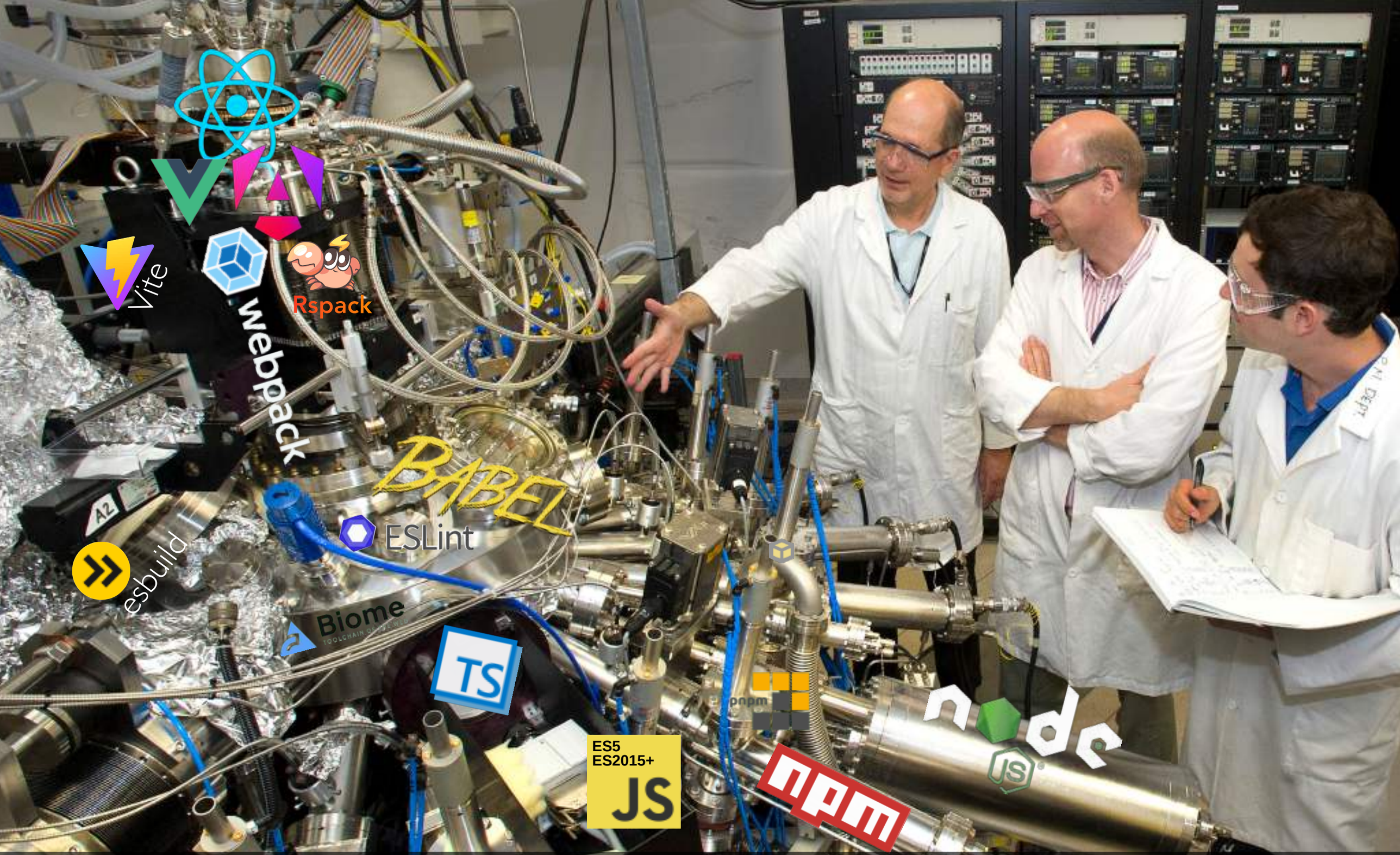
Tooling for a modern Front-End Build



JavaScript Ecosystem Today

Default Tooling

Setting up a modern JavaScript project is difficult.
Most framework ecosystems provide a scaffolding to make this task easier.

```
npm install -g @angular/cli
ng new awesome-ng
```

Angular CLI: https://angular.io/cli

using vite:

```
npm create vite@latest my-react-project -- --template react-swc-ts
```

The official React documentation is recommending a framework: Next.js or ReactRouter
https://react.dev/learn/creating-a-react-app#full-stack-frameworks

```
npm init vue@latest
```

create-vue: https://github.com/vuejs/create-vue (based on vite)

Vite for Build-Tooling

"The Build Tool for The Web"

https://vite.dev/

Frontend Build Tools/Bundlers

Legacy:

webpack
MODULE BUNDLER

REQUIRE.JS

browserify

GRUNT

Gulp

"ad-hoc bundling"

Snowpack
(using esbuild) (deprecated)
(using webpack or rollup for production)

Modern:

Vite
(using esbuild)
(using rollup for production)

PARCEL
Blazing fast, zero configuration web application bundler
(using SWC)

Lightning CSS
(written in Rust) https://lightningcss.dev/

Bleeding Edge:

Biome
TOOLCHAIN OF THE WEB
https://biomejs.dev/
(written in Rust)

Turbopack
https://turbo.build/pack
(written in Rust)

Bun
(written in Zig)
https://bun.sh/

Rspack
(written in Rust)
https://rspack.rs/

Rsbuild
https://rsbuild.rs/

Low-Level:

rollup.js

esbuild
https://esbuild.github.io/
(written in Go)

SWC
https://swc.rs/
(written in Rust)

Rolldown
rollup written in Rust
https://rolldown.rs/

Oxc

Niche:

jspm.io

WMR (preactjs/wmr)
(using esbuild / using rollup for production)

Bazel

@web/dev-server
https://modern-web.dev/
(using esbuild / using rollup for production)

https://bundlers.tooling.report/
https://css-tricks.com/comparing-the-new-generation-of-build-tools/

Vite is the default tooling for most modern frontend framework setups:

https://viteconf.org/

The Future?

`void(0)` - Next Generation Toolchain for JavaScript

Vite

Vitest

Oxc

Rolldown

October , 2024: VoidZero founded by Evan You (creator of Vue and Vite).
A company dedicated to building unified development toolchain for the JavaScript ecosystem.

October , 2025: Announcement of Vite+: the unified toolchain for web JavaScript ecosystem wit a commercial license.

https://voidzero.dev/posts/announcing-voidzero-inc
https://voidzero.dev/posts/announcing-vite-plus

https://voidzero.dev/
https://viteplus.dev/

Dependency Management & Script Runner

Declare & Resolve project dependencies.
Orchestrate other tools.

npm · pnpm · yarn · Bun

Build Automation & Bundling

Build one or several bundled asset files for deployment.
Resolve module dependencies.
Optimize asset files for production.

Vite · TurboPack · Rsbuild · Rspack · rollup.js · webpack (MODULE BUNDLER) · PARCEL (Blazing fast, zero configuration web application bundler) · Rolldown · Bun

Transpilation

Transform development sources (ES2015+ / TS / JSX) into ES5.

esbuild (written in go) · BABEL · SWC · TypeScript · Oxc

Static Type-System

Check type correctness of source code with (optional) static types at development time.

