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 🃜 💄 我的课程



Q

12: 框架实现: 划

?

··

索引目录

从所有教程的词条中查询…

//// 内外性的内外 | 三內

首页 > 慕课教程 > Vue3源码分析与构建方案 > 12:框架实现:处理根节点的转化,生成 JavaS...

全部开发者教程 :Ξ

08: 扩展知识: AST 到 JavaScript AST 的转化策略和 注意事项

09:源码阅读:编译器第二步:转化 AST,得到

JavaScript AST 对象

10: 框架实现: 转化 JavaScript AST,构建深度优 先的 AST 转化逻辑

11: 框架实现:构建 transformXXX 方法,转化对 成节点

12:框架实现:处理根节点的 转化、牛成 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

12: 框架实现: 处理根节点的转化, 生成 JavaScript AST

那么最后我们就只剩下根节点的处理了。

1. 在 transform 方法中:

```
<> 代码块
     export function transform(root, options) {
2
         createRootCodegen(root)
         root.helpers = [...context.helpers.keys()]
         root.components = []
         root.directives = []
         root.imports = []
8
         root.hoists = []
9
         root.temps = []
10
         root.cached = []
11
     }
```

2. 创建 createRootCodegen 方法:

```
<> 代码块
 1
     * 生成 root 节点下的 codegen
2
3
4
     function createRootCodegen(root) {
5
        const { children } = root
        // 仅支持一个根节点的处理
 8
        if (children.length === 1) {
            // 获取单个根节点
9
            const child = children[0]
1.0
            if (isSingleElementRoot(root, child) && child.codegenNode) {
11
12
               const codegenNode = child.codegenNode
                root.codegenNode = codegenNode
14
15
       }
     }
16
```

3. 创建 packages/compiler-core/src/hoistStatic.ts 模块, 创建 isSingleElementRoot 方法:

```
/**
/**
/**
/**
export function isSingleElementRoot(root, child) {
    const { children } = root
    return children.length === 1 && child.type === NodeTypes.ELEMENT
}
```

此时,整个 transform 处理完成。运行测试实例,可以得到如下打印:

✓ 意见反馈

♡ 收藏教程

□ 标记书签

```
<> 代码块
1
      type: 0,
2
      children: [
 3
       {
        type: 1,
        tag: 'div',
        tagType: 0,
8
        props: [],
        children: [{ type: 2, content: ' hello world ' }],
9
        codegenNode: {
10
11
          type: 13,
12
           tag: '"div"',
13
           props: [],
           children: [{ type: 2, content: ' hello world ' }]
14
15
      }
16
17
    ],
18 loc: {},
19
    codegenNode: {
20
      type: 13,
      tag: '"div"',
21
      props: [],
22
       children: [{ type: 2, content: ' hello world ' }]
23
24
    },
25
      helpers: [null],
26
      components: [],
27
     directives: [],
    imports: [],
28
    hoists: [],
29
    temps: [],
30
31
     cached: []
32 }
```

我们可以把如上打印放入到 vue 源代码中的 packages/compiler-core/src/compile.ts 中 baseCompile 方法中。

?

 \odot

注意: 需要把 helpers: [null] 改为 helpers: [CREATE_ELEMENT_VNODE]

此时,在 vue 源码中运行如下测试实例:

```
<> 代码块
1 <script>
     const { compile, h, render } = Vue
      // 创建 template
      const template = ``
     // 生成 render 函数
 6
     const renderFn = compile(template)
 8
9
    // 创建组件
10
    const component = {
11
      render: renderFn
12
13
14
    // 通过 h 函数, 生成 vnode
     const vnode = h(component)
15
17
     // 通过 render 函数渲染组件
18
      render(vnode, document.querySelector('#app'))
19 </script>
```

发现可以正常渲染 <div>hello world</div>

▶ 我要提出意见反馈

▶ 意见反馈

♡ 收藏教程

口 标记书签

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 A

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

⋮

?

0