Building the Monorepo

kchau@microsoft.com
Flywheel Team

About Monorepos

- JS codebases have grown, and are maintained by **thousands of devs**
- Modularity is needed: **package as unit of modularity**
- Related packages are updated at the **same commit**
- Monorepo require a **Monorepo Management Stack**

Monorepo Management Stack

1. Workspace-enabled package manager

- **Installs dependencies** for all packages
- **Links internal packages** to satisfy the node resolution algorithm
- Handles **dependency resolution** for all packages
- Optional: **hoisting**, **strictness** enforcement (phantom deps)
- On the market: `yarn`, `pnpm`, `rush`, `lerna + npm`

2. Task scheduler & runner

- **Runs npm scripts** for all packages
- **Optimize** task run speeds at the dev machine and CI
- Optionally in **topological** order or in **parallel**
- On the market: `lerna`, `wsrun`, `rush`, `pnpm recursive`, `lage`

3. Package publish tool

- Automated management of **semver**
  - Change description files or commit messages
- **Validation** of description of changes
- **Synchronize versions** between npm registry and git repository
- Automated **changelog creation**
- On the market: `rush`, `lerna`, `semantic-release`, `beachball`

Our Focus: Task scheduler & runner

Problem statement

Create a task runner that optimizes package tasks in a monorepo for a single machine

Current state

- JS monorepos in the wild run with **all kinds of workspaces**
- The state-of-the-art monorepo task runners are **not optimized**
- **CPU cores sit idle** for topological scripts
- Large monorepos generally have **clustered graph** of related packages

Philosphy

- Distribute work via smaller libraries with multiple owners
- Leverage OSS as much as possible
- Support package.json scripts as the script runner

Requirements of a task runner

- **Open sourced**
  - easily shared, public development demands **polish**
  - **easily contributed** to by many groups
- Works with all workspace implementations
- Easy setup
- Minimize idle CPU cores
- **Sublinear increase** in build time per package

Prior Art

- This should sound familiar because Vincent made a version for Midgard

Lage

“ *v. to make (Norwegian); pr. LAH-geh* ”

- Open sourced: https://github.com/microsoft/lage
- Easy to integrate with existing codebase
- Scales up with pipelining
- Scales out with caching and scoping

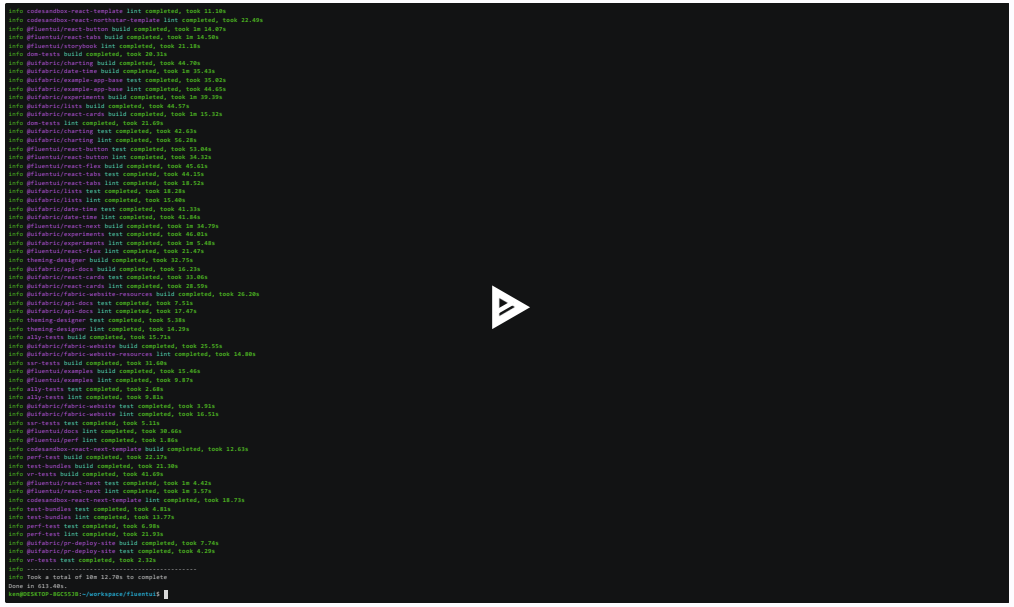
Collaboration

- **OneDrive/SharePoint**: rush showed us incremental builds
- **Midgard**: backfill cache, task-scheduler
- **Flywheel**: pipeline config, workspace-tools, `lage` tool
- **FluidX**: p-graph promise graph that supports priority queuing

What does it look like?