TypeScript

Linting

Static code analysis.

ESLint · Biome (TOOLCHAIN OF THE WEB) · Oxc

Source Formatting

Automtic formatting of source code.

Prettier · Biome (TOOLCHAIN OF THE WEB) · Oxc

Node Package Manger

- Packages can be local (for the current project) or global
- `package.json` describes a package or project including it's dependencies
- packages are stored in `node_modules`
- hierarchical dependencies: dependencies can include their own dependencies
  (you can have several versions of a package in your project)
- Dependencies are versioned according to semantic versioning (https://semver.npmjs.com/)
- Starting from npm 5, exact versions are listed in `package-lock.json`
  (note: `npm install` still upgrades top-level packages if no exact version is in `package.json`)
- Public Repository: npmjs.org
- Config: `.npmrc`

Tip: `npm config set save-exact true`

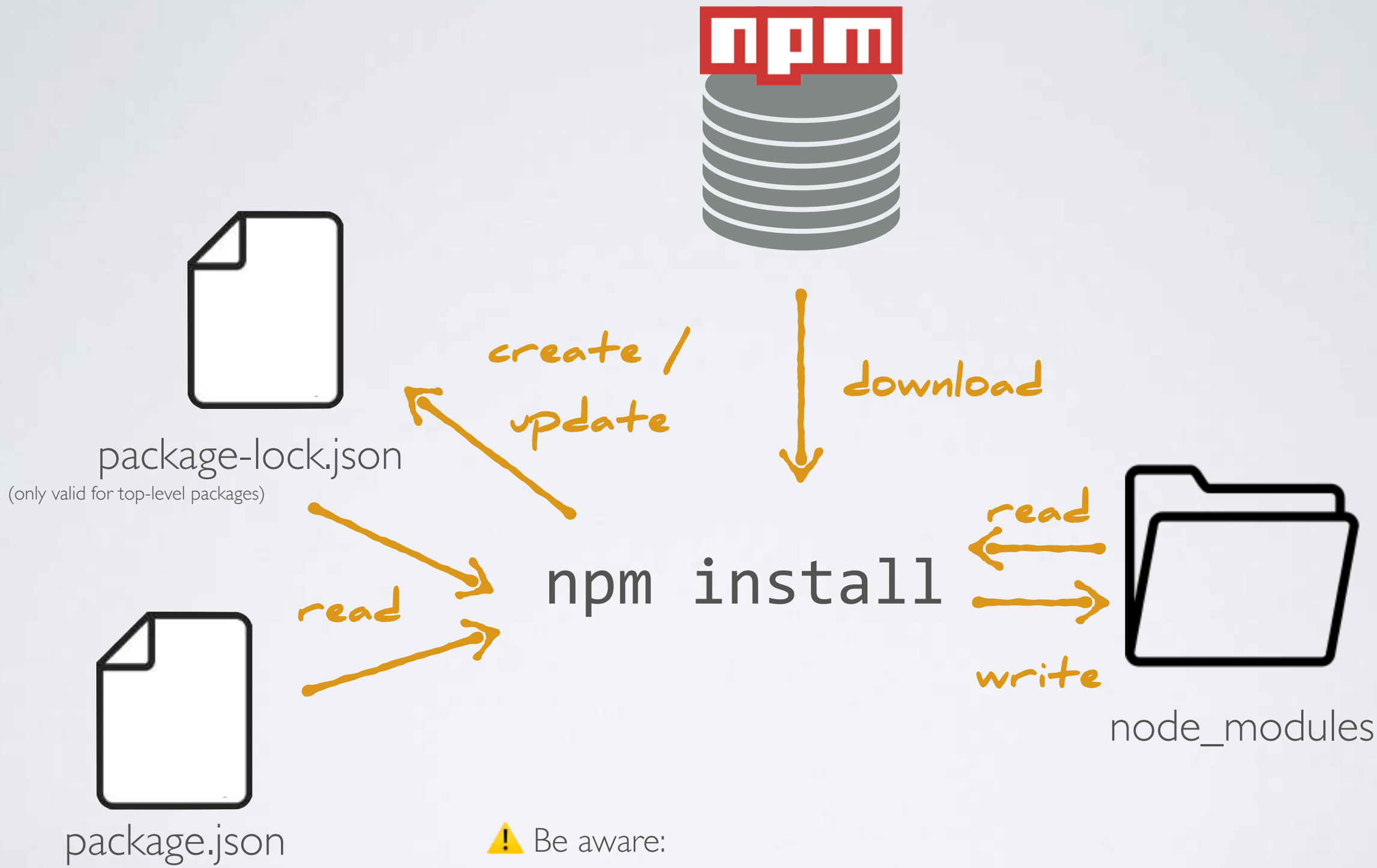
See: https://semver.npmjs.com/

Typical commands:

```
npm search
npm info
npm install
npm uninstall
npm list
npm update
npm init
npm root
npm config
npm ci
npm audit
```

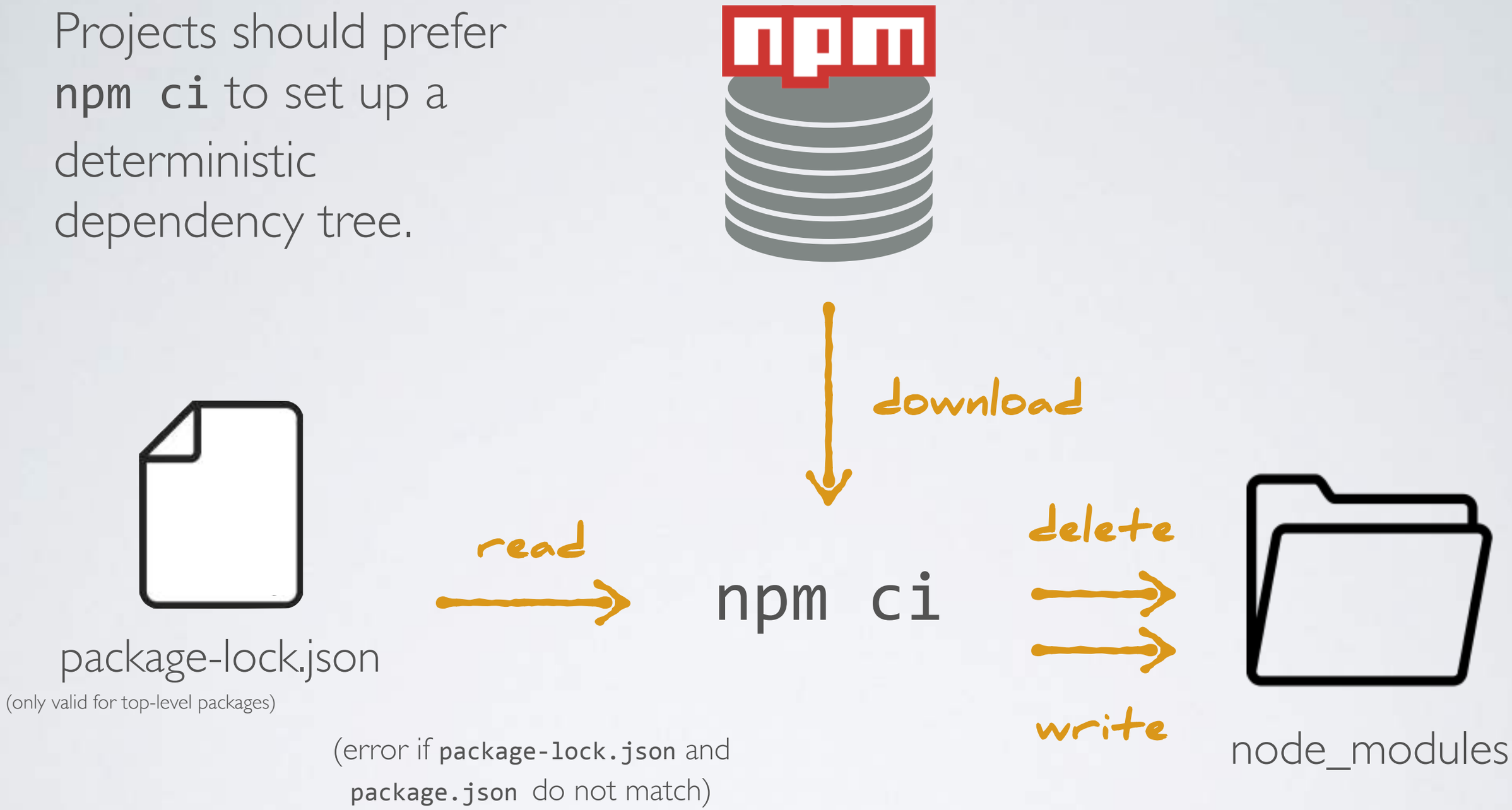
Flags:

```
--global / -g
--help
--save-dev / -D
--save-exact / -E
```



