*1. From Web Components To React Components*

This week, you've created pure Web components, React components, and learned how to use them interchangeably. In this activity, you'll solidify your understanding of components by converting a pure Web Component into a React Component.

## Why Focus on Web Components?

You might be wondering why you need to understand how to create Web components when React components appear to have greater functionality. As with most technologies and tools you'll come across, there are pros and cons to everything. One tool isn't always better than the other. You must learn to choose the best tool for the circumstances and requirements of what you're building.

## Choosing the Tool That Is Best for Your Project

For instance, React components have some drawbacks, including:

- They, for the most part, can only be used inside of an application that supports React
- They don't have as great a browser backwards compatibility that Web components have

If you're working on a single application, these aren't really issues. But what if you're working for a massive corporation with multiple applications that span different front-end technologies? That's where creating a Web component might be a better approach, since a Web component can run in just about any JavaScript library or framework. It is important that you understand how to use Web components and React components, and how they work together.

**In this activity, you'll start with an application that pulls movie showtime data from a JSON file named movies.json, just like the books example you saw in Dr. Sanchez's videos. Each movie you see in the web browser is a**

**Web component, and you need to convert them into React components instead.**

The file structure looks like this:

- `movies.json` This JSON file holds the movie data that will be displayed and does not need to be edited
- `styles.css` This contains some styles for the application and does not need to be edited
- `index.html` This is the only HTML file where you will import all the files for this activity. It does not need to be edited
- `main.js` This handles fetching the movie data from movies.json and passing that data to the Web components in movies.js
- `movies.js` This is where the Web component lives and where you'll be making most of your changes

Keep the following things in mind when converting the Movie Web component to a React component:

- Make sure the new React component keeps the same name, `Movie`.
- Instead of passing each movie data attribute (title, showTime, etc.), you'll want to pass the entire movie as an attribute called `data`.

This activity is intentionally similar to the books example in the videos. If you get lost or stuck, don't forget to go back and re-watch the videos for help.

*Notes:*

- *The starter code includes helpful comments to guide you through this activity*
- *The Web component in* `movies.js` *uses an inline style. Like HTML classes, you'll need to make some modifications when migrating to a React component. For help on how this is done, you can review - https://reactjs.org/docs/faq-styling.html*

*Hint: In the `main.js` file, don't forget to change the `nexttech-movie` to the name of the React component.*

*Hint: You can create an object of styles.*

```
function MyComponent ({data}) {
    const myStyle = {
        background: 'url(' + data.imageUrl + ')',
        backgroundSize: 'cover',
    };
    return (
        <div style={myStyle}>
    )
}
```