

全部开发者教程

第八章: runtime 运行时 - 运行时核心设计原则

01: 前言

02: HTML DOM 节点树与虚拟 DOM 树

03: 挂载与更新

04: h 函数 与 render 函数

05: 运行时核心设计原则

06: 总结

第九章: runtime 运行时 - 构建 h 函数, 生成 Vnode

01: 前言

02: 阅读源码: 初见 h 函数, 跟踪 Vue 3 源码实现基础逻辑

03: 框架实现: 构建 h 函数, 处理 ELEMENT + TEXT_CHILDREN 场景

04: 源码阅读: h 函数, 跟踪 ELEMENT +



Sunday • 更新于 2022-10-19

上一节 03: 挂载与更新 05: 运行时核心... 下一节

04: h 函数 与 render 函数

不知道大家还记不记得，我们在 第二章 讲解 运行时的概念时，曾经使用过这样的案例：

<> 代码块

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <meta charset="UTF-8">
6    <title>Document</title>
7    <script src="https://unpkg.com/vue@3.2.36/dist/vue.global.js"></script>
8  </head>
9
10 <body>
11   <div id="app"></div>
12 </body>
13
14 <script>
15   const { render, h } = Vue
16   // 生成 VNode
17   const vnode = h('div', {
18     class: 'test'
19   }, 'hello render')
20
21   // 承载的容器
22   const container = document.querySelector('#app')
23
24   // 渲染函数
25   render(vnode, container)
26 </script>
27
28 </html>
```

当时，我们说：“有些同学可能看不懂当前代码是什么意思，没有关系，这不重要，后面我们会详细去讲”，那么现在就是讲解这个时候了。

根据前两个小节介绍其实我们已经知道了，vue 的渲染分为：挂载和更新。两个步骤。

而无论是挂载还是更新，都是借助 VNode 来进行实现的。

在以上代码中，我们知道主要涉及到了两个函数：

1. h 函数
2. render 函数

那么下面我们一个一个来去说

h 函数

根据以上代码可知，我们可以通过 h 函数 得到一个 vnode：

<> 代码块

```
1  const vnode = h('div', {
```

索引目录

- 04: h 函数 与 re
- h 函数
- render 函数
- 总结



打印当前的 `vnode`，可以得到一下内容：

<> 代码块

```
1  {
2    "__v_isVNode": true,
3    "__v_skip": true,
4    "type": "div",
5    "props": { "class": "test" },
6    "key": null,
7    "ref": null,
8    "scopeId": null,
9    "slotScopeIds": null,
10   "children": "hello render",
11   "component": null,
12   "suspense": null,
13   "ssContent": null,
14   "ssFallback": null,
15   "dirs": null,
16   "transition": null,
17   "el": null,
18   "anchor": null,
19   "target": null,
20   "targetAnchor": null,
21   "staticCount": 0,
22   "shapeFlag": 9,
23   "patchFlag": 0,
24   "dynamicProps": null,
25   "dynamicChildren": null,
26   "appContext": null
27 }
```

以上内容，我们剔除掉无用的内容之后，得到一个精简的 `json`：

<> 代码块

```
1  {
2    // 是否是一个 VNode 对象
3    "__v_isVNode": true,
4    // 当前节点类型
5    "type": "div",
6    // 当前节点的属性
7    "props": { "class": "test" }
8    // 它的子节点
9    "children": "hello render"
10 }
```

那么由此可知 `h` 函数本质上其实就是一个：**用来生成 `VNode` 的函数**。

`h` 函数 最多可以接收三个参数：

1. `type: string | Component`：既可以是一个字符串 (用于原生元素) 也可以是一个 Vue 组件定义。
2. `props?: object | null`：要传递的 prop
3. `children?: Children | Slot | Slots`：子节点。

官方示例描述了它的详细使用方式，这个就不写在文档中了。

render 函数

那么在了解了 `h` 函数的作用之后，下面我们来看 `render` 函数。

<> 代码块

```
1  render(vnode, container)
```

从以上代码中我们可知，`render` 函数主要接收了两个参数：

1. `vnode`：虚拟节点树 或者叫做 虚拟 DOM 树，两种叫法皆可

[意见反馈](#)

[收藏教程](#)

[标记书签](#)



通过 `render` 函数，我们可以：**使用编程式地方式，创建虚拟 DOM 树对应的真实 DOM 树，到指定位置。**

总结

这小节，我们知道了 `h` 函数和 `render` 函数的作用，这两个函数是整个运行时的关键函数，后面我们实现运行时的代码，核心就是实现这两个函数。

03: 挂载与更新 ◀ 上一节 下一节 ▶ 05: 运行时核心设计原则

 我要提出意见反馈

[企业服务](#) [网站地图](#) [网站首页](#) [关于我们](#) [联系我们](#) [讲师招募](#) [帮助中心](#) [意见反馈](#) [代码托管](#)

Copyright © 2022 imooc.com All Rights Reserved | 京ICP备 12003892号-11 京公网安备11010802030151号



 意见反馈

 收藏教程

 标记书签