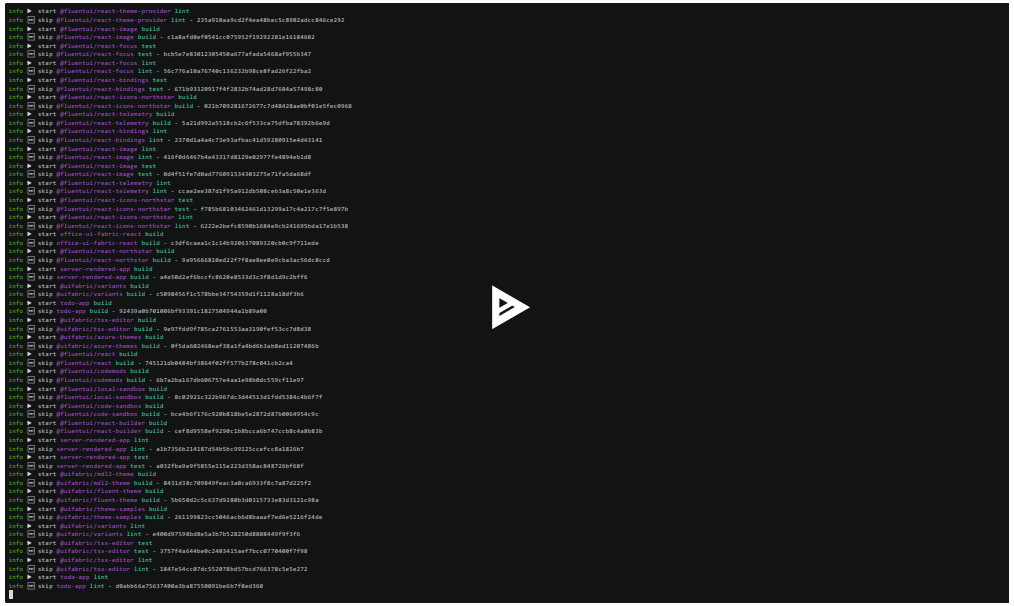
Full Build

```
$ lage build test lint --grouped --verbose --reset-cache
```



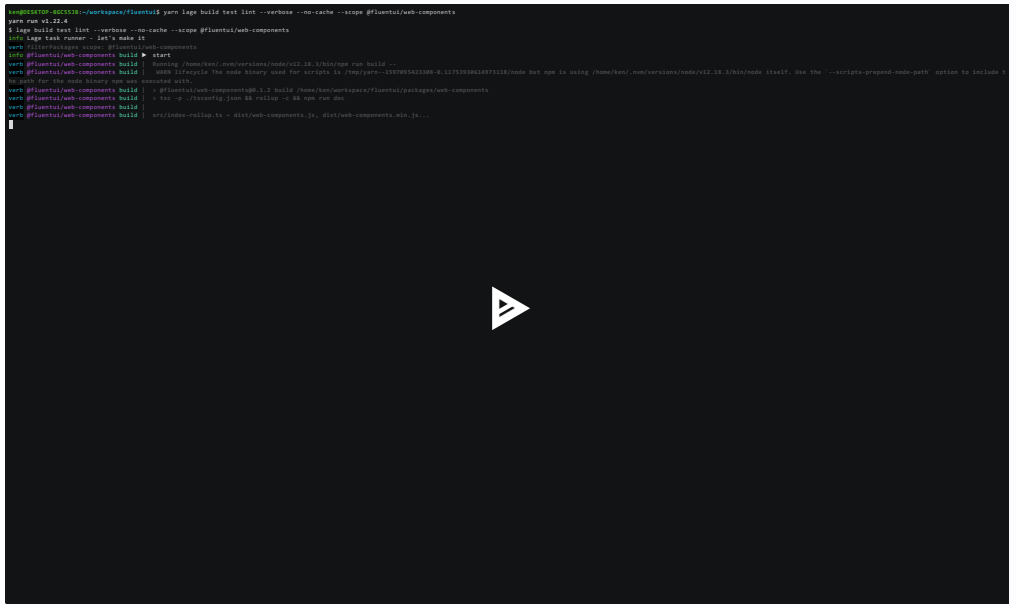
Cached Build

```
$ lage build test lint --grouped --verbose
```



