Futu5 Node.js 接入层脚手架方案

背景

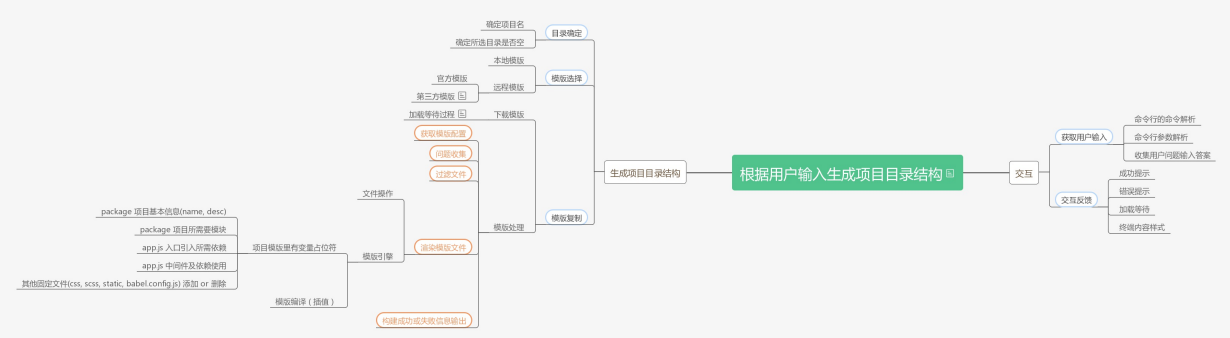
目前开发基于 futu5 node.js 接入层为基础的项目时，直接从之前的项目copy过来然后进行修改。表面上看问题不大，但其实存在很多问题：

- 重复性工作，繁琐而且浪费时间
- copy过来的模板容易存在与个人需求无关的代码
- 人工操作永远都有可能犯错，建新项目时，总要花时间去排错
- 项目中有很多需要配置的地方，容易忽略一些配置点，进而埋坑
- 缺乏版本控制：内部框架也在不停的迭代，人工建项目往往不知道框架最新的版本号是多少，使用旧版本的框架可能会重新引入一些bug
- 新的 feature 和 bug fix 难以同步

针对以上问题，开发人员可以根据交互动态生成项目结构、自动添加依赖和配置、移除不需要的文件的脚手架工具是有必要的。

像我们熟悉的 `vue-cli, react-native-cli` 等脚手架，只需要输入简单的命令 `vue init webpack project`，即可快速帮我们生成一个初始项目。因此，在实际工作中，我们可以定制一个属于自己的脚手架，来提高自己的工作效率。

需求分析



命令格式：

```
# 命令
futu5 init '<template-name>#[branch-name]' [project-name]
# options
# 使用 git clone 方式下载模版，默认 http
-c, --clone
```

```
# 输出帮助信息
-h, --help
# 使用本地缓存模版模式
-o, --offline
```

命令示例：

```
# 根据模版创建项目
futu5 init futu5_ipo my-project
# 根据指定模版版本创建项目 - 拉取指定分支
futu5 init futu5_ipo#1.0 my-project
# 使用本地缓存模版创建项目
futu5 init futu5_ipo my-project -o
# 使用 git clone 方式下载模版创建项目
futu2 init futu5_ipo my-project -c
```

方案介绍

Copy模版方案

1. 直接复制 futu5 node.js 接入层基础框架
2. 在此基础上根据自己业务需求进行修改或开发

express-generator方案

1. 将 futu5 node.js 接入层基础框架作为模版放在脚手架根目录下 template 文件夹
2. 模版目录可以是非严格的项目目录结构，目录结构的组织放在脚手架主逻辑中组织
3. 在脚手架中通过结合用户问题收集进行文件筛选并复制/渲染（利用模版引擎动态渲染依赖配置项及其相关配置）文件到目标项目的指定子目录下

vue-cli方案

1. 与 express—generator 类似
2. 但模版放在远程仓库，通过 clone 或 http 方式下载
3. 模版仓库具有严格的目录格式
4. 模版仓库根目录下有 meta.{json, js} 模版配置文件及 template 模版目录
5. meta. {json, js} 文件下定义了进行问题收集的相关依赖配置性问题以及文件筛选的基本根据。该文件必须导出为一个 js 对象
6. template 下必须为严格标准的目录结构。脚手架只关注依赖配置和文件过滤，不做目录结构的组织

方案比较

属性/方案	copy模版方案	express-generator方案	vue-cli方案
使用简单	Y	Y	Y
更新方便	N	N	Y
存在第三方依赖	N	Y	Y
维护成本	高	中	低
耦合程度	高	中	低
有良好的扩展性	低	中	高
可扩展第三方模版	Y	N	Y
扩展第三方模版难度	低	高	中

方案优劣

Copy模版方案

1. 使用简单

1. 重复性工作，繁琐而且浪费时间
2. copy过来的模板容易存在与个人需求无关的代码
3. 人工操作永远都有可能犯错，建新项目时，总要花时间去排错
4. 项目中有很多需要配置的地方，容易忽略一些配置点，进而埋坑

