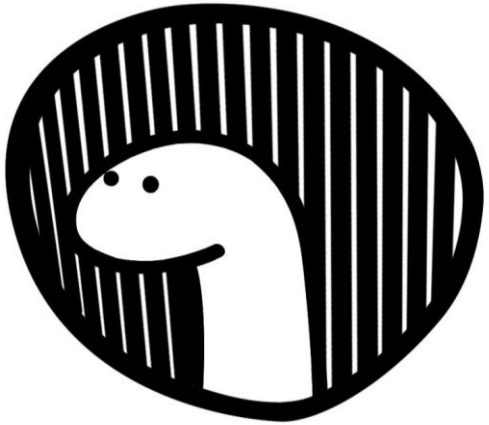
Denoにおける外部モジュールのバージョン指定についての分析

実証的ソフトウェア工学研究室

20T2001K　相沢華愛

Denoとは



Deno

- JavaScript/TypeScriptのランタイム環境
- Node.jsにおける反省を元に2018年に作られた

Deno

import文にURLを与えることで
外部モジュールを使用可能

Node.js

npmを使ってモジュールを
インストールする必要がある

バージョンを指定しないと

常に最新バージョンがインポートされる

例1)
import moduleA
→最新バージョンの**v3.0.0**がimportされる

例2)
import moduleA**@v2.0.0**
→**v2.0.0**がimportされる

v3.0.0がバグを抱えていた → 例1ではエラーが発生

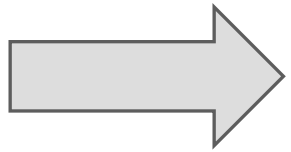
import時にエラーが発生した事例

Denoのstd/nodeというモジュールが削除
Deno自身の機能として統合された

その際、いくつかのモジュールが壊れてしまった

参照先が存在しないためエラー

https://deno.land/std/node/net.ts



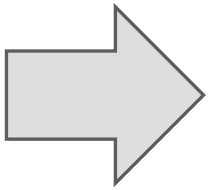
バージョンを指定することでエラーが起きない

https://deno.land/std/node/@0.177.0/net.ts

バージョンを指定することで解決

目的

・比較的新しい環境
・新しいimport方法
・バージョンが壊れた事例



バージョン管理の現状が不明

Denoのバージョン管理の現状を分析

Denoのimport方法

- 内部（自身の）モジュール（相対パス）

```
import { Context } from "./mod.ts";
```

✓ ●外部モジュール（バージョン指定なし　絶対パス）

```
import { Application } from "https://deno.land/x/oak/mod.ts";
```

✓ ●外部モジュール（バージョン指定あり　絶対パス）

```
import { send } from "https://deno.land/x/oak@v12.5.0/mod.ts";
```