⚠️ Be aware:
`npm install` does update your `package-lock.json` to keep it "automatically" in sync with the `package.json` 🤯
To enforce a deterministic dependency resolution use `npm ci` instead.

Clean Install:
Projects should prefer `npm ci` to set up a deterministic dependency tree.



Corresponding commands with other package managers:

```
pnpm install --frozen-lockfile
yarn install --frozen-lockfile
```

https://docs.npmjs.com/cli/ci

npm Ecosystem: Challenges

Sheer amount of libraries/frameworks/tools:
- how to choose?

Short life-spans of libraries/frameworks/tools:
- how to avoid avoid risk for switching/re-writing
- how to avoid re-learning for every project?

Controlling and auditing dependencies:
- How to avoid security risks ("supply chain attacks")?
  - 2016: Leftpad Incident
  - 2018: Event-Stream Hacked
  - 2021: Malware found in coa (6 mio weekly downloads) and rc (14 mio weekly downloads)
  - 2025: Several organized supply chain attacks on a wide range of packages (shai-hulu, s1ngularity -180+ npm packages with over 2.5 billion weekly downloads compromised)

Good read: https://hackernoon.com/im-harvesting-credit-card-numbers-and-passwords-from-your-site-here-s-how-9a8cb347c5b5

The JavaScript Dependency "Situation"

```
>npx create-react-app react-project

...

added 1909 packages from 732 contributors
found 0 vulnerabilities

> du -hs react-project/node_modules/
252M    react-project/node_modules/
```

```
> vue create vue-project

...

added 1324 packages from 987 contributors
found 0 vulnerabilities

> du -hs vue-project/node_modules/
175M    vue-project/node_modules/
```

```
>ng new angular-project

...

added 1600 packages from 1278 contributors
found 0 vulnerabilities

> du -hs angular-project/node_modules/
523M    angular/node_modules/
```

Note: recent project-starters like `create-vite` have massively reduced the initial dependencies ...

... recent improvements:

```
npm create vite@latest my-react-app -- --template react-ts

npm ci

added 88 packages, and audited 89 packages in 1s
```

```
npx @angular/cli@latest new awesome-ng
...
npm ci
...
added 920 packages, and audited 921 packages

du -hs node_modules/
375M    node_modules/
```

```
npm init vue@latest awesome-vue

Vue.js - The Progressive JavaScript Framework

✔ Add TypeScript? … Yes
✔ Add JSX Support? … No
✔ Add Vue Router for Single Page Application development? … Yes
✔ Add Pinia for state management? … No
✔ Add Vitest for Unit Testing? … Yes
✔ Add Cypress for End-to-End testing? … No
✔ Add ESLint for code quality? … Yes
✔ Add Prettier for code formatting? … No

...

added 331 packages, and audited 332 packages in 56s
```

Maintenance

A "small" Angular project from 2018:

```
>cd angular-project-from-2018
>nvm use 8
>npm install

...

added 1288 packages from 1314 contributors
found 1396 vulnerabilities
(985 low, 18 moderate, 391 high, 2 critical)
```

However `npm audit` vulnerabilities might not be a good metric: https://overreacted.io/npm-audit-broken-by-design/

Real-World example:
In-house React component library, 2 years untouched:

```
>npm audit

...

found 165426 vulnerabilities (109880 low, 526 moderate, 55018 high, 2 critical) in 4149 scanned packages
  run `npm audit fix` to fix 164670 of them.
  678 vulnerabilities require semver-major dependency updates.
  78 vulnerabilities require manual review. See the full report for details.
```

However npm audit vulnerabilities might not be a good metric: https://overreacted.io/npm-audit-broken-by-design/

npm belongs to a private company!

In April 2020 npm was acquired by GitHub.
(GitHub was acquired by Microsoft in 2018)

- the npm registry is a *centralized* system owned and operated by a private company (GitHub/Microsoft)

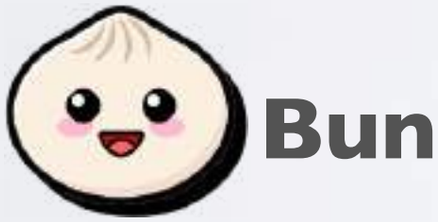
- the npm registry is not open-source

JSConf EU 2019: the economics of open source
https://www.youtube.com/watch?v=MO8hZIgK5zc

Alternatives to npm

yarn: https://yarnpkg.com
Initially faster and deterministic compared to npm. Today no big difference any more. Also using the public npm registry.

pnpn: https://pnpm.js.org/
A drop-in-replacement which keeps node_modules in a central local repository (similar to maven).



Bun is also a package manager
https://bun.com/package-manager

Deno 2 is also a package manager.
https://deno.com/blog/your-new-js-package-manager

Deno originally promoted to get rid of traditional node_modules / package-managers and to declare dependencies directly via urls..

Enterprise Concerns

There are options for a private npm registry:

- Nexus
  https://help.sonatype.com/repomanager3/nexus-repository-administration/formats/npm-registry
- Artifactory
  https://jfrog.com/help/r/jfrog-artifactory-documentation/npm-registry
- Azure Artifacts:
  https://docs.microsoft.com/en-us/azure/devops/artifacts/overview
- verdaccio: http://www.verdaccio.org/
- Github Packages: https://github.com/features/packages
- Gemfury: https://gemfury.com/

```
npm config list
npm config set registry <registry url>
```

Config files:
`<project>/.npmrc`
`$HOME/.npmrc`

Updating Dependencies

Get information about outdated npm-packages:

```
npm outdated
npm update
```

```
yarn outdated
yarn upgrade
yarn upgrade-interactive
```

`npm outdated` / `yarn outdated` also show the latest versions available.
`npm update` / `yarn upgrade` only update within the version range specified in `package.json`

Be careful with manually changing `package.json`: remember that `package-lock.json` resp. `yarn.lock` has to be updated too ...

