列表key与diffing学习总结

Created and last modified by 王文磊 on Mar 10, 2022

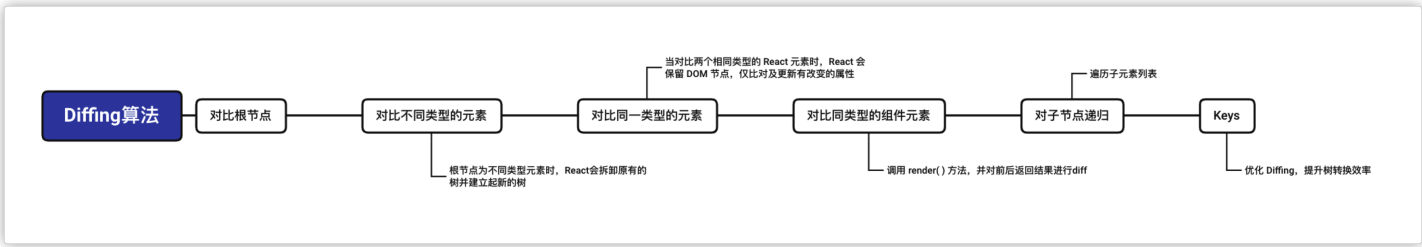
一、Diffing算法

背景：

不同状态下React的render( )方法会返回不同的React元素树，React需要基于这两颗树之间的差别来判断如何高效的更新UI，以保证当前UI与最新的树保持同步

使用常规的算法将一棵树转化为另一棵树时间复杂度为O(n³)，n为树中元素的数量，若有1000个元素则需要10亿次的比较，这个开销实在太过高昂，于是React在以下两个假设的基础上提出了一套O(n)的启发式算法：

1. 两个不同类型的元素会产生出不同的树；
2. 开发者可以通过设置 key 属性，来告知渲染哪些子元素在不同的渲染下可以保存不变；



对比不同类型的元素：

根节点为不同类型元素时，React会拆卸原有的树并建立起新的树。

对应的DOM节点会被销毁，组件实例将执行 componentWillUnmount( ) 方法。当建立一颗新的树时，对应的DOM节点会被创建以及插入到DOM中。组件实例将执行UNSAFE_componentWillMount( ) 方法，紧接着componentDidMount( ) 方法。所有与之前的树相关联的state也会被销毁。

对比同类型的组件元素：

组件更新时，组件实例会保持不变，React得以在不同渲染时保持state一致。

React将更新该组件的props以确保与最新元素一致，并调用该实例相应的生命周期方法。

最后调用 render( ) 方法，并对前后返回元素进行diff。

对子节点递归：

当递归DOM节点的子元素时，React会同时遍历两个子元素的列表，当产生差异时，生成一个mutation。

需要注意的是，当我们没有为列表中的元素绑定key时，在子元素末尾新增元素，更新开销比较小：

```
<ul>
  <li>first</li>
  <li>second</li>
</ul>

<ul>
  <li>first</li>
  <li>second</li>
  <li>third</li>
</ul>
```

React 会先匹配两个 `<li>first</li>` 对应的树，然后匹配第二个元素 `<li>second</li>` 对应的树，最后插入第三个元素的 `<li>third</li>` 树。

如果只是简单的将新增元素插入到表头，那么更新开销会比较大：

```
<ul>
  <li>Duke</li>
  <li>Villanova</li>
</ul>

<ul>
  <li>Connecticut</li>
  <li>Duke</li>
  <li>Villanova</li>
</ul>
```

React 并不会意识到应该保留 `<li>Duke</li>` 和 `<li>Villanova</li>`，而是会重建每一个子元素。这种情况会带来性能问题。

二、Key

为了解决上述问题，React 引入了 `key` 属性。当子元素拥有 key 时，React 使用 key 来匹配原有树上的子元素以及最新树上的子元素。

key 帮助 React 识别哪些元素改变了，如被添加或删除，通常我们使用数据中的 id 等唯一标识作为元素的 key。

这个 key 不需要全局唯一，但在列表中需要保持唯一。

编写注意：

1. **唯一性：一个元素的 key 最好是这个元素列表中独一无二的字符串，通常，我们会使用数据中的id**
2. **使用索引：万不得已使用索引作为 key 值需要注意，这样会导致性能变差，还可能引起组件状态的问题。当基于下标的组件进行重新排序时，组件 state 可能会遇到一些问题。由于组件实例是基于它们的 key 来决定是否更新以及复用，如果 key 是一个下标，那么修改顺序时会修改当前的 key，导致非受控组件的 state（比如输入框）可能相互篡改，会出现无法预期的变动。具体可参考此例：https://codepen.io/pen?&editors=0010&layout=left**
3. **用key提取组件：当列表中元素为 React 组件时，key 值应当作为该组件的属性而不是组件内部某个元素的属性**

```
function ListItem(props) {
  // 正确！这里不需要指定 key：   return <li>{props.value}</li>;}

function NumberList(props) {
  const numbers = props.numbers;
  const listItems = numbers.map((number) =>
    // 正确！key 应该在数组的上下文中被指定     <ListItem key={number.toString()} value={number
  return (
    <ul>
      {listItems}
    </ul>
  );
}

const numbers = [1, 2, 3, 4, 5];
ReactDOM.render(
  <NumberList numbers={numbers} />,
  document.getElementById('root')
);
```

4. **不可读：key 会传递信息给 React ，但不会传递给你的组件。如果组件需要知道并使用 key 可以使用其他 props 显示传递。**
5. **默认值：如果不显式指定元素的key，React 将默认使用该元素在兄弟节点中的索引作为 key。**

何时可以使用索引作为key：

> 1.列表以及其中每一项都是静态的—并非通过计算得出且不会改变
> 2. 列表使用的数据本就没有 id
> 3. 该列表永远不会被排序或过滤
>
> —— https://robinpokorny.medium.com/index-as-a-key-is-an-anti-pattern-e0349aece318

学习资料：

https://zh-hans.reactjs.org/docs/lists-and-keys.html#gatsby-focus-wrapper

https://robinpokorny.medium.com/index-as-a-key-is-an-anti-pattern-e0349aece318

https://zh-hans.reactjs.org/docs/reconciliation.html#elements-of-different-types

Like Be the first to like this