Sunday • 更新于 2022-10-19

上一节 25：框架实现：... 01：前言 下一节

26：总结

在本章中，我们主要讲解了 `render` 函数的渲染机制。

我们首先讲解了渲染器 `renderer` 的概念，并且知道 `render` 函数是渲染器对象中的一个函数。但是这个渲染函数我们是不可以直接被用户访问的。所以我们额外在 `packages/runtime-dom/src/index.ts` 中构建了一个 `render` 函数，用来给用户使用 `render` 渲染器 。

在整个的 `render` 渲染过程中，我们把所有的核心逻辑放入到了 `runtime-core` 文件夹内，把所有和浏览器相关的 `API` 操作，放到了 `runtime-dom` 文件夹下。以此来达到兼容性的目的。

针对不同的节点，具体的渲染逻辑会有所不同。我们分别从：

1. `ELEMENT`
2. `TEXT`
3. `COMMENT`

这三种节点类型入手，讲解了 挂载、更新、卸载 的对应操作逻辑。

对于这三种节点而言， `ELEMENT` 毫无因为是最复杂的。

对于 `ELEMENT` 而言，它的渲染除了本身的 挂载、更新、卸载 之外，还包含了 `props` 属性 的挂载、更新、卸载操作。

对于 `props` 属性而言，因为 `HTML Attributes` 和 `DOM Properties` 的原因，所以我们需要针对不同的 `props` 分别进行处理。

而对于 `event` 事件而言，这里 `vue` 通过了 `vei` 的方式进行了更新逻辑，这个设计是非常精妙的。

但是哪怕我们做了这么多事情之后，针对于 `render` 的渲染逻辑，我们依然只是处理了冰山一角而已，比如：

1. `diff`：在旧节点和新节点的子节点都为 `Array` 时，我们需要就行 `diff` 运算，已完成高效的更新。
2. `component` 组件：组件的处理也是 `vue` 中非常重要的一块内容，这个也需要我们进行单独的讲解。

25：框架实现：renderer渲染器下， Fragm... 上一节 下一节 01：前言

我要提出意见反馈

索引目录

26：总结