

全部开发者教程

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 构建 AST 结构的方案

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 02: 扩展知识: ... 04: 源码阅读: ... 下一节 ▶

03: 扩展知识: 扫描 tokens 构建 AST 结构的方案

在上一小节中, 我们已经知道可以通过自动状态机解析模板为 `tokens`, 那么解析出来的 `tokens` 就是生成 `AST` 的关键。

生成 `AST` 的过程, 就是 `tokens` 扫描的过程。

我们以以下 `html` 结构为例:

<> 代码块

```
1 <div>
2   <p>hello</p>
3   <p>world</p>
4 </div>
```

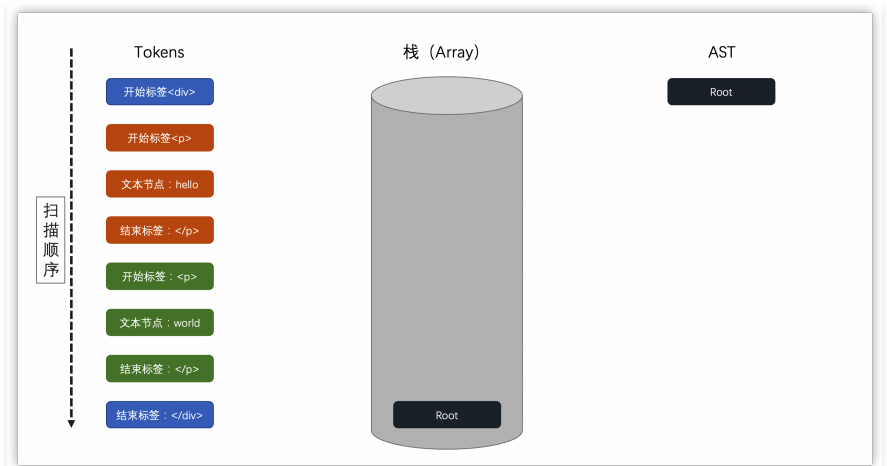
该 `html` 可以被解析为如下 `tokens` :

<> 代码块

```
1 开始标签: <div>
2 开始标签: <p>
3 文本节点: hello
4 结束标签: </p>
5 开始标签: <p>
6 文本节点: world
7 结束标签: </p>
8 结束标签: </div>
```

具体的扫描过程为 (文档中仅显示初始状态和结束状态, 具体扫描流程可以查看 [课程资料 PPT 第 7 页](#)) :

初始状态:



结束状态:

索引目录

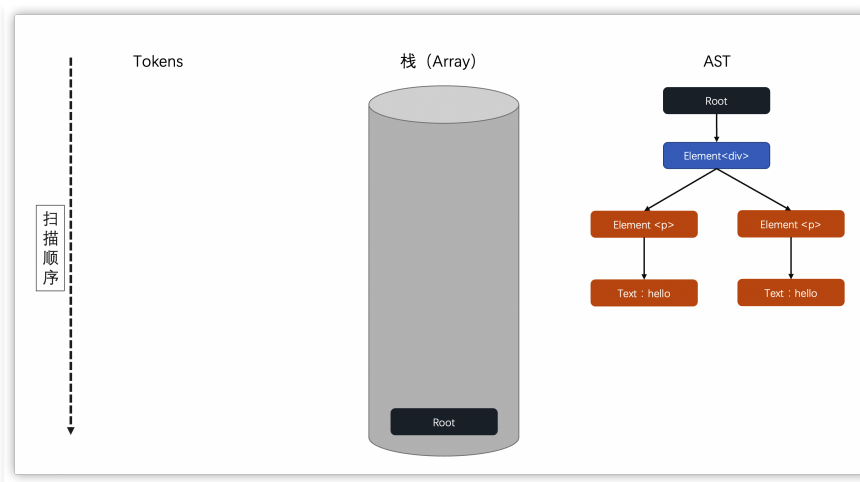
03: 扩展知识: 扫描 tokens 构建 AST 结构的方案

📄

?

📱

💬



在刚才的图示中，我们通过 [递归下降算法](#) 这样的一种扫描形式把 `tokens` 通过 **栈** 解析成了 **AST**（抽象语法树）。

02: 扩展知识: JavaScript与有限自动状态机 ◀ 上一节 下一节 ▶ 04: 源码阅读: 编译器第一步: 依据模板, ...

[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](#)