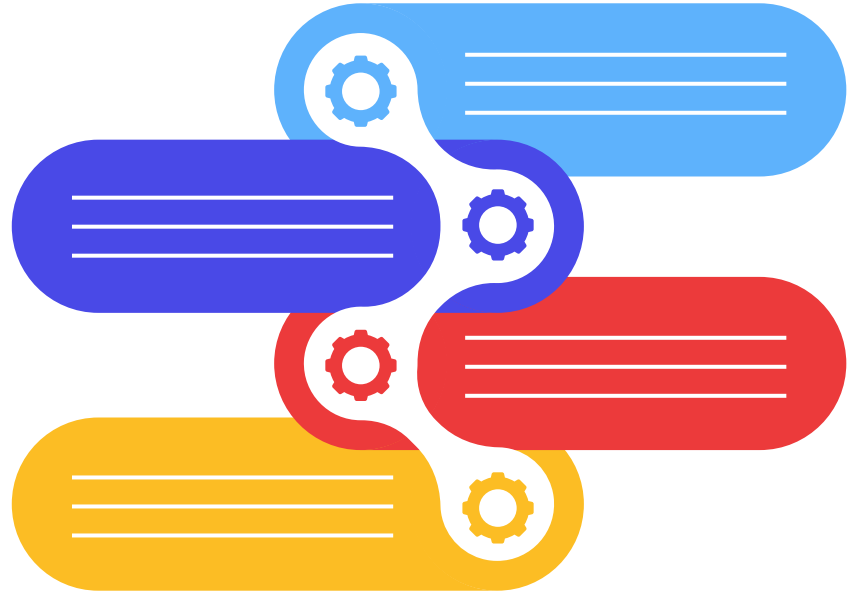


# Accelerating the Assembly Process

Reducing the Cycle Time



BUTTON



BUTTON

None



0 Rev

### Stage: checks



lint



label



requests



one



test



files

one

Time

```
-----  
1726 Test Suites: 11 skipped, 241 passed, 241 of 252 total  
1727 Tests:       24 skipped, 1560 passed, 1584 total  
1728 Snapshots:   198 passed, 198 total  
1729 Time:        348.178 s  
1730 Ran all test suites.
```

**Duration:** 11 minutes 32 seconds

**Finished:** 1 day ago

**Queued:** 2 seconds

**Timeout:** 1h (from project) 

**Runner:** #2610631 (39ydBTSSsE) buli



**Me** 10:23 AM

guys, check my [new MR!3914](#)



**Me** 10:23 AM

guys, check my [new MR!3914](#)



**Teammate** 10:40 AM

tests were failed 😞



**Me** 10:23 AM

guys, check my [new MR!3914](#)



**Teammate** 10:40 AM

tests were failed 😞



**Me** 10:42 AM

fixed





**Me** 10:23 AM

guys, check my [new MR!3914](#)



