Sunday • 更新于 2022-10-19

上一节 02：响应性数据… 04：响应性数据… 下一节

03：响应性数据的编辑器处理：AST 解析逻辑

代码块

```
// 需要新增的 AST 结构
{
  "type": 5, // NodeTypes.INTERPOLATION
  "content": {
    "type": 4, // NodeTypes.SIMPLE_EXPRESSION
    "isStatic": false,
    "constType": 0,
    "content": "msg",
    "loc": {}
  }
}
```

查看 `packages/compiler-core/src/parse.ts` 中的代码逻辑，找到 `parseChildren` 方法。

我们知道该方法的主要是用来解析子节点，内部存在如下的 `if` 逻辑：

代码块

```
    if (startsWith(s, '{{')) {
        ...
    }
```

对于该逻辑而言，它就是用来处理复合表达式的对应逻辑，我们可以在该逻辑中，生成对应的 `node`：

代码块

```
function parseChildren(context: ParserContext, ancestors) {
    ...
    while (!isEnd(context, ancestors)) {
        ...
        if (startsWith(s, '{{')) {
+           node = parseInterpolation(context)
        }
        // < 意味着一个标签的开始
        else if (s[0] === '<') {
            ...
        }
...

    return nodes
}
```

然后增加 `parseInterpolation` 方法：

代码块

```
/**
 * 解析插值表达式 {{ xxx }}
 */
function parseInterpolation(context: ParserContext) {
    // open = {{
    // close = }}
    const [open, close] = ['{{', '}}']
```

```
10
11      // 获取插值表达式中间的值
12      const closeIndex = context.source.indexOf(close, open.length)
13      const preTrimContent = parseTextData(context, closeIndex)
14      const content = preTrimContent.trim()
15
16      advanceBy(context, close.length)
17
18      return {
19          type: NodeTypes.INTERPOLATION,
20          content: {
21              type: NodeTypes.SIMPLE_EXPRESSION,
22              isStatic: false,
23              content
24          }
25      }
26  }
```

至此，我们成功解析了 `AST` 。

打印解析之后的 `AST` 可得：

代码块

```
1   const ast = {
2     type: 0,
3     children: [
4       {
5         type: 1,
6         tag: 'div',
7         tagType: 0,
8         props: [],
9         children: [
10          { type: 2, content: ' hello ' },
11          { type: 5, content: { type: 4, isStatic: false, content: 'msg' } },
12          { type: 2, content: ' ' }
13        ]
14      }
15    ],
16    loc: {}
17  }
```

我们可以把以上代码替换到源码的 `baseCompile` 中，发现可正常渲染。证明我们当前生成的 `AST` 没有问题。

02：响应性数据的编辑器处理：响应性数据... 上一节 下一节 04：响应性数据的编辑器处理：JavaScript ...