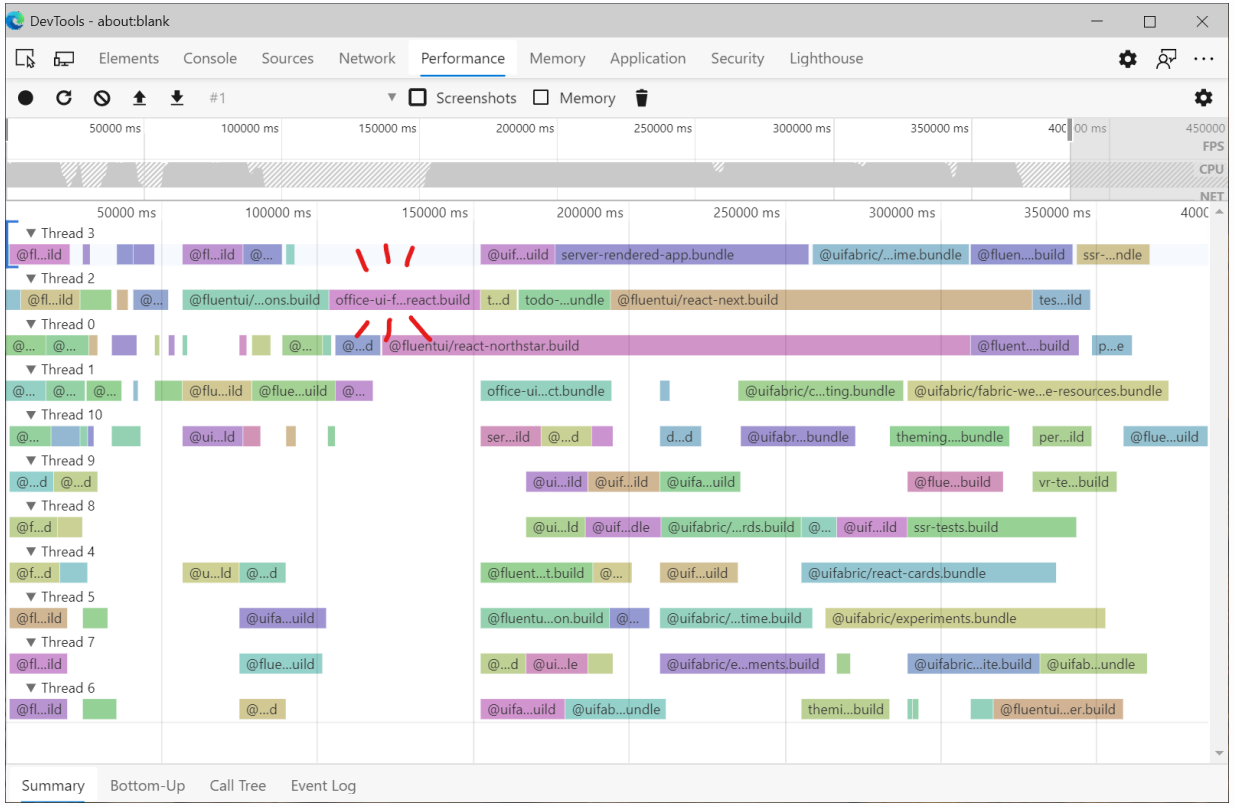
Scoped Build

```
$ lage build test lint --grouped --verbose --scope @fluentui/web-components
```



Profiling

```
$ lage build test lint --profile
```



How does it work?

https://microsoft.github.io/lage/guide/levels.html

How to try it at home?

https://microsoft.github.io/lage/guide/getting-started.html

1. npm scripts (build, test, lint) are at package level
2. `npx lage init`
   - creates a `lage.config.js` - configure it
   - adds `lage` as a dep
3. `yarn lage build` or `npm run lage build`

Future

- `lage` is a great solution for a **single machine**, not distributed builds
- Microsoft solution is buildxl
  - lage spits out info for `buildxl` now!

```
lage info build --reporter json
```

- Another popular distributed build solution: bazel
  - lage can potentially spit out WORKSPACE & BUILD

More info

Github:

https://github.com/microsoft/lage

Documentation:

https://microsoft.github.io/lage/

Complex Configuration:

https://github.com/microsoft/fluentui/blob/master/lage.config.js