

再惠通用 Hooks 库 h/hooks Q2 总结

项目背景

再惠通用 Hooks 库，一个通用、高效、更好管理及扩展的通用 React Hooks 库

项目地址

[仓库地址] <https://pasta.zaihui.com.cn/fe/h/h-hooks> (复制后打开)

安装运行

```
yarn add @h/h-hooks@1.0.1
```

项目进度






- 目前输出 22 个通用 Hooks

- 主要来源

- Saas 组项目 14 个
 - 收录部分 ahooks 6 个
 - 其他来源 1 个

- 基建进度

- 基于 vite 库模式打包  杨海啸
 - 实现 dts 提取及打包写入  杨海啸
 - CI/CD 集成  杨海啸
 - 接入 fe-lint、lint-staged 等实现 commit-lint  白亚鹏
 - CI/CD 实现项目发布 hnpm  王亚  杨海啸
 - 接入 jest 及 @testing-library/react-hooks 用于测试 React hooks  汪莹
 - Todo

- 基于 lerna + yarn/pnpm + gulp + vite 实现 monorepo 多包管理  杨海啸
 - 解决操纵 DOM、BOM 的自定义 Hook 的测试问题  杨海啸  汪莹
 - 基于 React 生成一个 h-hooks 静态网站可供查阅  杨海啸  王亚

Hooks CheckList

查询 Hook 的一些相关信息

Hooks Checklist表格

目录结构

项目目录结构起初参考 [ahooks](#) 目录结构

目录结构：

```
1 .
2 |— config
3 |   |— buildTypes.ts
4 |— docs
5 |— packages
6 |   |— common
7 |   |   |— hooks
8 |   |   |— utils
9 |   |— mp
10 |   |   |— hooks
11 |   |   |— utils
12 |   |— web
13 |   |   |— hooks
14 |   |   |— utils
15 |— types
16 |— utils
17 |— index.ts
18 |— tsconfig.json
19 |— tsconfig.node.json
20 |— tsconfig.types.json
21 |— Dockerfile
22 |— babel.config.js
23 |— vite.config.ts
24 |— jest.config.js
25 |— package.json
26 |— README.md
27 |— yarn-error.log
28 |— yarn.lock
```

Usage

提供了两种打包方式

```
1 |— cjs
2 |   |— packages
3 |— es
4 |   |— index.d.ts
5 |   |— packages
```

1. CommonJS 方式

2. 原生 ESM 方式

ESM 打包提供了 dts 类型文件，推荐使用 ESM 打包后的文件。

剥离了依赖

h-hooks 是纯粹的源码，不会额外提供任何依赖。使用 Vite 剥离了依赖，h-hooks 不会为 hooks 提供依赖，而是会在宿主包里寻找依赖，否则使用会报错（因此，注释里要标注 h-hooks 的前置依赖）

注意

1. 对于 Hook 的引用，推荐按需引用某个 Hook 或子包，而不是全量引入
2. 使用 Hook 前，请根据注释/Markdown 检验前置依赖条件是否满足
3. 注意区分运行环境去引入对应子包的 Hook

Contribution

如何提交一个 Hook 至 h/hooks 仓库

发起 Issue

如果你希望能新增某个功能的 Hook，或想发起 PR 新增一个自定义 Hook，可以先建立一个 issue，说明该 Hooks 的应用场景及用法。探讨 Hook 是否足够通用、是否有优化空间等

Pull Request

基于已有 Hook（React Hooks or 项目中已有的 Hooks）来初始化一个新的 Hook

开发一个符合规范的自定义 Hook

1. 确定 Hook 的存放位置

首先，根据 Hook 的使用场景，确定你的 Hook 应该放在哪个子包目录下。通常来说，我们把 Hooks 按照运行环境、使用场景分为了三个子包：

- packages/common 用于存放仅 React 环境下使用的 Hook，比如 `useDeepCompareEffect`
- packages/web 用于存放在 Web 端使用频率比较高的 Hook

- packages/mp 用于存放在小程序环境下（Taro + React）使用频率比较高的 Hook

一个完整的 Hook 由以下几个部分组成

```
1 |— __tests__
2 |— demo
3 |— index.md
4 |— index.ts
```

2. source file 需要有完整的注释，标明其输入、输出、依赖及 example

```
1 import { useEffect, DependencyList } from 'react'
2
3 import useLatest from '../../common/hooks/useLatest'
4
5 /**
6  * auto scroll to top
7  * @param {array} deps dependencies which trigger scrolling
8  * @param {string} selector scrollable element selector
9  * @param {string} behavior scrollable element behavior
10  * @example
11  * @dependencies
12  * ``tsx
13  * useScrollToTop([page, status], '.module-data-content')
14  *
15  */
16 const useScrollToTop = (
17   deps: DependencyList = [],
18   selector = '',
19   behavior?: ScrollBehavior,
20 ): void => {
21   const container = useLatest<HTMLElement | null>(null)
22   useEffect(() => {
23     if (!container.current) {
24       const child = document.querySelector(selector)
25       container.current = child?.parentElement as HTMLElement
26     }
27     if (container.current?.scrollTop === 0) return
28     if (behavior && container.current?.scrollTo) {
29       container.current.scrollTo({
30         top: 0,
31         behavior,
32       })
33       return
34     }
35   }, deps)
```

```

35     container.current.scrollTop = 0
36   }, [...deps])
37 }
38
39 export default useScrollToTop
40

```

3. 一个完整的 MarkDown 描述文件，需要有完整的注释，标明其输入、输出及前置依赖

```

1  ---
2  nav:
3    path: /web/hooks
4  ---
5
6  # useScrollToTop
7
8  ## 基础用法
9
10 自动划至顶部
11
12  ### API
13
14  ```typescript
15  useScrollToTop(
16    deps: any[],
17    selector: string,
18    behavior: "auto" | "instant" | "smooth"
19  )
20  ```
21
22  ### Params
23
24  | 参数 | 说明 | 类型 | 默认值 |
25  | --- | --- | --- | --- |
26  | deps | 触发滚动的依赖项 | `array` | [] |
27  | selector | 可滚动元素选择器 | `string` | "" |
28  | behavior | 指定滚动元素的过渡动画类型 | `"auto" \| "instant" \| "smooth" | "
29

```

4. 完整的测试用例

应有完整的测试用例。（当前 Hook 测试 环境并不够完善，导致很多 Hook 没有测试用例）

5. 在子包根文件下导出它

如果 hook 有一些外部依赖，请判断它的 dependencies 类型，放在合适的位置。如果你不想依赖源码被打包进库，请在 vite.config.js 的 build.rollupOptions.external 中配置该依赖

6. checkList 更新文档