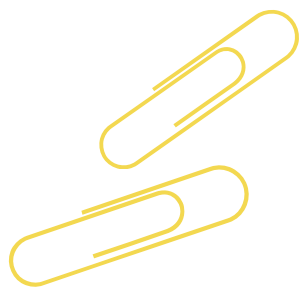


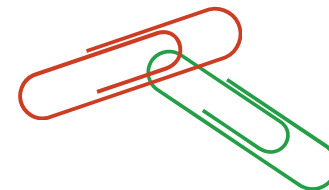
指南

React前端架构迁移



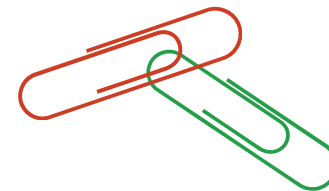
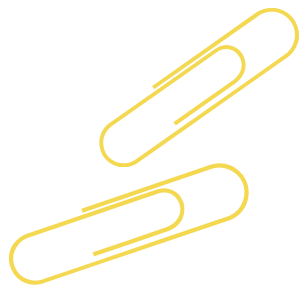


目录 CONTENTS



- ① Solv-libs 私仓
- ② Solv-libs State
- ③ Solv Components
- ④ SolvApp & React



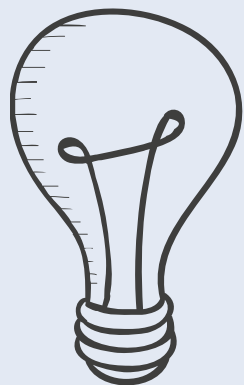


PART 01

solv-utils 私仓

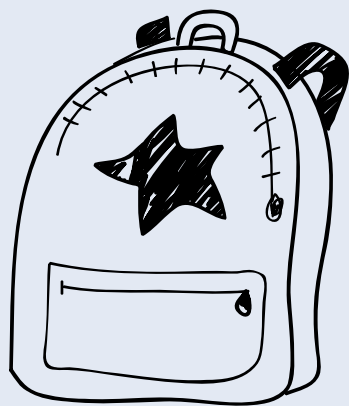


Lerna 核心命令大全



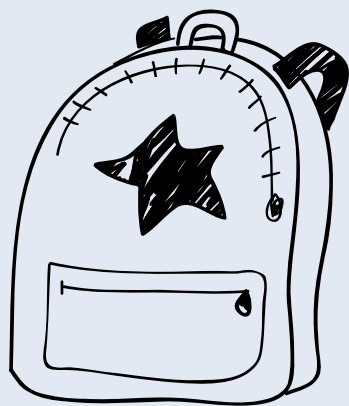
- | | | |
|----------------------|---------------|---------|
| 1. lerna init | #初始化项目 | ● 项目初始化 |
| 2. lerna create core | #添加新项目 | |
| 3. lerna add xxx | #安装依赖 | |
| 4. lerna link | #packages项目依赖 | |
| 5. lerna exec | #执行linux命令 | ● 开发测试 |
| 6. lerna run | #执行npm命令 | |
| 7. lerna clean | #清空依赖 | |
| 8. lerna bootstrap | #重装依赖 | ● 发布上线 |
| 9. lerna version | #管理版本号 | |
| 10. lerna changed | #查看变更 | |
| 11. lerna diff | #查看diff | |
| 12. lerna publish | #发布项目 | |

私仓部署核心命令[上]



- 1. `~/.config/verdaccio` 配置文件地址
- 2. 底部 `listen: 0.0.0.0:4873` 阿里云端口号
- 3. `verdaccio` 或者 `pm2 start verdaccio` 进行测试
- 4. <https://verdaccio.org/zh-cn/docs/webui/> 页面配置
- 5. 仓库文件 `~/.local/share/verdaccio/storage/`
- 6. `npm root -g` 查看全局包地址索引
- 7. 初始化项目 `npm init --scope <scope name>`
- 8. 为域(scope)添加用户
`npm adduser --registry=<registry> --scope=<scope name>`

私仓部署核心命令[中]



