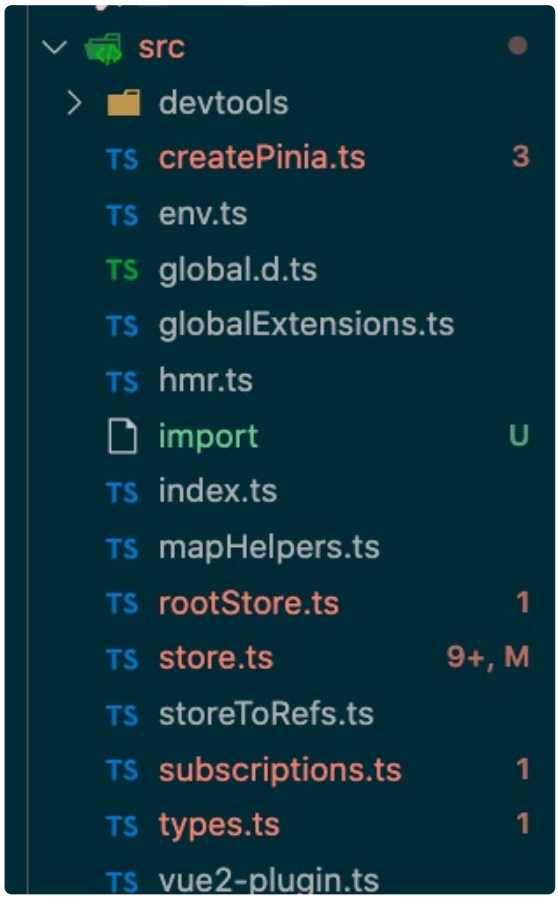
Pinia

Pinia源码-createPinia

createPinia函数的内部运行机制

pinia的源码放在packages/pinia/src



开发者使用pinia需要使用createPinia()让vue全局注册pinia插件为了可以让pinia在所有组件里面提供服务

> 每当vue插件被添加到应用程序中时，如果它是一个对象，就会调用 `install` 方法。如果它是一个 `function` ，则函数本身将被调用。在这两种情况下——它都会收到两个参数：由 Vue 的 `createApp` 生成的 `app` 对象和用户传入的选项。

那么先来捋一捋createPinia.ts 这个文件，

```
export function createPinia(): Pinia {
  const scope = effectScope(true)
  // NOTE: here we could check the window object for a state and directly set it
  // if there is anything like it with Vue 3 SSR
  const state = scope.run<Ref<Record<string, StateTree>>>(() =>
    ref<Record<string, StateTree>>({})
```

```
  )!

  let _p: Pinia['_p'] = []
  // plugins added before calling app.use(pinia)
  let toBeInstalled: PiniaPlugin[] = []

  const pinia: Pinia = markRaw({
    install(app: App) {
      // this allows calling useStore() outside of a component setup after
      // installing pinia's plugin
      setActivePinia(pinia)
      if (!isVue2) {
        pinia._a = app
        app.provide(piniaSymbol, pinia)
        app.config.globalProperties.$pinia = pinia
        /* istanbul ignore else */
        if (__DEV__ && IS_CLIENT) {
          registerPiniaDevtools(app, pinia)
        }
        toBeInstalled.forEach((plugin) => _p.push(plugin))
        toBeInstalled = []
      }
    },

    use(plugin) {
      if (!this._a && !isVue2) {
        toBeInstalled.push(plugin)
      } else {
        _p.push(plugin)
      }
      return this
    },

    _p,
    // it's actually undefined here
    // @ts-expect-error
    _a: null,
    _e: scope,
    _s: new Map<string, StoreGeneric>(),
    state,
  })

  // pinia devtools rely on dev only features so they cannot be forced unless
  // the dev build of Vue is used
  if (__DEV__ && IS_CLIENT) {
    pinia.use(devtoolsPlugin)
  }

  return pinia
}
```

代码简单，createPinia函数返回一个 pinia 实例对象，pinia对象有install方法，那么按照上面文档所说vue注

册pinia就会使用这个pinia 提供的install 方法，

---

pinia install 方法

```
install(app: App) {
// this allows calling useStore() outside of a component setup after
// installing pinia's plugin
  setActivePinia(pinia)
  if (!isVue2) {
    pinia._a = app
    app.provide(piniaSymbol, pinia)
    app.config.globalProperties.$pinia = pinia
    /* istanbul ignore else */
    if (__DEV__ && IS_CLIENT) {
      registerPiniaDevtools(app, pinia)
    }
    toBeInstalled.forEach((plugin) => _p.push(plugin))
    toBeInstalled = []
  }
}
```

1. setActivePinia(pinia)，内部实现就是让pinia 存储在一个全局的变量上面，为什么呢？为了让pinia可以在组件之外也可以使用useStore获取pinia
2. 当前使用的vue版本如果不是vue2，那么为pinia添加vue createApp创建的vue实例。
3. 使用app.provide方法添加所有组件可共享的数据，这里就是pinia。piniaSymbol是一个symbol ，为了不和其他app.provide提供的共享数据的key重名。app.provide的数据都可以在通过组件的 Inject使用。
4. app.config.globalProperties.$pinia, 注册一个全局可用的对象，所有组件实例都可以直接this.$pinia获取pinia
5. 注册Pinia，让其对Devtools可用
6. toBeInstalled 是一个数组类型，里面都是用户自定义的pinia的插件。_p是pinia对象的一个属性，用户自定义的插件通过toBeInstalled存储在 _p

注册Pinia插件

pinia对象提供了一个use方法接受一个参数，该参数就是用户自定义的插件逻辑，比如可以为一个函数。然后这个参数就会放入上面刚刚讲到的toBeInstalled。

```
use(plugin) {
  if (!this._a && !isVue2) {
    toBeInstalled.push(plugin)
  } else {
    _p.push(plugin)
  }
  return this
```

```
}
```

```
// 用户自定义pinia插件使用方法
function printMessage() {
    return {
        message: 'Hello Pinia Plugin'
    }
}

createPinia().use(printMessage)
```

上面的代码就是一个使用pinia 插件的例子，其最后的效果就是message 属性会挂在pinia实例，这么做就是让pinia可以通过用户自定义插件的功能进行功能扩展。

我们会在后面的章节讲解如何注册用户自定义插件。这里先知道一下插件的注册机制。

Pinia其他主要属性

pinia对象还加载了其他几个主要属性

```
// pinia的实例
_a: null,
// effectScope对象方便用来控制effect的取消，
// 使用场景
// vue的组件副作用会在unmount自动dispose
// 如果在组件之外使用则不会自动dispose
// effectScope帮助更容易的在组件之外使用响应式产生effect
// 的取消，比如当开发一个独立的package，
_e: scope,
// 多个store使用map结构配合id区分，
_s: new Map<string, StoreGeneric>(),
// 一个ref对象，存储了id: state的数据，即支持多个
// 不同id对state作出区分
state,
```

createPinia 就讲好了，