# NPM CLI


- Rationale
- Goals
- Benefits
- Constraints
- Delivery timeframe
- Solution
    - Jumpstart
    - CLI
- Future ideas
- Decisions
- Outstanding
- Next Steps
- Experimenting with Node.js and npm


DEV-159 - Getting issue details... STATUS

DEV-178 - Getting issue details... STATUS

**Update**

The POC setup tools has become a new Magnolia project. Here are its coordinates

Jira: https://git.magnolia-cms.com/scm/build/npm-jumpstart.git

GIT: https://git.magnolia-cms.com/projects/BUILD/repos/npm-jumpstart/browse (master [jetty+Magnolia web app ] and tomcat-bundle branches)

Jenkins: https://jenkins.magnolia-cms.com/job/build_npm-jumpstart/

Inspired by Tomáš's presentation at Lunch&Learn about Node.js and its package manager npmjs based module called CLI https://git.magnolia-cms.com/users/tgregovsky/repos/cli/browse.

## Rationale

Since enabling support for resource loading and yaml-based configuration, Magnolia has given front end developers extra tools for the simple creation of templates and tools for providing end users with updates in design and templates without the need for re-deployment. The combination of template definitions, templates and other resources created using this new offering is called Magnolia Light Modules (LM).
The recent introduction of NPM as a means for packaging and distribution of LMs additionally strengthens this offering by giving front-end developers tools for obtaining and distribution of such LMs. This encourages sharing them within different projects or even different teams.
While the enabling of resource loading and configuration via yaml has allowed front-end developers to use their favourite tools when working with Magnolia and NPM based tooling greatly simplified distribution of results of their work, the problem remains (same as previously with JCR based configuration) with the number of points at which even an experienced user might make mistakes by either placing configuration at the wrong location, unintentionally omitting some mandatory structures, or simply misspelling names of required structures in configuration files.
Aside from that, the wide range of configuration parameters offered by Magnolia makes it difficult for front-end developers to remember all possibilities, forcing them to frequently consult documentation in order to be able to proceed with template definition work.

## Goals

**Provide an npm package for front-end devs to get started fast and develop painlessly (*hopefully*) with Magnolia and light development**.

More precisely, provide command line based tooling (CLI) for front-end developers that can be used to automate common tasks, reducing the number of possible error points, as well as removing repetition and allowing front-end developers to concentrate on their goals: building websites that are visually stunning and easy to use.

## Benefits

- Increased efficiency by automation of daily tasks (creation of page and component templates, assignment of components to pages, creation of dialogs associated with pages or components)
- Reduced number of possible errors (by generating common structures automatically, limiting possible errors to misspellings of component names or dialog fields added by user, but ensuring that structure is correct)
- Lowered barriers of entry for becoming front-end developers using Magnolia (learning a couple of commands to generate basic website structure vs. learning the whole configuration structure for template definitions just to create first templates)
- Additional, secondary benefits are
    - Further increases in development efficiencies due to reduced number of testing configuration cycles (due to lower number of errors / regressions)
    - Learning advanced configuration is simpler (by allowing front-end developers to inspect generated structures)
    - Ensuring better understanding of custom work within project teams (generated structures are always same no matter which developer generated them; encourage people to place additions at the same locations)

## Constraints

- Whatever structures and placements for the resulting artifacts are generated by scripts need to be aligned with the structures generated by npm magnolia-build & distribution scripts
- Scripts need to operate correctly even when one chooses to not use npm for distribution (e.g., for small and internal projects)

## Delivery timeframe

- Final version should be released, at the latest, with Magnolia 5.5.
- Alpha version for the 2016 conference in Basel.

## Solution

Since Magnolia already supports npm as a tool to build and distribute LMs, automated tasks can and should be provided by additional npm-based scripts. Scripts for downloading and starting Magnolia distributions are already in the jumpstart project.
Prototype of remaining functions (and scripts) already exists (taken from original contribution by Tomáš) and is analyzed as part of jira@DEV-178.
Scripts described below can be either installed as global and executed without need for `npm run` prefix, or just locally with it. In the case of local execution only, one needs to have scripts defined in one's `package.json file.`

### Jumpstart

Functions supported by jumpstart

- `npm start` to download and start Magnolia initially
- `start-magnolia`
- `stop-magnolia`
- `restart-magnolia`

As part of `start` operation, `magnolia.properties` in downloaded and extracted instances are also adapted.