**Teammate** 10:40 AM

tests were failed 😞



**Me** 10:42 AM

fixed



**Teammate** 11:40 AM

linter is not okay 😬

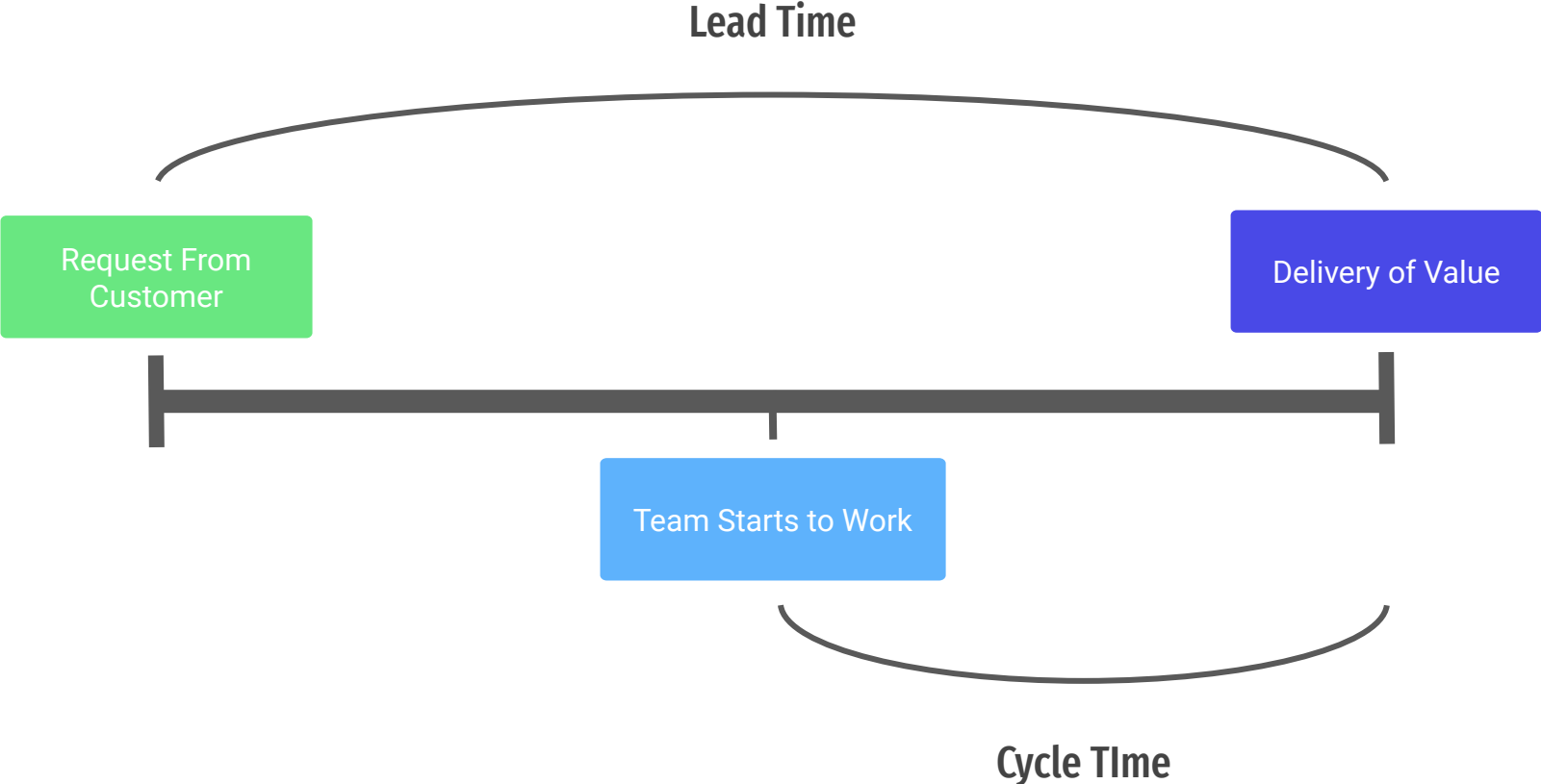
Lead Time

Request From  
Customer

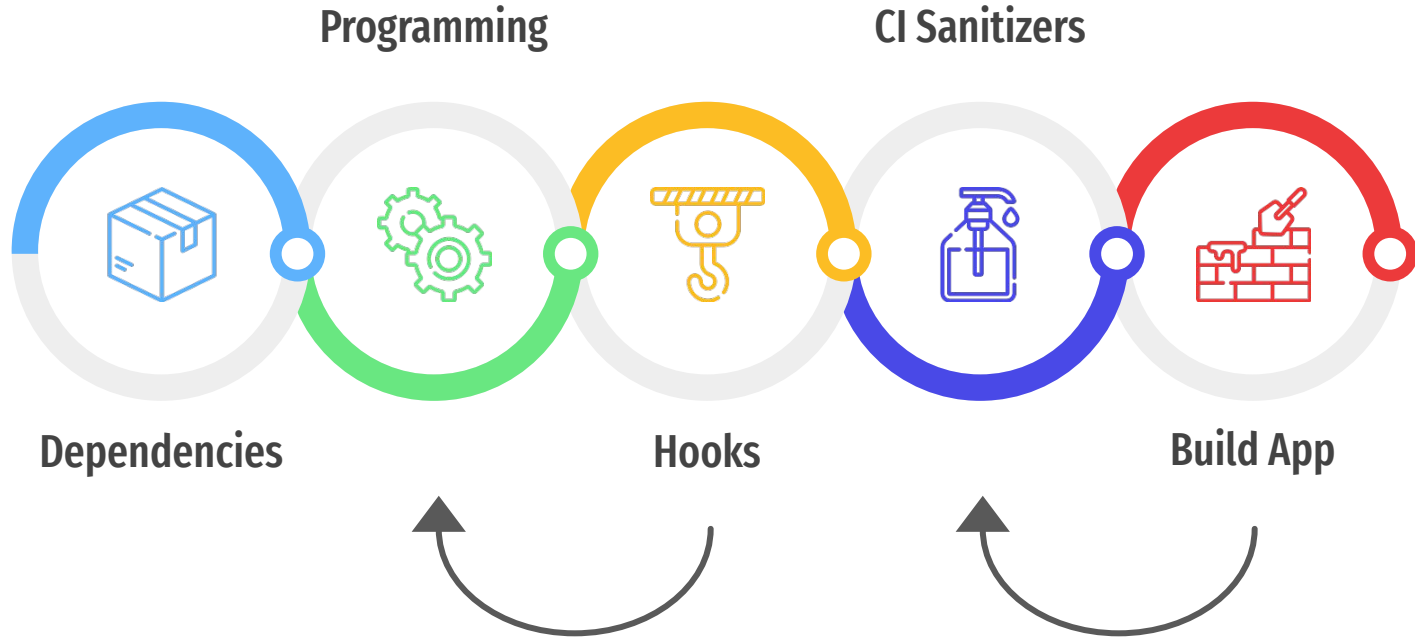
Delivery of Value

Team Starts to Work

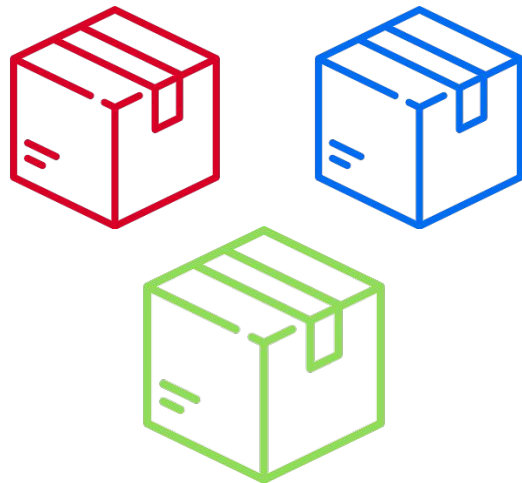
Cycle Time



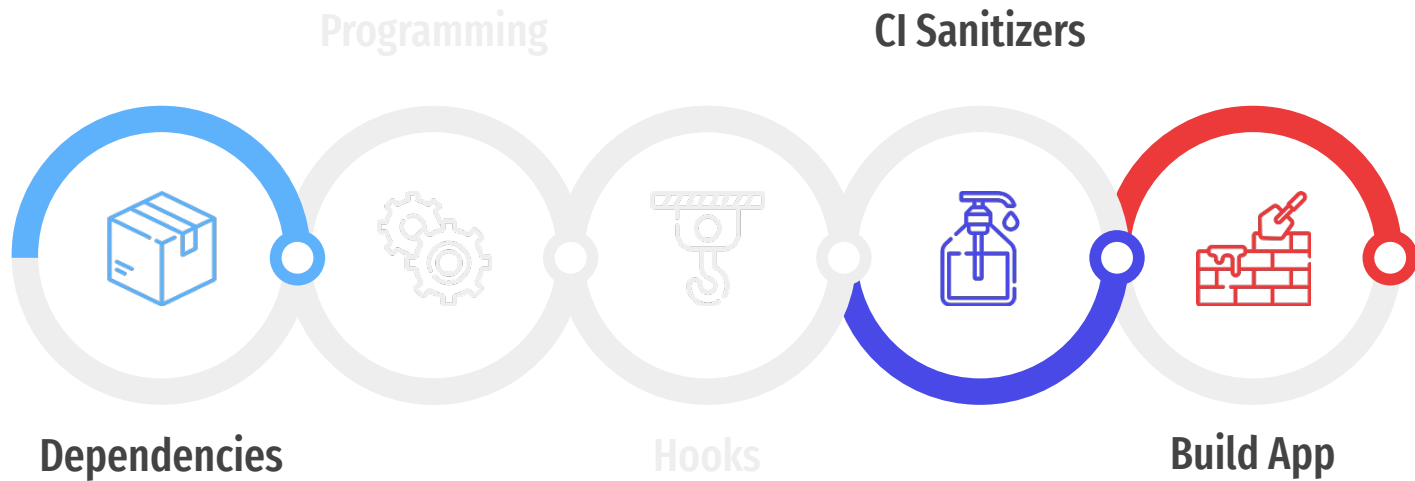
# Cycle Time



# Package Manager



# Package Manager



# Package Managers



# CI cache

```
test-job:  
  stage: build  
  cache:  
    key: $CI_COMMIT_REF_SLUG  
    paths:  
      - .npm/  
  script:  
    - npm ci --cache .npm --prefer-offline  
    - echo Run tests...
```

# CI cache

~1min



~29s



# Clean Install

`npm i` → `npm ci`

# Scripts life-cycle

**pre<event>**

**<event>**

**post<event>**

# npm ci

- preinstall
- install
- postinstall
- prepublish
- preprepare
- prepare
- postprepare

# npm ci --ignore-scripts

- ~~preinstall~~
- install
- ~~postinstall~~
- ~~prepublish~~
- ~~preprepare~~
- prepare
- ~~postprepare~~

# **npm ci --ignore-scripts**

- install
- prepare

**npm ci --ignore-scripts**



# npm ci --ignore-scripts

```
curl --silent -o-  
https://raw.githubusercontent.com/oresoftware/re  
alpath/master/assets/install.sh | bash
```

**npm ci --ignore-scripts**

**~29s**



**~25s**



# **npm ci --production**

- devDependencies
- dependencies

# npm ci --production

- ~~devDependencies~~
- dependencies

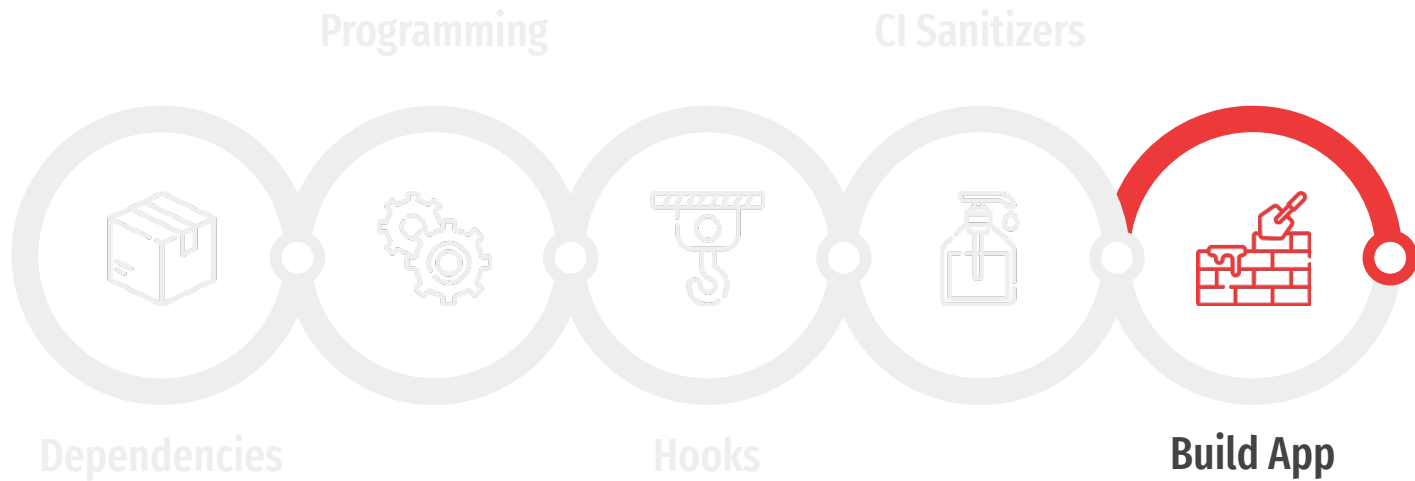
# npm ci --production

~29s



~10s

# npm ci --production



# why-npm-i-so-long

```
why-npm-i-so-long
▲ why-npm-i-so-long ~/project/package.json
my-project's dependencies:

resolve @ 1.14.2
  install size 104 kB
  publish size 94.1 kB

fs-extra @ 8.1.0
  install size 175 kB
  publish size 125 kB

fast-glob @ 3.1.1
  install size 446 kB
  publish size 74.0 kB

moment @ 2.24.0
  install size 2.66 MB
  publish size 2.66 MB

typescript @ 3.7.4
  install size 48.2 MB
  publish size 48.2 MB

~
▲
```

# why-npm-i-so-long

publish size

vs

install size

# depcheck

- Unused dependencies
- Unused devDependencies
- Missing dependencies

# NPM doppelgangers

```
{  
  "name": "library-a",  
  "version": "1.0.0",  
  "dependencies": {  
    "library-b": "^1.0.0",  
    "library-c": "^1.0.0",  
    "library-d": "^1.0.0",  
    "library-e": "^1.0.0"  
  }  
}
```



# NPM doppelgangers

```
{  
  "name": "library-b", // library-c  
  "version": "1.0.0",  
  "dependencies": {  
    "library-f": "^1.0.0"  
  }  
}
```

# NPM doppelgangers

```
{  
  "name": "library-d", // library-e  
  "version": "1.0.0",  
  "dependencies": {  
    "library-f": "^2.0.0"  
  }  
}
```

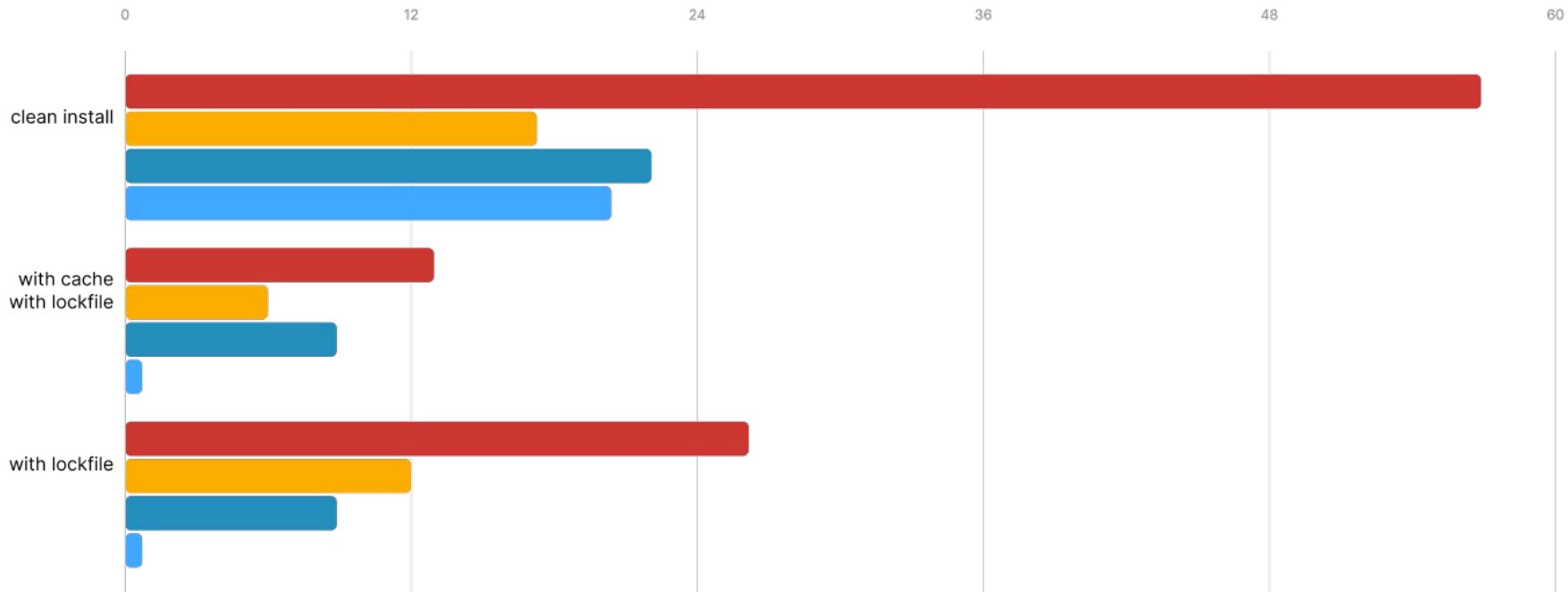
# NPM doppelgangers

```
{  
  "name": "library-a",  
  "version": "1.0.0",  
  "dependencies": {  
    "library-b": "^1.0.0", <--library-f@1.0.0  
    "library-c": "^1.0.0", <--library-f@1.0.0  
    "library-d": "^1.0.0", <--library-f@2.0.0  
    "library-e": "^1.0.0" <--library-f@2.0.0  
  }  
}
```

