Sunday • 更新于 2022-10-19

上一节 04：响应性数据… 06：响应性数据… 下一节

05：响应性数据的编辑器处理：render 转化逻辑分析

代码块

```
1  // render 函数，内容进行了简化
2  const _Vue = Vue
3
4  return function render(_ctx, _cache) {
5  +  with (_ctx) {
6  +    const { toDisplayString: _toDisplayString, createElementBlock: _createElementBlock
7
8  +    return _createElementBlock("div", null, " hello " + _toDisplayString(msg)
9  +  }
10 }
```

那么接下来我们就要处理 `render` 的转化逻辑了。由以上最终生成的方法可知，对于主要增加了以下两块代码：

1. `toDisplayString` 方法：该方法的作用非常简单，接收一个变量，返回对应的响应性数据。比如在以上代码和测试场景中，`_toDisplayString(msg)` 方法的调用代表着接收 `msg` 变量作为参数，返回 `world` 字符串
2. `with (_ctx)`：由刚才的代码我们可知，在使用 `_toDisplayString` 时，我们用到了一个 `msg` 变量。但是在整个的 `render` 代码中却没有 `msg` 变量的存在。那么为什么没有抛出对应的错误呢？这是因为 with 的作用，它会改变语句的作用域链，从而找到 `msg` 变量。

所以根据以上两点，我们在去处理时，就需要关注以下内容：

1. 在 `generate` 方法中，增加 `with` 的 `push` 和 `toDisplayString` 方法的调用
2. 完成 `toDisplayString` 方法
3. 因为 `with` 改变作用域，所以我们在 `runtime` 时，需要注意新的作用域会不会引发其他的错误。

04：响应性数据的编辑器处理：JavaScript … 上一节 下一节 06：响应性数据的编辑器处理：generate 生…

我要提出意见反馈

索引目录

05：响应性数据的