15

Faça como eu fiz: publicação no NPM

Agora que a **versão 1.0.0** da lib está fechada, você pode publicá-la, se quiser.

É possível publicar no NPM para que a biblioteca possa ser instalada com `npm install -g <nome-lib>`.

Preparando os arquivos da lib

Antes de fazer a publicação, temos que adicionar o ambiente para execução da lib.

Na linha 1 do arquivo `cli.js`, antes de tudo (inclusive das importações), adicione o seguinte código:

```
#!/usr/bin/env node
```

COPIAR CÓDIGO

Ela aparecerá no código como comentário, o que é normal.

Esta linha, que sempre inicia com os caracteres `#!`, é o que chamamos de **shebang**. Quando adicionamos esta linha a um arquivo que será executado através do terminal, estamos instruindo o sistema operacional sobre qual é o tipo de arquivo que será executado e de qual forma (no caso, estamos falando do Node.js).

O **shebang** só é interpretado por sistemas do tipo Unix, como distribuições Linux e MacOs. Sistemas Windows ignoram este tipo de comando, mas mesmo assim ele é necessário para que a instalação da lib e dos arquivos necessários do Node.js seja feita de forma correta.

> Você pode também acrescentar um arquivo `README.md` na raiz do repositório e na branch `main` de seu projeto no GitHub. Este arquivo serve para informar aos usuários do que se trata o projeto e como utilizá-lo.

Publicando no NPM

1) Crie uma conta no NPM (https://www.npmjs.com/);

2) Faça o login através da linha de comando do terminal com `npm add-user`. Você precisará do seu nome de user registrado no NPM e da senha que você utilizou para criar a conta (passo 1). Se der tudo certo, você verá no terminal a mensagem `Logged in as <seu nome de user> on https://registry.npmjs.org/.`

3) Você precisará atualizar o `package.json` com todas as informações necessárias:

`name`: Você precisará de um nome único para identificar sua lib no repositório do NPM. A recomendação é que seja um nome curto (se possível) e fácil de lembrar que não tenha `js` no nome, uma vez que isso está implícito - já que estamos utilizando o Node Package Manager. Você pode verificar quais nomes já estão em uso pesquisando no próprio site do NPM (https://www.npmjs.com/);

`version`: Para a primeira publicação, a versão deve ser `1.0.0`. Para as próximas - caso você queira fazer atualizações - veja mais instruções abaixo;

`description`: String com descrição da funcionalidade. Serve para ajudar outras pessoas que buscam por libs no NPM;

`keywords`: um array de strings com palavras-chave relativas à lib, também para facilitar a busca. Por exemplo: `['markdown', 'URL']`;

`homepage` : String com a URL do projeto no GitHub, ou outra página que você quiser informar (sua página pessoal de portfólio, por exemplo);

`bugs` : Um objeto com o contato para usuários reportarem bugs. Deve seguir o formato:

```
"bugs": {
  "url" : "https://github.com/<seu-user>/<nome-projeto>/issues
  "email" : "email@seu-email.com"
},
```

COPIAR CÓDIGO

Caso queira informar somente ou a `url` ou o `email` , não é preciso o objeto, apenas uma string: `"bugs": "string com o e-mail ou link do GitHub"` ,

`license` : Permissões e restrições de uso que você queira aplicar à sua lib. Você pode consultar a lista (https://spdx.org/licenses/) ou deixar a padrão, `ISC` .

`author` : String com suas informações, conforme o modelo: `"Seu Nome <email@sem-email.com> (http://seusite.com/)"` . O email e o site são opcionais;

`main` : String com o caminho do **ponto de entrada** da sua aplicação. No caso, será `./cli.js` .

`bin` : Este campo não aparece por padrão no `package.json` , você terá que criá-lo. É um objeto no qual você deve definir como chave o termo que será usado para executar sua lib pela linha de comando, e como valor o caminho para o *entrypoint*, ou seja, o arquivo que é o ponto de entrada da aplicação - no caso, o `./cli.js` . Por exemplo:

```
"bin": {
   "<nome-da-lib>": "./cli.js"
```

```
  },
```

COPIAR CÓDIGO

Seguindo o exemplo acima, para a execução da lib na linha de comando o usuário deverá inserir o comando `nome-da-lib ./caminho/do/arquivo.md`.

Em vez de `nome-da-lib` você pode utilizar outro termo, porém é padrão que seja o mesmo nome da lib para evitar confusões e ficar mais fácil de lembrar.

