React.js 小书

<-- 返回首页

动手实现 Redux（三）：纯函数（Pure Function）简介

- 作者：胡子大哈
- 原文链接： http://huziketang.com/books/react/lesson32

（本文未审核）

我们接下来会继续优化我们的 `createStore` 的模式，让它使我们的应用程序获得更好的性能。

但在开始之前，我们先用一节的课程来介绍一下一个函数式编程里面非常重要的概念 —— 纯函数（Pure Function）。

简单来说，**一个函数的返回结果只依赖于它的参数，并且在执行过程里面没有副作用，我们就把这个函数叫做纯函数**。这么说肯定比较抽象，我们把它掰开来看：

1. 函数的返回结果只依赖于它的参数。
2. 函数执行过程里面没有副作用。

函数的返回结果只依赖于它的参数

```
const a = 1
const foo = (b) => a + b
foo(2) // => 3
```

`foo` 函数不是一个纯函数，因为它返回的结果依赖于外部变量 `a`，我们在不知道 `a` 的值的情况下，并不能保证 `foo(2)` 的返回值是 3。虽然 `foo` 函数的代码实现并没有变化，传入的参数也没有变化，但它的返回值却是**不可预料**的，现在 `foo(2)` 是 3，可能过了一会就是 4 了，因为 a 可能发生了变化变成了 2。

```
const a = 1
const foo = (x, b) => x + b
foo(1, 2) // => 3
```

现在 `foo` 的返回结果只依赖于它的参数 `x` 和 `b`，`foo(1, 2)` 永远是 3。今天是 3，明天也是 3，在服务器跑是 3，在客户端跑也 3，不管你外部发生了什么变化，`foo(1, 2)` 永远是 3。只要 `foo` 代码不改变，你传入的参数是确定的，那么 `foo(1, 2)` 的值永远是**可预料的**。

这就是纯函数的第一个条件：**一个函数的返回结果只依赖于它的参数**。

函数执行过程没有副作用

一个函数执行过程对产生了**外部可观察的变化**那么就说这个函数是有副作用的。

我们修改一下 `foo`：

```
const a = 1
const foo = (obj, b) => {
  return obj.x + b
}
const counter = { x: 1 }
foo(counter, 2) // => 3
counter.x // => 1
```

我们把原来的 `x` 换成了 `obj`，我现在可以往里面传一个对象进行计算，计算的过程里面并不会对传入的对象进行修改，计算前后的 `counter` 不会发生任何变化，计算前是 1，计算后也是 1，它现在是纯的。但是我再稍微修改一下它：

```
const a = 1
const foo = (obj, b) => {
  obj.x = 2
  return obj.x + b
}
const counter = { x: 1 }
foo(counter, 2) // => 4
counter.x // => 2
```

现在情况发生了变化，我在 `foo` 内部加了一句 `obj.x = 2`，计算前 `counter.x` 是 1，但是计算以后 `counter.x` 是 2。`foo` 函数的执行对外部的 `counter` 产生了影响，它产生了**副作用**，因为它修改了外部传进来的对象，现在它是不纯的。

但是你在函数内部构建的变量，然后进行数据的修改不是副作用：

```
const foo = (b) => {
  const obj = { x: 1 }
  obj.x = 2
  return obj.x + b
}
```

虽然 `foo` 函数内部修改了 obj，但是 `obj` 是内部变量，外部程序根本观察不到，修改 `obj` 并不会产生外部可观察的变化，这个函数是没有副作用的，因此它是一个纯函数。

除了修改外部的变量，一个函数在执行过程中还有很多方式产生**外部可观察的变化**，比如说调用 DOM API 修改页面，或者你发送了 Ajax 请求，还有调用 `window.reload`

刷新浏览器，甚至是 `console.log` 往控制台打印数据也是副作用。

纯函数很严格，也就是说你几乎除了计算数据以外什么都不能干，计算的时候还不能依赖除了函数参数以外的数据。

总结

一个函数的返回结果只依赖于它的参数，并且在执行过程里面没有副作用，我们就把这个函数叫做纯函数。

为什么要煞费苦心地构建纯函数？因为纯函数非常“靠谱”，执行一个纯函数你不用担心它会干什么坏事，它不会产生不可预料的行为，也不会对外部产生影响。不管何时何地，你给它什么它就会乖乖地吐出什么。如果你的应用程序大多数函数都是由纯函数组成，那么你的程序测试、调试起来会非常方便。

> 因为第三方评论工具有问题，对本章节有任何疑问的朋友可以移步到 React.js 小书的论坛 发帖，我会回答大家的疑问。

下一节：动手实现 Redux（四）：共享结构的对象提高性能

上一节：动手实现 Redux（二）：抽离 store 和监控数据变化