Reverse Sort

Task: The sort feature works, but the ordering only includes one direction. Implement a reverse sort when the button is clicked twice, so it becomes a toggle between normal (ascending) and reverse (descending) sort.

Optional Hints:

- Consider that reverse or normal sort could be just another state (e.g. `isReverse`) next to the `sortKey`.
- Set the new state in the `handleSort` handler based on the previous sort.
- Use the new `isReverse` state for sorting the list with the sort function from the dictionary with the optionally applied `reverse()` function from JavaScript arrays.

The initial sort direction works for strings, as well as numeric sorts like the reverse sort for JavaScript numbers that arranges them from high to low. Now we need another state to track whether the sort is reversed or normal, to make it more complex:

```
const List = ({ list, onRemoveItem }) => {
  const [sort, setSort] = React.useState({
    sortKey: 'NONE',
    isReverse: false,
  });

  ...
};
```

src/App.js

Next, give the sort handler logic to see if the incoming `sortKey` triggers are a normal or reverse sort. If the `sortKey` is the same as the one in the state, it could be a reverse sort, but only if the sort state wasn't already reversed:

```
const List = ({ list, onRemoveItem }) => {
```

```
const List = ({ list, onRemoveItem }) => {
  const [sort, setSort] = React.useState({
    sortKey: 'NONE',
    isReverse: false,
  });

  const handleSort = sortKey => {
    const isReverse = sort.sortKey === sortKey && !sort.isReverse;

    setSort({ sortKey: sortKey, isReverse: isReverse });
  };

  const sortFunction = SORTS[sort.sortKey];
  const sortedList = sortFunction(list);

  return (
    ...
  );
};
```

src/App.js

Lastly, depending on the new `isReverse` state, apply the sort function from the dictionary with or without the built-in JavaScript reverse method for arrays:

```
const List = ({ list, onRemoveItem }) => {
  const [sort, setSort] = React.useState({
    sortKey: 'NONE',
    isReverse: false,
  });

  const handleSort = sortKey => {
    const isReverse = sort.sortKey === sortKey && !sort.isReverse;
    setSort({ sortKey, isReverse });
  };

  const sortFunction = SORTS[sort.sortKey];

  const sortedList = sort.isReverse
    ? sortFunction(list).reverse()
    : sortFunction(list);

  return (
    ...
  );
};
```

src/App.js

The reverse sort is now operational. For the object passed to the state updater function, we use what is called a **shorthand object initializer notation**:

```
const firstName = 'Robin';

const user = {
  firstName: firstName,
};

console.log(user);
// { firstName: "Robin" }
```

src/App.js

When the property name in your object is the same as your variable name, you can omit the key/value pair and just write the name:

```
const firstName = 'Robin';

const user = {
  firstName,
};

console.log(user);
// { firstName: "Robin" }
```

src/App.js

Here is the complete demonstration of the above concepts:

```
  □ □ □□  □   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.□□½+0□□;□□□!□fÊ□¦´Ó%Â JY·O□Â□'
```

If necessary, read more about JavaScript Object Initializers.

Exercises:

- Confirm the changes from the last section.
- Consider the drawback of keeping the sort state in the List instead of the App component. If you don't know, sort the list by "Title" and search for other stories afterward. What would be different if the sort state would be in the App component.
- Use your styling skills to give the user feedback about the current active

sort and its reverse state. It could be an arrow up or arrow down SVG next to each active sort button.