What's the Equivalent of an ng-repeat Directive in a React Component?

Michael Martin michael@angulartoreact.com

**Looking for a job?** With [Hired](https://hired.com/x/1mhm6), companies apply to you, not the other way around. You may hide your info from current and former employers. It's free for candidates, plus you get a **$1000 hiring bonus!** [Get started and find your dream job today!](https://hired.com/x/1mhm6)

Yes. React doesn’t use a proprietary construct to iterate over a collection of data. Instead, it relies on native Javascript iterators to generate repeating blocks of UI. Keep reading below for code samples and further explanation.

If you’ve been developing Angular applications for any amount of time, you’ve likely used the ng-repeat directive. Generating repeating blocks of UI from a data structure is a cornerstone of web development regardless of language or framework.

The AngularJS team wrapped up a really convenient and powerful piece of iterative magic in the ng-repeat directive. With very little code, you can create dynamic lists that stay in sync with a collection of data. Just so we’re on the same page, here’s what the Angular code looks like:

AngularJS Code Sample for ng-repeat

Assume you have an array like this:

**var** items = [

{ name: "Matthew", link: "https://bible.com/1/mat.1" },

{ name: "Mark", link: "https://bible.com/1/mrk.1" },

{ name: "Luke", link: "https://bible.com/1/luk.1" },

{ name: "John", link: "https://bible.com/1/jhn.1" }

];

In Angular 1.x, to create a ul with an li for each item in the array you could put the following code in your controller:

.controller("NgRepeatDemoCtrl", **function**($scope) {

$scope.items = items;

});

Then, in your view, you could use the ng-repeat directive like this:

<ul>

<li ng-repeat="item in items">

<a ng-href="{{item.link}}">{{item.name}}</a>

</li>

</ul>

React Alternative to ng-repeat

To perform the same task in React you just need to think natively. Under the hood ng-repeat is just using a native Javascript iterator. You can use the same sort of native iterator directly in React. For just example, Ill useArray.map`. Here’s an example:

**var** RepeatModule = React.createClass({

getInitialState: **function**() {

**return** { items: [] }

},

render: **function**() {

**var** listItems = **this**.props.items.map(**function**(item) {

**return** (

<li key={item.name}>

<a href={item.link}>{item.name}</a>

</li>

);

});

**return** (

<div>

<ul>

{listItems}

</ul>

</div>

);

}

});

In this example, Array.map iterates through every item contained in items and executes a function that returns a DOM element. (Technically, it’s a virtual DOM element, but that’s a different discussion.) Every li that is returned from inside of the map function gets added to a new array and finally returned as the value of the Array.map() function. The result of that function is stored in the listItemsvariable. When React calls the render method on our component, we generate an array of list items and then insert them into our ul using a single pair of enclosing curly braces.