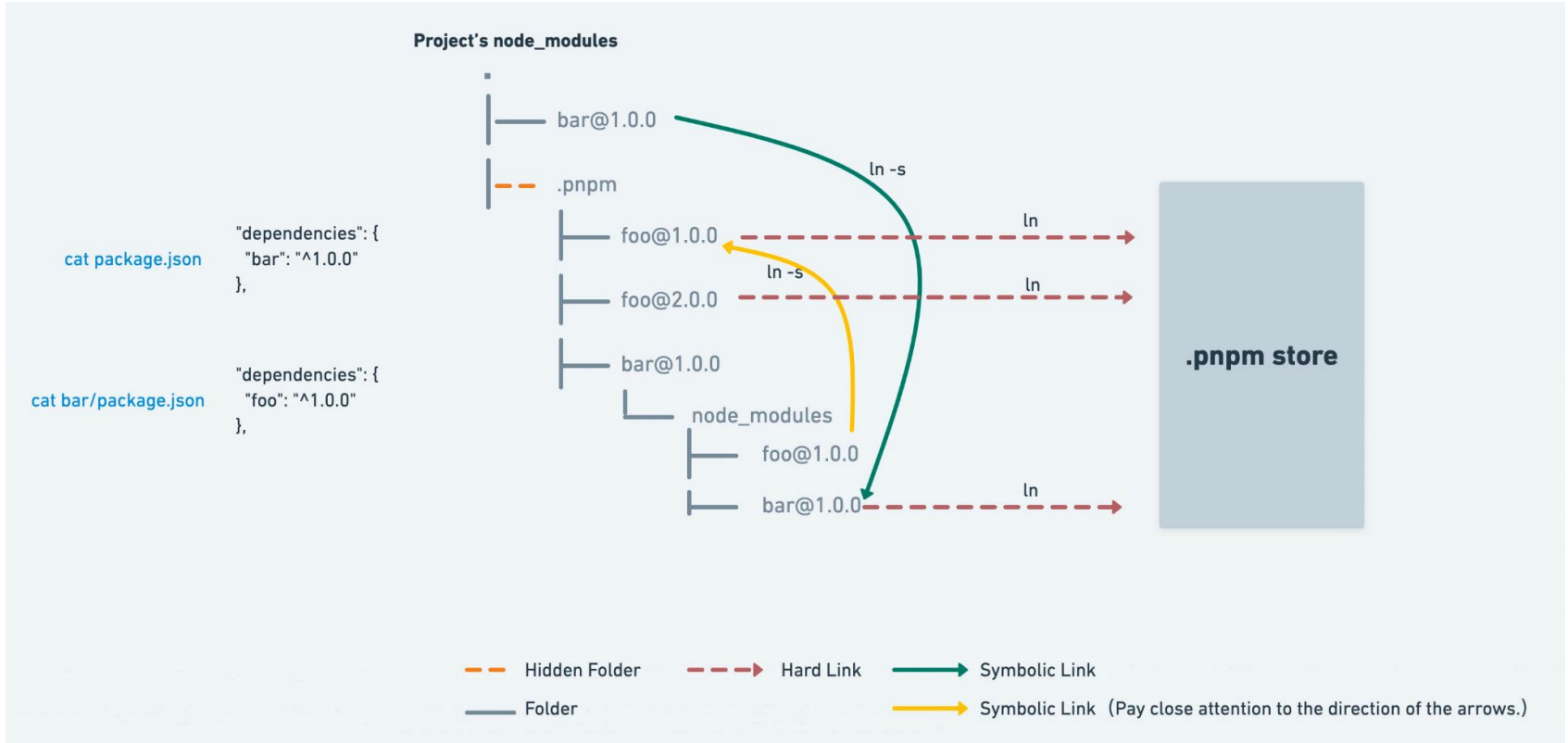
# NPM doppelgangers

```
- node_modules/  
  - library-b/  
  - library-c/  
  - library-d/  
    - node_modules/  
      - library-f/  
        - package.json <-- library-f@2.0.0  
  - library-e/  
    - node_modules/  
      - library-f/  
        - package.json <-- library-f@2.0.0  
  - library-f/  
    - package.json <-- library-f@1.0.0
```

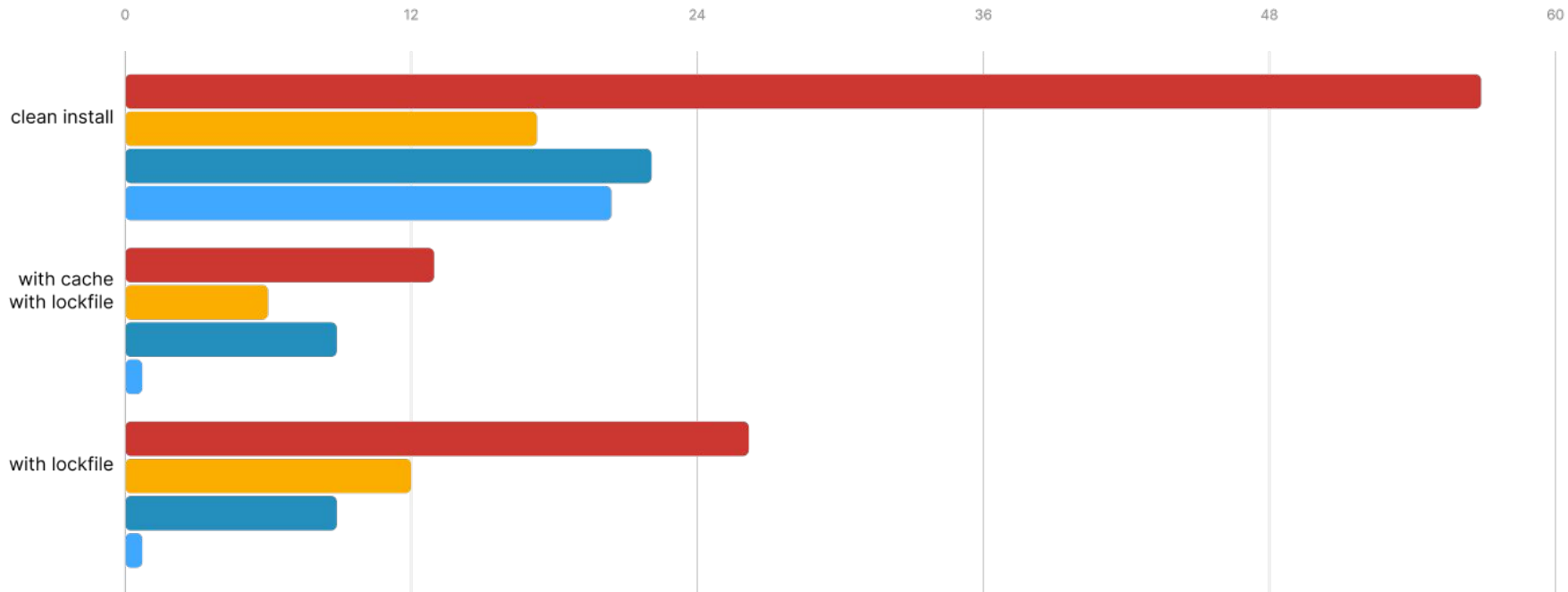
# Package Managers



# pnpm



# Package Managers



**Yarn PnP**

**ОГРОМНАЯ ПРОСЬБА, НЕ ПРЕДЛАГАЙТЕ МНЕ**



**NODE\_MODULES**



# Yarn PnP

~~node\_modules~~



.pnp.cjs



~/.yarn/berry/cache/[react-npm-18.2.0-1eae08fee2-10c0.zip](#)/node\_modules/react/

# Yarn PnP

```
2759     | ["punycode", "npm:2.3.0"]\
2760     ],\
2761     "linkType": "HARD"\
2762   }\
2763 ],\
2764 ["queue-microtask", [\
2765   ["npm:1.2.3", {\
2766     "packageLocation": "../..../..../.yarn/berry/cache/queue-microtask-npm-1.2.3-fcc98e4e2d-10c0.zip/node_modules/queue-microtask/",\
2767     "packageDependencies": [\
2768       ["queue-microtask", "npm:1.2.3"]\
2769     ],\
2770     "linkType": "HARD"\
2771   }]\
2772 ],\
2773 ["react", [\
2774   ["npm:18.2.0", {\
2775     "packageLocation": "../..../..../.yarn/berry/cache/react-npm-18.2.0-1eae08fee2-10c0.zip/node_modules/react/",\
2776     "packageDependencies": [\
2777       ["react", "npm:18.2.0"],\
2778       ["loose-envify", "npm:1.4.0"]\
2779     ],\
2780     "linkType": "HARD"\
2781   }]\
2782 ],\
2783 ["react-dom", [\
2784   ["npm:18.2.0", {\
2785     "packageLocation": "../..../..../.yarn/berry/cache/react-dom-npm-18.2.0-dd675bca1c-10c0.zip/node_modules/react-dom/",\
2786     "packageDependencies": [\
2787       ["react-dom", "npm:18.2.0"]\
2788     ],\
2789     "linkType": "SOFT"\
```

