慕课网首页 免费课 体系课 慕课教程 专栏 手记 企业服务 实战课

我的课程



Q

从所有教程的词条中查询…

全部开发者教程 Ξ

06:响应性数据的编辑器处

理: generate 生成 render 函

07: 响应性数据的编辑器处

理: render 函数的执行处理

08: 多层级模板的编辑器处

理: 多层级的处理逻辑

09: 基于编辑器的指令(v-xx) 处理: 指令解析的整体逻辑

10: 基于编辑器的指令(v-xx) 处理: AST 解析逻辑 (困难)

11: 基于编辑器的指令(v-xx) 处理: JavaScript AST,构建 vif 转化模块 (困难)

12: 基于编辑器的指令(v-xx) 处理: JavaScript AST , transform 的转化逻辑

13: 基于编辑器的指令(v-xx) 处理: 生成 render 函数

14: 总结

首页 > 慕课教程 > Vue3源码分析与构建方案 > 13: 基于编辑器的指令(v-xx)处理: 生成 rende...

Sunday • 更新于 2022-10-19

↓ 上一节 12: 基于编辑器...14: 总结 下一节 ▶

13: 基于编辑器的

?

··

索引目录

## 13: 基于编辑器的指令(v-xx)处理: 生成 render 函数

当 JavaScript AST 构建完成之后,最后我们只需要生成对应的 render 函数即可。

```
<>代码块
     const _Vue = Vue
    return function render(_ctx, _cache) {
      with (_ctx) {
 5
        const { createElementVNode: _createElementVNode, createCommentVNode: _createCommentV
 6
 8
 9
        return _createElementVNode("div", [], [
10
           " hello world ",
11
          isShow
             ? _createElementVNode("h1", null, ["你好, 世界"])
12
             : _createCommentVNode("v-if", true),
13
14
15
        1)
16
     }
17 }
```

依据以上模板,可以看出, render 的核心处理在于 当前的 三元表达式 (children) 处理:

```
<> 代码块
           " hello world ",
2
3
             ? _createElementVNode("h1", null, ["你好, 世界"])
4
5
             : _createCommentVNode("v-if", true),
6
         ]
```

而对于 codegen 模块而言,解析当前参数的函数为 genNode ,所以我们需要在 genNode 中增加对应的

1. 在 packages/compiler-core/src/codegen.ts 中的 genNode 方法下,增加节点处理:

```
<>代码块
1
     function genNode(node, context) {
        switch (node.type) {
            case NodeTypes.ELEMENT:
3
            case NodeTypes.IF:
                genNode(node.codegenNode!, context)
                break
            // JS调用表达式的处理
8
9
            case NodeTypes.JS CALL EXPRESSION:
                genCallExpression(node, context)
10
11
12
             // JS条件表达式的处理
```

✔ 意见反馈

♡ 收藏教程

口 标记书签

```
15 break
16 }
17 }
```

2. 创建 genCallExpression 方法:

```
<> 代码块
1 /**
     * JS调用表达式的处理
2
3
4
    function genCallExpression(node, context) {
        const { push, helper } = context
        const callee = isString(node.callee) ? node.callee : helper(node.callee)
        push(callee + `(`, node)
8
        genNodeList(node.arguments, context)
9
        push(`)`)
10
    }
```

3. 创建 genConditionalExpression 方法:

```
<> 代码块
1
     * JS条件表达式的处理。
2
    * 例如:
3
     * isShow
4
            ? _createElementVNode("h1", null, ["你好, 世界"])
5
            : _createCommentVNode("v-if", true),
 6
     function genConditionalExpression(node, context) {
9
        const { test, consequent, alternate, newline: needNewline } = node
10
        const { push, indent, deindent, newline } = context
       if (test.type === NodeTypes.SIMPLE_EXPRESSION) {
11
            // 写入变量
12
13
            genExpression(test, context)
14
        }
15
        // 换行
16
        needNewline && indent()
        // 缩进++
17
1.8
        context.indentLevel++
19
        // 写入空格
20
        needNewline || push(` `)
21
        // 写入 ?
22
        push(`? `)
        // 写入满足条件的处理逻辑
23
24
        genNode(consequent, context)
        // 缩进 --
25
26
        context.indentLevel--
27
        // 换行
28
        needNewline && newline()
29
        // 写入空格
        needNewline || push(` `)
30
        // 写入:
31
32
        push(`: `)
        // 判断 else 的类型是否也为 JS_CONDITIONAL_EXPRESSION
33
        const isNested = alternate.type === NodeTypes.JS_CONDITIONAL_EXPRESSION
35
        // 不是则缩进++
36
        if (!isNested) {
37
            context.indentLevel++
38
39
        // 写入 else (不满足条件)的处理逻辑
40
        genNode(alternate, context)
        // 缩进--
41
42
        if (!isNested) {
43
            context.indentLevel--
44
45
        // 控制缩进 + 换行
46
        needNewline && deindent()
47
```

?

0

## 此时生成的 render 函数为:

```
<> 代码块
const _Vue = Vue
2
3 return function render(_ctx, _cache) {
     with ( ctx) {
       const { createElementVNode: _createElementVNode, createCommentVNode: _createCommentV
     return _createElementVNode("div", [], [" hello world ", isShow
8
        ? _createElementVNode("h1", null, ["你好, 世界"])
9
        : _createCommentVNode("v-if", true)
10
       , " "])
11
12 }
13 }
```

在上述 render 中,因为使用了 createCommentVNode ,所以我们需要创建并导出该函数。

1. 在 packages/runtime-core/src/vnode.ts 中, 创建该函数:

```
/**
/**
/**
export function createCommentVNode(text) {
return createVNode(Comment, null, text)
}
```

2. 在 packages/runtime-core/src/index.ts 中导出:

```
1 export {
2 ...
3 createCommentVNode
4 } from './vnode'
```

?

0

3. 在 packages/vue/src/index.ts 中导出:

```
1 export {
2    ...
3    createCommentVNode
4  } from '@vue/runtime-core'
```

运行测试实例,效果可以正常展示。

同时,我们可以修改 isShow 的值,增加一个延迟的数据变化:

```
<> 代码块
1 <script>
    const { compile, h, render } = Vue
     // 创建 template
     const template = `<div> hello world <h1 v-if="isShow">你好, 世界</h1> </div>`
    // 生成 render 函数
 6
    const renderFn = compile(template)
 8
    console.log(renderFn.toString());
    // 创建组件
10 const component = {
11 data() {
12
      return {
       isShow: false
}
1.3
14
15 },
                          ♡ 收藏教程
                                         口 标记书签
             ╱ 意见反馈
```

```
created() {
    setTimeout
17
      setTimeout(() => {
 19
        this.isShow = true
       }, 2000);
   }
 20
21
22
23
24 // 通过 h 函数, 生成 vnode
25     const vnode = h(component)
26
27 // 通过 render 函数渲染组件
    render(vnode, document.querySelector('#app'))
28
29 </script>
响应式的数据渲染,依然可以正常展示。
   ✔ 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

**★ %** 6 **A** 

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

口 标记书签