- 1. `npm adduser --registry http://localhost:4873/`
- 2. `npm login --registry http://localhost:4873/`
`npm whoami(验证) npm logout (退出)`
- 3. `npm unpublish yd-libs --force`
- 4. `pnpm install @solv-utils/core -D`
- 5. `pnpm config set registry 地址`
- 6. `npm add laoyuan http://127.0.0.1:4873/` ● 省去配置
- 7. `npm use laoyuan` 或者直接配置
- 8. 下载配置 `.npmrc` 发布配置 `package.json` (`.npmrc` 优先)





私仓部署核心命令[下]monorepo









- 1. `pnpm i xxx --filter @solv-utils/demos`
- 2. `lerna add @solv-utils/core packages/demos/`
- 3. `pnpm add lodash -r/-w` 为每个项目或全局添加依赖
- 4. `pnpm run build --filter @solv-utils/demos`
- 5. `lerna run test --scope=@solv-utils/core`
- 6. `lerna publish from-package` 初次发版
- 7. `lerna changed` 发版前校验需要更改的包
- 8. 备选技术栈 `rush`、<https://nx.dev/>

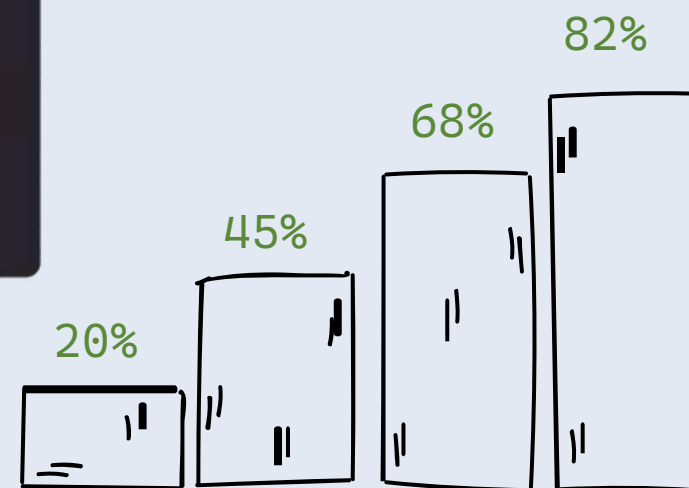


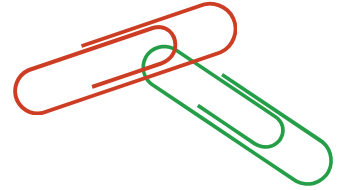
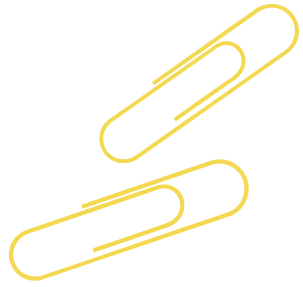
单元测试覆盖率

 **addUserToStore**   

Suggestions

| inputs | expectations |
|--------------------|---|
| user | return |
| { name: "array" } |   |
| { name: "object" } |   |
| { name: {} } |   |





