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 Ē 我的课程



Q

?

 \odot

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 > 10: 基于编辑器的指令(v-xx)处理: AST 解析逻...

全部开发者教程 Ξ

弗丁ユ早:complier 編年 器 - 深入编辑器处理逻辑

- 01: 前言
- 02:响应性数据的编辑器处
- 理:响应性数据的处理逻辑
- 03:响应性数据的编辑器处
- 理:AST 解析逻辑
- 04: 响应性数据的编辑器处
- 理: JavaScript AST 转化逻辑
- 05:响应性数据的编辑器处
- 理: render 转化逻辑分析
- 06: 响应性数据的编辑器处
- 理: generate 生成 render 函
- 数
- 07: 响应性数据的编辑器外
- 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

Sunday • 更新于 2022-10-19

↓ 上一节 09: 基于编辑器...11: 基于编辑器...下一节 ▶

索引目录

10: 基于编辑器的

10: 基于编辑器的指令(v-xx)处理: AST 解析逻辑 (困难)

那么首先我们先处理 AST 的解析逻辑。

我们知道 AST 的解析, 主要集中在 packages/compiler-core/src/parse.ts 中。在 该模块下, 存在 pa rseTag 方法,该方法主要用来解析标签。那么对于我们的属性解析,也需要在该方法下进行。

该方法目前的标签解析,主要分成三部分:

- 1. 标签开始
- 2. 标签名
- 3. 标签结束

根据标签 <div v-if="xx"> 的结构,我们的指令处理,应该在 标签名 - 标签结束 中间进行处理:

1. 在 parseTag 增加属性处理逻辑:

```
<> 代码块
 1
     * 解析标签
2
     function parseTag(context: any, type: TagType): any {
        // 属性与指令处理
 8
         advanceSpaces(context)
 9
        let props = parseAttributes(context, type)
10
        // -- 处理标签结束部分 --
11
12
         . . .
13
14
         return {
15
             type: NodeTypes.ELEMENT,
16
            tag,
17
            tagType,
18
            // 属性与指令
19
             props
20
         }
21
     }
```

2. 增加 advanceSpaces 方法, 处理 div v-if 中间的空格:

```
<> 代码块
1
      * 前进非固定步数
2
3
     function advanceSpaces(context: ParserContext): void {
5
         const \ match = /^[\t\n\f] + /.exec(context.source)
6
         if (match) {
7
             advanceBy(context, match[0].length)
8
         }
```

▶ 意见反馈

♡ 收藏教程

□ 标记书签

```
<> 代码块
     * 解析属性与指令
 1
2
     function parseAttributes(context, type) {
3
        // 解析之后的 props 数组
        const props: any = []
       // 属性名数组
        const attributeNames = new Set<string>()
8
        // 循环解析,直到解析到标签结束('>' || '/>')为止
9
        while (
10
            context.source.length > 0 &&
11
            !startsWith(context.source, '>') &&
12
            !startsWith(context.source, '/>')
13
      ) {
14
            // 具体某一条属性的处理
15
           const attr = parseAttribute(context, attributeNames)
16
            // 添加属性
17
           if (type === TagType.Start) {
18
               props.push(attr)
19
20
21
            advanceSpaces(context)
        }
22
2.3
        return props
24
     }
```

4. 创建 parseAttribute , 处理具体的属性:

```
<>代码块
1 /**
   * 处理指定指令,返回指令节点
2
3
    function parseAttribute(context: ParserContext, nameSet: Set<string>) {
4
        // 获取属性名称。例如: v-if
 6
        const match = /^[^\t\r\n\f />][^\t\r\n\f />=]*/.exec(context.source)!
        const name = match[0]
 8
        // 添加当前的处理属性
 9
        nameSet.add(name)
10
11
        advanceBy(context, name.length)
12
13
        // 获取属性值。
14
       let value: any = undefined
15
        // 解析模板,并拿到对应的属性值节点
16
17
        if (/^[\t\r\n\f ]*=/.test(context.source)) {
18
            advanceSpaces(context)
19
            advanceBy(context, 1)
            advanceSpaces(context)
21
            value = parseAttributeValue(context)
22
2.3
       // 针对 v- 的指令处理
24
       if (/^(v-[A-Za-z0-9-]|:|\.|@|#)/.test(name)) {
25
            // 获取指令名称
26
27
            const match =
               /(?:^v-([a-z0-9-]+))?(?:(?::|^\.|^@|^#)(\[[^\]]+\]|[^\.]+))?(.+)?$/i.exe
28
29
                  name
              )!
30
31
32
            // 指令名。v-if 则获取 if
33
            let dirName = match[1]
34
            // TODO: 指令参数 v-bind:arg
35
            // let arg: any
36
            // TODO: 指令修饰符 v-on:click.modifiers
37
            // const modifiers = match[3] ? match[3].slice(1).split('.') : []
38
            return {
                                           口 标记书签
           ▶ 意见反馈
                           ♡ 收藏教程
```

?

 \odot

```
exp: value && {
43
44
                   type: NodeTypes.SIMPLE_EXPRESSION,
45
                    content: value.content,
46
                   isStatic: false,
47
                   loc: value.loc
48
               },
               arg: undefined,
49
               modifiers: undefined,
51
               loc: {}
52
            }
53
      }
54
5.5
      return {
56
            type: NodeTypes.ATTRIBUTE,
57
            name,
58
            value: value && {
59
              type: NodeTypes.TEXT,
60
               content: value.content,
61
              loc: value.loc
62
           },
            loc: {}
63
64
       }
65 }
```

5. 创建 parseAttributeValue 方法处理指令值:

```
<> 代码块
    /**
     * 获取属性 (attr) 的 value
2
3
4
    function parseAttributeValue(context: ParserContext) {
5
       let content = ''
 6
        // 判断是单引号还是双引号
 8
        const quote = context.source[0]
        const isQuoted = quote === `"` || quote === `'`
9
       // 引号处理
10
      if (isQuoted) {
11
12
           advanceBy(context, 1)
           // 获取结束的 index
13
14
         const endIndex = context.source.indexOf(quote)
          // 获取指令的值。例如: v-if="isShow",则值为 isShow
15
16
          if (endIndex === -1) {
17
               content = parseTextData(context, context.source.length)
18
           } else {
19
               content = parseTextData(context, endIndex)
20
                advanceBy(context, 1)
21
22
23
24
        return { content, isQuoted, loc: {} }
25
```

?

0

至此 AST 的处理完成。解析出来的 AST 为:

```
<> 代码块
1 {"type":0,"children":[{"type":1,"tag":"div","tagType":0,"props":[],"children":[{"type":2
```

把当前 AST 替换到 vue 源码中,发现指令可以被正常渲染。

09: 基于编辑器的指令(v-xx)处理: 指令解析... ← 上一节 下一节 → 11: 基于编辑器的指令(v-xx)处理: JavaScri...

✓ 我要提出意见反馈

♪ 意见反馈

♡ 收藏教程

口 标记书签

6 6

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

∷

?

0