NodeJS

LABA

PACKAGE MANAGERS & VERSIONING

Solvd NodeJS LABA

PACKAGE MANAGERS

Is a collection of software tools that automates the process of installing, upgrading, configuring and removing packages.

A `package` (dependency) is one or more files, neatly packaged together, that can be downloaded from a package registry.

A `registry` is the center of code sharing.

NPM

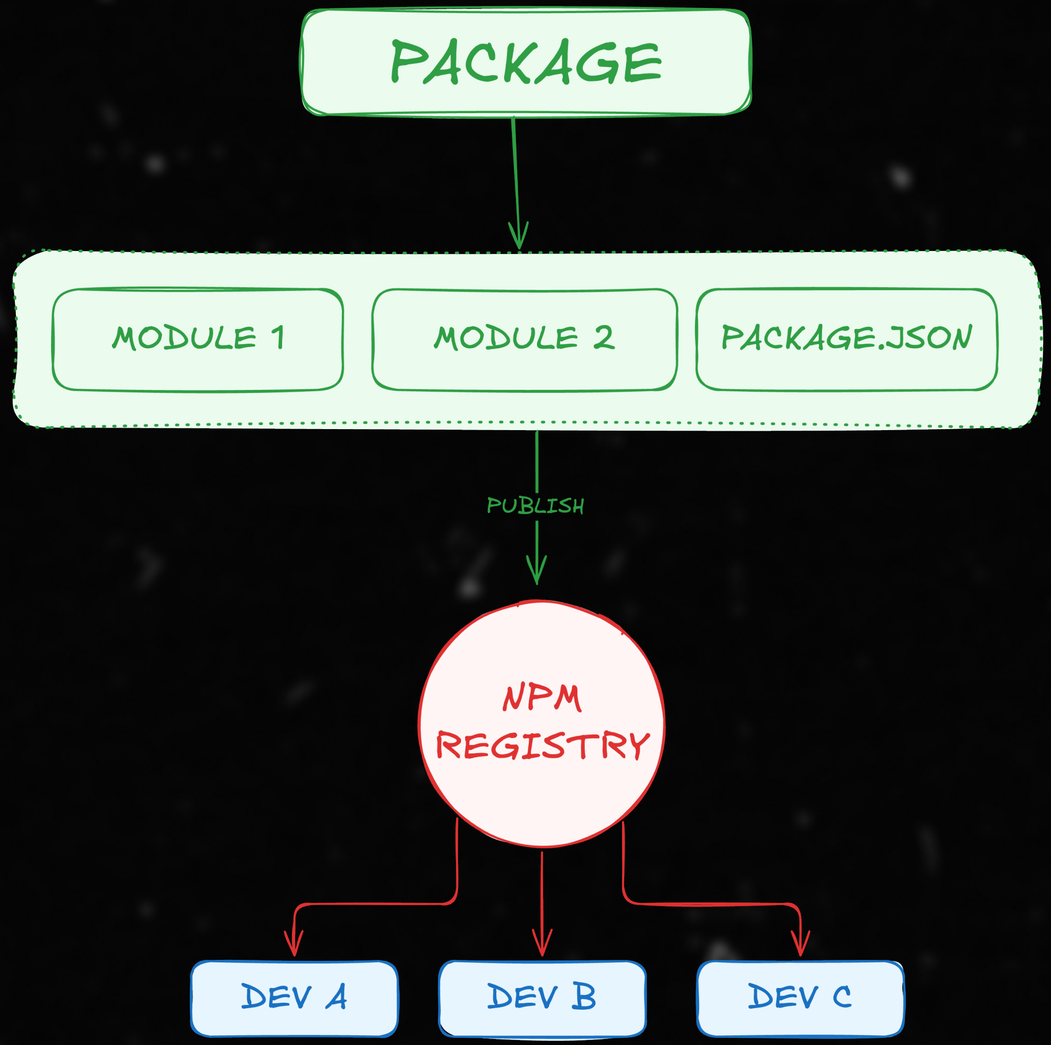
`NPM` is the official package manager for `NodeJS`, which comes bundled with node and installed by default.

`NPM registry`, is an online database of packages.

`NPM` consists of three core components:

- The website
- The Command Line Interface (CLI)
- The registry

NPM PACKAGES



Solvd NodeJS LABA

NPM

check the current version `NPM` of available on your system with: `npm -v`

A `package.json` file have the following roles on a project:

- List packages your project depends on
- Specify versions that your project can use following semantic versioning specification.
- Makes your build reproducible, allowing ease to share and distribute

`npm init` `npm init --yes`

Solvd NodeJS LABA

PACKAGE INSTALLATION

You can install npm on your system both globally and locally

Locally:

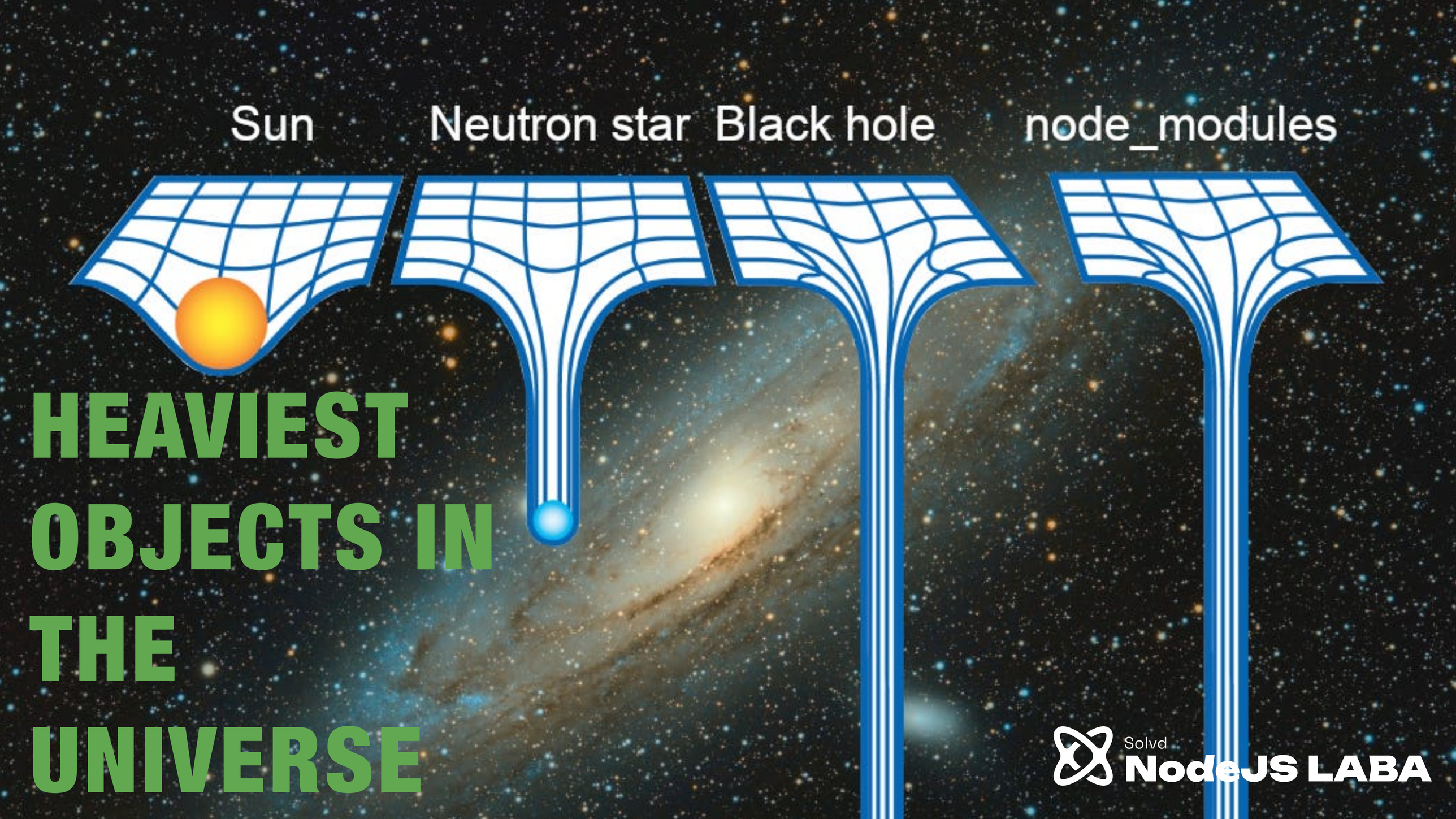
```
npm install <package_name>
```

Globally:

```
npm install <package_name> -g
```

To install a dependency only during development you can use: `npm install <package_name> —save-dev`

Solvd NodeJS LABA


Sun
Neutron star
Black hole
node_modules
HEAVIEST OBJECTS IN THE UNIVERSE
Solvd
NodeJS LABA

NPM CLI

Install a specific version of a package:

```
npm install [package_name]@[version-number]
```

Update a single package:

```
npm update <package_name>
```

List all installed packages as a dependency tree

```
npm list
```

Solvd NodeJS LABA

NPM CLI

Delete local package:

```
npm uninstall <package_name>
```

Delete global package:

```
npm uninstall <package_name> -g
```

PROS AND CONS - NPM

Pros:

- Vast package repository
- Default choice for NodeJS
- Mature Ecosystem
- Semantic Versioning
- Custom Scripts

Cons:

- Performance Issues
- Dependency Bloat
- Security Concerns
- Reliance on centralized registry
- Limited offline support

PROS AND CONS - YARN

Pros:

- Improved performance
- Offline support
- Improved error handling

Cons:

- Resource consumption
- Community fragmentation
- Potential for lock file drift
- Limited configuration
- Maintenance overhead

PROS AND CONS - PNPM

Pros:

- Shared dependencies
- Efficient Installation
- Reduced network bandwidth
- Improved cache efficiency

Cons:

- Compatibility issues
- Resource consumption
- Lockfile handling
- Community support

VERSIONING

Software versioning is the process of assigning unique name or numbers to different versions of software products.

Given a version number `major.minor.patch`, increment:

`major:` when you make incompatible API changes
`minor:` when you add backward compatible functionality
`patch:` when you make backward compatible bug fixes

VERSIONING

There different versions in software versioning represent different stages in the development and release process.

`alpha:` (0 / a)
`beta:` (1 / b)
`release candidate:` (2 / rc)
`public release:` (3)

VERSIONING

1.2.3.5 = 1.2.5

1.2.0.2 = 1.2a2

Beta with several bugs fixed

1.2.1.2 = 1.2b2

Release candidate

1.2.2.0 = 1.2rc

1.0.0-alpha < 1.0.0

1.0.0 < 2.0.0 < 2.1.0 < 2.1.1

Solvd
NodeJS LABA