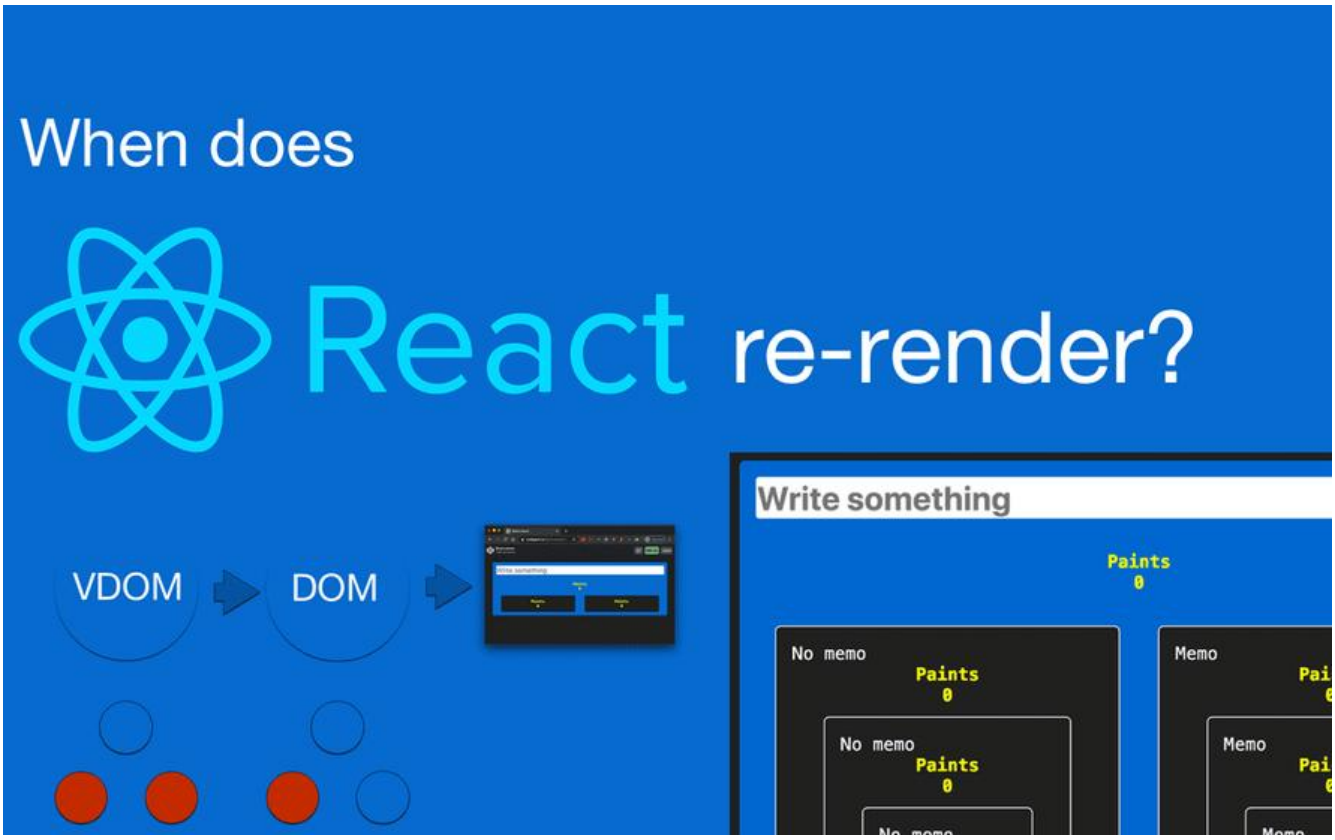
链滴

React 什么时候重新渲染组件?

作者：LiuKang
原文链接：https://ld246.com/article/1588752724523
来源网站：链滴



React仅通过更新已更改的UI部分来提供快速的用户体验而闻名。

在查看React的渲染性能时，有一些术语和概念可能很难理解。在相当长的一段时间内，我还不清楚10%是什么VDOM或React如何决定重新渲染组件。

在这篇文章的第一部分，我会向你解释的**有关呈现最重要的概念作出反应和 如何反应决定重新渲染一给定的组件。**

在第二部分中，我将向您展示如何**优化React应用程序的渲染性能。**

如果在阅读此书后仍有疑问或发现错误，请随时通过电子邮件或Twitter与我联系。

目录

在React中渲染

什么是渲染?