express-generator方案

1. 简单、具有针对性、易于实现
2. 适用于模版简单、变动频率不高的、可选配置项少的项目

1. 模版与脚手架深度耦合、需共同维护
2. 当模版发生变更时，修改内容可能触及脚手架主逻辑代码
3. 不方便扩展第三方模版

vue-cli方案

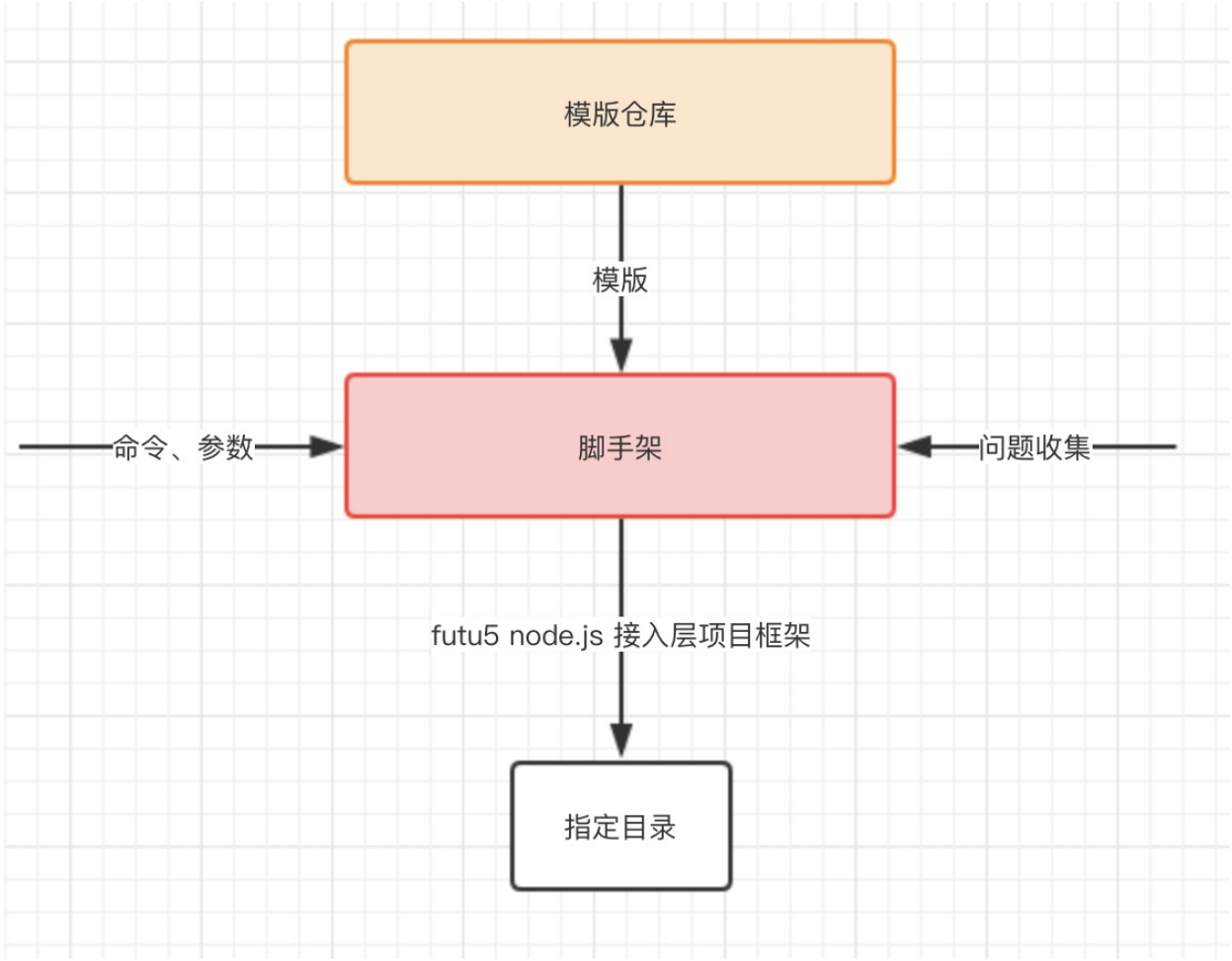
1. 脚手架更具通用性
2. 脚手架与模版分离，确保两部分独立维护
3. 脚手架只负责项目的构建流程
4. 模版负责项目的结构和依赖配置
5. 当项目结构和依赖发生变化时，只需对模版进行更新
6. 用户可根据自己需要定制模版

