

全部开发者教程

☰

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

05: 框架实现：构建 parse 方
法，生成 context 实例

06: 框架实现：构建有限自动
状态机解析模板，扫描 token
生成 AST 结构

07: 框架实现：生成 'AST'，
构建测试

08: 扩展知识：AST 的



Sunday • 更新于 2022-10-19

◀ 上一节 04：源码阅读：... 06：框架实现：... 下一节 ▶

05：框架实现：构建 parse 方法，生成 context 实例

从这一小节开始，我们将实现 `vue-next-mini` 中的编辑器模块。首先我们第一步要做的就是生成 `AST` 对象。但是我们知道 `AST` 对象的生成颇为复杂，所以我们把整个过程分为成三步进行处理。

1. 构建 `parse` 方法，生成 `context` 实例
2. 构建 `parseChildren`，处理所有子节点（最复杂）

1. 构建有限自动状态机解析模板
2. 扫描 token 生成 `AST` 结构

3. 生成 `AST`，构建测试

那么本小节，我们就先处理第一步。

1. 创建 `packages/compiler-core/src/compile.ts` 模块，写入如下代码：

<> 代码块

```
1 export function baseCompile(template: string, options) {
2   return {}
3 }
```

2. 创建 `packages/compiler-dom/src/index.ts` 模块，导出 `compile` 方法：

<> 代码块

```
1 import { baseCompile } from 'packages/compiler-core/src/compile'
2
3 export function compile(template: string, options) {
4   return baseCompile(template, options)
5 }
```

3. 在 `packages/vue/src/index.ts` 中，导出 `compile` 方法：

<> 代码块

```
1 export { compile } from '@vue/compiler-dom'
```

4. 创建 `packages/compiler-core/src/parse.ts` 模块下创建 `baseParse` 方法：

<> 代码块

```
1 /**
2  * 基础的 parse 方法，生成 AST
3  * @param content template 模板
4  * @returns
5  */
6 export function baseParse(content: string) {
7   return {}
8 }
```

5. 在 `packages/compiler-core/src/compile.ts` 模块下的 `baseCompile` 中，使用 `baseParse` 方法：

索引目录

05: 框架实现：构



<> 代码块

```
1 import { baseParse } from './parse'
2
3 export function baseCompile(template: string, options) {
4   const ast = baseParse(template)
5   console.log(JSON.stringify(ast))
6
7   return {}
8 }
```

那么至此，我们就成功的触发了 `baseParse`。接下来我们去生成 `context` 上下文对象。

1. 在 `packages/compiler-core/src/parse.ts` 中创建 `createParserContext` 方法，用来生成上下文对象：

<> 代码块

```
1 /**
2  * 创建解析器上下文
3  */
4 function createParserContext(content: string): ParserContext {
5   // 合成 context 上下文对象
6   return {
7     source: content
8   }
9 }
```

2. 创建 `ParserContext` 接口：

<> 代码块

```
1 /**
2  * 解析器上下文
3  */
4 export interface ParserContext {
5   // 模板数据源
6   source: string
7 }
```

3. 在 `baseParse` 中触发该方法：

<> 代码块

```
1 export function baseParse(content: string) {
2   // 创建 parser 对象，未解析器的上下文对象
3   const context = createParserContext(content)
4   console.log(context)
5   return {}
6 }
```

那么至此我们成功得到了 `context` 上下文对象。

我们可以创建测试实例 `packages/vue/examples/compiler/compiler-ast.html`：

<> 代码块

```
1 <script>
2   const { compile } = Vue
3   // 创建 template
4   const template = `<div> hello world </div>`
5
6   // 生成 render 函数
7   const renderFn = compile(template)
8 </script>
```

可以成功打印 `context`

 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