Meet Another React Component

So far we've only been using the App component to create our applications. We used the App component in the last section to express everything needed to render our list in JSX, and it should scale with your needs and eventually handle more complex tasks. To help with this, we'll split some of its responsibilities into a standalone List component:

```
const list = [ ... ];

function App() { ... }

function List() {
  return list.map(function(item) {
    return (
      <div key={item.objectID}>x
        <span>
          <a href={item.url}>{item.title}</a>
        </span>
        <span>{item.author}</span>
        <span>{item.num_comments}</span>
        <span>{item.points}</span>
      </div>
    );
  });
}
```

src/App.js

Optional: If this component looks odd, because the outermost part of the returned JSX starts with JavaScript. We could use it with a wrapping HTML element as well, but we'll continue with the previous version.

```
const list = [ ... ];

function List() {

  return (
    <div>
      {list.map(function(item) {

        return (...);
      })}
    </div>

  );
}
```

src/App.js

Now the new List component can be used in the App component:

```
  □ □ □□  □   ã□  F   □□  □   □  )□      □   9□  5□  @@  □   °□  n□  □PNG
□
   IHDR   □   □□□   (-□S   äPLTE"""""""""""""""""2PX=r□)7;*:>H□¤-BGE□□8do5Xb6[eK□®K□¯1MU
□
   IHDR   □   □□□   ×©ÍÊ  □ePLTE"""""""""""""""""""""""2RZN¢¹J□«3R[J□¬)59YÁÞ0KS4W`Q«ÄL□²%
?^q÷ñíÛ□ï.},□ìsæÝ_TttÔ¾ □1#□□/(ì□-[□□□è`□è`Ì□ÚïÅðZ□d5□□□□?ÎebZ¿Þ□i.Ûæ□□□ìqÎ□+1°□}Â□5
□
   IHDR        □□   D¤□Æ  □APLTE   """"""""""""""""""""""""""2RZVºÖ_ÔôU·Ñ=r□$()'25]ÎíC□□0
□
   IHDR   @   @□□   □·□ì  □:PLTE   """"""""""""""""""""""""""""""""""""""""""""""""""""""
¢ßqÇ8Ù□´□mKË±mÆ¶mÛü·yi!è□ÎªYÏuë ÀÏ_Àï?i÷□ý+ò□□ÄA□|□ù{□□´?¿□_En□).□JËD¤<□
©¬¢Z\Ts©R*□(□  ¯©□J□□□□u□X/□4J□9□¡5·DEµ4kÇ4□&i¥V4Ú□¡®Ð□□¯□vsf:àg,□¢èBC»î$¶□ºÍùî□□á□@
-ê>Û□º«¢XÕ¢î}ß¨ëÛÑ;□ÃöN´□ØvÅý□Î¸ÿ1 □ë×ÄO@&v/Äþ_□ö\ô□Ç\í.□□¼+0□□;□□□!□fÊ□¦´Ó%Â JY·O□Â□'
```

You've just created your first React component! With this example, we can see how components that encapsulate meaningful tasks can work for larger React applications.

Larger React applications have **component hierarchies** (also called **component trees**). There is usually one uppermost **entry point component** (e.g. App) that spans a tree of components below it. The App is the **parent component** of the List, so the List is a **child component** of the App. In a component tree, the App is the **root component**, and the components that don't render any other components are called **leaf components** (e.g. List). The App can have multiple children, as can the List. If the App has another child component, the additional child component is called a **sibling component** of the List.

Exercises:

- Confirm the changes from the last section.
- Draw your components – the App component and List component – as a component tree on a sheet of paper. Extend this component tree with other possible components (e.g. extracted Search component for the input field and label in the App component). Try to figure out which

other parts can be extracted as standalone components.

- If a Search component is used in the App component, would the Search component be a sibling, parent, or child component for the List component?
- Ask yourself what problems could arise if we keep treating the `list` variable as global variable. We will cover how to handle these problems in the upcoming sections.