1. 对模版目录要求严格
2. 在扩展第三方模版时，需模版定义人员有一定前置知识

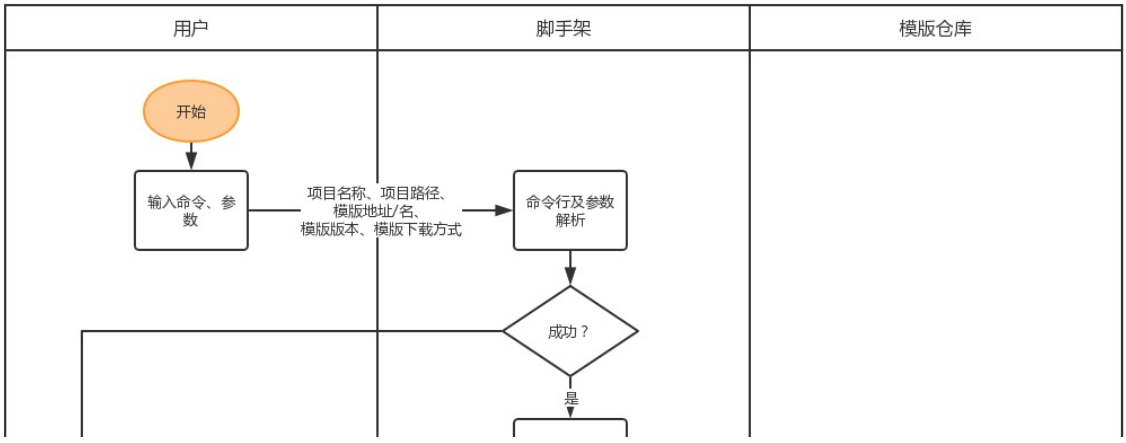
因此，本次将采用 vue-cli 脚手架方案进行开发，以下是脚手架分析：

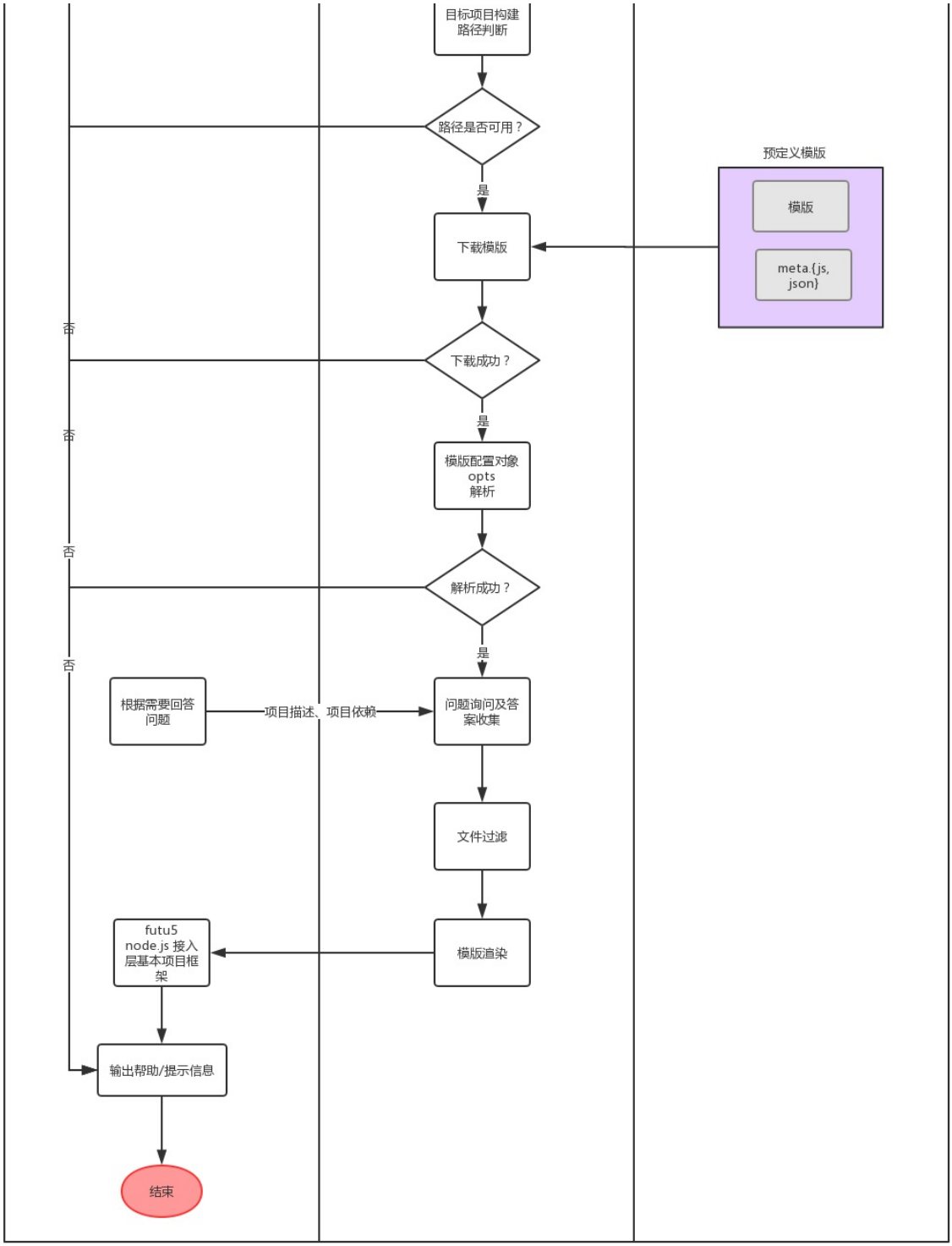
系统结构

- 用户输入指定命令及参数，脚手架解析出目标项目名称、目标项目构建路径、模版仓库
- 脚手架从模板仓库拉取模板文件，然后根据模板仓库根目录下定义描述文件进行收集问题
- 脚手架结合所收集问题答案过滤文件并生成接入层项目基本框架



