Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'dependency-selectors.7' command

$ man dependency-selectors.7

DEPENDENCY() DEPENDENCY()

NAME

Dependency

Description

The npm help query commmand exposes a new dependency selector syntax (informed by & respecting many aspects of the CSS Selectors 4 Spec https://dev.w3.org/csswg/selectors4/#relational) which:

? Standardizes the shape of, & querying of, dependency graphs with a robust object model, metadata & selector syntax

? Leverages existing, known language syntax & operators from CSS to make disparate package information broadly accessible

? Unlocks the ability to answer complex, multi-faceted questions about dependencies, their relationships & associative metadata

? Consolidates redundant logic of similar query commands in npm (ex. npm fund, npm ls, npm outdated, npm audit ...)

Dependency Selector Syntax v1.0.0

Overview:

? there is no "type" or "tag" selectors (ex. div, h1, a) as a depen? dency/target is the only type of Node that can be queried

? the term "dependencies" is in reference to any Node found in a tree returned by Arborist

Combinators

? > direct descendant/child

? any descendant/child

? ~ sibling

Selectors

? * universal selector

? #<name> dependency selector (equivalent to [name="..."])

? #<name>@<version> (equivalent to [name=<name>]:semver(<version>))

? , selector list delimiter

? . dependency type selector

? : pseudo selector

Dependency Type Selectors

? .prod dependency found in the dependencies section of package.json, or is a child of said dependency

? .dev dependency found in the devDependencies section of package.json, or is a child of said dependency

? .optional dependency found in the optionalDependencies section of package.json, or has "optional": true set in its entry in the peerDependenciesMeta section of package.json, or a child of said dependency

? .peer dependency found in the peerDependencies section of package.json

? .workspace dependency found in the workspaces https://docs.npmjs.com/cli/v8/using-npm/workspaces section of package.json

? .bundled dependency found in the bundleDependencies section of package.json, or is a child of said dependency

Pseudo Selectors

? :not(<selector>) https://developer.mozilla.org/en-US/docs/Web/CSS/:not

? :has(<selector>) https://developer.mozilla.org/en-US/docs/Web/CSS/:has

? :is(<selector list>) https://developer.mozilla.org/en-US/docs/Web/CSS/:is

? :root https://developer.mozilla.org/en-US/docs/Web/CSS/:root matches

the root node/dependency

? :scope https://developer.mozilla.org/en-US/docs/Web/CSS/:scope matches node/dependency it was queried against

? :empty https://developer.mozilla.org/en-US/docs/Web/CSS/:empty when a dependency has no dependencies

? :private https://docs.npmjs.com/cli/v8/configuring-npm/package-json#private when a dependency is private

? :link when a dependency is linked (for instance, workspaces or manually linked https://docs.npmjs.com/cli/v8/commands/npm-link packages)

? :deduped when a dependency has been deduped (note that this does not always mean the dependency has been hoisted to the root of node_modules)

? :overridden when a dependency has been overridden

? :extraneous when a dependency exists but is not defined as a dependency of any node

? :invalid when a dependency version is out of its ancestors specified range

? :missing when a dependency is not found on disk

? :semver(<spec>) matching a valid node-semver https://github.com/npm/node-semver spec

? :path(<path>) glob https://www.npmjs.com/package/glob matching based on dependencies path relative to the project

? :type(<type>) based on currently recognized types https://github.com/npm/npm-package-arg#result-object

Attribute Selectors https://developer.mozilla.org/en-US/docs/Web/CSS/Attribute_selectors

The attribute selector evaluates the key/value pairs in package.json if they are Strings.

? [] attribute selector (ie. existence of attribute)

? [attribute=value] attribute value is equivalant...

? [attribute~=value] attribute value contains word...

? [attribute*=value] attribute value contains string...

? [attribute|=value] attribute value is equal to or starts with...

? [attribute^=value] attribute value starts with...

? [attribute$=value] attribute value ends with...

Array & Object Attribute Selectors

The generic :attr() pseudo selector standardizes a pattern which can be used for attribute selection of Objects, Arrays or Arrays of Objects accessible via Arborist's Node.package metadata. This allows for itera? tive attribute selection beyond top-level String evaluation. The last argument passed to :attr() must be an attribute selector or a nested :attr(). See examples below:

Objects

```
/* return dependencies that have a `scripts.test` containing `"tap"` */
*:attr(scripts, [test~=tap])
```

Nested Objects

Nested objects are expressed as sequential arguments to :attr().

```
/* return dependencies that have a testling config for opera browsers */
*:attr(testling, browsers, [~=opera])
```

Arrays

Arrays specifically uses a special/reserved . character in place of a typical attribute name. Arrays also support exact value matching when a String is passed to the selector.

Example of an Array Attribute Selection:

```
/* removes the distinction between properties & arrays */
/* ie. we'd have to check the property & iterate to match selection */
*:attr([keywords^=react])
*:attr(contributors, :attr([name~=Jordan]))
```

Example of an Array matching directly to a value:

```
/* return dependencies that have the exact keyword "react" */
/* this is equivalent to `*:keywords([value="react"])` */
*:attr([keywords=react])
```

Example of an Array of Objects:

```
/* returns */
*:attr(contributors, [email=ruyadorno@github.com])
```

Groups

Dependency groups are defined by the package relationships to their an?
cestors (ie. the dependency types that are defined in package.json).
This approach is user-centric as the ecosystem has been taught to think
about dependencies in these groups first-and-foremost. Dependencies are
allowed to be included in multiple groups (ex. a prod dependency may
also be a dev dependency (in that it's also required by another dev de?
pendency) & may also be bundled - a selector for that type of depen?
dency would look like: *.prod.dev.bundled).

? .prod

? .dev

? .optional

? .peer

? .bundled

? .workspace

Please note that currently workspace deps are always prod dependencies.
Additionally the .root dependency is also considered a prod dependency.

Programmatic Usage

? Arborist's Node Class has a .querySelectorAll() method

? this method will return a filtered, flattened dependency Arborist
Node list based on a valid query selector

```
const Arborist = require('@npmcli/arborist')
const arb = new Arborist({})
// root-level
arb.loadActual().then(async (tree) => {
  // query all production dependencies
  const results = await tree.querySelectorAll('.prod')
  console.log(results)
})
// iterative
arb.loadActual().then(async (tree) => {
  // query for the deduped version of react
  const results = await tree.querySelectorAll('#react:not(:deduped)')
  // query the deduped react for git deps
```

```
  const deps = await results[0].querySelectorAll(':type(git)')
  console.log(deps)
})
```

See Also

? npm help query

? @npmcli/arborist https://npm.im/@npmcli/arborist