React 条件渲染

React 列表 & Keys

我们可以使用 JavaScript 的 map() 方法来创建列表。

React 实例

使用 map() 方法遍历数组生成了一个 1 到 5 的数字列表:

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((numbers) =>
<li>{numbers}</li>
);
ReactDOM.render(
<ul>{listItems}</ul>,
document.getElementById('example')
);
```

尝试一下 »

我们可以将以上实例重构成一个组件，组件接收数组参数，每个列表元素分配一个 key，不然会出现警告 `a key should be provided for list items`，意思就是需要包含 key：

React 实例

```
function NumberList(props) {
const numbers = props.numbers;
const listItems = numbers.map((number) =>
<li key={number.toString()}>
{number}
</li>
);
return (
<ul>{listItems}</ul>
);
}
const numbers = [1, 2, 3, 4, 5];
ReactDOM.render(
<NumberList numbers={numbers} />,
document.getElementById('example')
);
```

尝试一下 »

Keys

Keys 可以在 DOM 中的某些元素被增加或删除的时候帮助 React 识别哪些元素发生了变化。因此你应当给数组中的每一个元素赋予一个确定的标识。

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
  <li key={number.toString()}>
    {number}
  </li>
);
```

一个元素的 key 最好是这个元素在列表中拥有的一个独一无二的字符串。通常，我们使用来自数据的 id 作为元素的 key:

```
const todoItems = todos.map((todo) =>
  <li key={todo.id}>
    {todo.text}
  </li>
);
```

当元素没有确定的 id 时，你可以使用他的序列号索引 index 作为 key：

```
const todoItems = todos.map((todo, index) =>
  // 只有在没有确定的 id 时使用
  <li key={index}>
    {todo.text}
  </li>
);
```

如果列表可以重新排序，我们不建议使用索引来进行排序，因为这会导致渲染变得很慢。

用keys提取组件

元素的 key 只有在它和它的兄弟节点对比时才有意义。

比方说，如果你提取出一个 ListItem 组件，你应该把 key 保存在数组中的这个 `<ListItem />` 元素上，而不是放在 ListItem 组件中的 **<li>** 元素上。

错误的示范

```
function ListItem(props) {
  const value = props.value;
  return (
    // 错啦！你不需要在这里指定key:
    <li key={value.toString()}>
      {value}
    </li>
  );
}

function NumberList(props) {
```

```
  const numbers = props.numbers;
  const listItems = numbers.map((number) =>
    //错啦！元素的key应该在这里指定：
    <ListItem value={number} />
  );
  return (
    <ul>
      {listItems}
    </ul>
  );
}

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

key的正确使用方式

React 实例

```
function ListItem(props) {
// 对啦！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) =>
// 又对啦！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('example')
);
```

尝试一下 »

当你在 map() 方法的内部调用元素时，你最好随时记得为每一个元素加上一个独一无二的 key。

元素的 key 在他的兄弟元素之间应该唯一

数组元素中使用的 key 在其兄弟之间应该是独一无二的。然而，它们不需要是全局唯一的。当我们生成两个不同的数组时，我们可以使用相同的键。

React 实例

```
function Blog(props) {
const sidebar = (
<ul>
{props.posts.map((post) =>
<li key={post.id}>
{post.title}
</li>
)}
</ul>
);
const content = props.posts.map((post) =>
<div key={post.id}>
<h3>{post.title}</h3>
<p>{post.content}</p>
</div>
);
return (
<div>
{sidebar}
<hr />
{content}
</div>
);
}
const posts = [
{id: 1, title: 'Hello World', content: 'Welcome to learning React!'},
{id: 2, title: 'Installation', content: 'You can install React from npm.'}
];
ReactDOM.render(
<Blog posts={posts} />,
document.getElementById('example')
);
```

尝试一下 »

key 会作为给 React 的提示，但不会传递给你的组件。如果您的组件中需要使用和 key 相同的值，请将其作为属性传递：

```
const content = posts.map((post) =>
  <Post
    key={post.id}
    id={post.id}
    title={post.title} />
);
```

上面例子中，Post 组件可以读出 props.id，但是不能读出 props.key。

在 jsx 中嵌入 map()

在上面的例子中，我们声明了一个单独的 listItems 变量并将其包含在 JSX 中:

```
function NumberList(props) {
  const numbers = props.numbers;
  const listItems = numbers.map((number) =>
    <ListItem key={number.toString()}
              value={number} />

  );
  return (
    <ul>
      {listItems}
    </ul>
  );
}
```

JSX 允许在大括号中嵌入任何表达式，所以我们可以在 map() 中这样使用：

React 实例

```
function NumberList(props) {
const numbers = props.numbers;
return (
<ul>
{numbers.map((number) =>
<ListItem key={number.toString()}
value={number} />
)}
</ul>
);
}
```

尝试一下 »

这么做有时可以使你的代码更清晰，但有时这种风格也会被滥用。就像在 JavaScript 中一样，何时需要为了可读性提取出一个变量，这完全取决于你。但请记住，如果一个 map() 嵌套了太多层级，那你就可以提取出组件。

React 条件渲染

1 篇笔记

写笔记

JSX 允许在大括号中嵌入任何表达式，需要注意的事项（请看注释）：

```
var ListItem = (props) => {       //es6中箭头函数
    return <li>{props.value}</li>;
```

```
}

function NumberList(props) {
    var numbers;    //声明在外面是因为 {} 中不能出现var,const,let等这种关键字
    return (
    <ul>
      {
        numbers = props.numbers,   //注意这里要加逗号

        numbers.map((number) =>
        <ListItem key={number}
         value={number} />
      )}
    </ul>
    );
}

var arr = [1,2,3];   //要传递的参数
ReactDOM.render(
    <NumberList numbers={arr}/>,  //这里的numbers就是props下的numbers,即props.numbers
    document.all('example')
);
```

阿凯 7个月前 (08-22)