主流程图



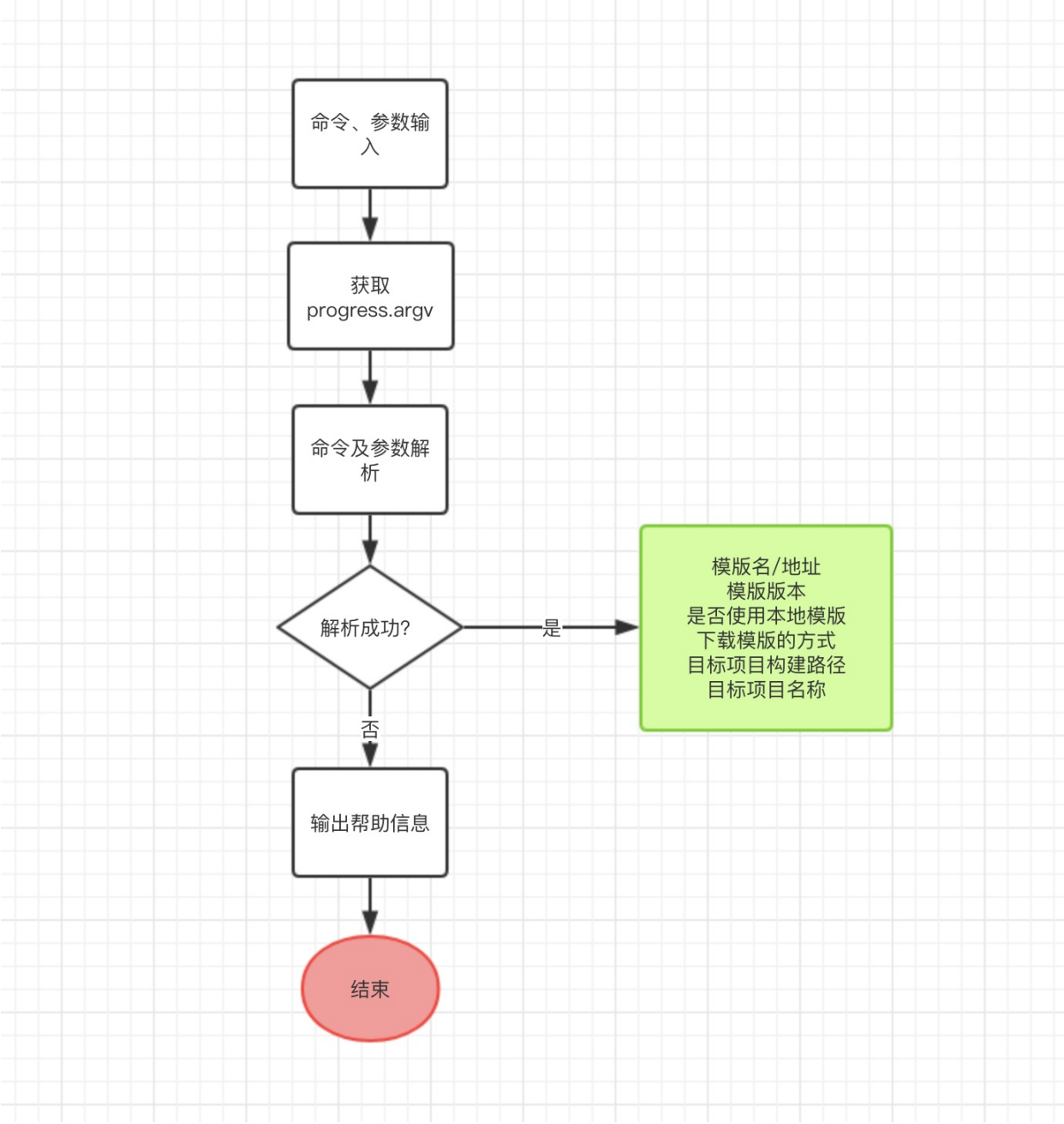


核心功能

1. 命令及参数解析

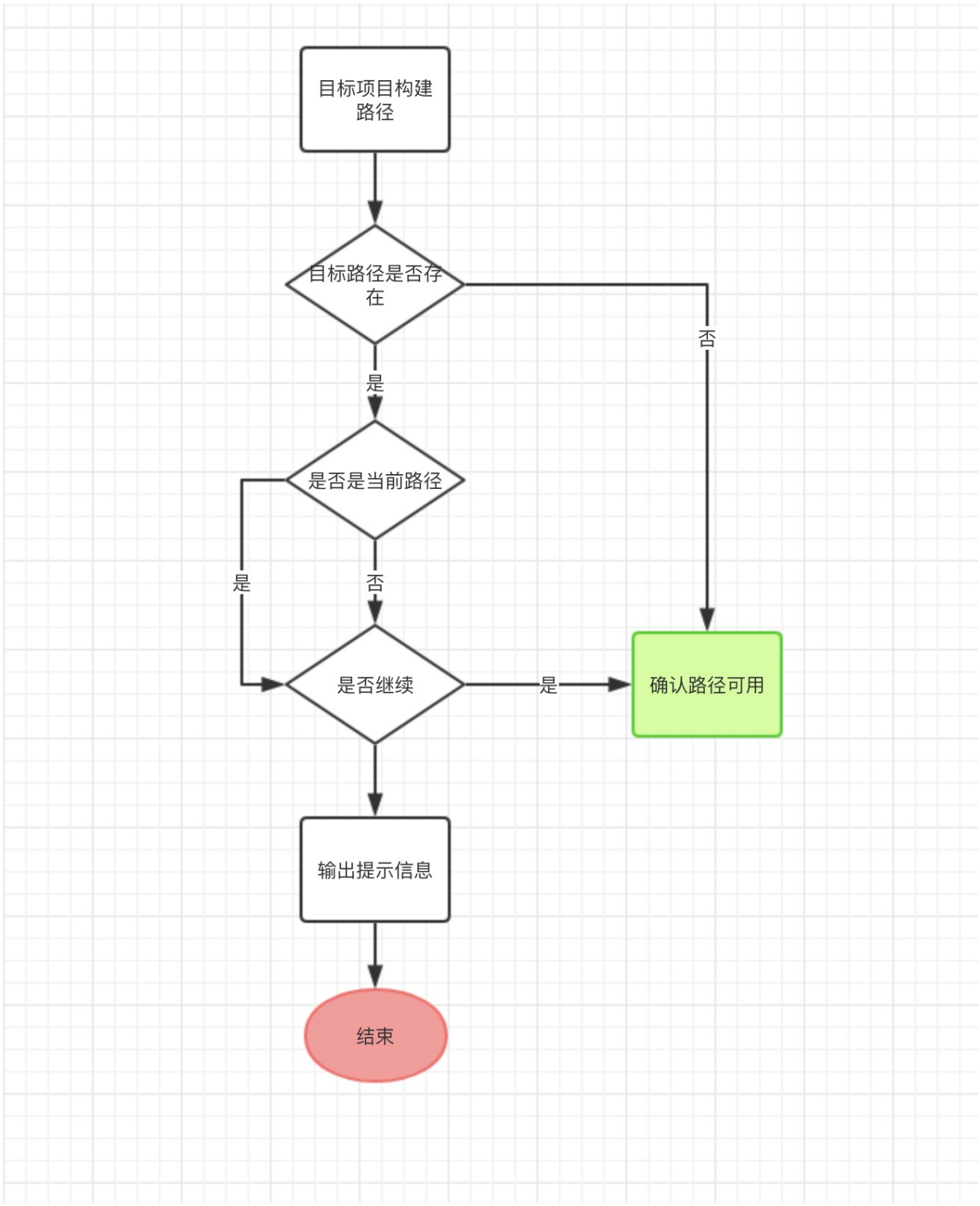
脚手架根据用户输入的命令及参数(progress.argv 包含当启动 Node.js 进程时传入的命令行参数)进行解析获得模版仓库地址/模版名（模版仓库地址为第三方扩展模版）、模版版本、下载模版

