

# Nodejs 工具开发入门

lingxufeng@bytedance.com

# 一些原理

# node工具

- 比如webpack/eslint/mocha
- 基于 Node 引擎运行, 通过 Command Line Interface 交互的工具
- 抽象自业务, 较低的使用成本
- 用工具造轮子, 而不是轮子的堆砌

# 了解node工具需要的技能储备

- OS常识
- 命令行常识
- 文件系统常识
- 网络通信常识

# shell script 介绍

- 什么是 shell

一种可操控操作系统的程序，将用户的输入与 kernel 沟通，控制硬件工作。

- 什么是 bash、zsh、iterm

bash 是加强 shell，遵循标准、操作系统默认；zsh 兼容 bash

- 什么是 shell script

一个程序，纯文本文件，可利用 shell 与相关工具指令来进行一些操作，无需编译。

《linux 私房菜》10.1、12.1

# 一个示例(mvp.sh)

mvp.sh x

```
#!/bin/bash
```

非常重要!

文件处理

```
i=0
rm -rf ~/vote
mkdir ~/vote
cd ~/vote
```

```
while [ "${i}" != "50" ]
```

```
do
```

```
    i=$((i+1))
```

```
    curl "https://bbs.hupu.com/vote-${i}" -O
```

```
done
```

一些编程逻辑

```
cat ~/vote/vote-* | grep '<a id="" href="' | cut -d '>' -f 2 | cut -d '<' -f 1 | tr -s '\n' | grep 'mvp' | grep '威' | tee weishao.txt | wc -l
cat ~/vote/vote-* | grep '<a id="" href="' | cut -d '>' -f 2 | cut -d '<' -f 1 | tr -s '\n' | grep 'mvp' | grep '哈登' | tee hadeng.txt | wc -l
```

连续指令单一化

简单的数据处理

# #! /bin/bash 的作用

#不是注释！#! 读作 hashbang, 后接的是解释器的查找路径。

- 宣告这个文件的语法使用 bash 语法
- 运行时能够加载bash 的相关环境配置文件
- 没有的话, 操作系统得去猜测使用什么 bash

# 执行 shell

- 执行方式

- 指令直接下达
  - 文件必须可以执行(rx)
  - 相对路径、绝对路径
  - 将 shell 让道 path 指定的目录内部, 比如 ~/bin 文件夹
- 用 bash 程序执行
  - sh test.sh

