

全部开发者教程

第11章: compiler 编译器 - 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



Sunday • 更新于 2022-10-19

◀ 上一节 05: 响应性数据... 07: 响应性数据... 下一节 ▶

06: 响应性数据的编辑器处理: generate 生成 render 函数

这一小节我们来完成 generate 的函数拼接:

1. 在 generate 方法中, 新增 with 的处理:

<> 代码块

```
1  export function generate(ast) {
2    ...
3
4    // 增加 with 触发 (加到 hasHelpers 之前)
5    push(`with (_ctx) {`)
6    indent()
7
8    // 明确使用到的方法。如: createVNode
9    const hasHelpers = ast.helpers.length > 0
10   ...
11
12   // with 结尾
13   deindent()
14   push(`}`)
15
16   // 收缩缩进 + 换行
17   ...
18
19   return {
20     ast,
21     code: context.code
22   }
23 }
```

2. 在 genNode 中处理其他节点类型:

<> 代码块

```
1  function genNode(node, context) {
2    switch (node.type) {
3      ...
4      // 复合表达式处理
5      case NodeTypes.SIMPLE_EXPRESSION:
6        genExpression(node, context)
7        break
8      // 表达式处理
9      case NodeTypes.INTERPOLATION:
10       genInterpolation(node, context)
11       break
12      // {{}} 处理
13      case NodeTypes.COMPOUND_EXPRESSION:
14        genCompoundExpression(node, context)
15        break
16    }
17  }
```

3. 增加 genExpression 方法, 处理复合表达式

索引目录

06: 响应性数据的



意见反馈

收藏教程

标记书签

<> 代码块

```
1  /**
2   * 复合表达式处理
3   */
4  function genCompoundExpression(node, context) {
5    for (let i = 0; i < node.children!.length; i++) {
6      const child = node.children![i]
7      if (isString(child)) {
8        context.push(child)
9      } else {
10       genNode(child, context)
11     }
12   }
13 }
```

4. 增加 genExpression 方法, 处理 表达式

<> 代码块

```
1  function genExpression(node, context) {
2    const { content, isStatic } = node
3    context.push(isStatic ? JSON.stringify(content) : content, node)
4  }
```

5. 增加 genInterpolation 方法, 处理 {{}}

<> 代码块

```
1  /**
2   * {{}} 处理
3   */
4  function genInterpolation(node, context) {
5    const { push, helper } = context
6    push(`${helper(TO_DISPLAY_STRING)}()`)
7    genNode(node.content, context)
8    push(`${}`)
9  }
```

此时运行测试实例, 可以得到如下 render 函数:

<> 代码块

```
1  const _Vue = Vue
2
3  return function render(_ctx, _cache) {
4    with (_ctx) {
5      const { toDisplayString: _toDisplayString, createElementVNode: _createElementVNode }
6
7
8      return _createElementVNode("div", [], [" hello " + _toDisplayString(msg) + " "])
9    }
10 }
```

05: 响应性数据的编辑器处理: render 转化... < 上一节 下一节 > 07: 响应性数据的编辑器处理: render 函数...

✍ 我要提出意见反馈