的方式（http/clone）、目标项目构建路径、目标项目名称等。



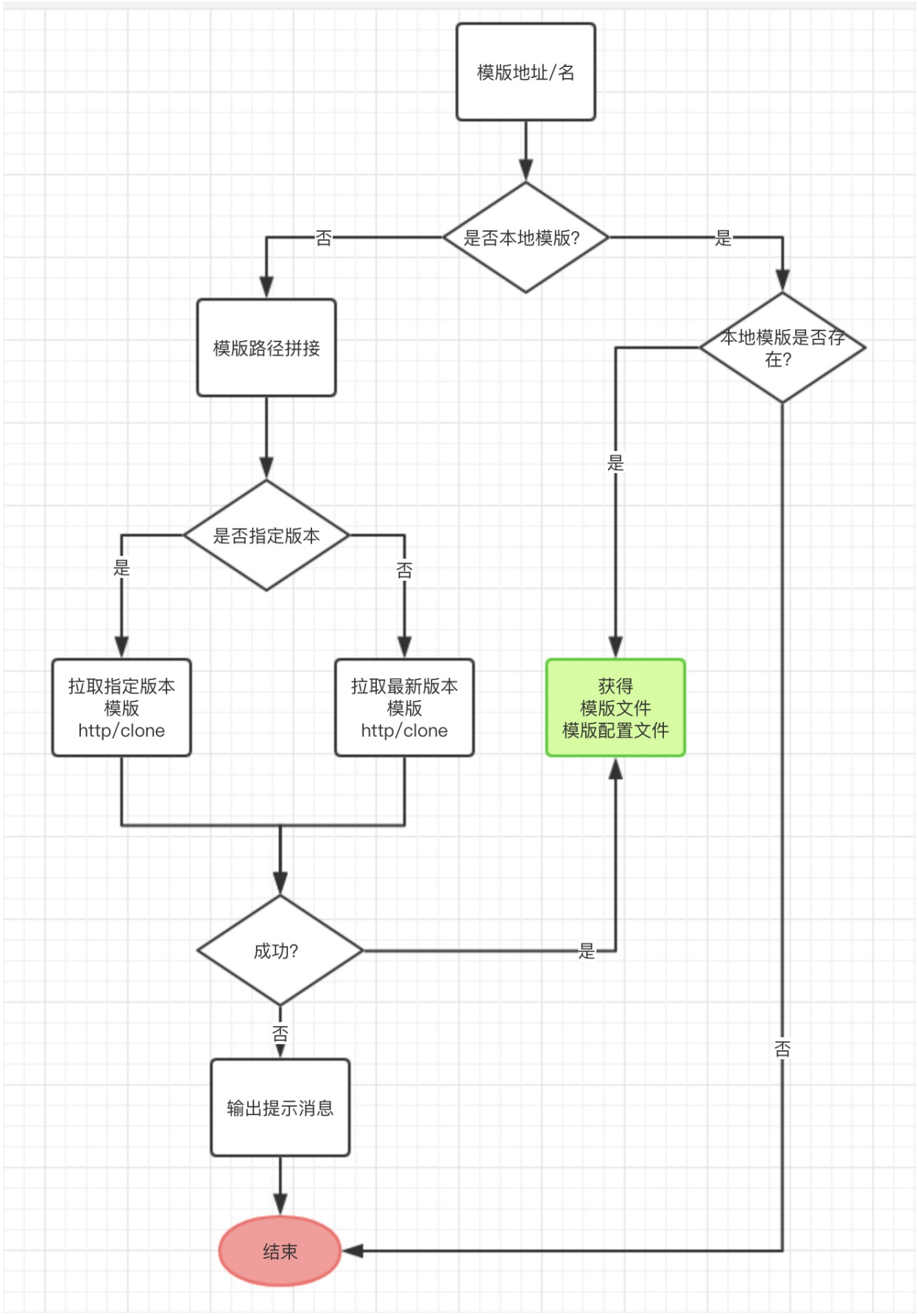
2. 目标项目构建路径判断

脚手架根据解析用户输入所得 **目标项目构建路径** 进行判断（是否已存在、是否是当前目录）。



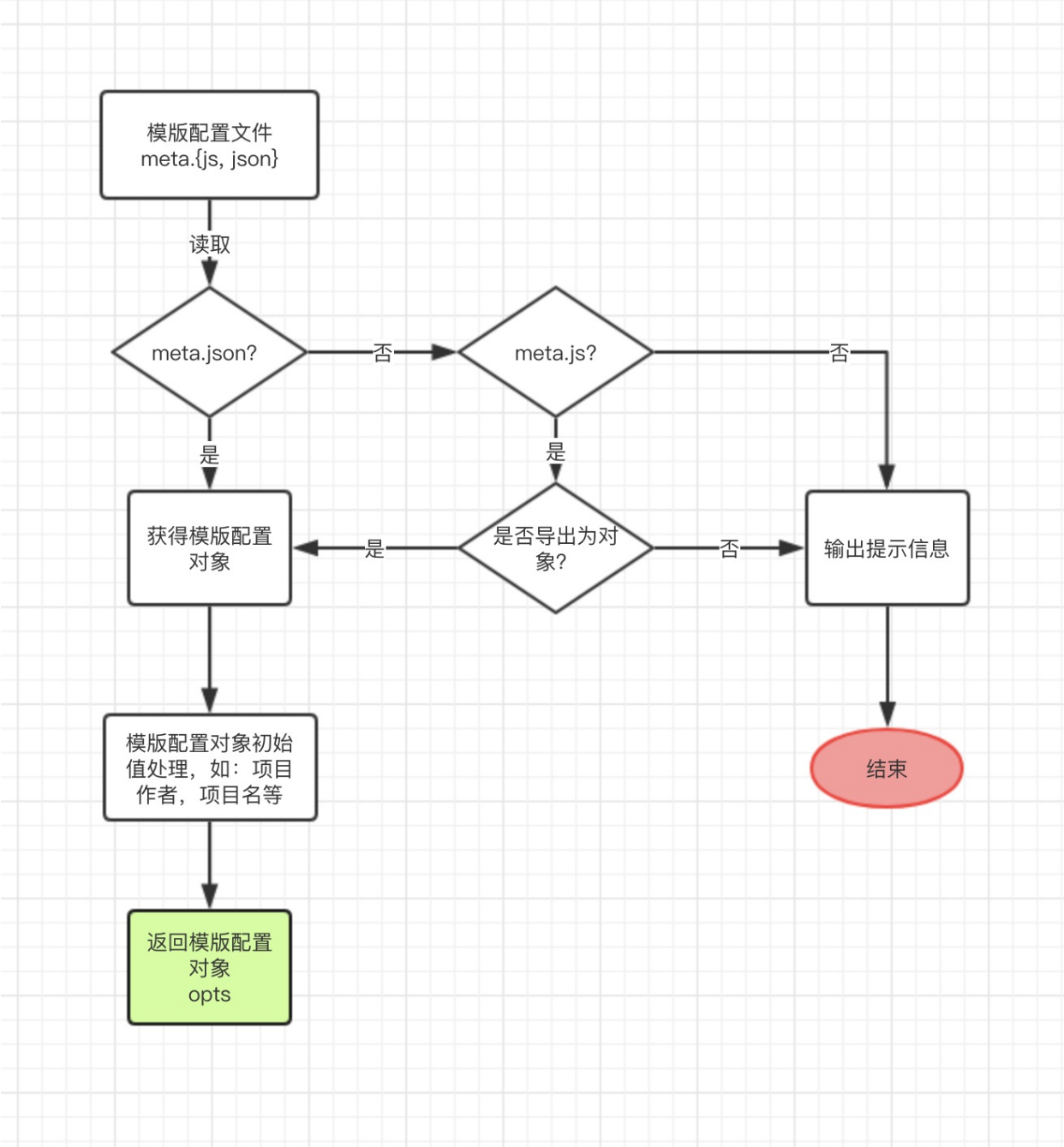
3. 模板下载

脚手架根据解析用户输入所得 **模版仓库地址/名** 拉取模版文件。


模版地址/名
是否本地模版?
否
是
模版路径拼接
本地模版是否存在?
是
是否指定版本
是
否
拉取指定版本
模版
http/clone
拉取最新版本
模版
http/clone
获得
模版文件
模版配置文件
成功?
是
否
输出提示消息
否
结束

