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

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 `parseTag` 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

```
/**
 * 解析标签
 */
function parseTag(context: any, type: TagType): any {
    ...

    // 属性与指令处理
    advanceSpaces(context)
    let props = parseAttributes(context, type)

    // -- 处理标签结束部分 --
    ...

    return {
        type: NodeTypes.ELEMENT,
        tag,
        tagType,
        // 属性与指令
        props
    }
}
```

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

代码块

```
/**
 * 前进非固定步数
 */
function advanceSpaces(context: ParserContext): void {
    const match = /^[\t\r\n\f ]+/.exec(context.source)
    if (match) {
        advanceBy(context, match[0].length)
    }
}
```

代码块

```
 * 解析属性与指令
 */
function parseAttributes(context, type) {
    // 解析之后的 props 数组
    const props: any = []
    // 属性名数组
    const attributeNames = new Set<string>()

    // 循环解析，直到解析到标签结束（'>' || '/>'）为止
    while (
        context.source.length > 0 &&
        !startsWith(context.source, '>') &&
        !startsWith(context.source, '/>')
    ) {
        // 具体某一条属性的处理
        const attr = parseAttribute(context, attributeNames)
        // 添加属性
        if (type === TagType.Start) {
            props.push(attr)
        }
        advanceSpaces(context)
    }
    return props
}
```

4. 创建 `parseAttribute` ，处理具体的属性：

代码块

```
/**
 * 处理指定指令，返回指令节点
 */
function parseAttribute(context: ParserContext, nameSet: Set<string>) {
    // 获取属性名称。例如：v-if
    const match = /^[^\t\r\n\f />][^\t\r\n\f />=]*/.exec(context.source)!
    const name = match[0]
    // 添加当前的处理属性
    nameSet.add(name)

    advanceBy(context, name.length)

    // 获取属性值。
    let value: any = undefined

    // 解析模板，并拿到对应的属性值节点
    if (/^[\t\r\n\f ]*=/.test(context.source)) {
        advanceSpaces(context)
        advanceBy(context, 1)
        advanceSpaces(context)
        value = parseAttributeValue(context)
    }

    // 针对 v- 的指令处理
    if (/^(v-[A-Za-z0-9-]|:|\.|@|#)/.test(name)) {
        // 获取指令名称
        const match =
            /(?:^v-([a-z0-9-]+))?(?:(?::|^\.|^@|^#)(\[[^\]]+\]|[^\.]+))?(.+)?$/i.exe
                name
            )!

        // 指令名。v-if 则获取 if
        let dirName = match[1]
        // TODO：指令参数  v-bind:arg
        // let arg: any

        // TODO：指令修饰符  v-on:click.modifiers
        // const modifiers = match[3] ? match[3].slice(1).split('.') : []

        return {
```

```
            exp: value && {
                type: NodeTypes.SIMPLE_EXPRESSION,
                content: value.content,
                isStatic: false,
                loc: value.loc
            },
            arg: undefined,
            modifiers: undefined,
            loc: {}
        }
    }

    return {
        type: NodeTypes.ATTRIBUTE,
        name,
        value: value && {
            type: NodeTypes.TEXT,
            content: value.content,
            loc: value.loc
        },
        loc: {}
    }
}
```

5. 创建 `parseAttributeValue` 方法处理指令值：

代码块

```
/**
 * 获取属性（attr）的 value
 */
function parseAttributeValue(context: ParserContext) {
    let content = ''

    // 判断是单引号还是双引号
    const quote = context.source[0]
    const isQuoted = quote === `"` || quote === `'`
    // 引号处理
    if (isQuoted) {
        advanceBy(context, 1)
        // 获取结束的 index
        const endIndex = context.source.indexOf(quote)
        // 获取指令的值。例如：v-if="isShow"，则值为 isShow
        if (endIndex === -1) {
            content = parseTextData(context, context.source.length)
        } else {
            content = parseTextData(context, endIndex)
            advanceBy(context, 1)
        }
    }

    return { content, isQuoted, loc: {} }
}
```

至此 `AST` 的处理完成。解析出来的 `AST` 为：

代码块

```
{"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　标记书签