

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

01: 前言

02: 扩展知识: JavaScript与有限自动状态机

03: 扩展知识: 扫描 tokens

04: 源码阅读: 编译器第一步: 依据模板, 生成 AST 抽象语法树



Sunday • 更新于 2022-10-19

◀ 上一节 06: 总结 02: 扩展知识: ... 下一节 ▶

01: 前言

在上一章我们了解了 `compiler` 的作用和大致流程之后，那么这一章我们将要在 `vue-next-mini` 中实现一个自己的编译器。

但是我们也知道，`compiler` 是一个非常复杂的概念，我们无法在有限的课程中实现一个完善的编译器，所以，我们将会和之前一样，严格遵循：**没有使用就当做不存在** 和 **最少代码的实现逻辑** 这两个标准。只关注 **核心** 和 **当前业务** 相关的内容，而忽略其他。

那么明确好了以上内容之后，接下来，就让我们进入编辑器的学习吧~~~

06: 总结 ◀ 上一节 下一节 ▶ 02: 扩展知识: JavaScript与有限自动状态机

 我要提出意见反馈

索引目录

01: 前言

