是路径（如上面），也可以是一个命名路由。
     ```
     redirect:{name:'foo'}
     ```
6. 别名又是什么意思？举例说明。
   - /a 的别名是/b 表示，当用户访问/b 是，URL 会保持在/b，但是实际路由匹配的是/a，就像用户访问的是/a 一样。sss
7. 如何去掉地址中的#？
   1. 需要在路由中，给出 404 页面。如：
      - `{ path: '*', redirect: '/404', hidden: true }`
   2. 需要后台进行对应配置。思路是：如果 URL 匹配不到任何静态资源，则应该返回同一个 index.html 页面，这个页面就是你 app 依赖的页面。
      - 以 ngix 为例，如下。如果没有匹配的，就返回 index 页面

8. 如何注册一个全局前置守卫？to，from,next 三个参数分别解释下？next 的三种调用参数说明下？
   - router.beforeEach((to,from,next) => {})
   - to:Route(即将要进入的目标路由对象) from:Route(正要离开的目标路由对象) next:Function(调用此方法，resolve 钩子，必须调用)

   - next()进行管道中的下一个参数 next(false) 中断当前导航 next('/')或 next({path:'/'}) 当前的导航被中断，并且跳转到不同的地址。 next(error) 传入的如果是个 Error 实例。导航会终止且错误会被传给 router.onError()注册过的回调
9. 如何注册全局后置钩子？这些够自己会改变导航本身吗？
   - router.afterEach((to,from) =>{})
   - 不会
10. 如何给路由配置直接定义独享的守卫？
    - 在定义 routes 时，加一个 beforeEnter 属性，和 patch、component 属性并列。deforeEnter 属性的参数和 beforeEach 的参数相同。都是 to,from 和 next。
11. 完整的导航解析流程？
    1. 导航被触发。
    2. 在失活的组件里调用离开守卫。
    3. 调用全局的 beforeEach 守卫。
    4. 在重用的组件里调用 beforeRouteUpdate 守卫 (2.2+)。
    5. 在路由配置里调用 beforeEnter。
    6. 解析异步路由组件。
    7. 在被激活的组件里调用 beforeRouteEnter。
    8. 调用全局的 beforeResolve 守卫 (2.5+)。
    9. 导航被确认。
    10. 调用全局的 afterEach 钩子。
    11. 触发 DOM 更新。
    12. 用创建好的实例调用 beforeRouteEnter 守卫中传给 next 的回调函数。
12. 什么是路由懒加载？如何做？
    - 将不同路由对应的组件分割成不同的代码块，然后当路由被访问的时候才加载对应组件。
    - 定义一个能够被 Webpack 自动代码分割的异步组件 `const Foo = () => import('./Foo.vue')`