There are also 3rd party tools that can help with the task:

```
npx npm-check -u
```

https://github.com/dylang/npm-check

```
npx npm-check-updates
```

https://github.com/raineorshine/npm-check-updates

npm run <script>

execute npm scripts

```
npm start
npm test
npm run build
npm run lint
```

package.json

```
"scripts": {
    "start": "lite-server",
     "lint": "eslint src/**/*.js"
  },
```

In addition to the shell's pre-existing `PATH`, `npm run` adds `node_modules/.bin` to the `PATH` provided to scripts.

Passing arguments to the script command:

```
npm run test -- --grep="pattern"
```

https://docs.npmjs.com/cli/run-script

Running npm Binaries

`npx <command>` (npm 5.2 or later) https://docs.npmjs.com/cli/v8/commands/npx

- Example: `npx npm-check@latest -u`
- executes npm package binaries
- executes the binaries either from a local node_modules/.bin or from a central cache, installing any packages needed in order for <command> to run.

`npm create <initializer>` (npm 6 or later) https://docs.npmjs.com/cli/v8/commands/npm-init

- alias for `npm init <initializer>`
- Example: `npm create vite@latest` => `create-vite` is an npm package
- executes npm package binaries of the package `create-<initializer>`
- executes the binaries either from a local node_modules/.bin, or from a central cache, installing any packages needed in order for <command> to run.

For `npx` and `npm create` it is a good practice to specify the `@latest` package version.

`npm audit` (npm 6 or later)

npm maintains a databse of known JavaScript package vulnerabilities.

Scan your project for vulnerabilities:

```
npm audit
```

Automatically install any compatible updates to vulnerable dependencies:

```
npm audit fix
```

For typical SPA toolchains npm audit reports many "false positives":
- *npm audit: Broken by Design:* https://overreacted.io/npm-audit-broken-by-design/

https://docs.npmjs.com/cli/audit

Managing Node Versions

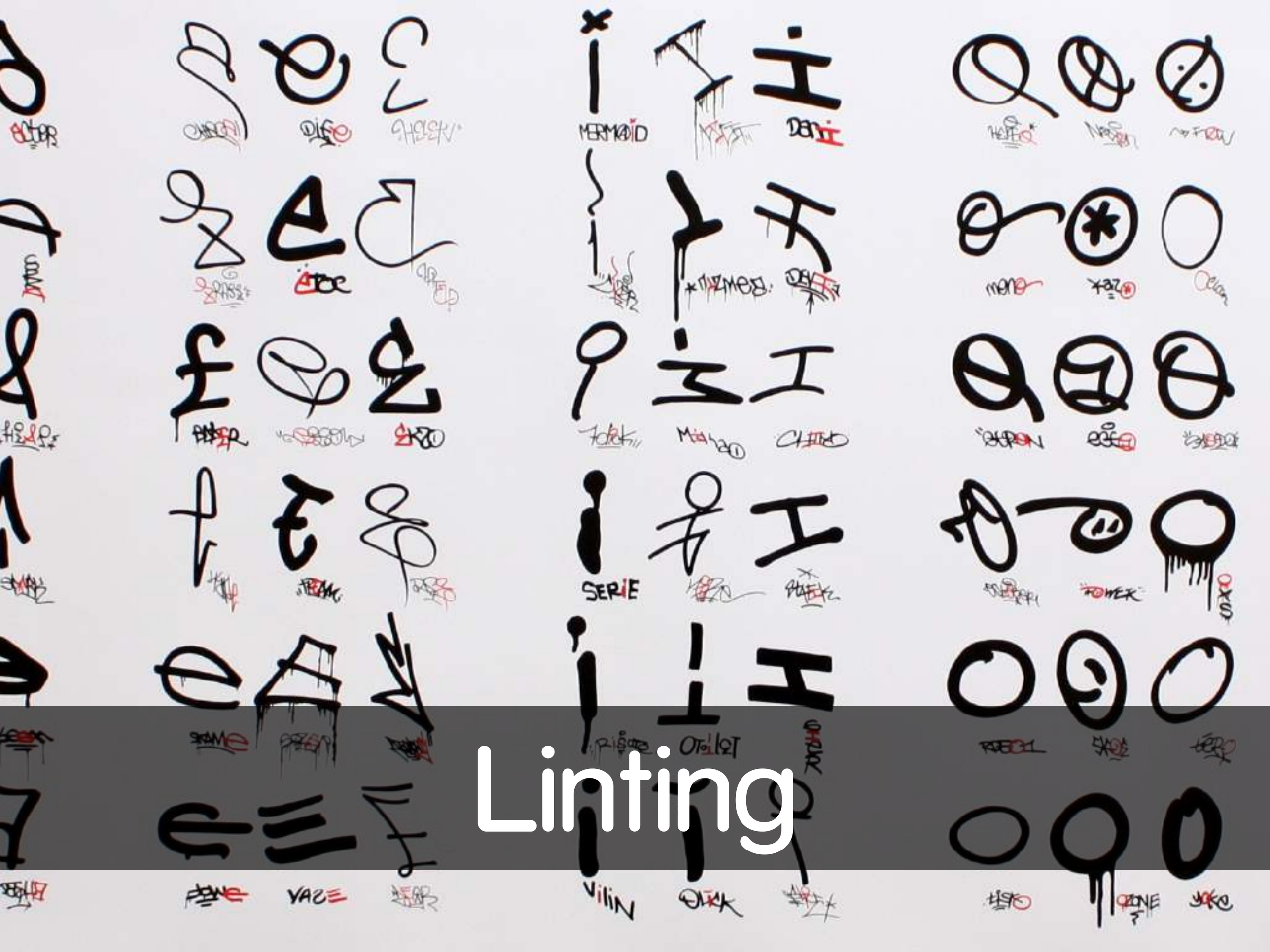
`nvm` - Node Version Manager:

A CLI tool for managing multiple Node installations

- nvm for macOS
  https://github.com/nvm-sh/nvm
- nvm-windows
  https://github.com/coreybutler/nvm-windows

Alternatives:

- fnm: Fast Node Manager - https://github.com/Schniz/fnm
- Volta - https://volta.sh/
- N - https://github.com/tj/n



Linting

ESLint

http://eslint.org/

ESLint is the state of the art linter for JavaScript, TypeScript and JSX.

ESLint is very configurable and extensible to use a "standard" style or to adapt to your style.

There are popular rule sets by goolge, airbnb or StandardJS.