PART 02

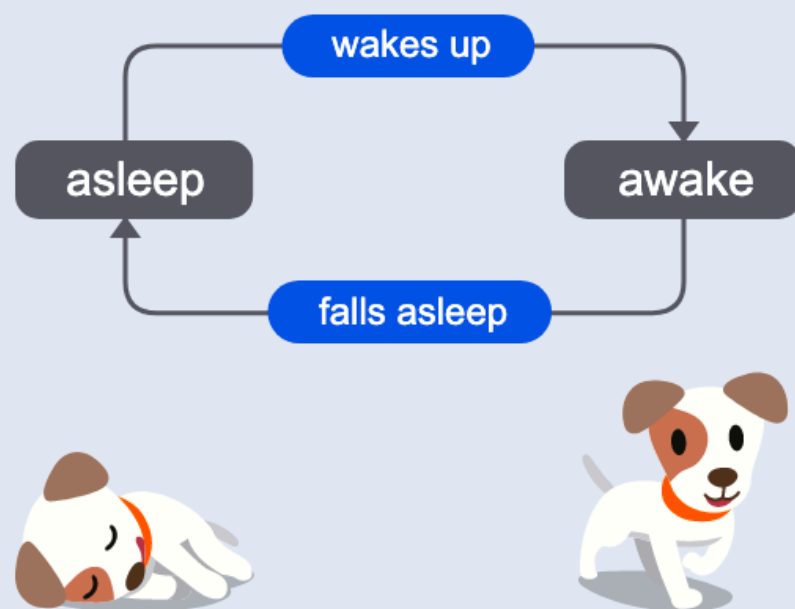
`solv-utils` state



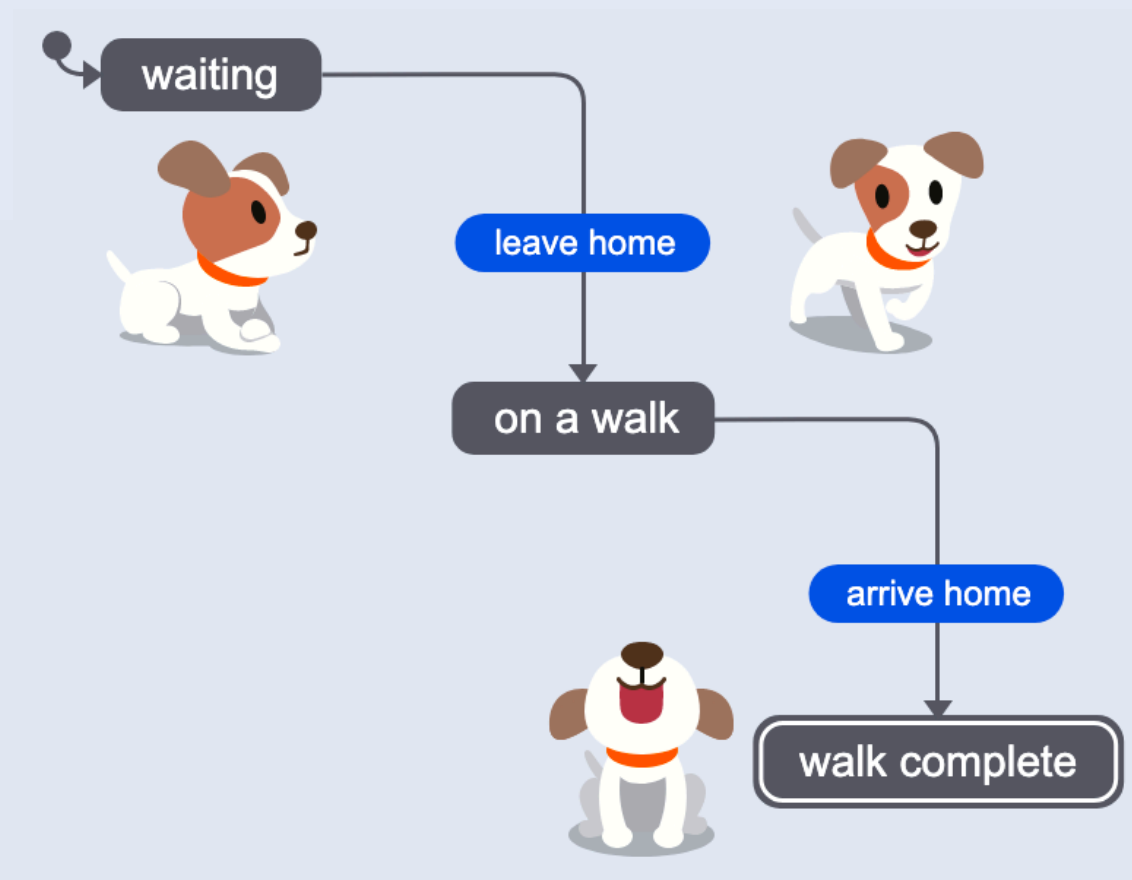


xstate 与 状态机

<https://xstate.js.org/viz/>

