我的课程



Q

从所有教程的词条中查询…

首页 > 慕课教程 > Vue3源码分析与构建方案 > 03: 创建测试实例: 在 Vue 源码中运行测试实例

全部开发者教程 \equiv

13: 扩展: 所谓良好的 `TypeScript` 支持,是如何提

供的?

14: 总结

第三章: Vue 3源码结构 -搭建框架雏形

01: 前言

02: 探索源码设计: Vue3 源 码设计大解析

03: 创建测试实例: 在 Vue 源 码中运行测试实例

04: 跟踪解析运行行为: 为 vue 开启 SourceMap

05: 授人以鱼: 如何针对源码

进行 debugger

06: 授人以渔: 如何阅读源码

07: 开始搭建自己的框架: 创 建 vue-next-mini

08: 为框架进行配置: 导入 ts

Sunday • 更新于 2022-10-19

◆ 上一节 02: 探索源码设...

04: 跟踪解析运... 下一节 ▶

索引目录

03: 创建测试实例

运行 vue3 源代 运行测试实例

?

 \odot

03: 创建测试实例: 在 Vue 源码中运行测试实例

那么现在我们已经大致的了解了 vue 源代码的基本结构,那么接下来我们来看一下,如何在 Vue3 的代 码中运行测试实例,并进行 debugger

运行 vue3 源代码

- 1. 因为 vue3 是通过 pnpm 作为包管理工具的,所以想要运行 vue3 那么首先需要 安装 pnpm
- 2. 我们可以直接通过如下指令安装 pnpm

```
<> 代码块
    npm install -g pnpm
```

- 3. pnpm 会通过一个 集中管理 的方式来管理 电脑中所有项目 的依赖包,以达到 节约电脑磁盘 的目 的, 具体可点击 这里 查看
- 4. 安装完 pnpm 之后,接下来就可以在 项目根目录 下通过:

```
<> 代码块
    pnpm i
```

的形式安装依赖

5. 等待所有的依赖包安装完成之后, 执行:

```
<> 代码块
    npm run build
```

进行项目打包

6. 执行完成之后,可以发现在 packages/vue/dist 下,将生成如下文件:

```
<> 代码块
1
     dist
     ├─ vue.cjs.js
     ├─ vue.cjs.prod.js
     ├─ vue.esm-browser.js
     — vue.esm-browser.prod.js
     — vue.esm-bundler.js
     ├─ vue.global.js
8
     ├─ vue.global.prod.js
9
     vue.runtime.esm-browser.js
     ├─ vue.runtime.esm-browser.prod.js
10
11
     — vue.runtime.esm-bundler.js
12
     ├─ vue.runtime.global.js
     └─ vue.runtime.global.prod.js
```

7. 那么至此,项目打包完成

运行测试实例



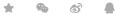
- 1. 创建 packages/vue/examples/imooc 文件夹
- 2. 在该文件夹中创建第一个测试实例: reactive.html , 并写入如下代码 (这些代码可能大家会不太理解 , 没有关系! 后面会进行详细讲解 , 这里只是为了测试 vue 代码的 debugger) :

```
<> 代码块
    <!DOCTYPE html>
1
    <html lang="en">
    <head>
      <meta charset="UTF-8">
     <meta http-equiv="X-UA-Compatible" content="IE=edge">
 6
     <meta name="viewport" content="width=device-width, initial-scale=1.0">
     <title>Document</title>
    <script src="../../dist/vue.global.js"></script>
10 </head>
11
12
    <body>
      <div id="app"></div>
13
    </body>
14
15
    <script>
16
     // 从 Vue 中结构出 reactie、effect 方法
17
      const { reactive, effect } = Vue
18
     // 声明响应式数据 obj
19
    const obj = reactive({
20
      name: '张三'
21
22
    })
23
    // 调用 effect 方法
24
25
    effect(() => {
2.6
      document.querySelector('#app').innerText = obj.name
27
28
29
      // 定时修改数据,视图发生变化
30
      setTimeout(() => {
      obj.name = '李四'
31
     }, 2000);
32
33
34
    </script>
35
    </html>
```

- 3. 注意: 该代码不可以直接放入到浏览器中运行。
- 4. 如果想要运行改代码,那么我们需要启动一个服务
- 5. 大家可以在 VSCode 中安装 Live Server 的插件,该插件可以帮助我们直接启动一个 本地服务
- 6. 安装完成之后,直接在 html 文件上:右键 -> open with Live Server 即可:
- 7. 此时,可以发现 等待两秒钟之后, 展示的文本由张三变为李四

那么至此我们已经成功了运行了一个测试的实例

▶ 我要提出意见反馈



?

 \odot