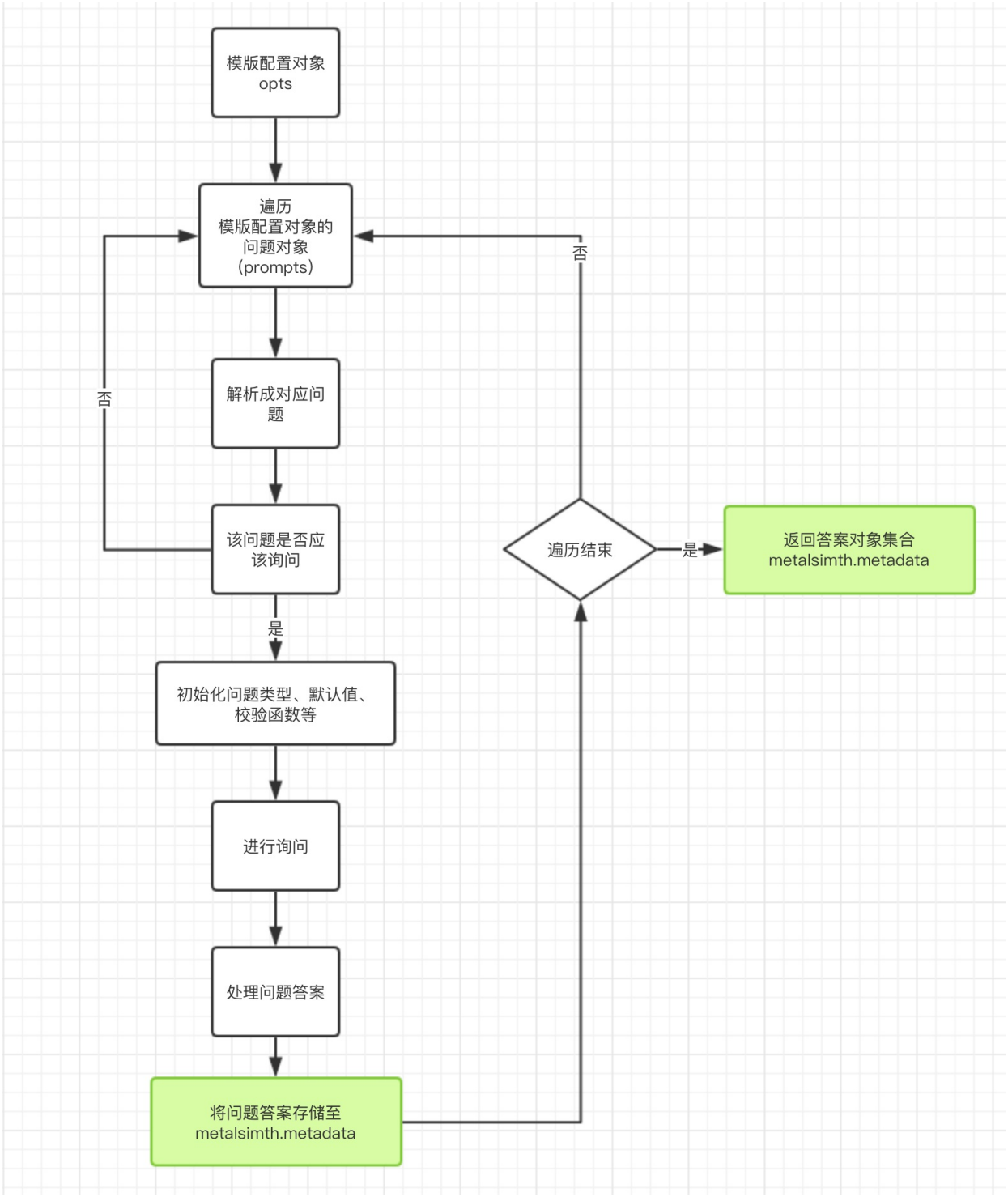
4. 模版配置文件解析

解析所拉取模版文件下包含的模版定义数据文件（即 `[meta.{js, json}](https://github.com/vuejs-templates/webpack/blob/develop/meta.js)`）。该文件描述初始化项目时命令行的交互动作及文件过滤依据（即关于目标项目名称、目标项目描述、目标项目依赖配置选择等问题的定义）。返回模版配置对象。



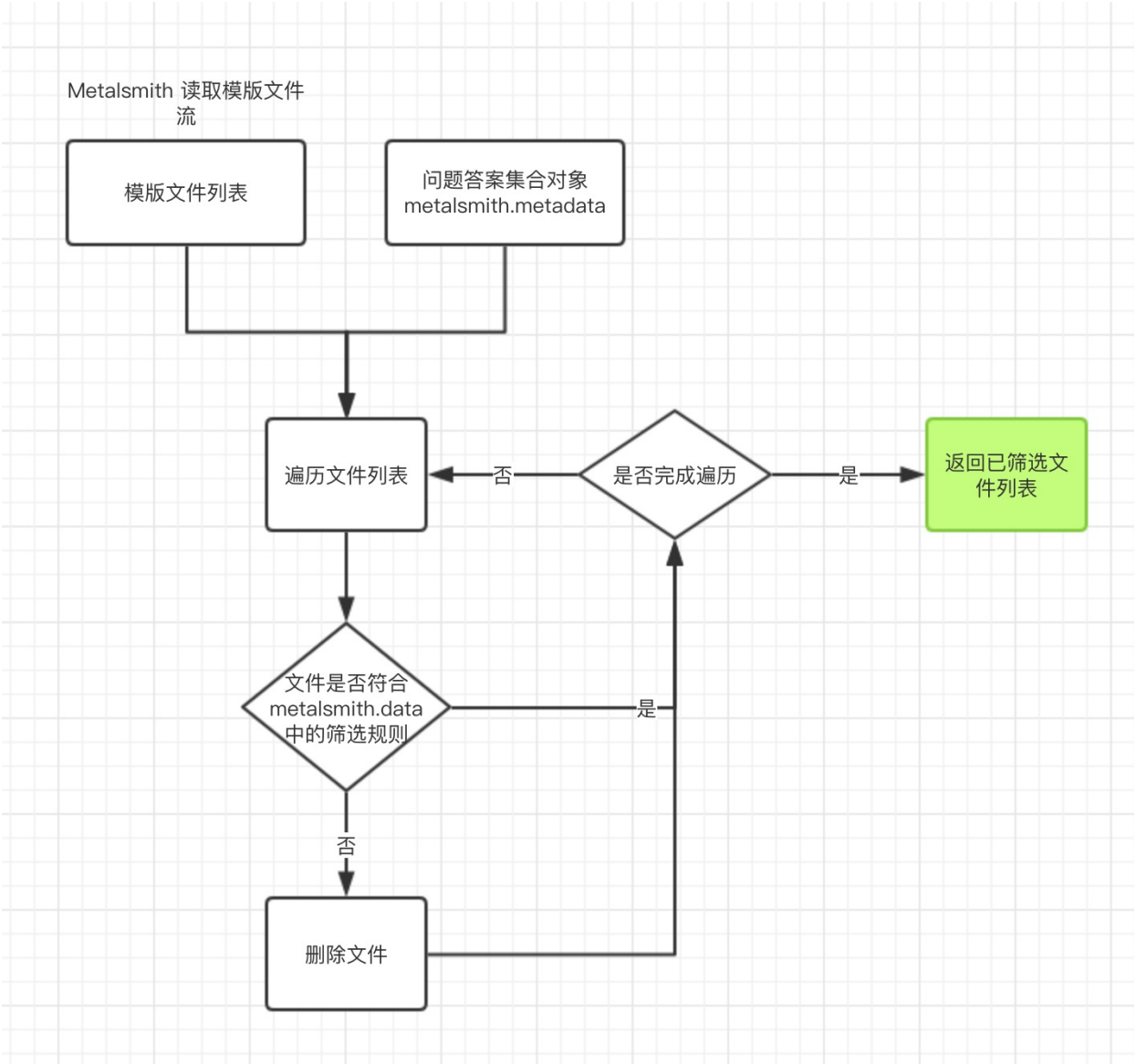
5. 问题收集

将 `meta.js` 或者 `meta.json` 模版配置对象中的 `prompts` 字段解析成对应的问题询问，并收集答案存于 `metalsmith.metadata()`。



