Sunday • 更新于 2022-10-19

上一节 06：框架实现：… 08：扩展知识：… 下一节

07：框架实现：生成 AST，构建测试

当 parseChildren 处理完成之后，我们可以到 children，那么最后我们就只需要利用 createRoot 方法，把 children 放到 ROOT 节点之下即可。

1. 创建 createRoot 方法：

代码块

```
/**
 * 生成 root 节点
 */
export function createRoot(children) {
    return {
        type: NodeTypes.ROOT,
        children,
        // loc: 位置，这个属性并不影响渲染，但是它必须存在，否则会报错。所以我们给了他一个
        loc: {}
    }
}
```

2. 在 baseParse 中使用该方法：

代码块

```
/**
 * 基础的 parse 方法，生成 AST
 * @param content tempalte 模板
 * @returns
 */
export function baseParse(content: string) {
    // 创建 parser 对象，未解析器的上下文对象
    const context = createParserContext(content)
    const children = parseChildren(context, [])
    return createRoot(children)
}
```

至此整个 parse 解析流程完成。我们可以在 packages/compiler-core/src/compile.ts 中打印得到的 AST：

代码块

```
export function baseCompile(template: string, options) {
    const ast = baseParse(template)
    console.log(JSON.stringify(ast))

    return {}
}
```

得到的内容为：

代码块

```
{
  "type": 0,
```

```
        "type": 1,
        "tag": "div",
        "tagType": 0,
        "props": [],
        "children": [{ "type": 2, "content": " hello world " }]
      }
    ],
    "loc": {}
  }
```

我们可以把得到的该 AST 放入到 vue 的源码中进行解析，以此来验证是否正确。

在 vue 源码的 packages/compiler-core/src/compile.ts 模块下 baseCompile 方法中：

代码块

```
export function baseCompile(
  template: string | RootNode,
  options: CompilerOptions = {}
): CodegenResult {
  ...

- const ast = isString(template) ? baseParse(template, options) : template
+ const ast = {
+   type: 0,
+   children: [
+     {
+       type: 1,
+       tag: 'div',
+       tagType: 0,
+       props: [],
+       children: [{ type: 2, content: ' hello world ' }]
+     }
+   ],
+   loc: {}
+ }

  ...
}
```

运行源码的 compile 方法，浏览器中应该可以渲染 hello world：

代码块

```
<script>
  const { compile, h, render } = Vue
  // 创建 template
  const template = ``

  // 生成 render 函数
  const renderFn = compile(template)

  // 创建组件
  const component = {
    render: renderFn
  }

  // 通过 h 函数，生成 vnode
  const vnode = h(component)

  // 通过 render 函数渲染组件
  render(vnode, document.querySelector('#app'))
</script>
```

成功运行，标记着我们的 AST 处理完成。