レジストリについて

●レジストリ：モジュールを配信しているサーバー

例1）https://deno.land/x/oak/mod.ts

→　レジストリ名：deno.land/x

例2）https://esm.sh/jszip

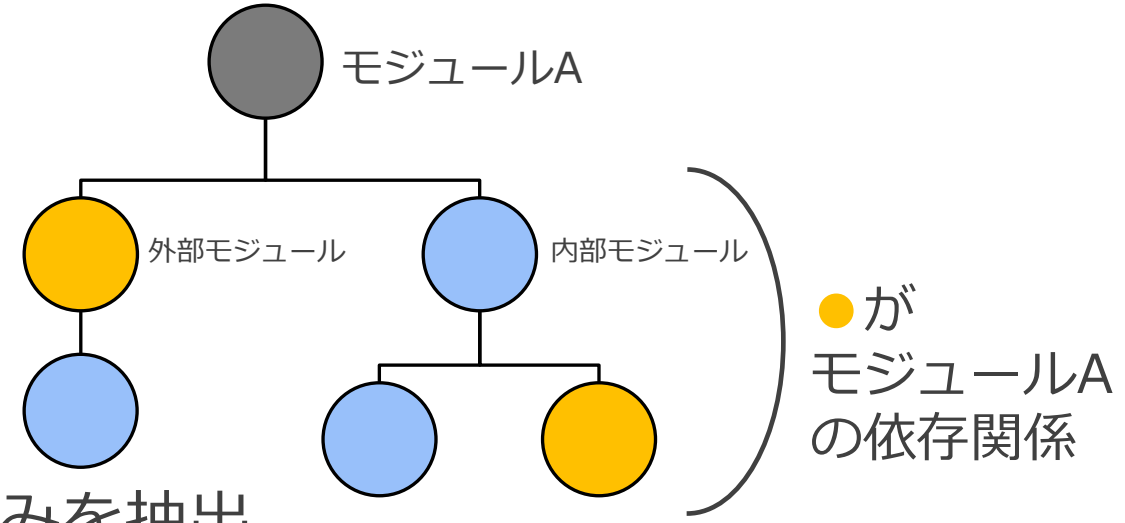
→　レジストリ名：esm.sh

取得するデータ

対象：2023年7月12日時点でdeno.landに存在する6,616件のモジュール

① 対象モジュールを起点に呼び出される
すべてのモジュールURLを取得

② ①の中から外部モジュール
(絶対パスでimportされているURL)のみを抽出
＝ **対象モジュールの依存関係**



依存関係について

③　依存関係にあるモジュールからレジストリ名、バージョンを取得

対象モジュール：https://deno.land/x/oak@v12.5.0/mod.tsの場合

https://deno.land/x/oak@v12.5.0/mod.ts

- https://deno.land/std@0.188.0/bytes/mod.ts
  - レジストリ：deno.land/std
  - バージョン：0.188.0
- https://deno.land/x/path_to_regexp@v6.2.1/index.ts
  - レジストリ：deno.land/x
  - バージョン：6.2.1
  - ⋮

Research Questions

RQ1：モジュールごとのバージョン指定率に特徴はあるか

RQ2：依存関係の数とバージョン指定率に関係性はあるか

RQ3：バージョンが指定されていないモジュール(not_pinned)
についてどのような特徴があるか

RQ1 モジュールごとのバージョン指定率

●依存関係にあるモジュールのバージョンと最新バージョンの比較結果

✓ ・new：最新バージョンを指定している
✓ ・old：最新バージョンと比較して古いバージョンを指定している
・not_pinned：バージョンの指定をしていない
・error：import時にエラーが発生する
・deprecated：importは出来るが、最新バージョンが消滅している
・hash_pinned：バージョン情報をハッシュ値で指定している

呼び出し関係に外部モジュールを含むモジュールは6,616件のうち2,645件

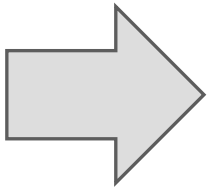
RQ1 モジュールごとのバージョン指定率

- モジュールごとのバージョン指定率について分析

$$\text{適切なバージョン指定率} = \frac{\text{newとoldの合計}}{\text{依存関係の数}} \times 100 \qquad \text{最新バージョン指定率} = \frac{\text{newの数}}{\text{依存関係の数}} \times 100$$

例) moduleAには依存関係が10個存在

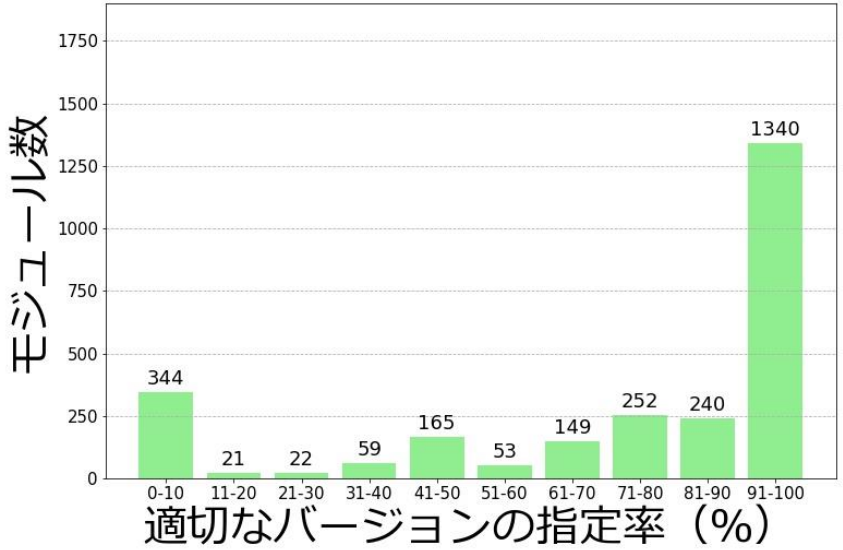
[new, new, new, new,
old, old, old,
deprecated, hash_pinned, not_pinned]