**Yarn PnP**

**ОГРОМНАЯ ПРОСЬБА, НЕ ПРЕДЛАГАЙТЕ МНЕ**



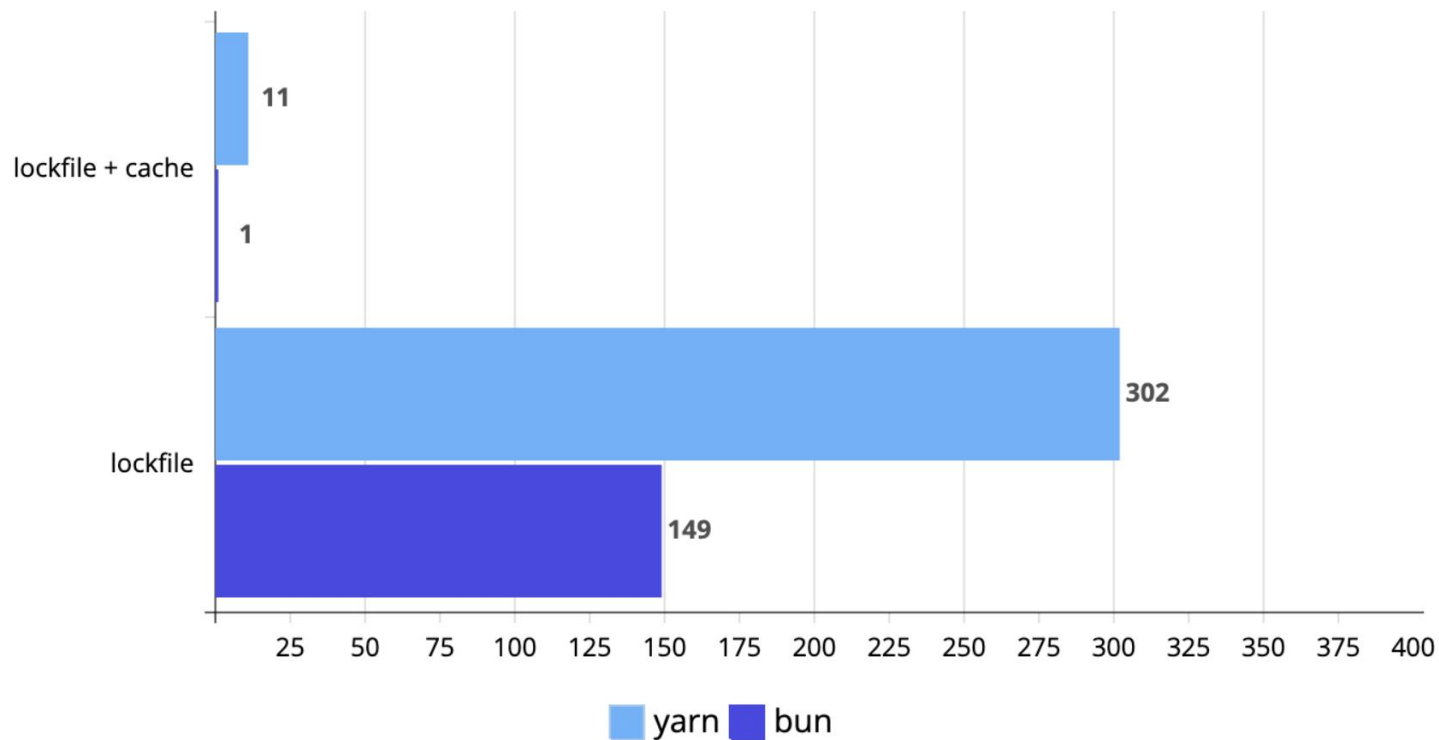
**YARN PNP**

# Bun



Installing dependencies from cache for a Remix app.