- 执行方式的差异

直接执行会在子程序的 bash 中执行, 变量无法回传, exit 值可以; source 会在父程序中执行

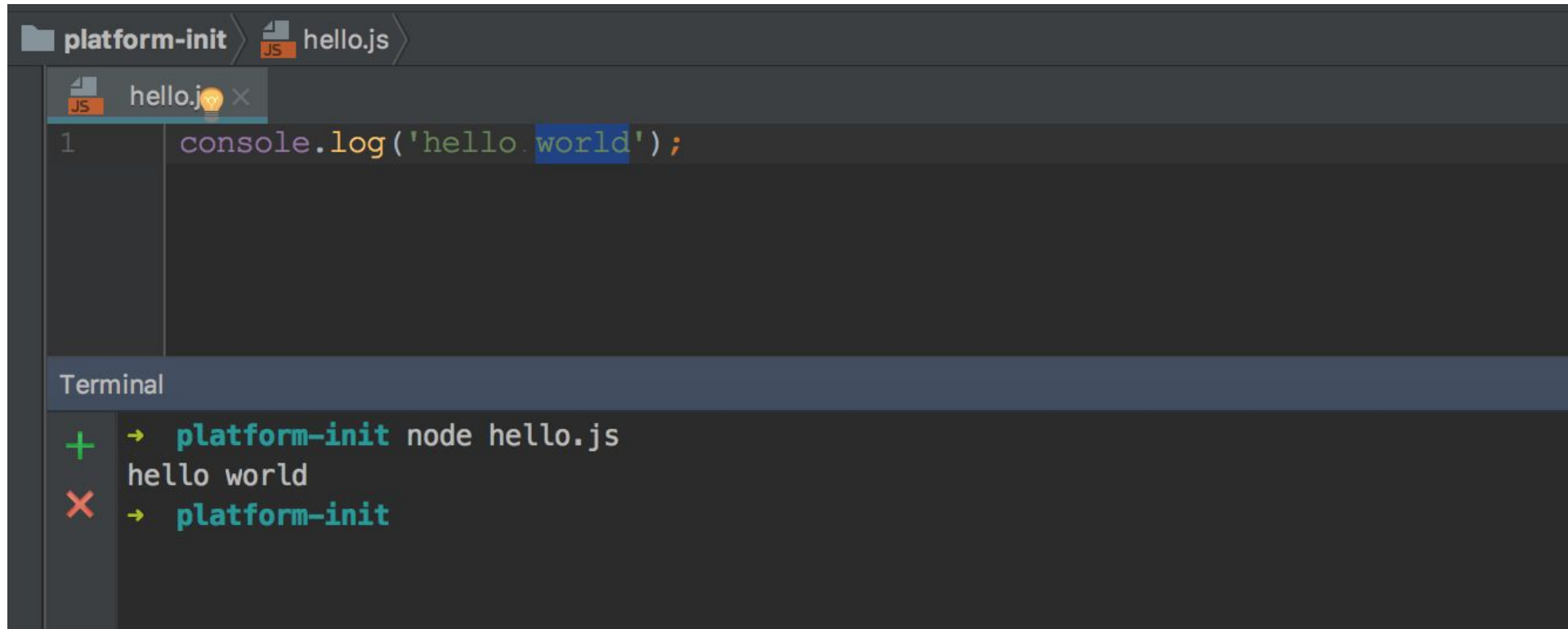
《linux 私房菜》12.2.2



# 几个疑问

- 环境变量 PATH 到底是什么？
- 如果定义为 `#!/bin/node`, 内容用 js 写, 程序会不会运行
- 比如用 js 写一个 hello, 能不能在任意位置执行 `$ hello` ？

# 执行node hello.js



```
platform-init > hello.js
```

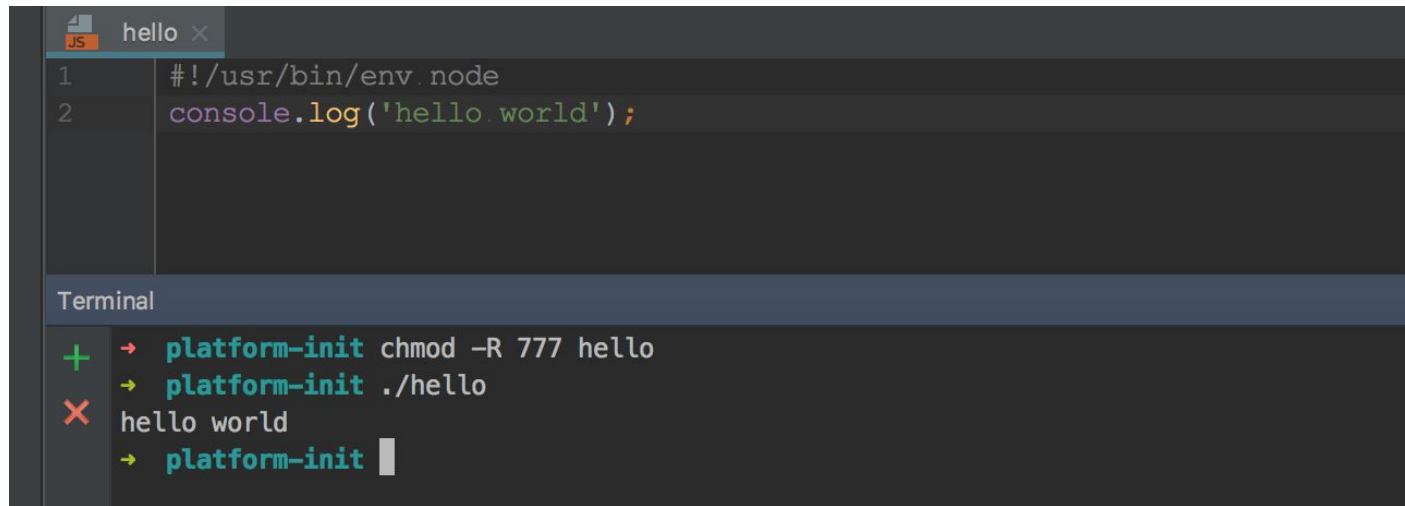
```
hello.js x
```

```
1 console.log('hello world');
```

```
Terminal
```

```
+ → platform-init node hello.js
hello world
✗ → platform-init
```

# 执行 ./hello



The screenshot shows a code editor with a file named 'hello.js' containing two lines of code:

```
1 #!/usr/bin/env node
2 console.log('hello world');
```

Below the code editor is a terminal window. It shows the following commands and output:

```
+ → platform-init chmod -R 777 hello
→ platform-init ./hello
X hello world
→ platform-init
```