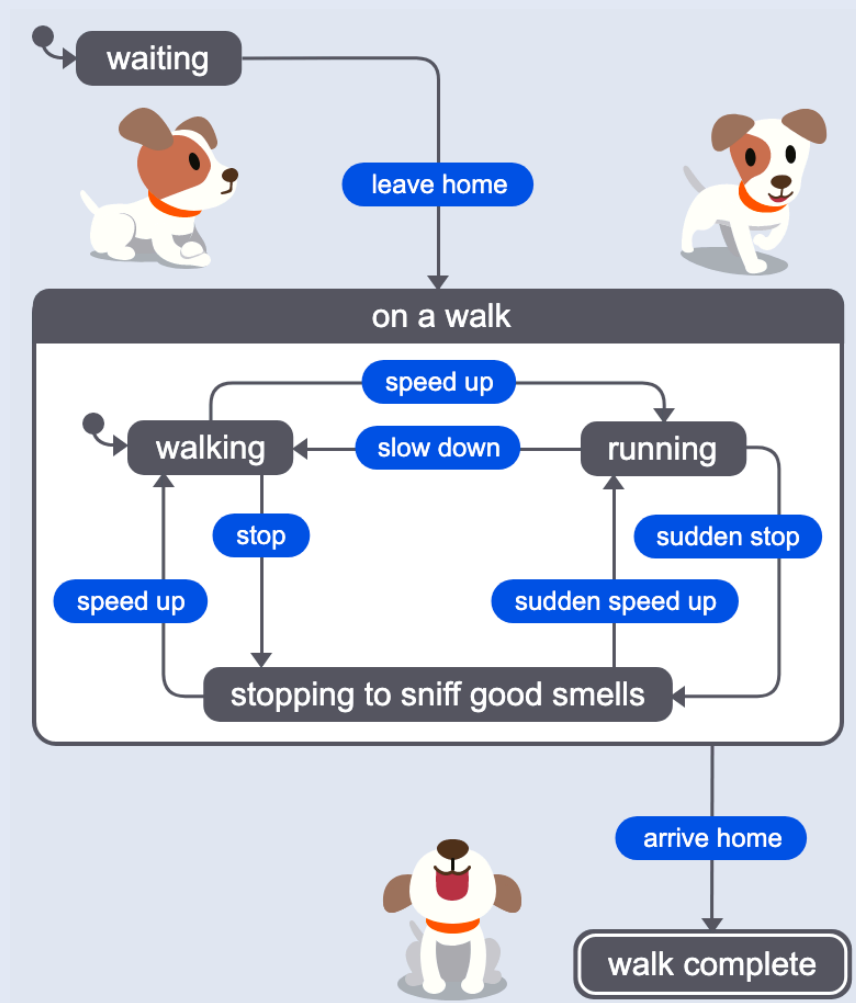


Transitions and events

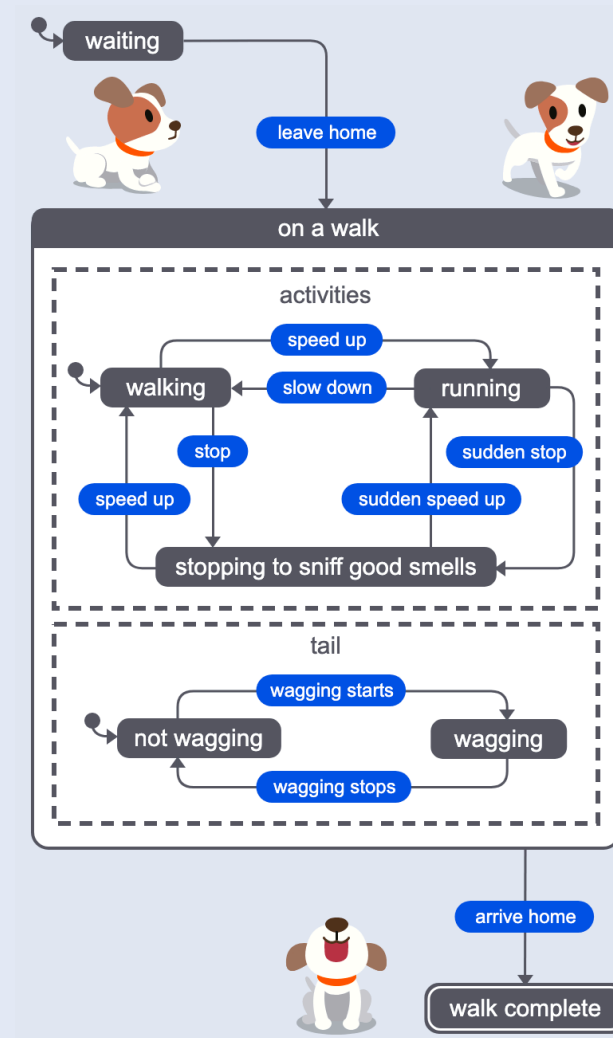


初始状态和最终状态

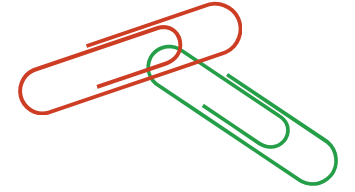
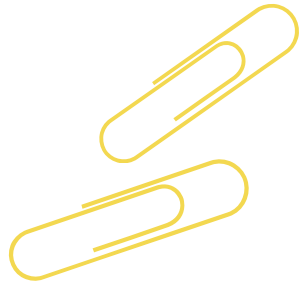
