Map, Reduce & Filter in React

In this lesson, we'll demonstrate how the map, reduce, and filter methods are used in JavaScript and applied in React.

What's the best approach for teaching the JSX syntax for React newcomers? Usually, I start out with defining a variable in the `render()` method and using it as JavaScript in HTML in the return block.

```
import React from 'react';
require('./style.css');

import ReactDOM from 'react-dom';
import App from './app.js';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

You only have to use the curly braces to get your JavaScript in HTML. Going from rendering a string to a complex object isn't any different.

```
import React from 'react';
require('./style.css');

import ReactDOM from 'react-dom';
import App from './app.js';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

Usually, the next question then is: How to render a list of items? That's one of the best parts about explaining React in my opinion. There is no React specific API such as a custom attribute on an HTML tag which enables you to render multiple items in React. You can use plain JavaScript for iterating over the list of items and returning HTML for each item.

```
import React from 'react';
require('./style.css');

import ReactDOM from 'react-dom';
import App from './app.js';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

I'm sure there were countless times in your life when you've had to iterate over an array in a for-loop, pass the array's elements through a function, and put them in a new array. You'll realize that other than the function, the code to do this is repetitive and can be standardized into a function of its own. Well, today's your lucky day because the `map()` method does exactly that - it creates a new array with the results of calling a user-written function on every element in the calling array. You can use the `map()` method to render elements like in the following code block. Having used the JavaScript arrow function before, you can get rid of the function body and the return statement which leaves your rendered output way more concise.

```
import React from 'react';
require('./style.css');

import ReactDOM from 'react-dom';
import App from './app.js';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

Pretty soon, every React developer becomes used to the built-in JavaScript `map()` methods for arrays. It just makes so much sense to map over an array and return the rendered output for each item. The same can be applied for custom-tailored cases where `filter()` or `reduce()` make more sense rather than rendering an output for each mapped item.

```
import React from 'react';
require('./style.css');

import ReactDOM from 'react-dom';
import App from './app.js';

ReactDOM.render(
  <App />,
```

```
  document.getElementById('root')
);
```

In general, that's how the React developers are getting used to these JavaScript built-in functions without having to use a React specific API for it. It is just JavaScript in HTML.

Coding Challenge: Use the map() method

In the following, write a program using the map function that squares the numbers in the hidden given array called `notSquared`. It contains random integers. Call the new array `squared` or your code won't compile.

```
// Write-your-code-here
squared = -1;
```