Getting started: `eslint --init`

Configuration: `eslint.config.mjs`

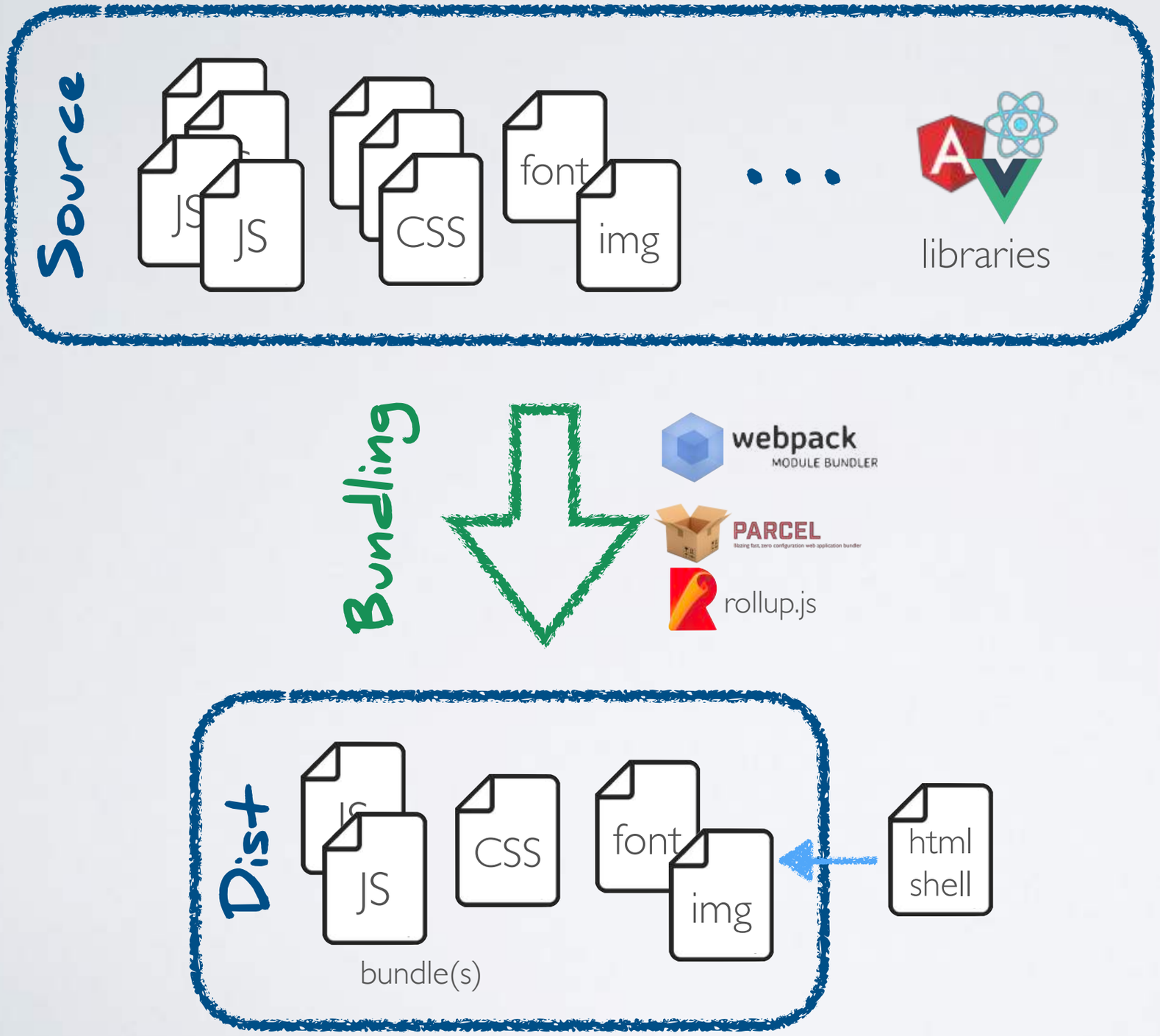
Biome
TOOLCHAIN OF THE WEB

https://biomejs.dev/

Biome is an alternative to ESLint, that is quickly gaining popularity ...

Bundling

Bundling



Development artifacts:
- fine-grained
- not optimized
- contain unused code

Deployment artifacts:
- coarse-grained
- optimized (size, performance ...)
- unused code is eliminated
- cacheable

Bundling

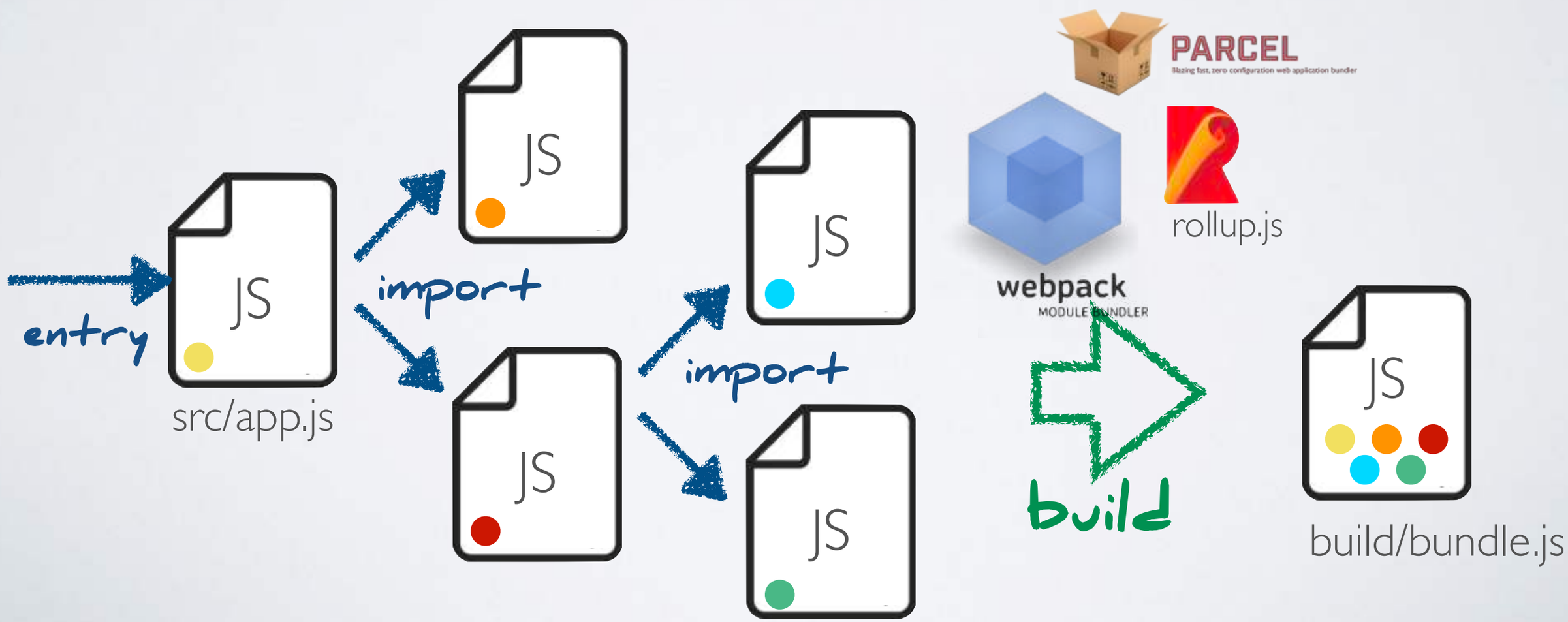
(Development Time Build Toolchain)