Xstate与状态机-复合状态、原子状态、并行状态



复合状态是可以包含更多状态的状态，也称为子状态。



并行状态是一种复合状态，其中所有子状态（也称为区域）同时处于活动状态。

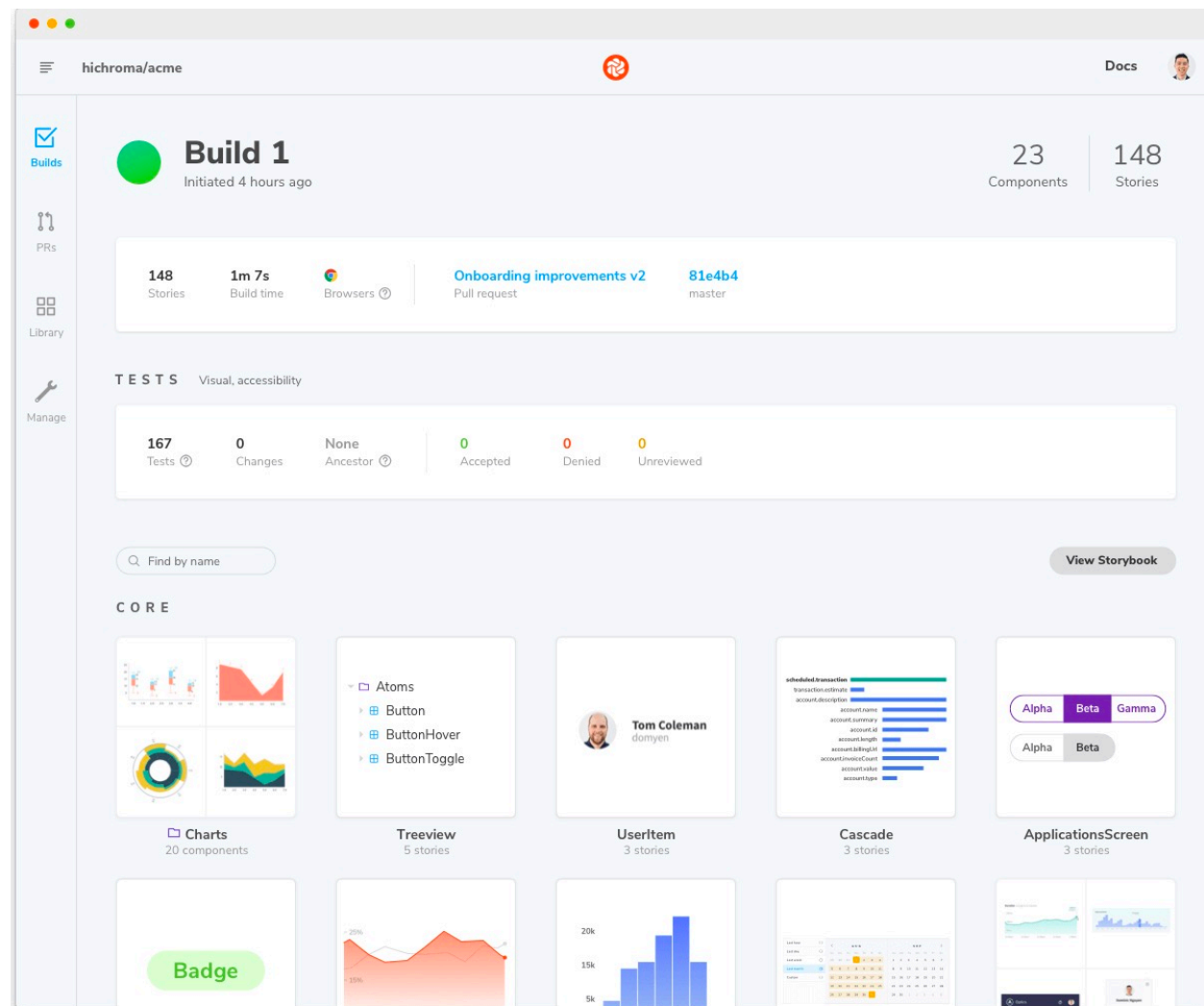
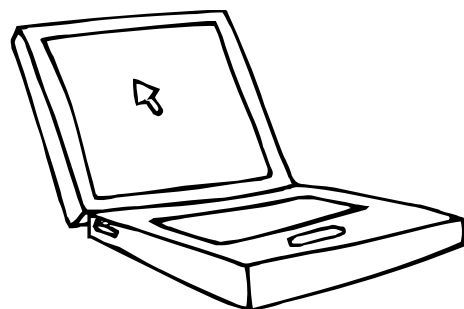