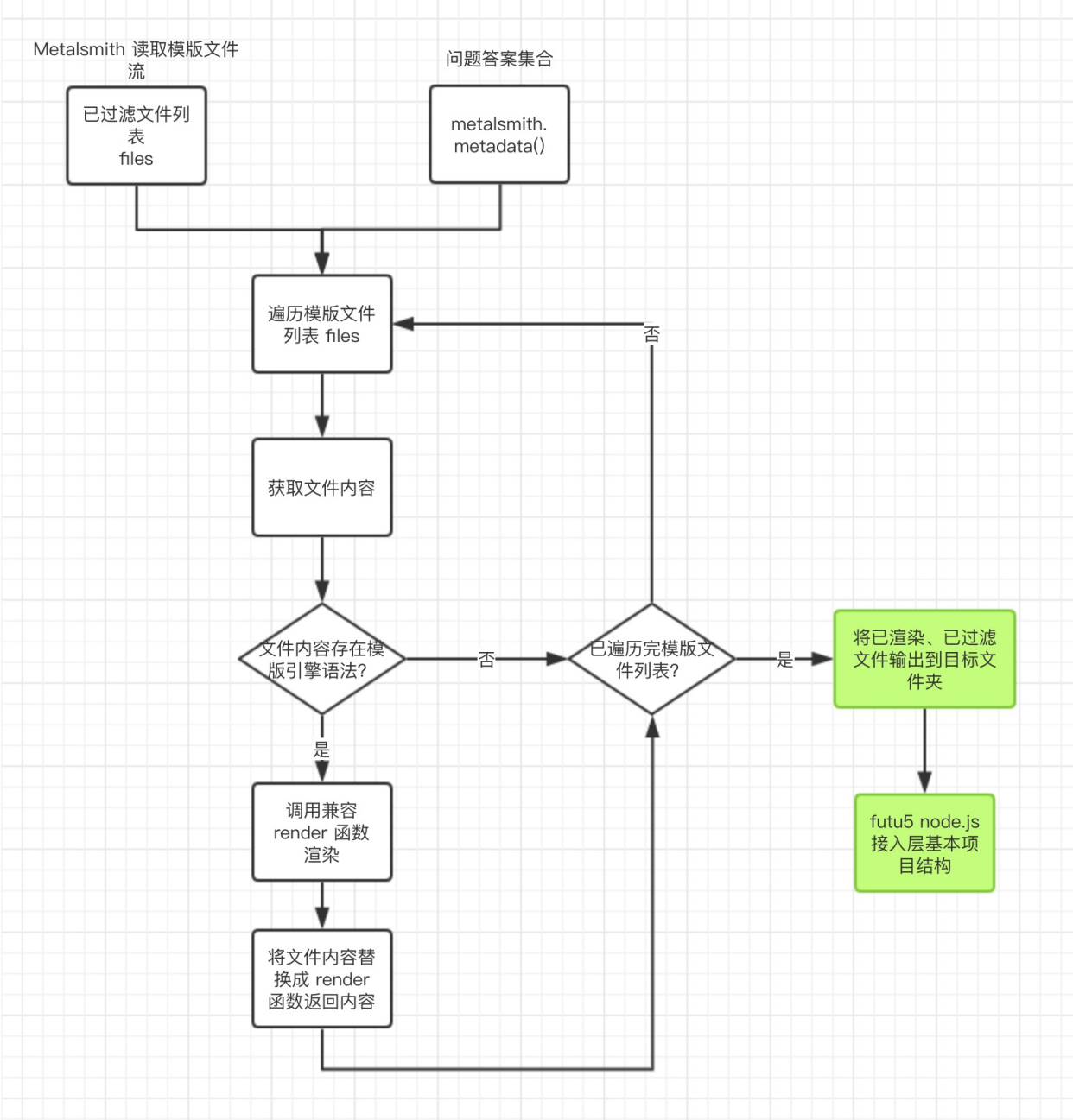
6. 文件过滤

根据问题答案收集所得 `metalsmith.metadata()` 删除一些不需要的模版文件。



7. 模板渲染

将已过滤模版文件中内容含模版引擎语法的文件内容替换成 `render` 函数（对模版文件的变量占位符实现动态插值）的返回，其他文件保持不变。



8. 输出提示信息

输出项目启动帮助信息。

技术选型

1. 使用语言：Node.js

2. 命令及参数解析：commander.js

- 将用户输入参数项解析成一个对象的属性
- 如，可直接通过 `program.clone` 知道模版的下载方式
- 简化了命令行开发

3. 路径存在判断：fs.existsSync(path)

4. 模板下载：download-git-repo

- 下载并提取 git 仓库，用于下载项目模板
- 支持 clone 以及 http 模式

5. 问题收集：inquirer.js

- 用于与用户进行交互，接受一个问题对象的数组
- 在用户与终端交互过程中，将用户的输入存放在一个答案对象
- 支持多种类型的问题形式：confirm、数组、字符串、checkbox、password、editor等
- 支持根据前面问题回答情况决定后面问题是否需要进行提问

6. 文件读取：metalsmith

- 一个简单的静态网站生成器，可插拔。所有的逻辑都是由插件来处理的。以文件流的形式链接起来。
- 工作原理：
  - 读取源目录中的所有文件。
  - 调用一系列操作文件的插件。可在处理文件时使用中间件来对模版进行处理
  - 将结果写入目标目录
- 在本项目中：
  - 读取模版目录中所有文件
  - 调用自定义文件处理中间件：询问问题、过滤文件、渲染模版
  - 将结果写入目标目录

7. 过滤文件：minimatch

- 文件匹配

8. 模板引擎：handlerbars

- 模板文件中通过 handlerbars 引擎去处理与项目依赖等相关的模版文件。

9. 模板渲染：consolidate

- 模板引擎结合体。
- 在使用时，输入统一的模版数据对象及配置即使后期更换模版，也不必在主逻辑中修改，只需把引入的 render 函数改成其他模版引擎的 render 函数

10. 模版配置对象解析

- 读取 json 文件：read-metadata
- 检验项目名是否合法：validate-npm-package-name

11. 其他

- 加载等待动画：ora
- 高亮终端打印输出信息： chalk