☰ 逐行剖析 Vue.js 源码

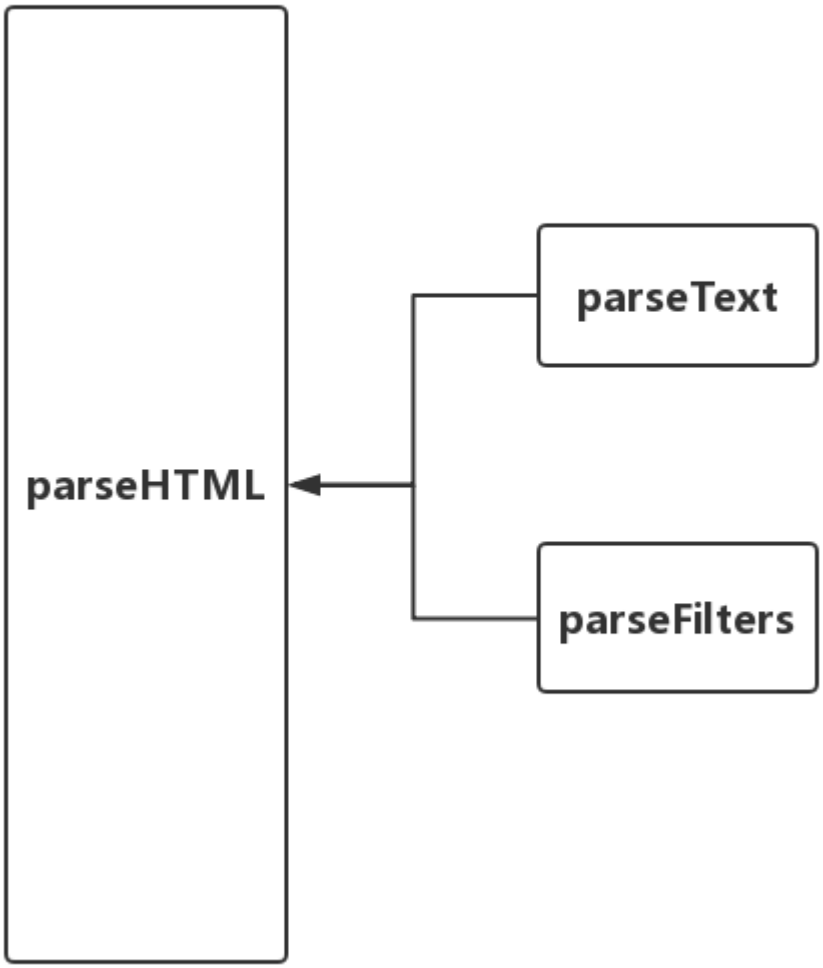
1. 整体流程

上篇文章中我们说了，在模板解析阶段主要做的工作是把用户在 `<template></template>` 标签内写的模板使用正则等方式解析成抽象语法树（ `AST` ）。而这一阶段在源码中对应解析器（ `parser` ）模块。

解析器，顾名思义，就是把用户所写的模板根据一定的解析规则解析出有效的信息，最后用这些信息形成 `AST` 。我们知道在 `<template></template>` 模板内，除了有常规的 `HTML` 标签外，用户还会一些文本信息以及在文本信息中包含过滤器。而这些不同的内容在解析起来肯定需要不同的解析规则，所以解析器不可能只有一个，它应该除了有解析常规 `HTML` 的HTML解析器，还应该有解析文本的文本解析器以及解析文本中如果包含过滤器的过滤器解析器。

另外，文本信息和标签属性信息却又是存在于HTML标签之内的，所以在解析整个模板的时候它的流程应该是这样子的：HTML解析器是主线，先用HTML解析器进行解析整个模板，在解析过程中如果碰到文本内容，那就调用文本解析器来解析文本，如果碰到文本中包含过滤器那就调用过滤器解析器来解析。如下图所示：

☰ 逐行剖析 Vue.js 源码



2. 回到源码

解析器的源码位于 `/src/complier/parser` 文件夹下，其主线代码如下：

```js
// 代码位置：/src/complier/parser/index.js

/**
 * Convert HTML string to AST.
 */
export function parse(template, options) {
   // ...
  parseHTML(template, {
    warn,
    expectHTML: options.expectHTML,
    isUnaryTag: options.isUnaryTag,
    canBeLeftOpenTag: options.canBeLeftOpenTag,
    shouldDecodeNewlines: options.shouldDecodeNewlines,
    shouldDecodeNewlinesForHref: options.shouldDecodeNewlinesForHref,
```

```
    },
    end () {

    },
    chars (text: string) {

    },
    comment (text: string) {

    }
  })
  return root
}
```

从上面代码中可以看到， `parse` 函数就是解析器的主函数，在 `parse` 函数内调用了 `parseHTML` 函数对模板字符串进行解析，在 `parseHTML` 函数解析模板字符串的过程中，如果遇到文本信息，就会调用文本解析器 `parseText` 函数进行文本解析；如果遇到文本中包含过滤器，就会调用过滤器解析器 `parseFilters` 函数进行解析。

3. 总结

本篇文章主要梳理了模板解析的整体运行流程，模板解析其实就是根据被解析内容的特点使用正则等方式将有效信息解析提取出来，根据解析内容的不同分为HTML解析器，文本解析器和过滤器解析器。而文本信息与过滤器信息又存在于HTML标签中，所以在解析器主线函数 `parse` 中先调用HTML解析器 `parseHTML` 函数对模板字符串进行解析，如果在解析过程中遇到文本或过滤器信息则再调用相应的解析器进行解析，最终完成对整个模板字符串的解析。

了解了模板解析阶段的整体运行流程后，接下来，我们就对流程中所涉及到的三种解析器分别深入分析，逐个击破。

← 综述 模板解析阶段(HTML解析器) →