- `#!/usr/bin/env node` 会在运行时被重定向到 `${PREFIX}/bin/node`
- 但是依然还是带上路径, 没有解决环境变量的问题

# 执行\$ hello

The image shows a code editor interface for a project named 'platform-init'. The left sidebar displays the project structure, including files like 'package.json', 'package-lock.json', 'README.md', and 'scm\_build.sh'. The main editor area shows the 'package.json' file with the following content:

```
10  "scripts": {
11    "lint-all": "eslint ./src; exit 0"
12  },
13  "bin": {
14    "hello": "./hello"
15  },
16  "license": "ISC"
17 }
```

A red annotation '命令与文件的映射' (Command and file mapping) points to the 'hello' entry in the 'bin' field.

Below the editor, the 'hello' file is shown with the following content:

```
1  #!/usr/bin/env node
2  console.log('hello world');
```

A red annotation '什么都没有改变' (Nothing has changed) points to the content of the 'hello' file.

The terminal at the bottom shows the command 'platform-init npm link' and its output:

```
+ platform-init npm link
> fsevents@1.1.2 install /Users/lingxufeng/bytedance/platform-init/node_modules/fsevents
> node install

[fsevents] Success: "/Users/lingxufeng/bytedance/platform-init/node_modules/fsevents/lib/binding/Release/node-v57-darwin-x64/fse.node" already installed
Pass --update-binary to reinstall or --build-from-source to recompile
npm WARN platform-init@1.0.0 No repository field.

added 355 packages in 12.319s
/Users/lingxufeng/.npm/versions/node/v8.1.2/bin/hello -> /Users/lingxufeng/.npm/versions/node/v8.1.2/lib/node_modules/platform-init/hello
/Users/lingxufeng/.npm/versions/node/v8.1.2/lib/node_modules/platform-init -> /Users/lingxufeng/bytedance/platform-init
platform-init hello
hello world
platform-init
```

A red box highlights the two lines of symbolic links created, with a red annotation '两个符号链接' (Two symbolic links) pointing to them.

# 原理解读

`/Users/lingxufeng/.nvm/versions/node/v8.1.2/lib/node_modules/platform-init -> /Users/lingxufeng/bytedance/platform-init`

- 相当于执行 `npm install -g platform-init`

`/Users/lingxufeng/.nvm/versions/node/v8.1.2/bin/hello -> /Users/lingxufeng/.nvm/versions/node/v8.1.2/lib/node_modules/platform-init/hello`

- 搜索 `platform-init` `bin` 字段内注册的可执行文件, 符号链接到 `${PREFIX}/bin` 文件夹下

```
→ ~  
→ ~  
→ ~ echo $PATH  
/Users/lingxufeng/.nvm/versions/node/v8.1.2/bin:/Users/lingxufeng/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin  
→ ~
```

# 全局安装 vs 本地安装

- 区别

全局安装直接将『可执行文件』安装到PATH 能访问到的 bin 文件夹中;本地安装会产生.bin 文件夹, 符号链接, 运行时加入 path

- 一个疑问, 项目依赖于 webpack, 但是项目本地和全局都安装了, 执行 webpack 时, 会用到哪一个?

```
➔ platform-init type -a webpack  
webpack is /Users/lingxufeng/.nvm/versions/node/v8.1.2/bin/webpack  
webpack is /usr/local/bin/webpack
```

# 链接

- inode

文件实体，记录文件属性以及该文件实体放置在哪儿几个 block

- hard link

在某个目录下增加一个文档名，该文档链接到一个 inode，只能 link 文件，安全。

- 符号链接

即快捷方式，创建一个独立文件，指向 link 的文档名，不安全

《linux 私房菜》7.2

# 工具与实践



# 写一个命令行工具 bytebone

```
→ ~ j inter
/Users/lingxufeng/bytedance/interview
→ interview git:(master) cd
→ ~ bytebone init redux kitty 参数处理
✓ 模板下载成功, 进入模板配置流程 下载功能

? 项目名称 kitty

? 项目具体描述 基于 ES6 + React + Redux + React-Router + Webpack 的前端项目

? 项目作者 lingxufeng <lingxufeng@bytedance.com>

? 是否为你的代码配置 ESLint 功能 ? Yes 交互式表单

? 确定 ESLint 预置方案 Airbnb 编译工具

✓ 初始化模板文件: .eslintignore
✓ 初始化模板文件: .babelrc
✓ 初始化模板文件: .eslintrc.json
✓ 初始化模板文件: .gitignore
✓ 初始化模板文件: README.md 醒目的输出
✓ 初始化模板文件: scm_build.sh
✓ 初始化模板文件: package.json
✓ 初始化模板文件: webpack.config.js 完成一项预定的任务!!!
✓ 初始化模板文件: webpack.prod.config.js
```