PART 03

Solv Components



React 技术细节



React 技术细节

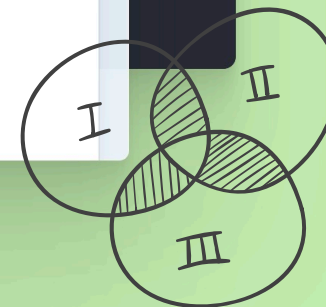
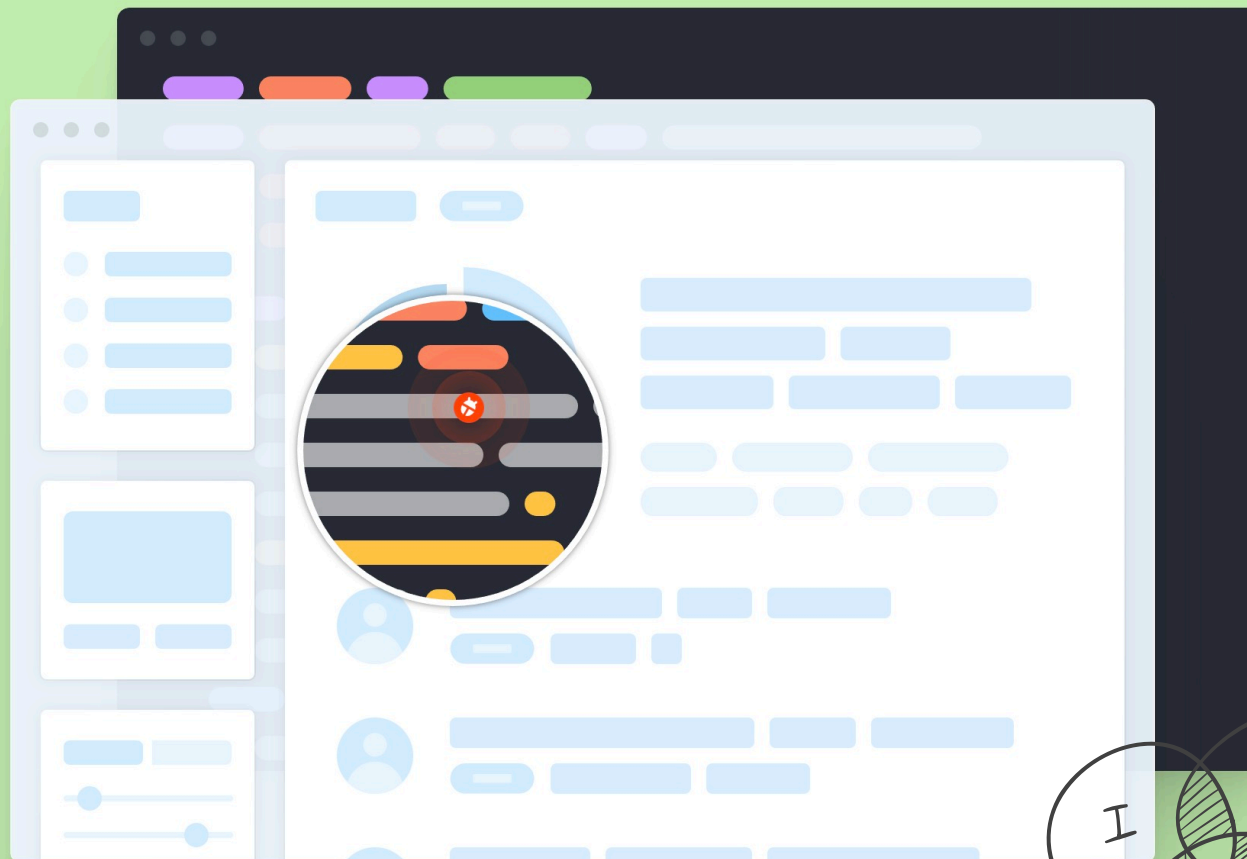
[Features](#) ▾[Pricing](#)[Use cases](#) ▾[Docs](#)[Sign in](#)[Sign up](#)