`"preferGlobal": true,` : Booleano com o valor `true`. Uma vez que o propósito desta lib é ser executada a partir do terminal e consultar listas de arquivos que podem estar em qualquer diretório, faz mais sentido que ela seja instalada globalmente por padrão. Desta forma, é possível executá-la a partir de qualquer diretório, não apenas no diretório onde ela foi instalada.

Existem vários outros campos com todas as informações que podem ou não ser pertinentes a um projeto em Node.js. Você pode consultar a lista completa aqui (https://docs.npmjs.com/cli/v7/configuring-npm/package-json), mas com os campos acima já é possível publicar sua lib.

O `package.json` para publicação ficará semelhante ao abaixo:

```
{
 "name": "nome-lib",
 "version": "1.0.0",
 "description": "sua descrição",
 "keywords": ["markdown", "URL"],
 "main": "./cli.js",
 "bin": {
   "<nome_da_lib>": "./cli.js"
 },
 "preferGlobal": true,
 "scripts": {
```

```
    "test": "jest ./test",
    "cli": "node ./src/cli.js"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/<seu-user>/<seu-repo>.git"
  },
  "author": "Juliana Amoasei",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/<seu-user>/<seu -repo>/issues"
  },
  "homepage": "https://github.com/<seu -user>/<seu-repo>#readme"
  "dependencies": {
    "chalk": "^4.1.2",
    "node-fetch": "^2.6.1"
  },
  "devDependencies": {
    "jest": "^27.0.6"
  }
}
```

COPIAR CÓDIGO

4) Publique sua lib com o comando `npm publish`;

5) Confira no site do NPM! `https://www.npmjs.com/package/<nome-da-lib>`;

6) Instale em seu sistema com o comando `npm install -g <nome-da-lib>` (veja que a instalação está sendo feita localmente com a flag `-g`);

7) Teste se está tudo Ok executando a lib com o comando `nome-da-lib caminho/do/arquivo.md`.

> Lembre-se que o caminho do arquivo pode ser **relativo** ao diretório de onde você está executando a lib, ou **absoluto**.

Novas versões

O NPM trabalha com o conceito de versionamento semântico. Você já deve ter visto que as sequências numéricas das versões de todas as dependências que usamos (inclusive o próprio Node.js e o NPM) seguem o mesmo padrão de três números separados por pontos, por exemplo `1.0.0` (a versão inicial que o `package.json` cria) ou `14.17.5` (última versão recomendada do Node.js no momento em que este texto foi escrito). O que significa cada número desta sequência?

O versionamento semântico utiliza os seguintes critérios:

O primeiro número da sequência (o `14` em `14.17.5`) se refere a **breaking changes**, ou seja, atualizações de versão que têm potencial para "quebrar" códigos que utilizem as versões anteriores. Ou seja, uma aplicação que utiliza códigos (métodos, funções, etc) da versão `13.X.X` de determinada lib pode deixar de funcionar com a versão `14.X.X`, pois haverá diferenças significativas entre as versões. Estas atualizações são conhecidas como `major`.

O número do meio (o `17` em `14.17.5`) se refere a novas funcionalidades adicionadas, mas que não causam "quebra" em relação a códigos das versões anteriores. Este tipo de atualização é conhecida como `minor`.

O último número (o `5` em `14.17.5`) se refere a correção de código: resolução de bugs, melhoramento de performance ou alterações similares que não alteram as funcionalidades atuais (ou exceção da correção de bugs) e nem introduzem novas. É conhecida como `patch`.

Atualizando a lib no NPM

Se você fizer alterações no código da sua lib após o curso e quiser atualizar a versão no NPM, verifique de acordo com a lista acima qual é o tipo de atualização e rode o comando `npm version <patch | minor | major>`, escolhendo apenas uma das três opções de acordo com o tipo de atualização. Por exemplo, caso você esteja atualmente na versão `1.0.0`, após rodar `npm version patch` você deverá receber no terminal a mensagem `v.1.0.1` e o valor também já estará alterado no `package.json`.

Após este processo, você pode atualizar o repositório do NPM com o comando `npm publish`.

> Atenção: o repositório da sua lib no GitHub precisa estar atualizado para o comando `npm version` funcionar.

Se quiser saber mais sobre versionamento semântico, você pode consultar a documentação aqui (https://semver.org/lang/pt-BR/).

Este é um guia básico para a primeira publicação, mas você sempre pode consultar a documentação sobre publicação do NPM (https://docs.npmjs.com/creating-and-publishing-scoped-public-packages) para ver todas as opções.