- Resources are optimized
  - Code is minimized
  - Bundles are coarse grained, network overhead is minimized
- Cache-Busting mitigates caching problems
- ES2015 modules prevent polluting the global namespace

Modules at Build Time

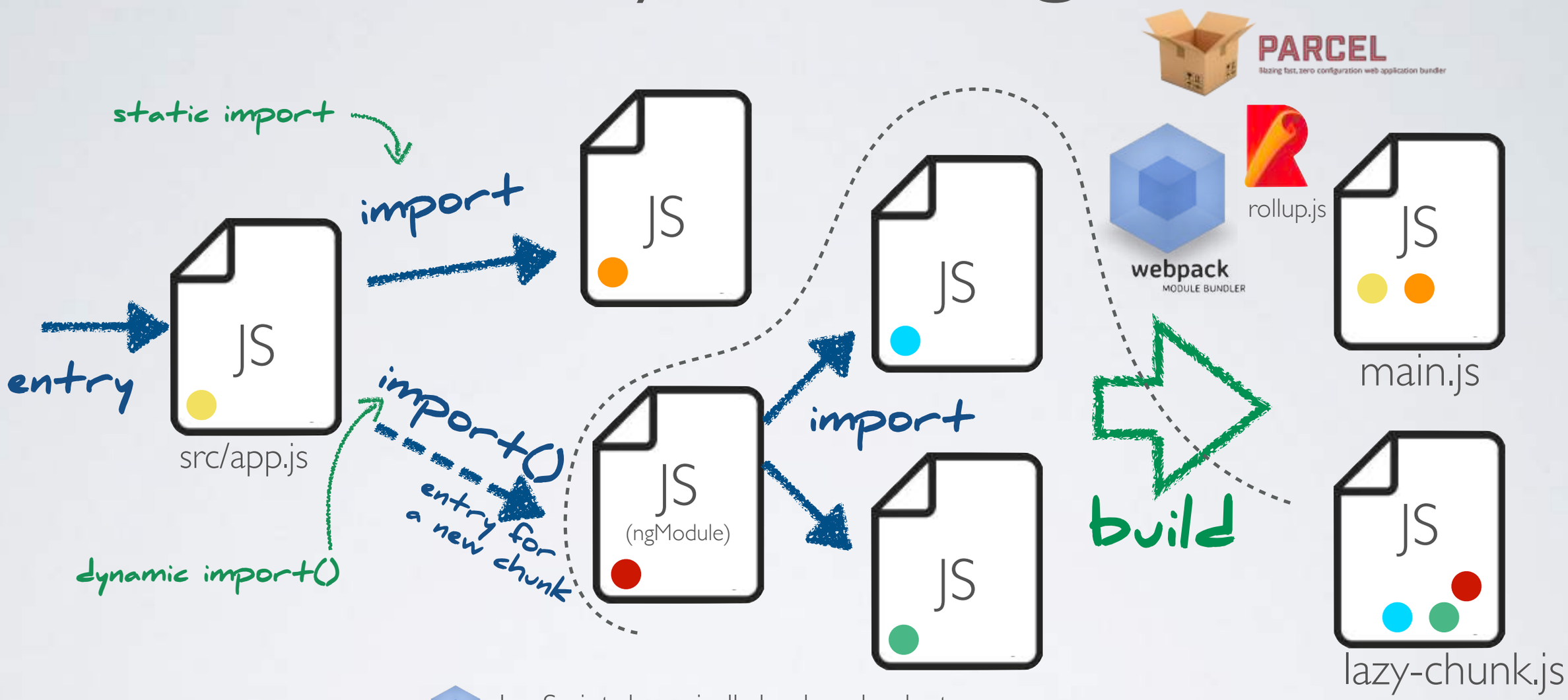
The primary use-case for modules today is build-tooling.
Modern *bundlers* (WebPack, Rollup, Parcel ...) work with modules:

- build a dependency tree based on fine grained ecma script modules (import/export)
- create coarse grained JavaScript bundles which are optimized to be loaded by a browser.



traditional

Lazy Loading



JavaScript dynamically loads a chunk at *runtime*. The code is (partially) generated by the bundler.
Frameworks have additional abstractions (Angular Router, React.lazy ...)

All the application parts (source, npm-packages ...) must be available at build time.

One build creates a *single* deployment artifact consisting of several chunks.

Vite

https://vitejs.dev/

Next Generation Frontend Tooling

A "no-bundler" toolchain: built on top of native ES Modules.
Native ES Modules are loaded during development.
Dependencies are "pre-bundled" with esbuild.
Bundling and optimzation during production build (based on Rollup).

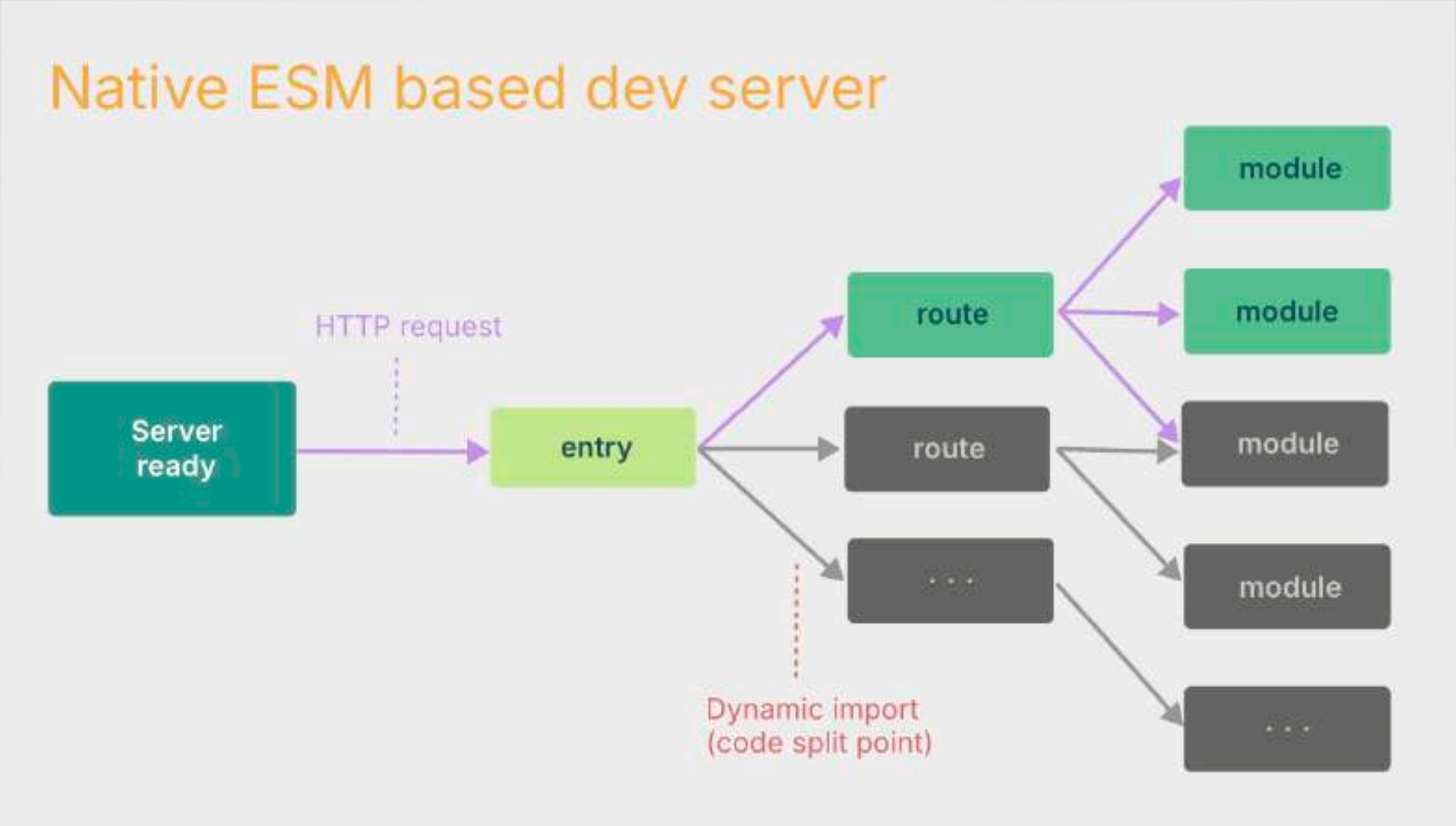
Works with Typescript, React, Preact, Vue, Svelte, lit …

