go modules

Gilvan Ritter
github.com/gritt

Consultant Developer - ThoughtWorks

histórico

`go get` baixa dependências sem conhecer versões, não consegue encontrar a melhor versão de uma dependência mútua entre dois pacotes

`go mod` - go modules é o gerenciador oficial de dependências, disponível a partir da v1.11.x

características

- **retrocompatibilidade**
  - "...a especificação do Go v1, vai continuar compilando e executando corretamente no futuro."

características

- imports **explicitos**
  - `import "github.com/pborman/uuid"`

características

- utilizar a última versão de uma biblioteca não é uma "feature", é uma **escolha** (.lock files)
- o algorítmo do go modules é conversador nesse sentido, e utiliza a **versão mínima que satisfaz as dependências** ao invés da latest

características

- o **versionamento semântico** propõe que quebras de compatibilidade são aceitáveis incrementando a versão major. consequentemente você acaba precisando conhecer histórico de uma dependência, para saber qual versão tem a implementação que você precisa
- **semantic import versioning**

características

- **semantic import versioning**
- "se um pacote antigo e um pacote novo tem o **mesmo path,** então o pacote novo **deve ser retrocompatível**"
- equivalente a versões **minor**

características

- e versões **major?**
- **sufixo** identificando a versão
    - `import "github.com/pborman/uuid/v2"`

características

- trata versões major como se fossem pacotes **diferentes** e **independentes**
- **elimina** a necessidade de uma versão **best fit**
- possibilita utilizar **duas versões major do mesmo pacote**, por exemplo, fazer uma migração gradual de SDK v1 -> SDK v2

utilizando go modules

- `go mod init`
- `go mod tidy`
- `import`

publicando um go module

- `go.mod:`

```
module example.com/hello

go 1.12

require rsc.io/quote/v3 v3.1.0
```

- `git tag v1.0.0 && git push origin v1.0.0`

publicando um go module >v2

- a estratégia recomendada é criar um novo diretório `/v2` no seu projeto, que será o **sufixo** do pacote **indicando a versão major**
- desenvolver `v2` de maneira independente da `v1`

publicando um go module >v2

- `go.mod:`

```
/go.mod    → gisthub.com/googleapis/gax-go

/v2/go.mod → github.com/googleapis/gax-go/v2
```

- `git tag v2.0.0 && git push origin v2.0.0`

prefira built-in

- pondere o uso de dependências
- revise o source
  - testes
  - mantainers
  - stars
  - qualidade (*goreport*)
- auditoria de dependências (*gosec*)

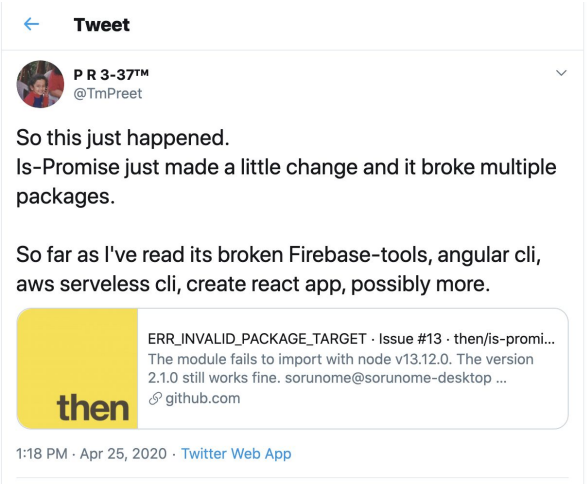
engraçado mas trágico

Another one-line npm package breaks the JavaScript ecosystem

An update to tiny "is-promise" library impacted millions of JavaScript projects.

6 lines (5 sloc) | 201 Bytes

```
1  module.exports = isPromise;
2  module.exports.default = isPromise;
3
4  function isPromise(obj) {
5    return !!obj && (typeof obj === 'object' || typeof obj === 'function') && typeof obj.then === 'function';
6  }
```



referências

- https://blog.golang.org/versioning-proposal
- https://blog.golang.org/using-go-modules
- https://blog.golang.org/publishing-go-modules
- https://blog.golang.org/v2-go-modules
- https://www.youtube.com/watch?v=F8nrpe0XWRg