Changelog Generator

Generates changelogs based on Gatsby Release Process conventions:

1. Each `minor` release has its own release branch (e.g. `release/2.32`)
2. Each `minor` release branch starts from master with pre-minor `next` tag (e.g. minor `2.32.0` starts from `2.32.0-next.0`)
3. So all commits of a minor release are in a range like this: 2.32.0-next.0...2.32.0.
4. Commits of a patch release are in the usual range: 2.32.0...2.32.1.

Conventional Commits

This generator relies on conventional commits and under the hood uses tooling from conventional-changelog (adapted for Gatsby Release Process conventions).

Usage

```
node scripts/gatsby-changelog-generator/cli.js
```

```
cli.js <command>

Commands:
  cli.js regenerate <pkg>  Regenerates changelog of a given package
  cli.js regenerate-all    Regenerate changelogs of all packages in the monorepo
                           (slow)
  cli.js update <pkg>      Add new versions to the changelog of a given package
  cli.js update-all        Update changelogs of all packages in the monorepo

Options:
  --help     Show help                                                 [boolean]
  --version  Show version number                                       [boolean]
```

regenerate

Completely re-writes all entries of the package changelog published using the new release process conventions. Old versions are kept intact (relies on `<a name="before-release-process"></a>` separator inside `CHANGELOG.md` for demarcation).

This action is idempotent.

update

Add recent package versions to the top of changelog. Under the hood it reads git tags and compares with the latest recorded version in the changelog.

> *`regenerate-all` and `update-all` are the same as `regenerate` and `update` respectively but applied to all packages in the monorepo*

Automation

Updates changelogs on `lerna publish`

> *Default changelogs generated by `lerna` are disabled globally in `lerna.json`.*

This tool uses `version` lifecycle hook of lerna to automatically update changelogs on publish.

It only updates changelogs when stable releases are published. Pre-releases and canaries are ignored (caveat: `rc`, `alpha`, `beta` versions won't get their changelog entries).

See `version` field in `package.json` on actual setup (uses `lerna-version-lifecycle.js` script)

Updates changelogs in `master` automatically

Actual publishing happens in `release/*` branches, so changelogs in `master` get out of sync.

To sync them we need to update changelogs in `master` separately. This happens automatically via `update_changelogs` job in CircleCI. This job calls `update-and-open-pr.js` to update changelogs in the current branch and open a PR with suggested updated.