# Bun

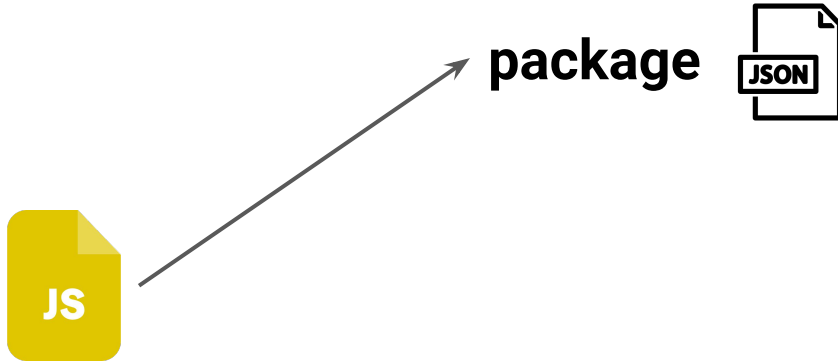




# Custom script

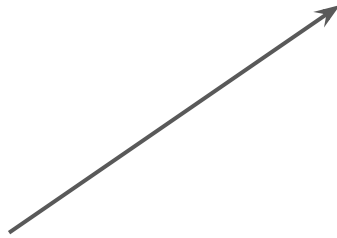


# Custom script





# Custom script



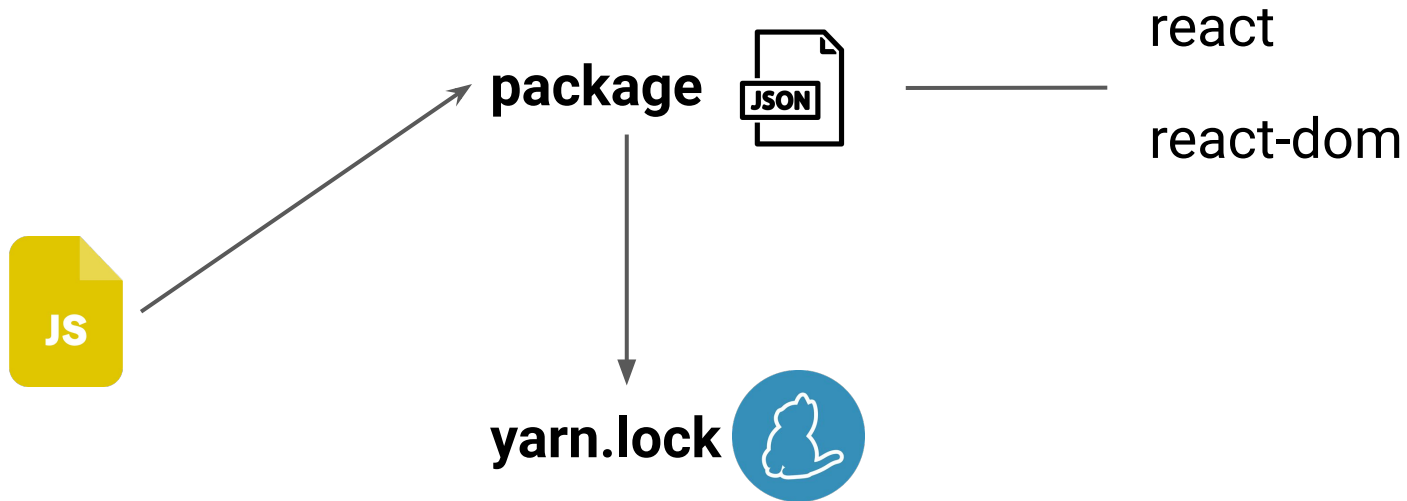
package



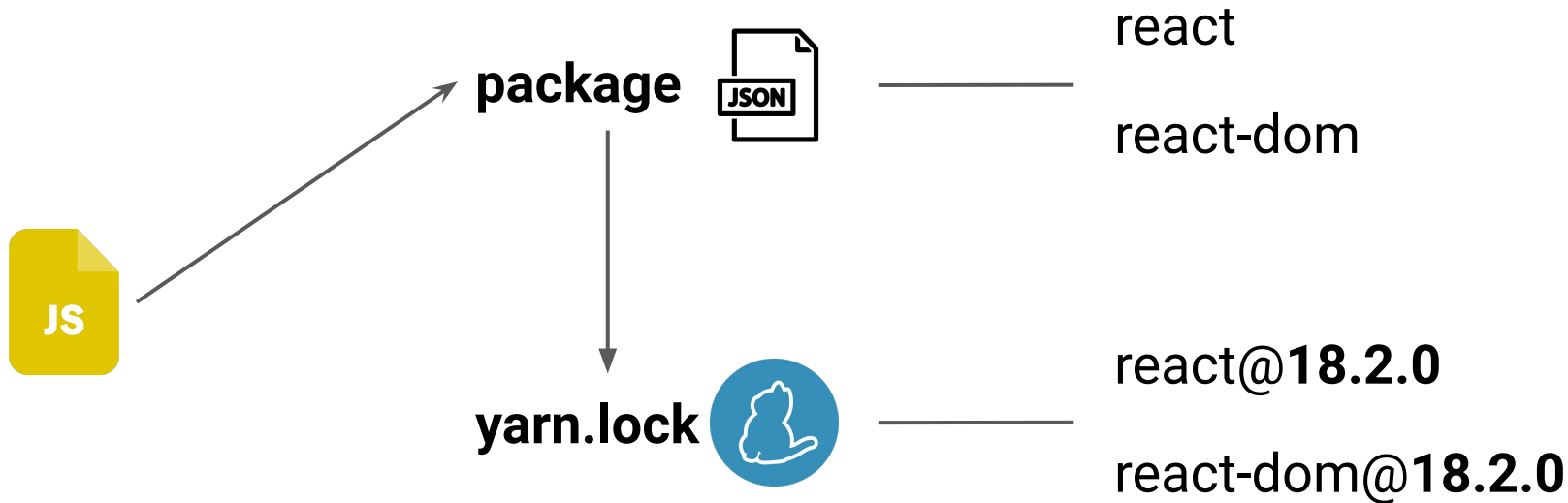
react

react-dom

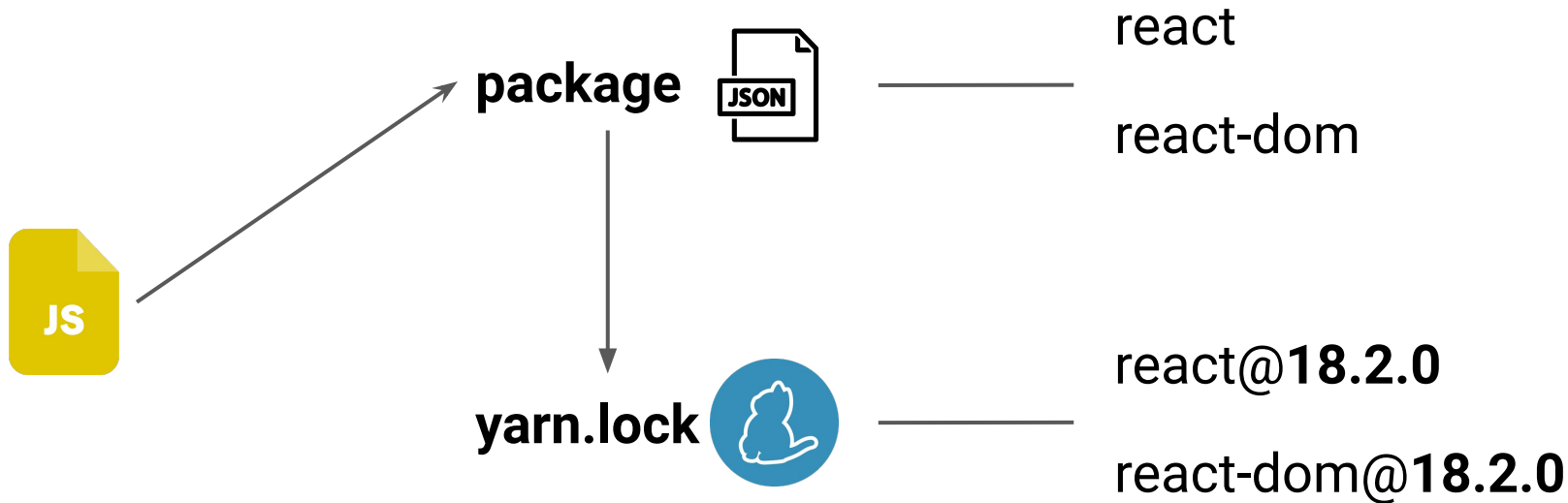
# Custom script



# Custom script



# Custom script



**bun install** react@18.2.0 react-dom@18.2.0

# Sum Up

## Improve current PM

- ci cache
- test install flags
- check your deps

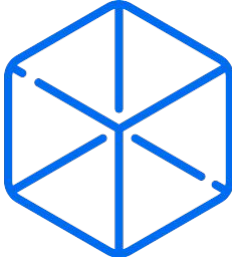
# Sum Up

## Improve current PM

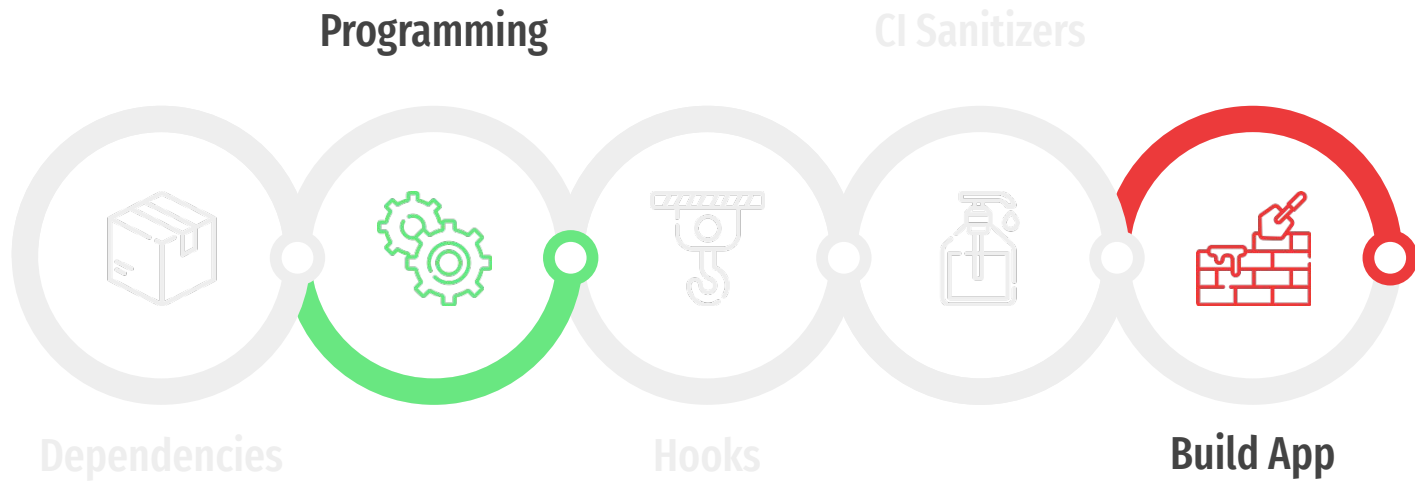
- ci cache
- test install flags
- check your deps

## Choose your new PM

# Bundler



# Bundler





# Parcel



**Webpack**

**РЕАЛЬНО УДОБНО**



**РЕАЛЬНО ПОД СЕБЯ**

**Rollup**



# Vite



**EXTREMELY BLAZING FAST**

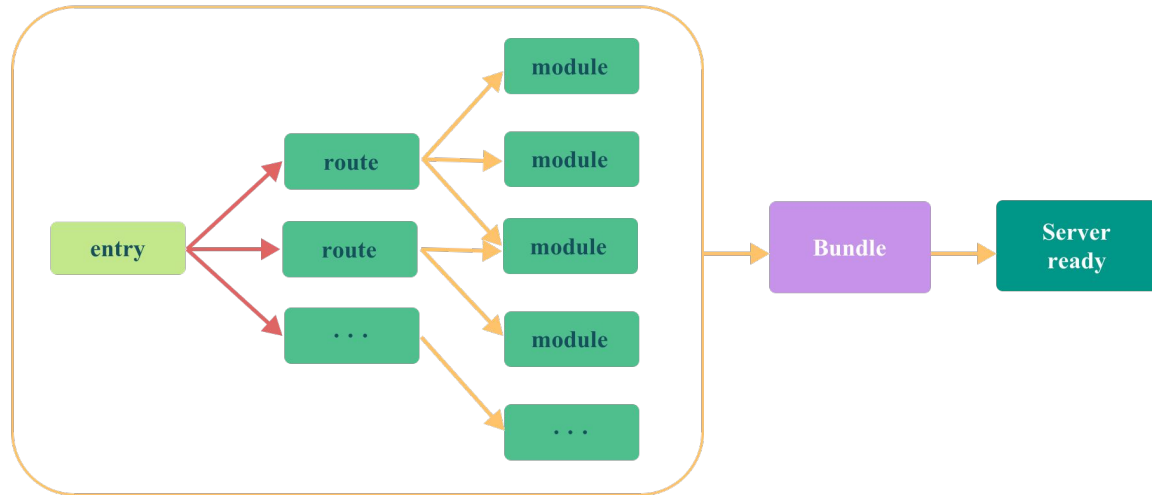


**ZERO CONFIGURATION WEB  
APPLICATION BUNDLER**



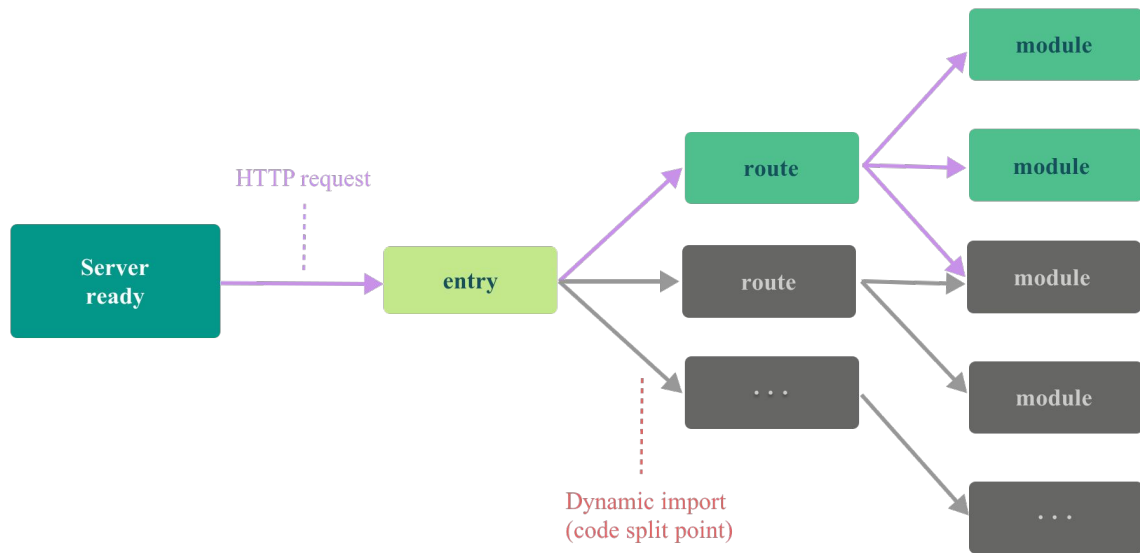
# Vite

## Bundle based dev server



# Vite

## Native ESM based dev server



# Vite

Esbuild **10-100x faster** than JavaScript-based bundlers.

Native ESM === fast HRM

# Vite + React

**@vitejs/plugin-react(esbuild + babel)**

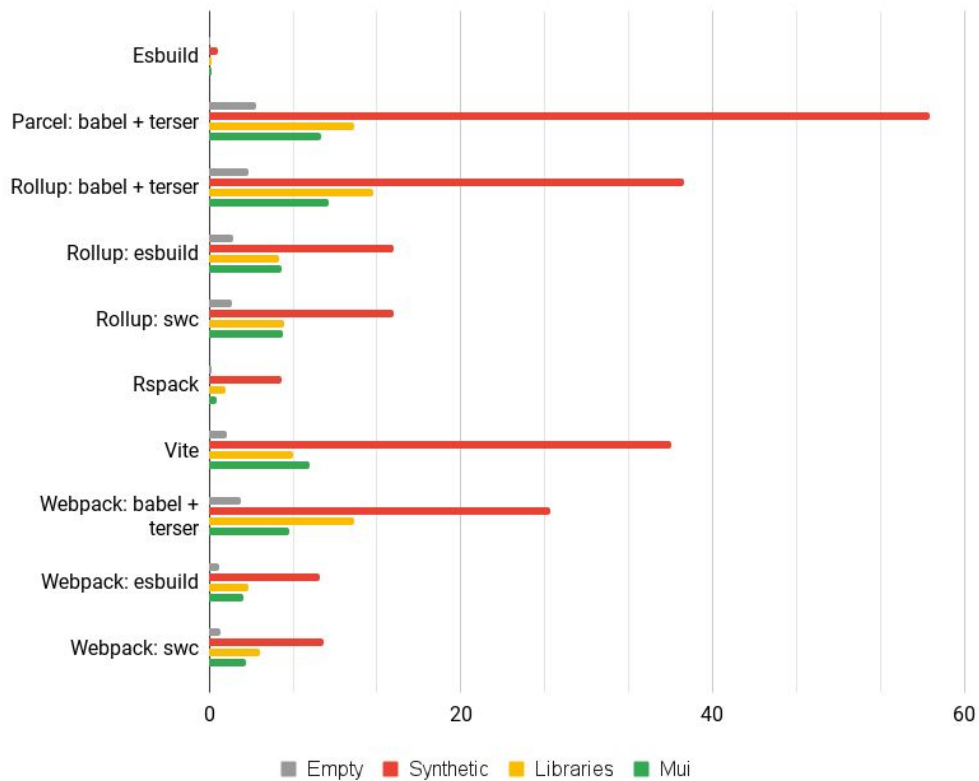


# Vite + React

**@vitejs/plugin-react(esbuild + babel)**

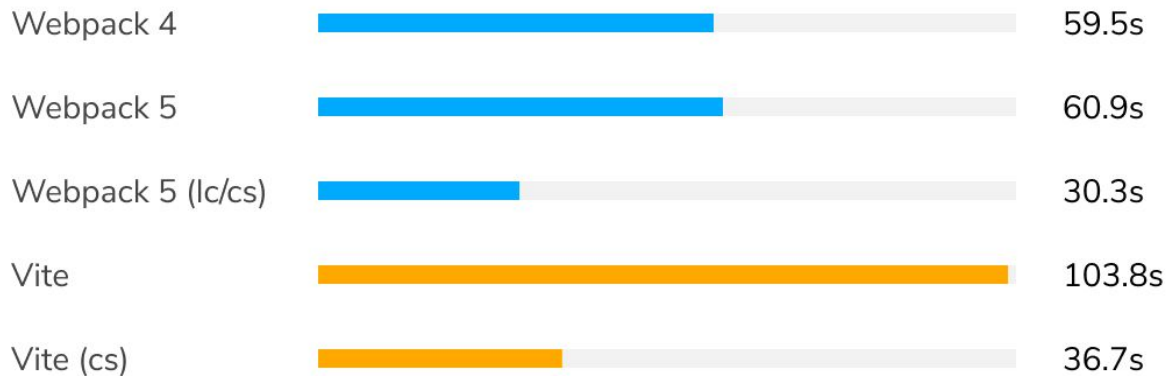
**@vitejs/plugin-react-swc(SWC + esbuild)**

