Bower - wnioski

Nie chcę wraz z projektem commitować jego zależności. Nie chcę ręcznie rozpakowywać ZIP-a np. z biblioteką JQuery lub Angular. Nie chcę pilnować, czy wyszła najnowsza wersja jQuery i jej podmieniać w projekcie. Dlatego używam bowera. Instaluję w nim jQuery, ono jest zapisywane do pliku z listą zależności. Do repozytorium commituję tylko plik z listą zależności. Gdy inna osoba w zespole pobierze z repozytorium mój projekt, nie będzie w nim ściągniętych zależności. Ta osoba sama je sobie ściągnie poleceniem **bower install**. Sama je zaktualizuje do ich najnowszych wersji, polecemeniem **bower update.**

NodeJs

Jeżeli chcemy by nasza paczka która instalujemy znalazła się w package.json musimy dodać opcję:

npm install <package name> --save

npm install <package name> --save-dev

npm install <package name> --save-optional

Trzy opcje, trzy różna miejsca do których trafi nam paczka. Zaleca się by --save był wykorzystany wtedy kiedy paczki są niezbędne do uruchamiania środowiska/aplikacji (na przykład express.js lub jakiś inny framework), --save-dev wtedy kiedy paczka niezbędna jest do rozwijania i testowania aplikacji (na przykład unit testing framework), --save-optional jak sama nazwa wskazuje ;)

Dzięki wykorzystaniu tych opcji sami nie będziemy musieli tego wpisywać do pliku package.json – zaoszczędzi nam to czas. Nawet jeżeli nie chcemy kodu publikować, warto stworzyć sobie plik package.json i wykorzystywać opcje save w trakcie instalacji paczki, chociażby dlatego, że bardzo często folder w którym instalowane są paczki (node\_modules) ma zagłębienie większe niż 255 znaków, co skutecznie uniemożliwia przenoszenie folderu gdziekolwiek. Dobrą praktyką jest też nie wrzyucanie go na github :)

Dodatkowo, kiedy mamy te paczki w pliku możemy usunąć sobie katalog node\_modules, i wykonać polecenie:

npm install

które ściągnie nam wszystkie paczki na nowo :)

Bower

Bower jest menadżerem paczek klienckich i umożliwia nam pobieranie takich paczek jak jQuery, angular, itp.

By móc z niego korzystać musimy jednak go zainstalować:

npm install bower -g

bower init

stworzy nam plik bower.json w którym możemy opisać naszą paczkę, jeżeli będziemy chcieli ją publikować jak i da nam możliwość przechowywania zależności (od jakich paczek nasz projekt jest uzależniony) – nie jest to wymagane, ale jest przydatne.

bower install <package name>

zainstaluje nam package name paczkę w katalogu bower\_components (wyszukać paczkę możemy przez [stronę](http://bower.io/search/) jak i za pomocą komendy search). To co fajnego jest w bower a czego mi brakuje w nuget, to możliwość konfiguracji gdzie będą instalowane nasze zależności (nie wiem jak wy, ale ja nieznoszę katalogu Scripts – kod który piszę napewno do skrytpów się nie zalicza :)).

touch .bowerrc

notepad .bowerrc

i wpisujemy:

{

"directory": "sciezka/do/katalogu"

}

mami@NB-MMIOTK /cygdrive/d/development/git\_public/learning-bower/bower\_helloworld

$ cat bower.json

{

"name": "bower\_helloworld",

"description": "Eksperymentowanie",

"main": "mami",

"keywords": [

"nauka"

],

"authors": [

"Marcin Miotk"

],

"license": "MIT",

"homepage": "",

"ignore": [

"\*\*/.\*",

"node\_modules",

"bower\_components",

"test",

"tests"

]

}

mami@NB-MMIOTK /cygdrive/d/development/git\_public/learning-bower/bower\_helloworld

$ bower install jquery --save

bower jquery#\* not-cached https://github.com/jquery/jquery-dist.git#\*

bower jquery#\* resolve https://github.com/jquery/jquery-dist.git#\*

bower jquery#\* download https://github.com/jquery/jquery-dist/archive/3.2.1.tar.gz

bower jquery#\* extract archive.tar.gz

bower jquery#\* resolved https://github.com/jquery/jquery-dist.git#3.2.1

bower jquery#^3.2.1 install jquery#3.2.1

jquery#3.2.1 bower\_components\jquery

Podobnie jak w npm, jeżeli chcemy zaktualizować plik bower.json by zawierał on zależności to:

bower install <package name> --save

bower install <package name> --save-dev

Ponownie, --save wtedy kiedy jest to niezbędne do uruchomienia aplikacji, --save-dev jeżeli do jej rozwoju.

