给定下面的代码，我们使用 [map()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map) 函数使 numbers 数组中的元素值翻倍。我们将 map()返回的新数组分配给变量 doubled，并且打印这个它：

const numbers = [1, 2, 3, 4, 5];

const doubled = numbers.map((number) => number \* 2);

console.log(doubled);

这段代码在控制台中打印为 [2, 4, 6, 8, 10]。

### 多组件渲染

下面，我们用 JavaScript 的 [map()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map) 函数将 numbers 数组循环处理。对于每一项，我们返回一个 <li> 元素。最终，我们将结果元素数组分配给 listItems：

const numbers = [1, 2, 3, 4, 5];

const listItems = numbers.map((number) =>

<li>{number}</li>

);

把整个 listItems 数组包含到一个 <ul> 元素，并渲染到 DOM：

ReactDOM.render(

<ul>{listItems}</ul>,

document.getElementById('root')

);

## 键(Keys)

键(Keys) 帮助 React 标识哪个项被修改、添加或者移除了。数组中的每一个元素都应该有一个唯一不变的键(Keys)来标识：

挑选 key 最好的方式是使用一个在它的同辈元素中不重复的标识字符串。多数情况你可以使用数据中的 IDs 作为 keys：

const todoItems = todos.map((todo) =>

<li key={todo.id}>

{todo.text}

</li>

);

当要渲染的列表项中没有稳定的 IDs 时，你可以使用数据项的索引值作为 key 的最后选择：

const todoItems = todos.map((todo, index) =>

// Only do this if items have no stable IDs

<li key={index}>

{todo.text}

</li>

);

### 使用 keys 提取组件

keys 只在数组的上下文中存在意义。

例如，如果你[提取](http://www.css88.com/react/docs/components-and-props.html" \l "extracting-components) 一个 ListItem 组件，应该把 key 放置在数组处理的 <ListItem /> 元素中，不能放在 ListItem 组件自身中的 <li> 根元素上。

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 应该在这里被指定

<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')

);

### 在 JSX 中嵌入 map()

function NumberList(props) {

const numbers = props.numbers;

return (

<ul>

{numbers.map((number) =>

<ListItem key={number.toString()}

value={number} />

)}

</ul>

);

}