

全部开发者教程

最长递增子序列

14: 源码阅读：场景五：乱序下的 diff 比对

15: 框架实现：场景五：乱序下的 diff 比对

16: 总结

第十三章：compiler 编译器 - 编译时核心设计原则

01: 前言

02: 模板编译的核心流程

03: 抽象语法树 - AST

04: AST 转化为 JavaScript AST，获取 codegenNode

05: JavaScript AST 生成 render 函数代码

06: 总结

第十四章：compiler 编译器 - 构建 compile 编译器



Sunday • 更新于 2022-10-19

上一节 01: 前言 03: 抽象语法树 ... 下一节

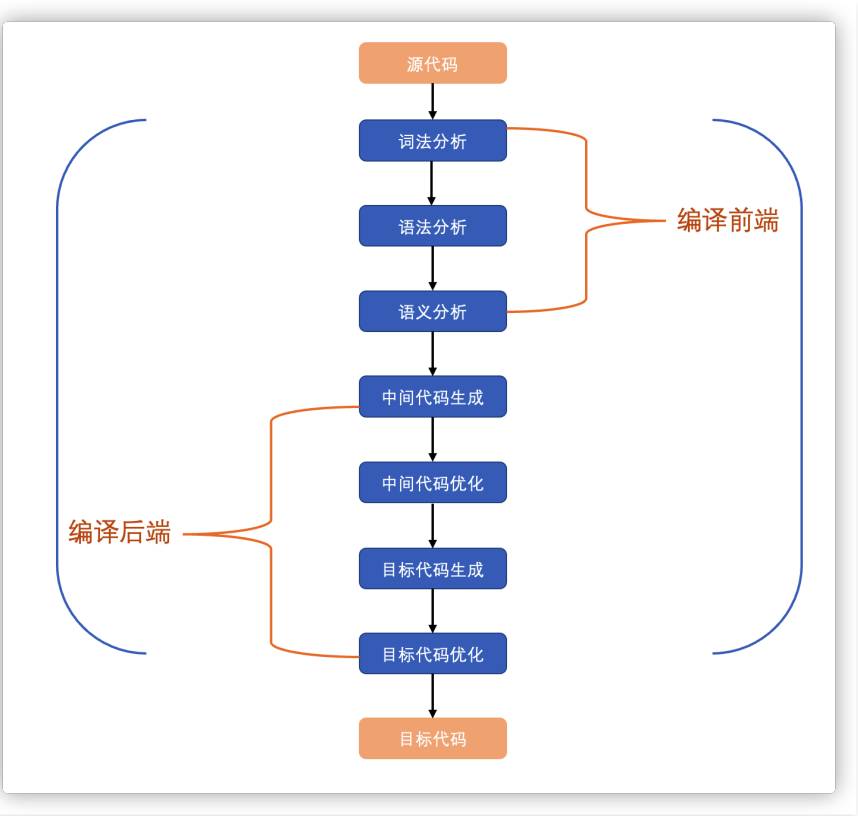
02：模板编译的核心流程

我们知道，对于 vue 中的 compiler 而言，它的核心作用就是把 template 模板 编译成 render 函数，那么在这样一个编译过程中，它的一个具体流程是什么呢？这一小节我们来看一下。

从上一小节的源码中，我们可以看到 编译器 compiler 本身只是一段程序，它的作用就是：把 A 语言，编译成 B 语言。

在这样的一个场景中 A 语言，我们把它叫做 源代码。而 B 语言，我们把它叫做 目标代码。整个的把源代码变为目标代码的过程，叫做 编译 compiler。

一个完整的编译过程，非常复杂，下图大致的描述了完整的编译步骤。



由图可知，一个完善的编译流程非常复杂。

但是对于 vue 的 compiler 而言，因为他只是一个领域特定语言（DSL）编译器，所以它的一个编译流程会简化很多，如下图所示：

索引目录

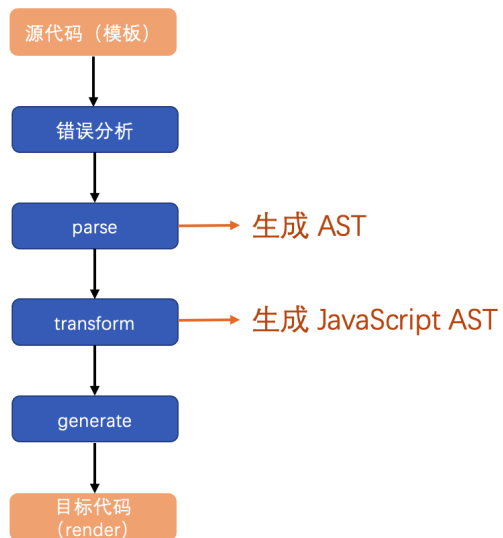
02: 模板编译的形

📖

?

📱

💬



由上图可知，整个的一个编译流程，被简化为了 4 步。

其中的错误分析就包含了词法分析、语法分析。这个我们不需要过于关注。

我们的关注点只需要放到 `parse`、`transform`、`generate` 中即可。

01: 前言 ◀ 上一节 下一节 ▶ 03: 抽象语法树 - AST

✎ 我要提出意见反馈

[企业服务](#) [网站地图](#) [网站首页](#) [关于我们](#) [联系我们](#) [讲师招募](#) [帮助中心](#) [意见反馈](#) [代码托管](#)

Copyright © 2022 imooc.com All Rights Reserved | 京ICP备 12003892号-11 京公网安备11010802030151号



✎ 意见反馈

♥ 收藏教程

🔖 标记书签