

全部开发者教程

12: 框架实现：处理根节点的转化，生成 JavaScript AST

13: 扩展知识：render 函数的生成方案

14: 源码阅读：编译器第三步：生成 render 函数

15: 框架实现：构建 CodegenContext 上下文对象

16: 框架实现：解析 JavaScript AST，拼接 render 函数

17: 框架实现：新建 compat 模块，把 render 转化为 function

18: 总结

第十五章：compiler 编译器 - 深入编辑器处理逻辑

01: 前言

02: 响应性数据的编辑器处理：响应性数据的处理逻辑

Sunday • 更新于 2022-10-19

◀ 上一节 17: 框架实现：... 01: 前言 下一节 ▶

18: 总结

到这里我们就已经完成了一个基础的编辑器处理。

我们知道整个编辑器的处理过程分成了三部分：

1. 解析模板 `template` 为 `AST`

1. 在这一步过程中，我们使用了

1. 有限自动状态机解析模板得到了 `tokens`

2. 通过扫描 `tokens` 最终得到了 `AST`

2. 转化 `AST` 为 `JavaScript AST`

1. 这一步是为了最终生成 `render` 函数做准备

2. 利用了深度优先的方式，进行了自下向上的逐层转化

3. 生成 `render` 函数

1. 这一步是最后的解析环节，我们需要对 `JavaScript AST` 进行处理，得到最终的 `render` 函数

整个一套编辑器的流程非常复杂，我们目前只完成了最基础的编辑逻辑，目前只能支持 `<div>文本</div>` 的处理。那么如果我们想要处理更复杂的逻辑，比如：

1. 响应性数据

2. 多个子节点

3. 指令

的话，对比编辑器而言，还需要做更多的事情才可以。

下一章，我们会深入编辑器，来看一下，以上问题应该如何进行处理。

17: 框架实现：新建 compat 模块，把 ren... ◀ 上一节 下一节 ▶ 01: 前言

我要提出意见反馈

索引目录

18: 总结

📖

?

📱

💬