# Benchmarks

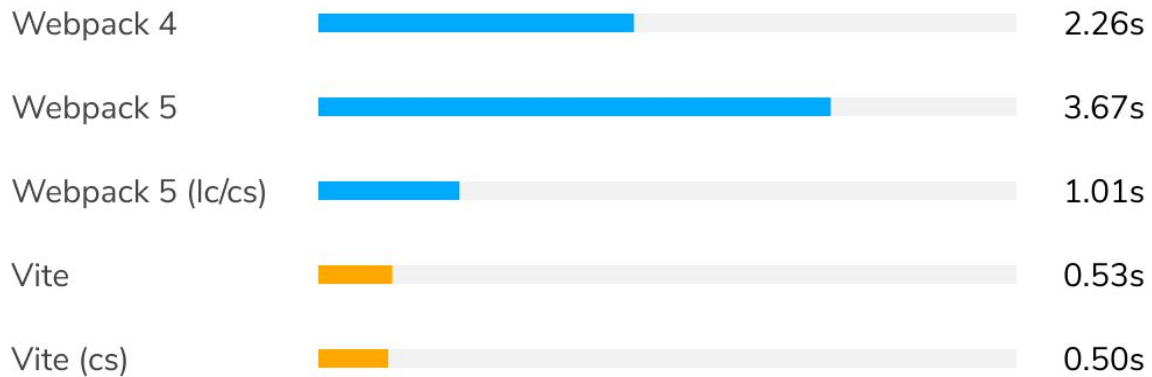


# Development (cold start)

Time to first story dev start + story render



# Rebuild time (Hot Module Reloading)



# Production build



## Bundlers

**BABEL**



**ESBUILD**



**SWC**



# Sum Up

## Bundlers

- avoid babel, use esbuild or SWC

## Vite

- good for small projects
- good dev experience

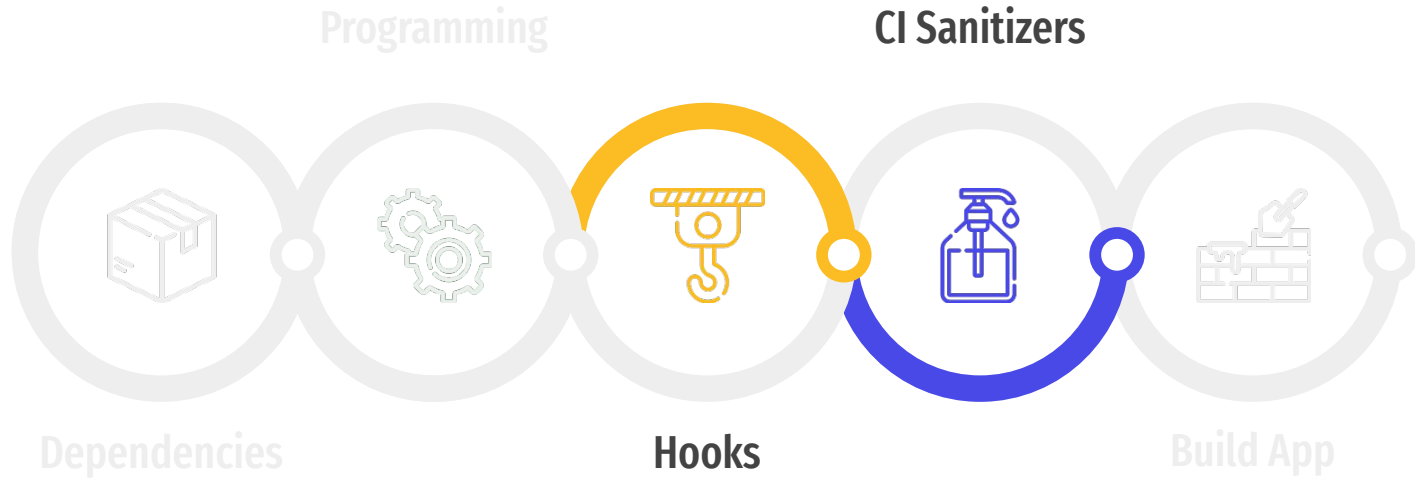
# Hooks

And sanitizers





# Shift Left



# Hooks

- Linters
- Prettier
- Unit tests
- Security
- Unused/duped node\_modules
- Unused files

**Husky**



# Husky

- Can't run commands in parallel
- Have 30 direct dependencies
- Can't filter files

**Husky +   lint-staged**

**!** concurrent mode is on

# Husky + lint-staged

- In general works only with pre-commit hook
- Have 80 direct dependencies

**Lefthook**



# Unit Test





# Jest

pre-commit hook	
merge request	
custom	

# Jest

pre-commit hook

--onlyChanged

merge request

custom

# Jest

pre-commit hook	--onlyChanged
merge request	--changedSince=master
custom	

# Jest

pre-commit hook	--onlyChanged
merge request	--changedSince=master
custom	--findRelatedTests


# Jest parallelization

- `--runInBand` `-i` 

# Jest parallelization

- `--runInBand -i` ⚠
- `--maxWorkers=50%`

# Jest parallelization

- `--runInBand` `-i` 
- `--maxWorkers=50%` [`~20% Faster`]

# Jest parallelization

- `--runInBand -i` ⚠
- `--maxWorkers=50%` [~20% Faster]
- `--workerThreads` ⚠



**babel-jest**  **@swc/jest**

# @swc/jest

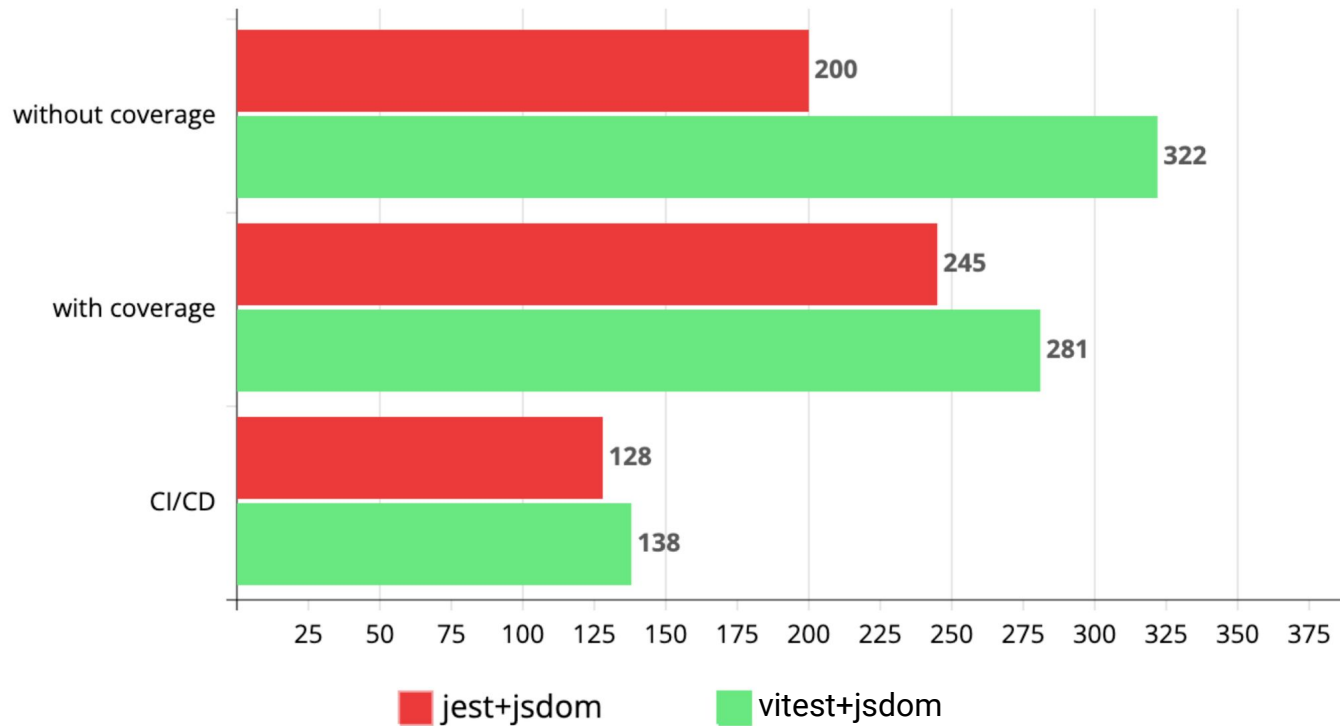
```
module.exports = {
  transform: {
    "^.+\\.\\.?(t|j)sx?$": "@swc/jest",
  },
};
```

**jsdom**  **happy-dom**

# happy-dom

```
module.exports = {  
  "testEnvironment": "@happy-dom/jest-environment"  
};
```

# vitest



# Sum Up

## Hooks

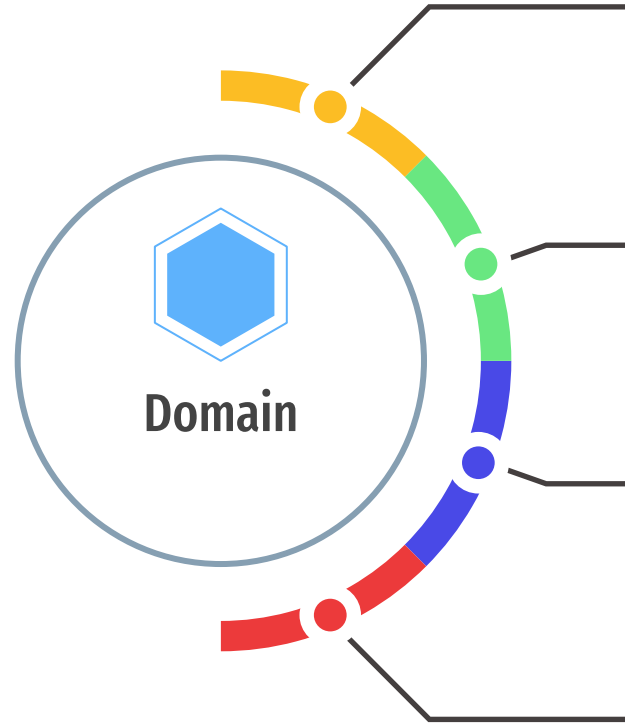
- run jobs in parallel
- split to different hooks
- or disable hooks