如果我们想了解React渲染和重新渲染的工作方式，那么最好了解库幕后发生的事情。

渲染是可以在不同抽象级别上理解的术语。根据上下文，它的含义略有不同。无论如何，最终**它描述生成图像的过程。**

首先，我们需要了解什么是DOM（文档对象模型）：

“ W3C文档对象模型（DOM）是一种平台和语言无关的接口，它允许程序和脚本动态访问和更新文的内容，结构和样式。”

用简单的英语来说，这意味着**DOM代表您在**打开网站时**在屏幕上看到的内容**，并通过标记语言HTM表示。JavaScript还具有一个DOM，它表示为一个对象，其中根元素为 document。

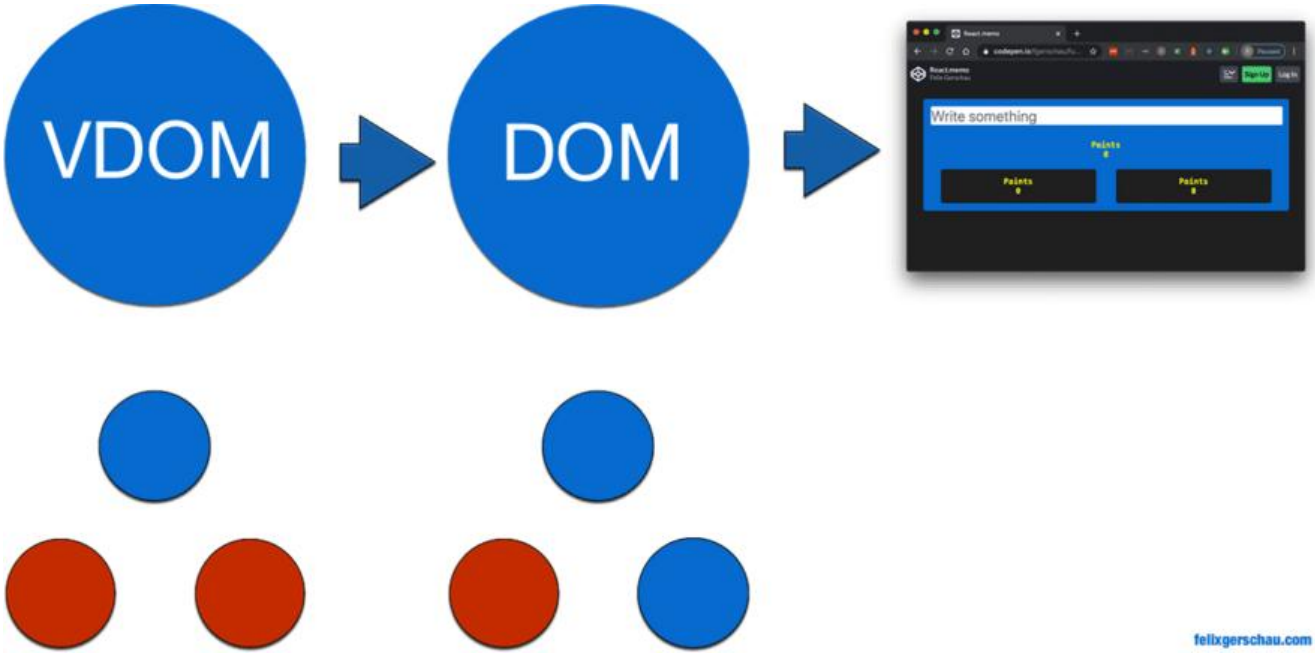
您可以通过DOM编程界面使用JavaScript修改DOM，该界面包含 document.write，Node.appendhild或等功能 Element.setAttribute。

什么是VDOM?

