TypeScript

下面是对路由的meta属性进行类型声明，这样就可以在组件中拥有类型提示了。
创建 `src/router.d.ts` 声明文件，内容如下

```
import 'vue-router'

declare module 'vue-router' {
  interface RouteMeta {
    // 是可选的
    isAdmin?: boolean
    // 每个路由都必须声明
    requiresAuth: boolean
  }
}

```

#自动加载

每次单独配置路由比较麻烦，我们可以将路由自动化加载配置。

```
//路由配置
const routes = []
//所有布局组件
const layouts = import.meta.globEager('../layouts/*.vue')
//所有页面组件
const views = import.meta.globEager('../views/**/*.vue')

//布局组件路由定义
Object.entries(layouts).forEach(([file, component]) => {
    //布局路由路径
    const path = file.split('/').pop().slice(0, -4)
    const route = {
        path: `/${path}`,
        component: component.default,
        children: [],
```

```
    }
    routes.push(route)
})

//页面组件路由定义
Object.entries(views).forEach(([file, component]) => {
    //视图路由
    const route = viewRoute(file, component.default)

    //视图路由添加到路由配置
    const group = viewGroup(file)
    group ? group.children.push(route) : routes.push(route)
})

//视图组件链接地址
function viewRoute(file, component) {
    const route = { path: '', component, route: component.route || {} }

    //页面自定义的路由
    if (route.route.path) {
        route.path = route.route.path
        return route
    }

    //跟据页面地址自动声明路由
    if (viewGroup(file)) {
        route.path = file.split('/').splice(3).join('/').slice(0, -4)
    } else {
        route.path = '/' + file.split('/').splice(2).join('/').slice(0, -4)
    }

    return route
}

//视图所在布局组件
function viewGroup(file) {
    return routes.find(group => group.children &&
file.includes(`views${group.path}/`))
}

```

```
export default routes

```

项目目录结构如下

```
src
├── layouts
|   └── admin.vue
├── router
|   ├── index.js
|   └── routes.js
└── views
    ├── admin
    |   └── home.vue
    └── index.vue
```