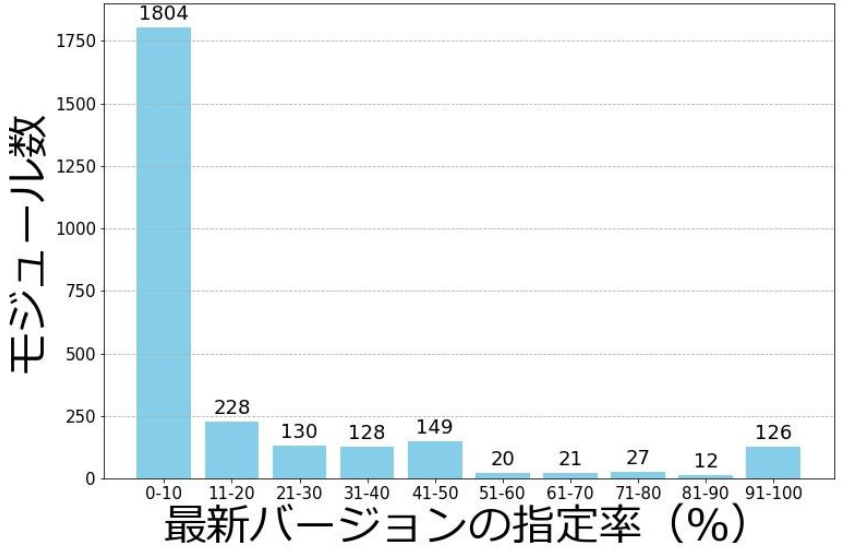


バージョン指定率：
(4 + 3) / 10 × 100 = 70%

最新バージョン指定率：
4 / 10 × 100 = 40%

RQ1 モジュールごとのバージョン指定率





- 半分以上のモジュールでバージョン指定率は90%~100%
- 一部バージョンの指定率が低いモジュールが存在する
- 最新バージョンの指定率は低い

RQ2 依存関係の数とバージョン指定率

- 依存関係の数を用いて分析

  - 依存関係の数に応じてモジュールを分類

    ・・・0~5個、6~10個、11~20個、21~50個、51~100個、101個~

  - 依存関係にあるモジュールのバージョンの比較結果を取得

  - 依存関係の数とバージョンの比較結果から表を作成

RQ2 依存関係の数とバージョン指定率

依存関係の数(個)	該当するモジュール(個)	合計した依存関係の数(個)	new(%)	old(%)	new+old(%)	not_pinned(%)	error(%)	deprecated(%)	hash_pinned(%)
0~5	**1,601**	3,610	13.9	58.8	**72.7**	**11.1**	2.2	13.6	0.3
6~10	398	3,049	14.5	63.6	78.1	8.2	1.1	12.4	0.3
11~20	365	5,356	15.6	66.6	82.2	5.4	0.6	11.6	0.3
21~50	226	6,963	17.5	62.4	79.9	6.1	1.6	12.1	0.3
51~100	45	3,088	23.8	54.9	78.7	6.3	5.2	9.6	0.3
101~	10	6,574	4.9	92.6	97.5	1.0	0.2	1.2	0.0

・全体としてバージョンの指定率は高い

・依存関係の数が5個以下のモジュールが半分以上
バージョン指定率が他と比べて低い

RQ3 バージョンが指定されていないモジュール

- 依存関係にあるモジュールをレジストリで分類
- レジストリごとにnot_pinnedの割合を分析

not_pinnedを依存関係に含むモジュールは2,645件のうち617件
また、すべての依存関係28,640件のうちnot_pinnedは1,624件

RQ3 バージョンが指定されていないモジュール

レジストリ名	分類
deno.land/x	deno公式のモジュールレジストリ
deno.land/std	
github.com	GitHub関連のレジストリ
denopkg.com	
ghuc.cc	
esm.sh	npmパッケージを配布するレジストリ
jspm.dev	
unpkg.com	
skypack.dev	
pika.dev	
jsdelivr.net	CDNパッケージを配布するレジストリ
nest.land	ブロックチェーン上のパッケージレジストリ
other	その他レジストリ

RQ3 バージョンが指定されていないモジュール

レジストリ名	importされた回数(回)	not_pinnedの回数(回)	not_pinnedの割合(%)
deno.land/x	11,872	561	**4.7**
deno.land/std	10,818	468	**4.3**
github.com	372	119	**32.0**
denopkg.com	223	22	9.9
ghuc.cc	120	120	**100**
esm.sh	3,740	155	4.1
jspm.dev	225	89	39.6
unpkg.com	101	3	3.0
skypack.dev	721	74	10.3
pika.dev	206	0	0.0
nest.land	93	0	0.0
jsdelivr.net	62	5	8.1
other	87	8	9.2

- deno公式レジストリでは回数自体は多いがnot_pinnedの割合は低い
- GitHub関連のレジストリではnot_pinnedの割合が高い

まとめ

- バージョンの指定率自体は高いが、最新バージョンの指定率は低い
  →依存関係のモジュールのバージョンを管理していない可能性

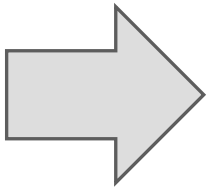
- 依存関係の数はバージョン指定率に大きくは影響しないものの
  依存関係が少ないモジュールではバージョン指定率が低い傾向があった

- レジストリによってnot_pinnedの割合は大きく異なる
  GitHub関連のレジストリでは高い傾向があった

今後の展望

・モジュールの作られた時期や制作者によって
　バージョンの指定率に偏りがあるのか

・利用者が多いモジュールほどバージョンが管理されているのか



バージョン管理の現状をより正確に把握
適切なバージョン管理に繋げる