NAME

Querying - Dependency Selector Syntax & Querying

Description

The npm help query command 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 **npm fund**, **npm ls**, **npm outdated**, **npm audit** ...)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

Overview:

- there is no "type" or "tag" selectors (ex. **div, h1, a**) as a dependency/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 **\fBworkspaces\fR** ⟨*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

- **\fB:not(<selector>)\fR** ⟨*https://developer.mozilla.org/en-US/docs/Web/CSS/:not*⟩
- **\fB:has(<selector>)\fR** ⟨*https://developer.mozilla.org/en-US/docs/Web/CSS/:has*⟩
- **\fB:is(<selector list>)\fR** ⟨*https://developer.mozilla.org/en-US/docs/Web/CSS/:is*⟩
- **\fB:root\fR** ⟨*https://developer.mozilla.org/en-US/docs/Web/CSS/:root*⟩ matches the root node/dependency
- **\fB:scope\fR** ⟨*https://developer.mozilla.org/en-US/docs/Web/CSS/:scope*⟩ matches node/dependency it was queried against
- **\fB:empty\fR** ⟨*https://developer.mozilla.org/en-US/docs/Web/CSS/:empty*⟩ when a dependency has no dependencies
- **\fB:private\fR** ⟨*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 packages manually **\fBlinked\fR** ⟨*https://docs.npmjs.com/cli/v8/commands/npm-link*⟩
- **: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>, [selector], [function])** match a valid **\fBnode-semver\fR** ⟨*https://github.com/npm/node-semver*⟩ version or range to a selector
- **: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*⟩
- **:outdated(<type>)** when a dependency is outdated
- **:vuln(<selector>)** when a dependency has a known vulnerability

:semver(<spec>, [selector], [function])

The **:semver()** pseudo selector allows comparing fields from each node's **package.json** using **semver** ⟨*https://github.com/npm/node-semver#readme*⟩ methods. It accepts up to 3 parameters, all but the first of which are optional.

- **spec** a semver version or range
- **selector** an attribute selector for each node (default **[version]**)
- **function** a semver method to apply, one of: **satisfies**, **intersects**, **subset**, **gt**, **gte**, **gtr**, **lt**, **lte**, **ltr**, **eq**, **neq** or the special function **infer** (default **infer**)

When the special **infer** function is used the **spec** and the actual value from the node are compared. If both are versions, according to **semver.valid()**, **eq** is used. If both values are ranges, according to **!semver.valid()**, **intersects** is used. If the values are mixed types **satisfies** is used.

Some examples:

- **:semver(^1.0.0)** returns every node that has a **version** satisfied by the provided range **^1.0.0**
- **:semver(16.0.0, :attr(engines, [node]))** returns every node which has an **engines.node** property satisfying the version **16.0.0**
- **:semver(1.0.0, [version], lt)** every node with a **version** less than **1.0.0**

:outdated(<type>)

The **:outdated** pseudo selector retrieves data from the registry and returns information about which of your dependencies are outdated. The type parameter may be one of the following:

- **any** (default) a version exists that is greater than the current one
- **in-range** a version exists that is greater than the current one, and satisfies at least one if its parent's dependencies
- **out-of-range** a version exists that is greater than the current one, does not satisfy at least one of its parent's dependencies
- **major** a version exists that is a semver major greater than the current one
- **minor** a version exists that is a semver minor greater than the current one
- **patch** a version exists that is a semver patch greater than the current one

In addition to the filtering performed by the pseudo selector, some extra data is added to the resulting objects. The following data can be found under the **queryContext** property of each node.

- **versions** an array of every available version of the given node
- **outdated.inRange** an array of objects, each with a **from** and **versions**, where **from** is the on-disk location of the node that depends on the current node and **versions** is an array of all available versions that satisfies that dependency. This is only populated if **:outdated(in-range)** is used.
- **outdated.outOfRange** an array of objects, identical in shape to **inRange**, but where the **versions** array is every available version that does not satisfy the dependency. This is only populated if **:outdated(out-of-range)** is used.

Some examples:

- **:root > :outdated(major)** returns every direct dependency that has a new semver major release
- **.prod:outdated(in-range)** returns production dependencies that have a new release that satisfies at least one of its parent's dependencies

:vuln

The **:vuln** pseudo selector retrieves data from the registry and returns information about which if your dependencies has a known vulnerability. Only dependencies whose current version matches a vulnerability will be returned. For example if you have **semver@7.6.0** in your tree, a vulnerability for **semver** which affects versions **<=6.3.1** will not match.

You can also filter results by certain attributes in advisories. Currently that includes **severity** and **cwe**. Note that severity filtering is done per severity, it does not include severities "higher" or "lower" than the one specified.

In addition to the filtering performed by the pseudo selector, info about each relevant advisory will be added to the **queryContext** attribute of each node under the **advisories** attribute.

Some examples:

- **:root > .prod:vuln** returns direct production dependencies with any known vulnerability

- **:vuln([severity=high])** returns only dependencies with a vulnerability with a **high** severity.
- **:vuln([severity=high],[severity=moderate])** returns only dependencies with a vulnerability with a **high** or **moderate** severity.
- **:vuln([cwe=1333])** returns only dependencies with a vulnerability that includes CWE-1333 (ReDoS)

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 **String**s.

- **[]** attribute selector (ie. existence of attribute)
- **[attribute=value]** attribute value is equivalent...
- **[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 **Object**s, **Array**s or **Arrays** of **Object**s accessible via **Arborist**'s **Node.package** metadata. This allows for iterative 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](https://ci.testling.com/guide/advanced_configuration) 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 **Object**s:

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

Groups

Dependency groups are defined by the package relationships to their ancestors (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**

dependency) & may also be **bundled** - a selector for that type of dependency 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*⟩