## Tests

- do not run all tests
- use new tools for jest
- vitest is overrated

# Architecture

And Design



**Architecture**

**Modular**



# Architecture

## Modular

- Reusability

# Architecture

## Modular

- Reusability
- Quality Assurance

# Architecture

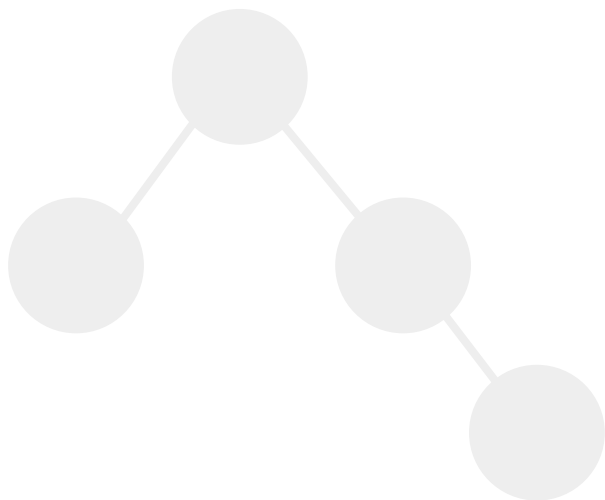
## Modular

- Reusability
- Quality Assurance
- Scalability



# Architecture

App Rootable Components



Reusable Components



Libs & Data



# Architecture

## Module#1

own hooks

own tests

own build

# Architecture

## Module#1

own hooks

own tests

own build

## Module#2

own hooks

own tests

own build

# Architecture

## Modular

- Reusability
- Quality Assurance
- Scalability

## Communication



# Architecture



**Teammate #1** @teammate1 · 1 hour ago

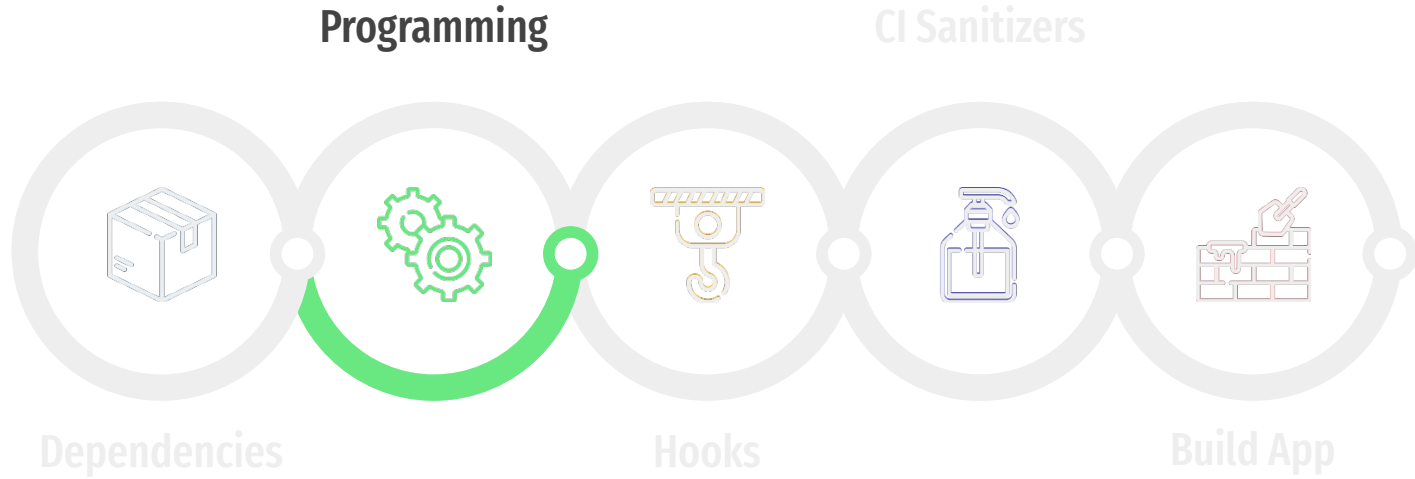


So we can either: `renderToPipeableStream` immediately to early flush, and wrap the skeleton in `Suspense`, which means the page is blank without Javascript Await the network queries first to avoid the `Suspense` boundary, which gets the skeleton displayed without Javascript, but will delay early flush