Jeżeli nasz plik bower.json już zawiera odpowiednie wpisy to możemy wykonać polecenie

bower install

tak jak w npm spowoduje to zainstalowanie wszystkich zależności z uwzględnieniem .bowerrc.

Aktualizacja paczek zainstalowanych też jest prosta:

bower update

Bower can manage components that contain HTML, CSS, JavaScript, fonts or even image files. Bower doesn’t concatenate or minify code or do anything else - it just installs the right versions of the packages you need and their dependencies.

A package can be a GitHub shorthand, a Git endpoint, a URL, and more. Read more about [bower install](https://bower.io/docs/api/#install).

*# installs the project dependencies listed in bower.json*

$ bower install

*# registered package*

$ bower install jquery

*# GitHub shorthand*

$ bower install desandro/masonry

*# Git endpoint*

$ bower install git://github.com/user/package.git

*# URL*

$ bower install http://example.com/script.js

How you use packages is up to you. We recommend you use Bower together with [Grunt, RequireJS, Yeoman, and lots of other tools](https://bower.io/docs/tools/) or build your own workflow with [the API](https://bower.io/docs/api/). You can also use the installed packages directly, like this, in the case of jquery:

<script src="bower\_components/jquery/dist/jquery.min.js"></script>

Similarly, using bower install <package> --save-dev will add <package> to your project’s bower.json devDependencies array.

*# install package and add it to bower.json devDependencies*

$ bower install <package> --save-dev

Registering your package allows others to install it with a short name, like bower install <my-package-name>.

To register a new package:

* The package name **must** adhere to the [bower.json spec](https://github.com/bower/spec/blob/master/json.md#name).
* There **must** be a valid bower.json in the project’s root directory.
* Your package should use [semver](http://semver.org/) Git tags. The v prefix is allowed.
* Your package **must** be publically available at a Git endpoint (e.g., GitHub). Remember to push your Git tags!
* For private packages (e.g. GitHub Enterprise), please consider running a private [Bower registry](https://github.com/bower/registry).

Then use [bower register](https://bower.io/docs/api#register):

$ bower register <my-package-name> <git-endpoint>

*# for example*

$ bower register example git://github.com/user/example.git

You can unregister packages with [bower unregister](https://bower.io/docs/api/#unregister). You first need to authenticate with GitHub with [bower login](https://bower.io/docs/api/#login) to confirm you are a contributor to the package repo.

bower login

*# enter username and password*

? Username:

? Password:

*# unregister packages after successful login*

bower unregister <package>

You’ll likely want to [bower cache clean](https://bower.io/docs/api#cache-clean) after your change.

Specyfikacja dla pliku Bower.JSON

<https://github.com/bower/spec/blob/master/json.md>

API do Bowera

<https://bower.io/docs/api/>

|  |  |
| --- | --- |
| Registered package name | jquery normalize.css |
| Git endpoint | https://github.com/user/package.git git@github.com:user/package.git |

|  |  |
| --- | --- |
| Shorthand (defaults to GitHub) | user/package |

**install options**

* -F, --force-latest: Force latest version on conflict
* -p, --production: Do not install project devDependencies
* -S, --save: Save installed packages into the project’s bower.json dependencies
* -D, --save-dev: Save installed packages into the project’s bower.json devDependencies
* -E, --save-exact: Configure installed packages with an exact version rather than semver

### search

$ bower search

$ bower search <name>

Finds all packages or a specific package.

### json

-j, --json

Output consumable JSON

bower list --paths

*# or*

$ bower list --paths --json

{

"backbone": "bower\_components/backbone/backbone.js",

"jquery": "bower\_components/jquery/dist/jquery.js",

"underscore": "bower\_components/underscore/underscore.js"

}

## Running on a continuous integration server

Bower will skip some interactive operations if it finds a CI environmental variable set to true. You will find that the CI variable is already set for you on many continuous integration servers, e.g., [CircleCI](https://circleci.com/docs/environment-variables#basics) and [Travis-CI](http://docs.travis-ci.com/user/ci-environment/#Environment-variables).

You may try to set the CI variable manually before running your Bower commands. On Mac or Linux, export CI=true and on Windows set CI=true

If for some reason you are unable to set the CI environment variable, you can alternately use the --config.interactive=false flag.

$ bower install --config.interactive=false