然后，我们有了React 的**虚拟DOM**（或VDOM），它是其之上的另一个抽象层。它由您的React应程序的元素组成。

您应用程序中的状态更改将首先应用于VDOM。如果VDOM的新状态需要更改UI，ReactDOM库将过尝试仅更新需要更新的内容来有效地做到这一点。

例如，如果仅元素的属性发生变化，React将仅通过调用 document.setAttribute（或类似方法）来新HTML元素的属性。



红点表示DOM树的更新。

更新VDOM不一定会触发真实DOM的更新。

原文链接：React 什么时候重新渲染组件?

更新VDOM后，React会将其与VDOM 的先前快照进行比较，然后仅更新实际DOM中已更改的内容如果没有任何变化，则完全不会更新真实的DOM。 **比较旧的VDOM和新的VDOM的过程称为diffin**
。

实际的DOM更新速度很慢，因为它们会导致UI的实际重绘。React通过更新实际DOM中可能的最小量来提高效率。

因此，我们必须意识到本地和虚拟DOM更新之间的区别。

在React的关于对帐的文档中阅读有关此工作原理的更多信息。

这对性能意味着什么?

当我们在React中谈论渲染时，实际上是在谈论**render函数的执行**，这**并不总是意味着UI的更新。**

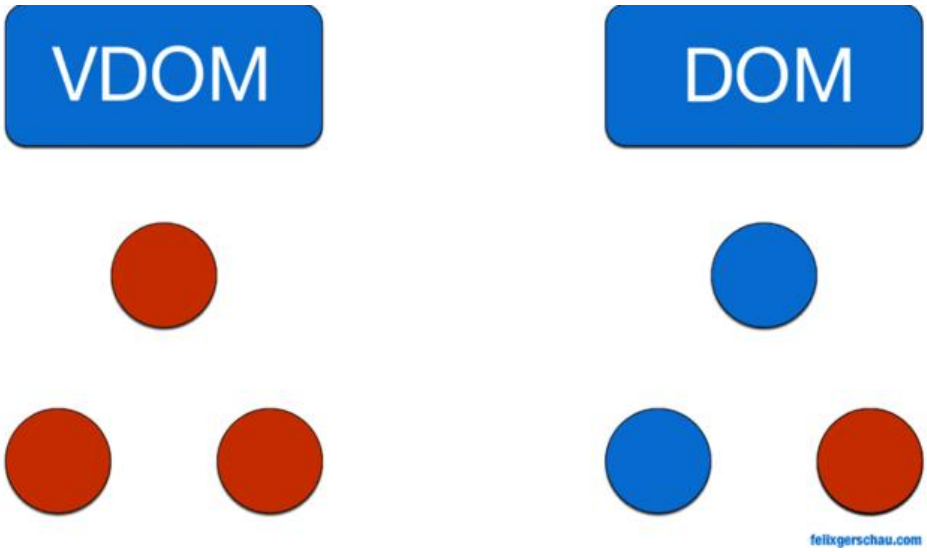
我们来看一个例子：

```
const App = () => {
  const [message, setMessage] = React.useState('');
  return (
    <>
      <Tile message={message} />
      <Tile />
    </>
  );
};
```

在函数组件中，整个函数的执行与类组件中的render函数等效。

当父组件的状态发生变化（在本例中为 App）时，这两个 Tile组件将重新渲染，即使第二个组件甚至有收到任何道具。

这意味着该 render函数被调用了3次，但是实际的DOM修改只在 Tile显示消息的组件中发生一次：



红点再次代表渲染。

原文链接：React 什么时候重新渲染组件?

在React中，这意味着调用render函数，在真实的DOM中，这意味着重新绘制UI。

好消息是您不必担心UI重绘的性能瓶颈。React已经为您优化了。

坏消息是：**左侧所有这些红点表示这些组件的渲染功能已执行。**

这些渲染函数的执行有两个缺点：

1. React必须在每个组件上运行其差异算法，以检查是否应更新UI。
2. 这些渲染函数或函数组件中的所有代码将再次执行。

可以说，第一点并不那么重要，因为React可以非常有效地计算出差异。危险在于您编写的代码在每个eact渲染上都会**一遍又一遍地执行。**

在上面的示例中，我们有一个非常小的组件树。但是想像一下，如果每个节点有更多的子代，而这些代又可能具有子代成分，会发生什么。我们将看到如何优化这一点

React什么时候重新渲染?