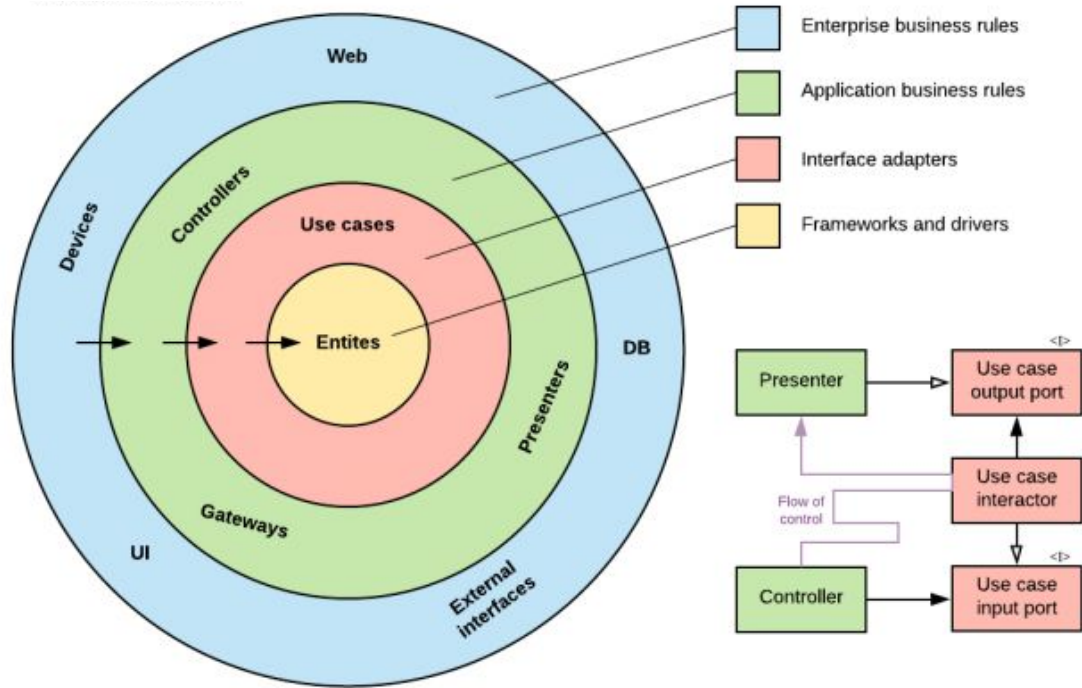


>    [42 replies](#) Last reply by Teammate #3 10 minutes ago

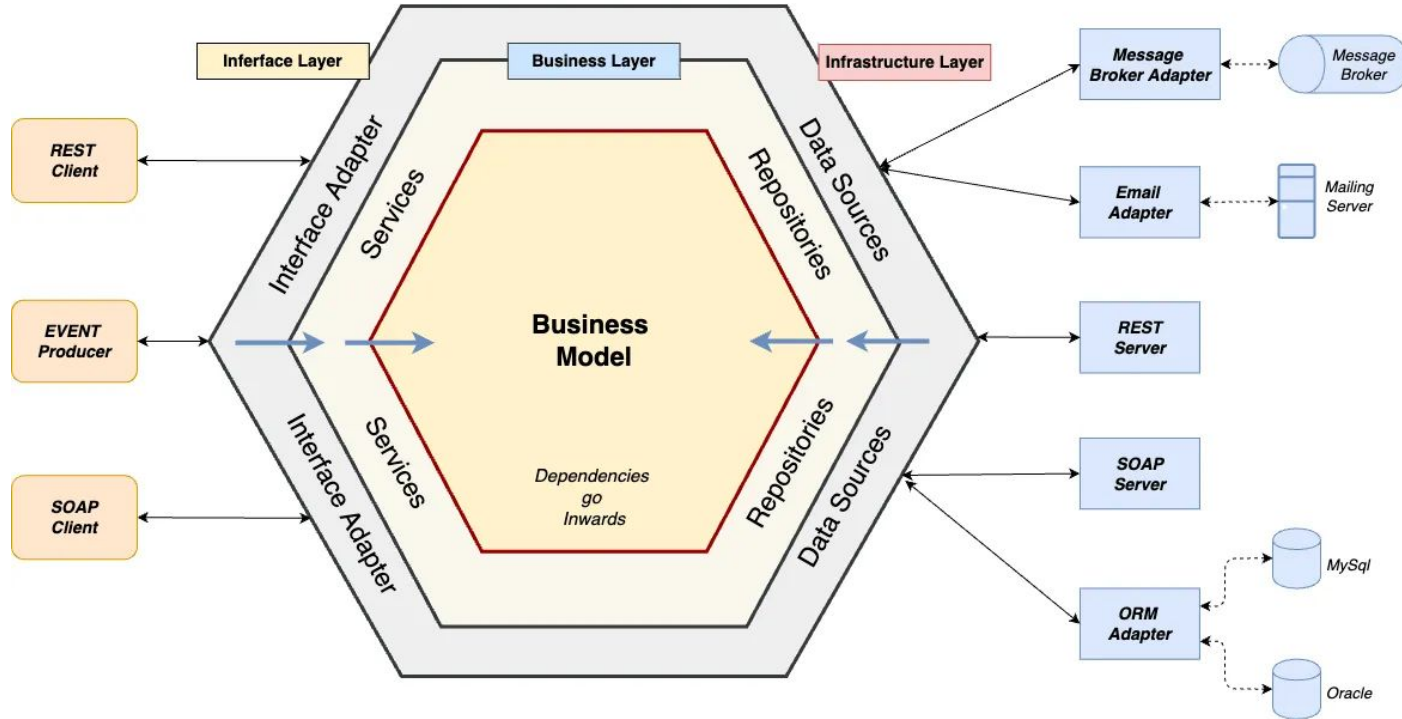
# Architecture



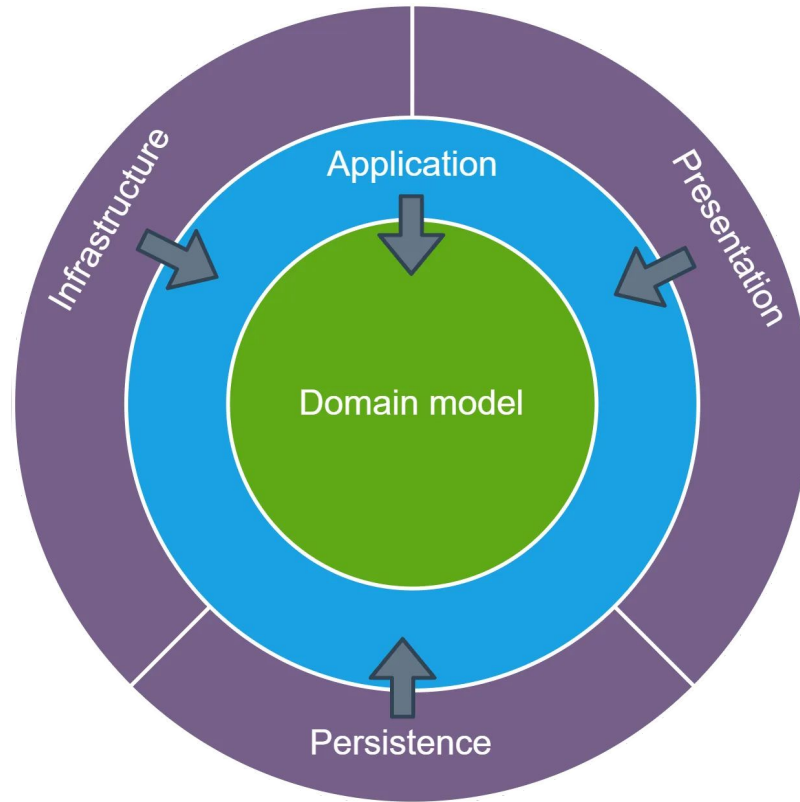
# Clean Architecture



# Hexagonal Architecture



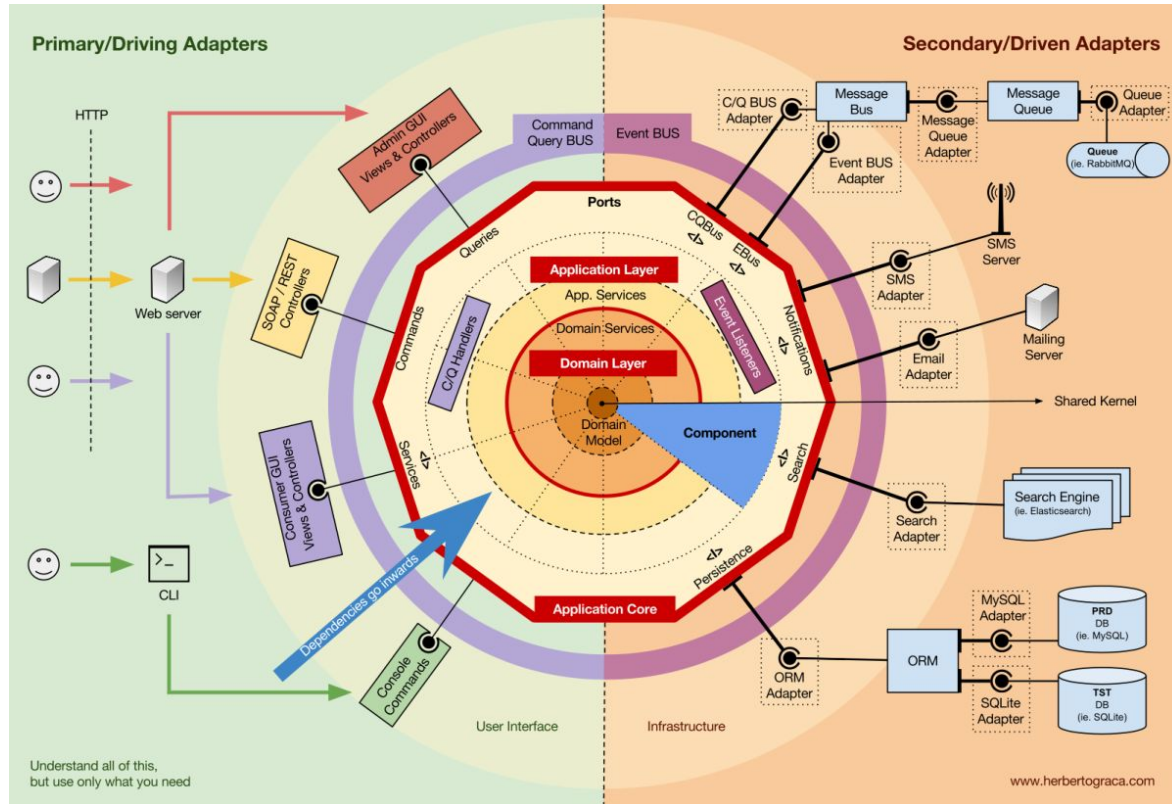
# Onion Architecture



# Everything Everywhere All at Once



# Everything Everywhere All at Once















# Architecture

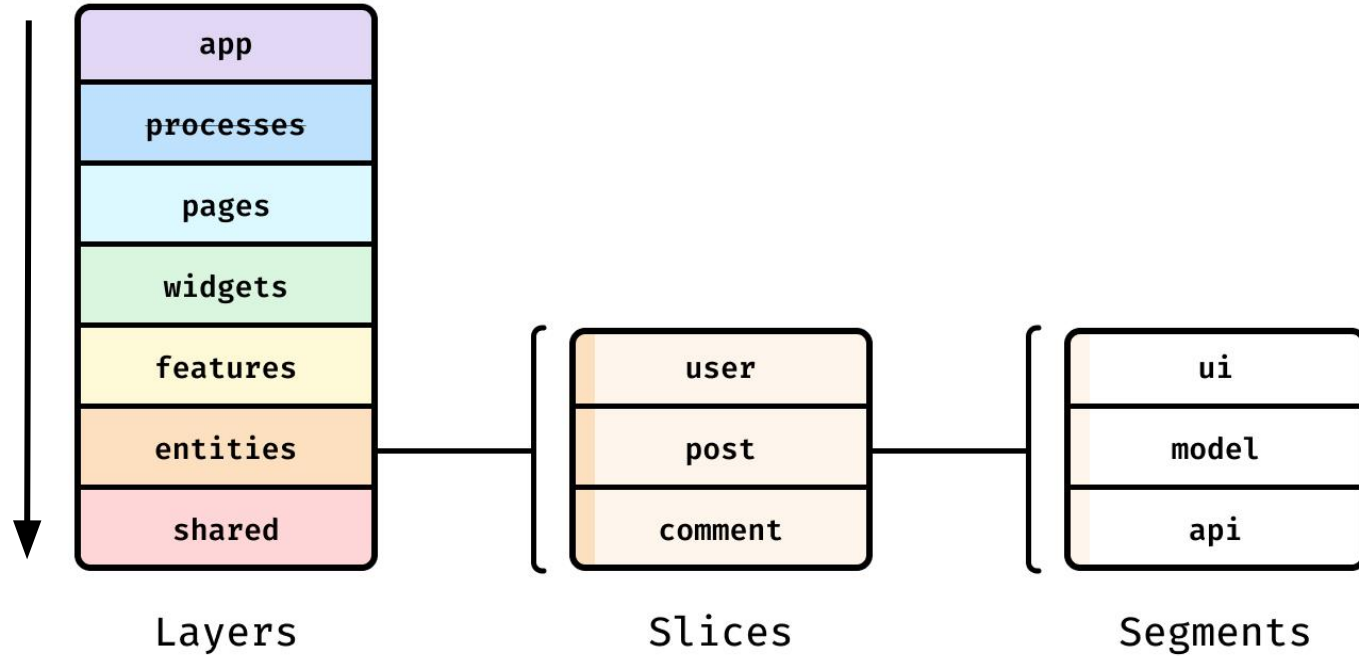
Clean Architecture in Frontend 🤔



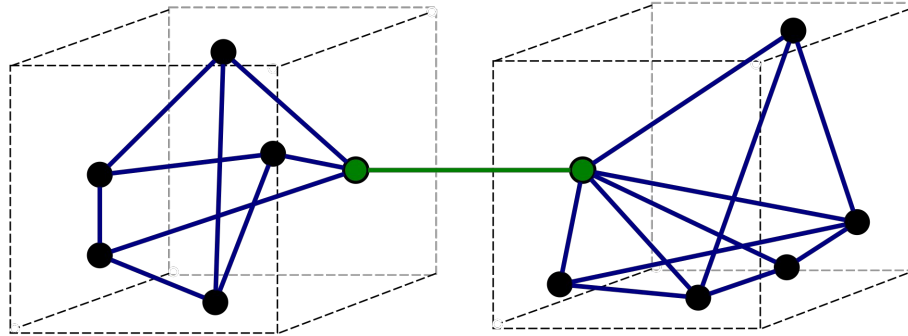
# Architecture

- >  api
- >  assets
- >  components
- >  context
- >  features
- >  fragments
- >  hooks
- >  infrastructure
- >  libs
- >  pages
- >  server
- >  store

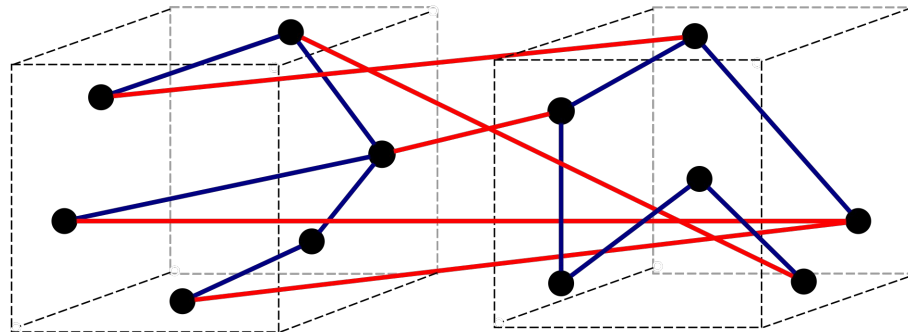
# Feature-Sliced Design



# Feature-Sliced Design



a) Good (loose coupling, high cohesion)



b) Bad (high coupling, low cohesion)

**Recap**

**Set up your npm or switch to X**



**Switch to a new gen transpiler**



**Run jobs in parallel**



**Do not run all tests**





**Do not create yet another architecture**



# Recap

Set up your npm or switch to X



Switch to a new gen transpiler



Run jobs in parallel



Do not run all tests



Do not create yet another architecture



**Have a good one**