Getting started:

```
npm create vite@latest
```

Vite in 100 Seconds: https://www.youtube.com/watch?v=KCrXgy8qtjM
Intro to Vite: https://www.youtube.com/watch?v=DkGV5F4XnfQ

The unbundled development workflow



The browser takes over the job of the bundler by loading code as esm modules. Vite just transforms single resources on demand.

https://vite.dev/guide/why.html

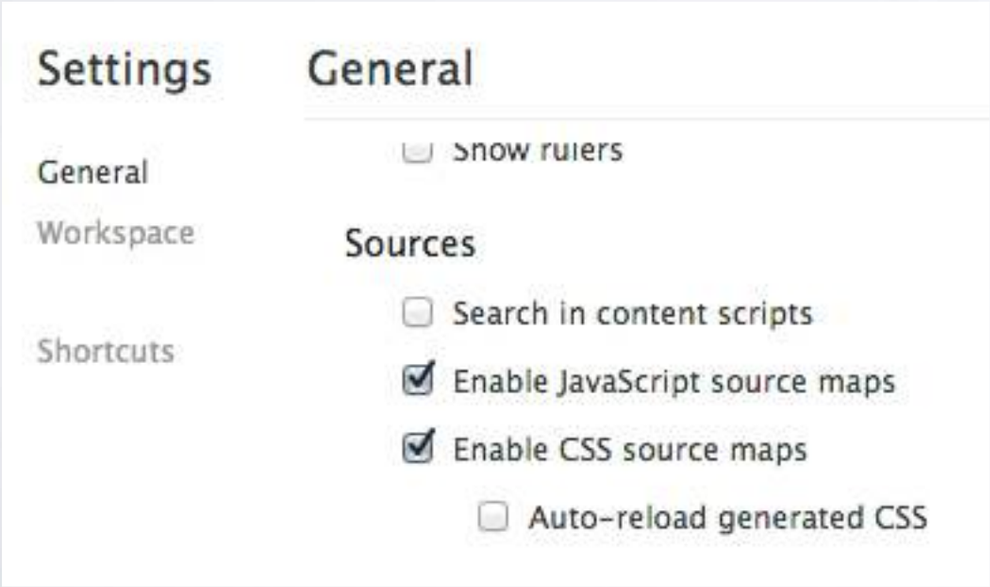
Vite uses esbuild and rollup for the production build.

In March 2025 it was announced that Vite plas to introduce "Full Bundel Mode" that will serve bundled files also in development: https://vite.dev/guide/rolldown.html#why-introducing-a-full-bundle-mode

Sourcemaps

- Build: Tools that transform sources generate a map of from the resulting artefact to the original sources:
  - `jquery.js -> jquery.min.js & jquery.min.map`
  - `main.ts -> main.js & main.min.map`

- Runtime: Browsers perform the mapping when debugging => the executed code is mapped to the original code, which is displayed for debugging

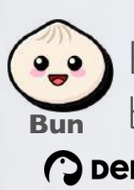


Testing

JS Testing Scenarios

(where to run tests)

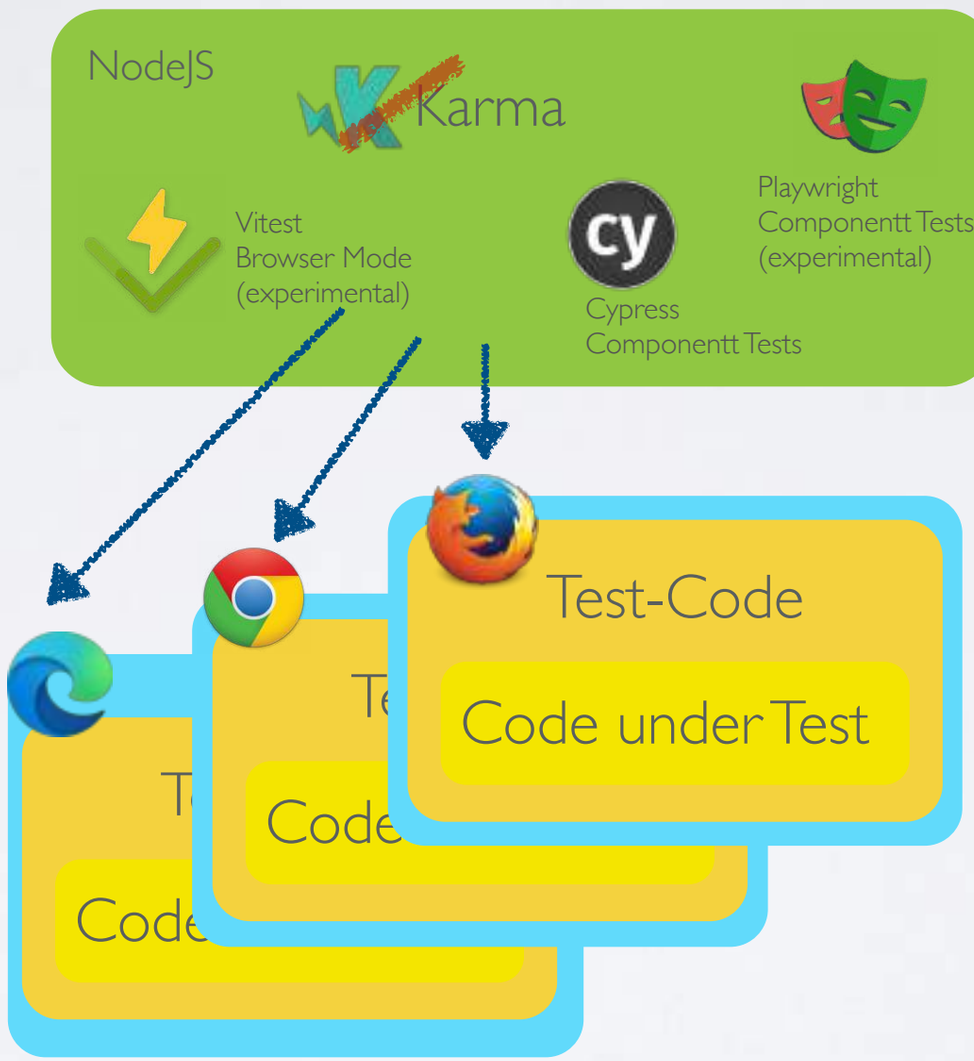
Unit-Tests in Node



Bun and Deno have built in test runners.