- 输入参数定义 & 处理
- 能执行网络任务
- 可提供交互式表单处理功能
- 有逻辑的输出提示信息
- 有流程, 目的是完成一项任务!
- 具有 help 信息

# 两个仓库

- 脚手架 bytebone 目录

一个npm 包, 用户全局安装, 提供一个命令行工具的全部能力。

几乎不会更新

- redux-template 目录

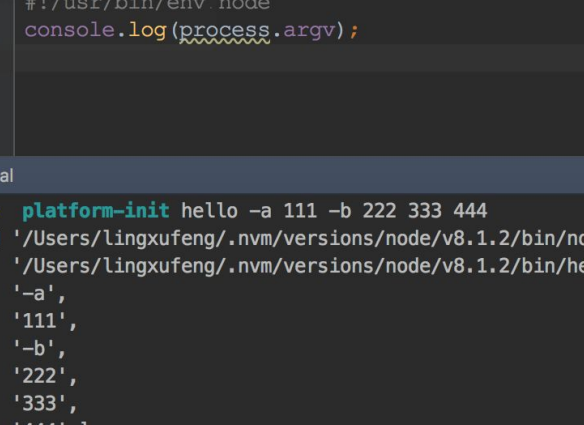
一个完备的基于 redux 的 react 项目模板。

用 handlebars 插入本多变量, 后续根据用户的输入编译成目标文件。

## 选项与参数

- `process.argv`

一个全局对象, 提供当前 node 的进程信息; argv 属性提供运行时参数。



The screenshot shows a code editor with two tabs: 'package.json' and 'hello'. The 'hello' tab is active, displaying a Node.js script:

```
1 #!/usr/bin/env node
2 console.log(process.argv);
3
```

Below the code editor is a terminal window with the title 'Terminal'. It shows the command `platform-init hello -a 111 -b 222 333 444` being executed. The output is an array of arguments:

```
[ '/Users/lingxufeng/.nvm/versions/node/v8.1.2/bin/node',
  '/Users/lingxufeng/.nvm/versions/node/v8.1.2/bin/hello',
  '-a',
  '111',
  '-b',
  '222',
  '333',
  '444' ]
```

The terminal prompt `platform-init` is visible at the bottom.

- commander.js

命令行接口解决方案, 提供参数定义、  
help 信息、优雅呈现输入

```

28 commander.on('--help', function () {
29   console.log();
30   console.log('  Examples:');
31   console.log();
32   console.log(chalk.gray('    # 创建一个 redux 项目'));
33   console.log(chalk.blue('    $ bytebone init redux my-project'));
34   console.log();
35 });
36 commander.parse(process.argv);
37
38 // **

```

terminal

```

→ bytebone git:(master) x bytebone

Usage: bytebone <command> [options]

Options:

  -V, --version  output the version number
  -h, --help     output usage information


Commands:

  init          选择一个模板来创建一个新的项目
  list          显示目前可用的模板列表
  help [cmd]    display help for [cmd]

```

# 执行 linux 命令

- `child_process`

新建子进程，运行结果保存在系统缓存，子进程退出后可通过回调返回运行结果。

- `exec()`

执行 bash 命令的常见用法

- `example`

```
var name = exec('git config --get user.name')  
var email = exec('git config --get user.email')
```

# 文件处理

- path 模块

`path.join`; `path.relative`; `path.resolve`

- fs 模块

`fs.existsSync`; `fs.readFile()`;

- 第三方工具模块

`require('user-home')`; `require('async')`

# 下载 & 等待 & color & 会话 & 编译

- down-git-repo: 用 node 下载一个 git 仓库
- ora: spin 动画与状态切换
- chalk: console 的美化输出, 可配置颜色
- requirer: 提供命令行上的 radio、checkbox、input
- handlebars: 编译模板文件

# 一些计划

- 开发一个命令行字典
- 一键生成 redux action, 并生成测试代码的工具
- 看懂一个比较大的轮子, 比如 mocha, 造轮子

# 参考资料

- linux 私房菜第四版(1~12章)
- node 官方文档
- npm 官方文档
- vue-cli
- create-react-app



thank you!