Extracting a Component

This lab is all about best practices and clean coding. We wanted to emphasize writing code that is a pleasure to maintain, easy and cheap to enhance and debug. Take a look through your Landing.js code. It's kind of complicated. We can make this a ton simpler by splitting it into two components.

Right now we're listing films and displaying a film object in the Landing component. That's too many things according to SRP[1]. Let's give the film its own component.

1. Create a new component called FilmBrief.js.

2. The component should receive a film and selectedDate in its props.

3. In its JSX, display <u>one</u> film. (Hint: you can cut some of your JSX from the Landing component. And don't forget the imports!).

4. Back in Landing.js, display a <FilmBrief film={film} key={film.id} selectedDate={selectedDate} /> instead of the <View>, <Text>s and the <Image>.

5. Run and test. You should still see your film data but now it is better designed.

6. Bonus! Go back in to Landing.js and FilmBrief.js with an eye toward even more simplicity. Make it all as simple as possible. Here's an example. This:

```
{films.map(film => {
  return <FilmBrief key={film.id} />
})}
```

7. Can be refactored to this:

```
{films.map(film =>  <FilmBrief key={film.id} />)}
```

I like the second one better. Look for any opportunities to improve your code and make them.

That's it! You're finished.

8. Take just one minute to appreciate what we did together. Look at Landing.js and how simple it is! When written as cleanly as this, any developer could very quickly reason about it and maintain it. And then go look at FilmBrief; it's even simpler! It does nothing except take in some props and display them. It' s a pure function[2] - easy to maintain.

---

[1] https://en.wikipedia.org/wiki/Single-responsibility_principle
[2] https://en.wikipedia.org/wiki/Pure_function