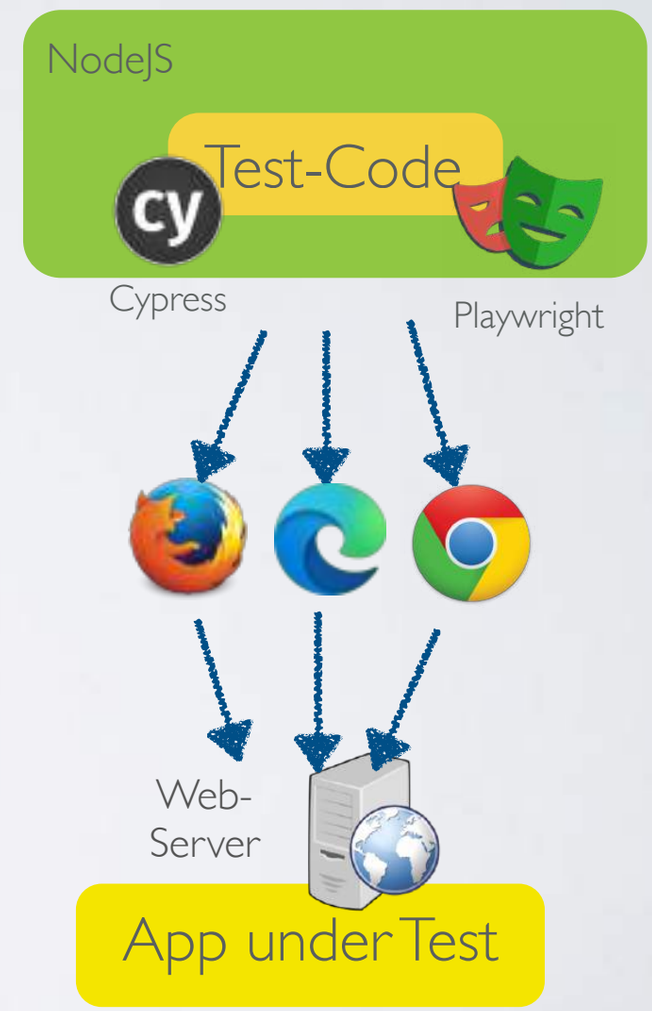
Run tests in node.
Optionaly use happy-dom or jsdom to simulate the DOM.

Unit-Tests in Browser



Run tests in browser.
Optionaly can use browsers in headless mode.

End-2-End Tests with Browser Automation



Script a browser to interact with a deployed app.

Vitest

https://vitest.dev/

A Vite-native unit test framework. It's fast!

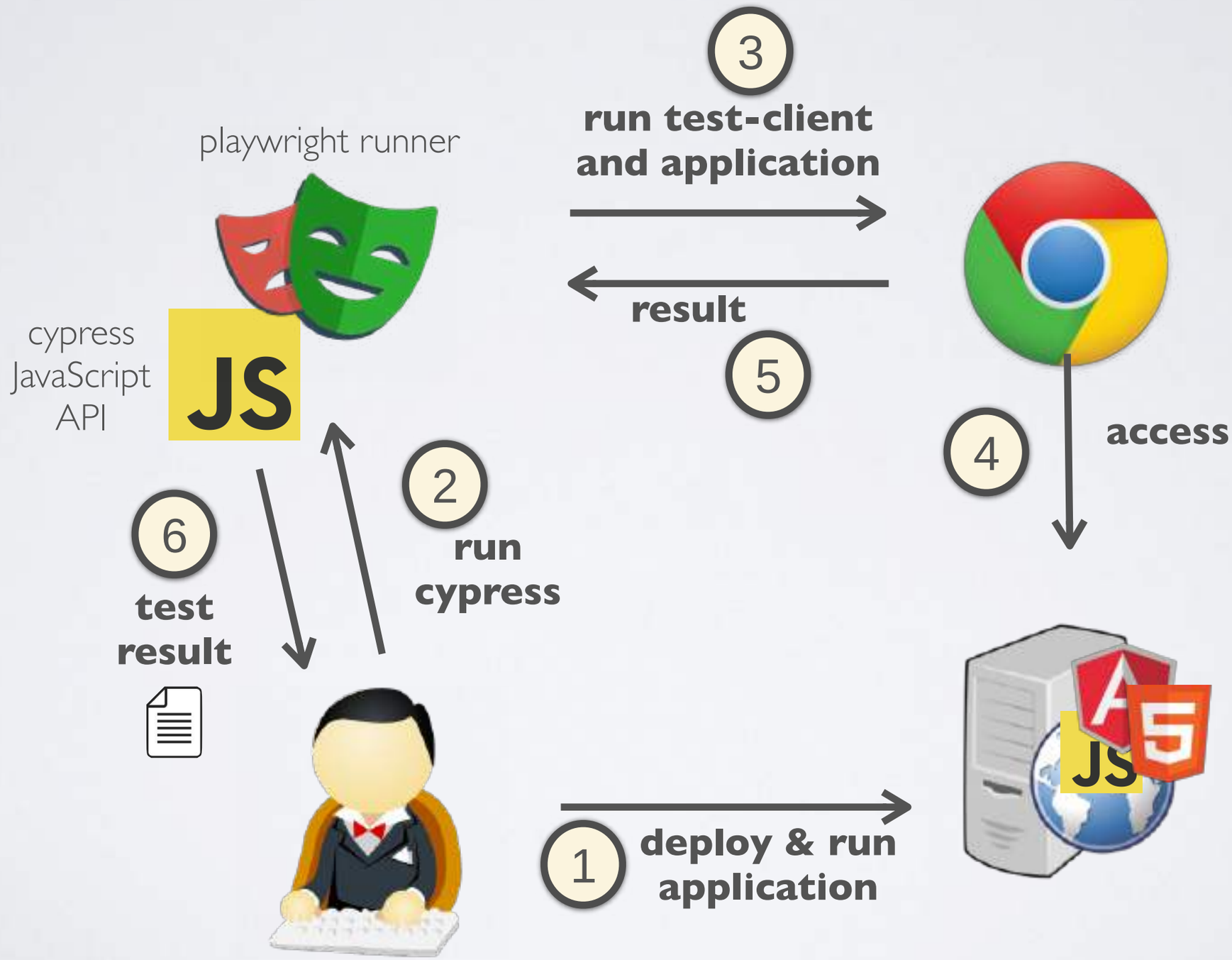
- "Zero Config": Out of the box support for ESM, TypeScript, JSX
- Re-using Vite configuration
- Jest Compatible

Playwright

https://playwright.dev/

An end-to-end test framework for modern web apps.

Playwright


playwright runner
cypress
JavaScript
API
JS
3
run test-client
and application
result
5
4
access
2
run
cypress
6
test
result
1
deploy & run
application
JS