PREV
10. Animations

NEXT
12. Server Side Rendering

Chapter 11. Performance Tuning

React's DOM diffing allows you to effectively discard your entire UI at any point in time with minimal impact on the DOM. How ever there are cases where delicate tuning of which component gets rendered can help make your application faster. For instance if you have a very deeply nested component tree. In this chapter we will cover a simple configuration you can provide to your component to help speed up your application.

shouldComponentUpdate

When a component is updated, either by receving new props, setState or forceUpdate being called, React will re-render each child component of that component. In most cases this performs without a hitch, but on pages with deeply nested component trees, you can experience some sluggishnes.

Sometimes a component gets re-rendered when it didn't need to be. For instance if your component doesn't use state or props or if the props or state didn't change when the parent re-rendered.

React provides the component lifecycle method `shouldComponentUpdate`, which gives you a way to help React make the right decision of whether or not to re-render a specific component.

`shouldComponentUpdate` should return a boolean. `false` tells React to skip the re-rendition of the component and use the previously rendered version. Returning `true` will tell React to re-render the component. By default `shouldComponentUpdate` returns true and components will thus always re-render.
Note that `shouldComponentUpdate` is only called on re-rendition, not on the initial rendition of a component.

`shouldComponentUpdate` receives the new props and state as arguments to help your make a decision of whether or not to re-render:

Find answers on the fly, or master something new. Subscribe today. See pricing options.

```
shouldComponentUpdate: function(nextProps, nextState) {
  return nextProps.id !== this.props.id;
}
```

In cases where you have deep complex props or state, comparison can be trick and slow. To help mitigate this, consider using immutable data structures like Immutable-js, which we cover in the Family chapter.

React.addons.Perf provides good insight into which components should have a manually added `shouldComponentUpdate` method by giving your a list of components that React wasted most time re-rendering. Read more about React.addons.Perf in the Addons chapter.

Key

Most often you'll find the key prop used in lists. Its purpose is to identify a component to react by more than the component class. For example if you have a div with key="foo" which later changes to "bar", react will skip the diffing and throw out the children completely, rendering from scratch.

This can be useful when rendering large subtrees to avoid a diff which you know is a waste of time.

In addition to telling it when to throw out a node, in many cases it can be used when the order of elements changes. For example, consider the following render which shows items based on a sorting function.

```
var items = sortBy(this.state.sortingAlgorithm, this.props.items);
return items.map(function(item){
  return <img src={item.src} />;
});
```

If the order changes, react will diff these and determine the most efficient operation is to change the src attribute on some img elements. This is very inefficient and will cause the browser to consult its cache, possibly causing new network requests.

To remedy this, we can simply use some string (or number) we know is unique to each item, and use it as a key.

```
  return <img src={item.src} key={item.id} />;
```

Now React will look at this and instead of changing src attributes, it will realize the minimum insertBefore operations to perform, which is the most efficient way to move DOM nodes.

Find answers on the fly, or master something new. Subscribe today. See pricing options.

key props must be unique to a given parent. This also means that no moves from one parent to another will be considered.

In addition to the order changing, insertions that aren't at the end also apply. Without correct key attributes prepending an item to the array would cause the src attribute of all following <img> tags.

Find answers on the fly, or master something new. Subscribe today. See pricing options.

PREV
10. Animations

NEXT
12. Server Side Rendering

Find answers on the fly, or master something new. Subscribe today. See pricing options.