It is possible to point scripts to existing Magnolia installations, and to start such existing installations (for users with custom locally-only available bundles).

Configuration and use of CE or EE instance of Magnolia is also possible by modification of the `package.json` file.

### CLI

Supported operations

- Creation of pages; e.g., `add-page myHome`
- Creation of components; e.g., `add-component myImage`
- Adding components availability to pages; e.g., `add-component myImage available@sampleModule:pages/myHome@main`
- Adding auto-generated components to pages; e.g., `add-component footer autogenerate@sampleModule:pages/myHome@footer`

Detailed syntax including example usage is described at https://git.magnolia-cms.com/projects/BUILD/repos/npm-cli/browse?at=refs%2Fheads%2Fcommander
Scripts allow for configuring availability and auto-generation only for page templates defined inside of the current light module/project; however components made available can come from anywhere (other light modules or other classic modules). No validation of errors against live instance of Magnolia is performed.

## Future ideas

- A custom command might allow users to download and place additional Magnolia modules automatically in `WEB-INF/lib` (resolving dependencies in that case could be tricky, or we need to limit this to downloading bundles that have all dependencies already packed).

- tl;dr parameters for experts:
  - **availability: create-component textImage -a pages/myHome@main**
  - **autogenerate: create-component textImage -g pages/myHome@main**

## Decisions

- include jumpstart part of the project
- do not use any other tools such as yeoman or gulp or others to avoid extra complexity
- none of the proposed solutions for install are ideal; explore yet another possibility. The one used by Tomáš is closer to how other tools install (e.g. gulp) and should be preferred in absence of better ideas
- use `create-page` and `create-component` instead of `add-`
- `add-availability` will stay as is
- syntax change for parameters - use `--available` and `--autogenerate` instead of `available@`
- module name is optional (if not provided, current is assumed, when provided we look for other project on the same level where current project is (in the same folder), when not found, simply fail with reasonable message)
- examples of updated syntax:
  - **create-component image**
    - Add/create component template (will create component yaml file, component ftl and component dialog)
  - **create-component textImage --available [sampleModule:]pages/myHome@main**
    - Same as above plus will set this component available on 'myHome' page in 'main' area (if area or availability configuration doesn't exist's in 'myHome' page, will create it as well)
  - **create-component footer --autogenerate [sampleModule:]pages/myHome@footer**
    - Same as above, just will do 'autogeneration' instead of availability
  - **add-availability [mtk:components/]html [sampleModule:]pages/myHome@main**
    - Make component 'html' available at 'myHome' in 'main' area
  - **create-light-module [-p path/to/light-modules-root-folder] myModule**
    - Create a new light module structure with the standard Magnolia light dev structure

## Outstanding

Items below require additional research and experimentation or discussion or decision or all of those

- Which bundle to use for jumpstart custom jetty or standard tomcat?

## Next Steps

- Exploratory task for better install of cli DEV-193 - Getting issue details... STATUS
- Refactoring for new syntax NPMCLI-8 - Getting issue details... STATUS
- Refactoring to remove need for use of "sudo" NPMCLI-5 - Getting issue details... STATUS
- Refactoring to remove need for YAML->JSON->YAML conversion NPMCLI-4 - Getting issue details... STATUS

---

## Experimenting with Node.js and npm

As I'm a newbie to Node.js I had first to get familiar with the new technology. The result of my trials and errors is the little module at ~~https://git.magnolia-cms.com/users/fgrilli/repos/mgnl-light-dev-poc/browse~~ https://git.magnolia-cms.com/projects/BUILD/repos/npm-jumpstart/browse

There are two flavours of the module on two different branches

- master: a module using Jetty + Magnolia community web app
- tomcat-bundle: a module using Magnolia's CE tomcat bundle

From the module's readme.txt

This is a PoC for a nodejs module (package) which installs and runs a Magnolia instance ready for light development.
...
Run:
- run "npm start". This will
-- connect to Magnolia's public repository on Nexus and download Magnolia (see package.json)
-- extract Magnolia
-- change some magnolia.properties (again see package.json)
-- start up Magnolia

Custom commands:
- Apart from the standard npm commands corresponding to the phases of a package lifecycle, e.g. install, start, etc.
this module provides these additional commands
-- start-magnolia, usage "npm run start-magnolia"
-- stop-magnolia, usage "npm run stop-magnolia"
-- restart-magnolia, usage "npm run restart-magnolia"