实现渲染器 render

前面我们说过渲染器的作用像一台机器：将传入组件转换为 dom。

至于怎么转换，可以提供不同平台的实现，这样就实现了跨平台。

我们来看一下如何实现?

整体思路

- 创建一个渲染器工厂 createRenderer()用于返回渲染器实例
- 渲染器实例提供 render()和 createApp()

代码编写

先搭个架子：

```JavaScript
// mini-vue/index.js
// customRenderer api
export function createRenderer(options) {
  // 负责渲染组件内容
  const render = () => {}

  return {
    render,
    // 创建createApp供用户调用
    createApp: createAppAPI(render)
  }
}

// createApp工厂，传入render，这样createApp不需要关心渲染细节
export function createAppAPI(render) {
  return function createApp() {}
}
```

完成 createAppAPI，只需要调用 render

```JavaScript
export function createAppAPI(render) {
  return function createApp(rootComponent) {
    // 之前的app移过来，但是不需要处理渲染逻辑
    const app = {
      mount(container) {
        render(rootComponent, selector);
      },
    };
    return app
  };
}
```

完成 render，实际上是之前的 mount 做的事情，但是不能直接调用 document 和 dom，否则就和浏览器平台产生耦合，失去了封装的意义。

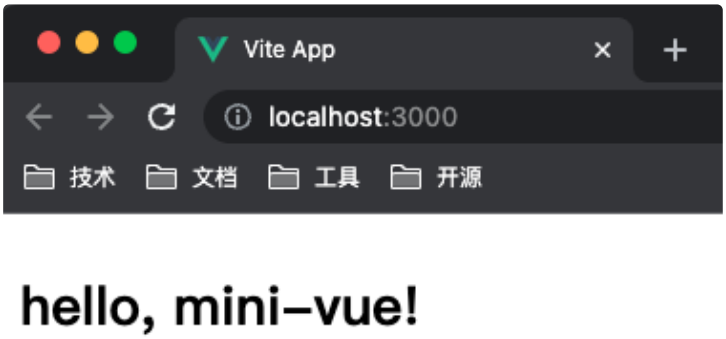
```JavaScript
const render = (rootComponent, selector) => {
  // 替换为options.querySelector
  const container = options.querySelector(selector);
  const el = rootComponent.render.call(rootComponent.data());
  // 替换为options.insert, 同时具备insertBefore和append功能
  // parent.appendChild(el);
  insert(el, container)
};
```

下面完成渲染器创建和 createApp 改写

```JavaScript
// 单例的渲染器
let renderer;
// 渲染器选项：dom节点操作、属性操作
const rendererOptions = {
```

```
  querySelector(selector) {
    return document.querySelector(selector)
  },
  insert(child, parent, anchor) {
    parent.insertBefore(child, anchor || null)
  }
};
function ensureRenderer() {
  // 确保平台仅有一个渲染器实例
  return renderer || (renderer = createRenderer(rendererOptions))
}
export function createApp(rootComponent) {
  // 之前的createApp只需要获取renderer并调用其createApp即可
  return ensureRenderer().createApp(rootComponent);
}
```

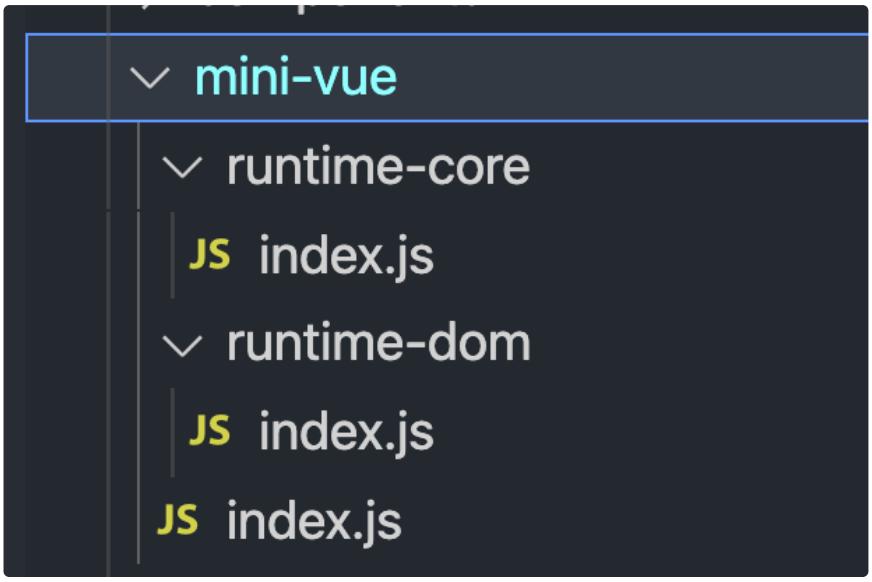
完成，测试效果！



拆分代码模块

通过阅读源码，我们知道设计渲染器的代码分布在 runtime-core 和 runtime-dom 两个模块，我们也将 mini-vue 做同样的拆分便于维护。

调整之后目录结构



mini-vue/index.js

```JavaScript
export * from './runtime-dom'
```

runtime-core/index.js

```JavaScript
export function createRenderer(options) {}

export function createAppAPI(render) {}
```

runtime-dom/index.js

```JavaScript
// 导入依赖createRenderer
import { createRenderer } from "../runtime-core";

let renderer;
const rendererOptions = {};
function ensureRenderer() {}
export function createApp(rootComponent) {}
```

三个模块的关系：