慕课网首页 免费课 实战课 体系课 **慕课教程** 专栏 手记 企业服务

Q 第 集 我的课程

3.013.11

从所有教程的词条中查询…

首页 > 慕课教程 > Vue3源码分析与构建方案 > 01: 前言

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

Su

Sunday • 更新于 2022-10-19

◆ 上一节 16: 总结02: 模板编译的... 下一节 →

索引目录

01: 前言

Q

?

··

01: 前言

从这一章开始我们就开始进入到编译器的学习。

编译器是一个非常复杂的概念,在很多语言中均有涉及。不同类型的编译器在实现技术上都会有较大的差异。

比如你要实现一个 java 或者 JavaScript 的编译器, 那就是一个非常复杂的过程了。

但是对于我们而言,我们并不需要设计这种复杂的语言编辑器,我们只需要有一个 领域特定语言 (DSL) 的编辑器即可。

DSL 并不具备很强的普适性,它是仅为某个适用的领域而设计的,但它也足以用于表示这个领域中的问题以及构建对应的解决方案。

那么我们这里所谓的特定语言指的就是: 把 template 模板,编译成 render 函数。这个就是 vue 中 **编译器 compiler **的作用。

我们可以先创建一个测试实例 packages/vue/examples/imooc/compiler/compiler.html ,以此来看一下 vue 中 compiler 的作用:

```
/>代码块

// const { compile } = Vue

const template = `

double div>hello world</div>

const renderFn = compile(template)

console.log(renderFn);

// c
```

查看最终的打印结果可以发现,最终 compile 函数把 template 模板字符串转化为了 render 函数。

那么我们可以借此来观察一下 compile 这个方法的内部实现。我们可以在 packages/compiler-dom/src/index.ts 中的 第40行 查看到该方法。

从代码中可以发现, compile 方法, 其实是触发了 baseCompile 方法, 那么我们可以进入到该方法。

该方法的代码比较简单,剔除掉无用的内容之后,可以得到如下内容:

```
c>代码块

1 export function baseCompile(
2 template: string | RootNode,
3 options: CompilerOptions = {}
4 ): CodegenResult {
5
6  // 1. 通过 parse 方法进行解析, 得到 AST
7 const ast = isString(template) ? baseParse(template, options) : template
8
9  // 2. 通过 transform 方法对 AST 进行转化, 得到 JavaScript AST
10 transform(

② 收職教程

□ 标记书签
```

```
extend({}, options, {
12
13
         prefixIdentifiers,
14
         nodeTransforms: [
15
          ...nodeTransforms,
16
           ...(options.nodeTransforms || []) // user transforms
        ],
17
18
        directiveTransforms: extend(
19
          {},
20
          directiveTransforms,
21
          options.directiveTransforms || {} // user transforms
22
        )
23
       })
2.4
    )
25
     // 3. 通过 generate 方法根据 AST 生成 render 函数
27
     return generate(
28
       ast,
       extend({}, options, {
29
30
        prefixIdentifiers
      })
31
32
    )
33 }
```

这段代码 (complie), 主要做了三件事情:

- 1. 通过 parse 方法进行解析,得到 AST
- 2. 通过 transform 方法对 AST 进行转化, 得到 JavaScript AST
- 3. 通过 generate 方法根据 AST 生成 render 函数

整体的代码解析,虽然比较清晰,但是里面涉及到的一些概念,我们可能并不了解。

比如: 什么是 AST?

所以我们需要先花费一些时间,来了解编译器中的一些基础知识,然后再去阅读对应的源码和实现具体的 逻辑。

那么本章节,我们就先来做第一件事情:了解编译时的基础知识。

▶ 我要提出意见反馈

① ②

.

0

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

*** *** 6 4

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