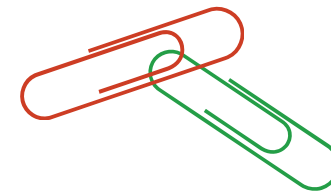
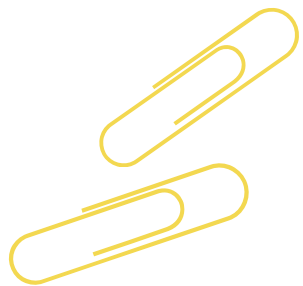
Pinpoint UI bugs in components instantly

Chromatic ensures consistency in UI components, down to the pixel. Every commit is automatically tested for visual changes in the cloud.

[Get started now](#)[Watch video](#)

 Made by Storybook maintainers - Setup in 2 minutes



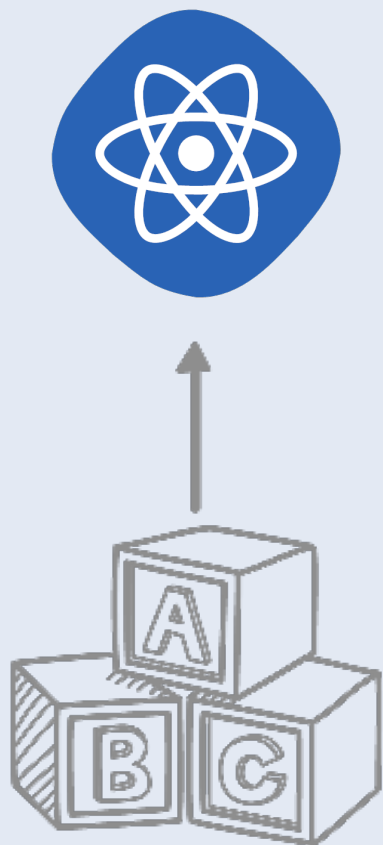


PART 04

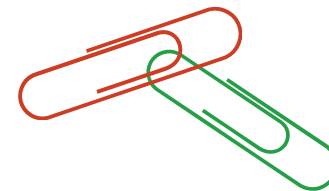
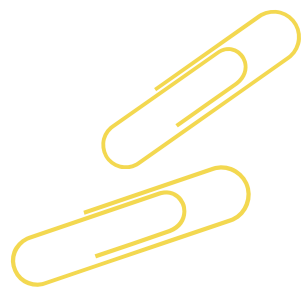
SolvApp & React



React 技术细节



- 1. 组件如果被嵌套外层添加memo
- 2. 组件传递的props是对象使用atom
- 3. 方法如需被传递外层添加useCallback



谢
谢

