Node.js (nvm)

Use the Node Version Manager (`nvm`) to install Node.js into your current user's path and switch Node.js versions on the fly.

Install `nvm` via the AUR:

```
yay -S nvm
```

Include the init script `/usr/share/nvm/init-nvm.sh` into your shell configuration to load it each time you start your terminal:

```
# bash
echo 'source /usr/share/nvm/init-nvm.sh' >> ~/.bashrc


# zsh
echo 'source /usr/share/nvm/init-nvm.sh' >> ~/.zshrc
```

Restart your terminal to reload all init scripts and you should be able to use `nvm` to install a Node.js version of your choice:

```
nvm install 12
```

Migrating globally installed `npm` packages

When you install and switch to a different `nvm` managed version of Node.js (`nvm install 14` or `nvm use 16`) you may find that your globally installed `npm` packages (e.g. `svgo`) are no longer available until you switch back to the specific version of Node.js you have been using before the upgrade or switch.

This is because globally installed `npm` packages are installed for the specific version of Node.js you happen to be using at the time of installation and placed in a directory i.e. `~/.nvm/versions/node/v16.14.0/lib/node_modules`. When you install a different version, e.g. `17.2.0` the path to your Node.js installation changes to `~/.nvm/versions/node/v17.2.0/lib/node_modules`.

Use the `--reinstall-packages-from=<version>` option to carry over globally installed packages to the new Node.js installation.

You can either pass a specific version you want to reinstall globally installed packages from or use bash string expansion to reinstall from the currently active one in use:

```
nvm install <new version> --reinstall-packages-from=<old version>


nvm install 17 --reinstall-packages-from=$(node -v)
```

Revision #6
Created 6 May 2020 20:50:04 by Sebin
Updated 26 March 2023 03:28:21 by Sebin