上面我们看到了导致用户界面重新绘制的原因，但是从什么开始调用React的render函数呢?

每当**组件状态**改变时，React都会调度渲染。

计划渲染意味着这不会立即发生。React将尝试为此找到最佳时机。

更改**状态**意味着当我们调用 setState函数时，React触发更新（在React Hooks中，您可以从获得此fnseState）。这不仅意味着将调用组件的render函数，而且还意味着 **将重新渲染其所有后续子组件，论其prop是否已更改。**

如果您的应用程序结构不良，那么您运行的JavaScript可能比您预期的要多得多，因为更新父节点意着要运行**所有子代**的 render功能。

在本文的下一部分，我们将看到一些技巧，可以帮助您**避免**这种开销。

如何优化重新渲染

低效重渲染的一个很好的例子是在高阶组件中控制输入字段时。

黄色数字render表示每个组件的功能已执行的次数：

即使我们仅更新了蓝色组件的状态，也触发了更多其他组件的渲染。

控制何时更新组件

React为我们提供了一些功能来防止这些不必要的更新。

让我们来看看它们，之后，我将向您展示另一种更有效的改善渲染性能的方法。

React.memo

我以前已经放弃的第一个是 React.memo。我已经写了一篇更深入的文章，但总而言之，它是一个功，**可以防止在道具不变时渲染React Hook组件。**

一个实际的例子如下：

```
const TileMemo = React.memo(({ children }) => {
  let updates = React.useRef(0);
  return (
    <div className="black-tile">
      Memo
      <Updates updates={updates.current++} />
      {children}
    </div>
  );
});
```

React类的等效项是使用 React.PureComponent。

shouldComponentUpdate

这个功能是React生命周期的功能之一，它允许我们通过告诉React什么时候更新类组件来优化渲染性。

它的参数是组件的下一个props和下一个状态：

```
shouldComponentUpdate(nextProps, nextState) {
// return true or false
}
```

这个函数很容易使用：返回 true会导致React调用render函数，返回会 false阻止这种情况。

设置键属性

在React中，执行以下操作非常普遍。找出问题所在：

```
<div>
  {
    events.map(event =>
      <Event event={event} />
    )
  }
</div>
```

在这里，我忘记了设置 key属性。大多数短毛绒都会对此发出警告，但是为什么它如此重要？

在某些情况下，React依靠 key属性来**识别组件并优化性能。**

在上面的示例中，如果将事件添加到数组的开头，React会认为**第一个**元素和所有后续元素均已更改并将触发这些元素的重新渲染。我们可以通过向元素添加键来防止这种情况：

```jsx
<div>
  {
    events.map(event =>
      <Event event={event} key={event.id} />
    )
  }
</div>
```

尽量避免将数组的索引用作键，并使用一些可标识内容的东西。

密钥仅在兄弟姐妹之间必须唯一。

组件的结构

改进重渲染的更好方法是对代码进行一些重组。

注意放置逻辑的位置。如果将所有内容都放在应用程序的根组件中 React.memo，那么世界上的所有能都将无法帮助您解决性能问题。

如果您将其放置在距离数据使用位置较近的位置，则可能甚至不需要 React.memo。

查看示例的优化版本并输入一些文本：

原文链接：React 什么时候重新渲染组件?

您会看到，即使状态更新，**其他组件也不会重新渲染。**

我所做的唯一更改是**将处理状态的代码移到单独的组件中：**

```
const InputSelfHandling = () => {
  const [text, setText] = React.useState('');
  return (
    <input
      value={text}
      placeholder="Write something"
      onChange={(e) => setText(e.target.value)}
    />
  );
};
```

如果您需要在应用程序的其他部分中使用状态，则可以通过使用React Context或MobX和Redux等代方法来使用。

结论

我希望更好地了解React的渲染机制如何工作以及如何才能充分利用这一点。对于本文，我必须对该题进行一些其他研究，以更好地了解React的渲染方式。
lush□blush

参考

原文摘自：When does React re-render components?

感谢观看，一起遨游知识的海洋□smiley□
miley