Recap

Now you've learned to interact with an API in React! This lesson will provide you a recap to what you have learned so far and will also share how your App.js code looks by now.

Let's recap the last chapter:

- **React**
  - ES6 class component lifecycle methods for different use cases
  - `componentDidMount()` for API interactions
  - Conditional renderings
  - Synthetic events on forms
  - Error handling
  - Aborting a remote API request
- **ES6 and beyond**
  - Template strings to compose strings
  - Spread operator for immutable data structures
  - Computed property names
  - Class fields
- **General**
  - Hacker News API interaction
  - Native fetch browser API
  - Client and server-side search
  - Pagination of data
  - Client-side caching
  - Axios as an alternative for the native fetch API

Again, it makes sense to take a break, internalize the lessons and apply them on your own. Experiment with the parameters for the API endpoint to query

different results. You can find the source code in the official repository.

Your `src/App.js` should look like the following by now:

```
import React, { Component } from 'react';
require('./App.css');

const DEFAULT_QUERY = 'redux';
const DEFAULT_HPP = '100';

const PATH_BASE = 'https://hn.algolia.com/api/v1';
const PATH_SEARCH = '/search';
const PARAM_SEARCH = 'query=';
const PARAM_PAGE = 'page=';
const PARAM_HPP = 'hitsPerPage=';

class App extends Component {

  constructor(props) {
    super(props);

    this.state = {
      results: null,
      searchKey: '',
      searchTerm: DEFAULT_QUERY,
      error: null
    };

    this.needsToSearchTopstories = this.needsToSearchTopstories.bind(this);
    this.setSearchTopstories = this.setSearchTopstories.bind(this);
    this.fetchSearchTopstories = this.fetchSearchTopstories.bind(this);
    this.onSearchChange = this.onSearchChange.bind(this);
    this.onSearchSubmit = this.onSearchSubmit.bind(this);
    this.onDismiss = this.onDismiss.bind(this);
  }

  componentDidMount() {
    const { searchTerm } = this.state;
    this.setState({ searchKey: searchTerm });
    this.fetchSearchTopstories(searchTerm);
  }

  setSearchTopstories(result) {
    const { hits, page } = result;
    const { searchKey, results } = this.state;

    const oldHits = results && results[searchKey]
      ? results[searchKey].hits
      : [];

    const updatedHits = [
      ...oldHits,
      ...hits
    ];

    this.setState({
      results: {
        ...results,
        [searchKey]: { hits: updatedHits, page }
```

```
    }
  });
}

fetchSearchTopstories(searchTerm, page = 0) {
  fetch(`${PATH_BASE}${PATH_SEARCH}?${PARAM_SEARCH}${searchTerm}&${PARAM_PAGE}${page}&${PAR
    .then(response => response.json())
    .then(result => this.setSearchTopstories(result))
      .catch(e => this.setState({ error: e }));
}

needsToSearchTopstories(searchTerm) {
  return !this.state.results[searchTerm];
}

onSearchChange(event) {
  this.setState({ searchTerm: event.target.value });
}

onSearchSubmit(event) {
  const { searchTerm } = this.state;
  this.setState({ searchKey: searchTerm });

  if (this.needsToSearchTopstories(searchTerm)) {
    this.fetchSearchTopstories(searchTerm);
  }

  event.preventDefault();
}

onDismiss(id) {
  const { searchKey, results } = this.state;
  const { hits, page } = results[searchKey];

  const isNotId = item => item.objectID !== id;
  const updatedHits = hits.filter(isNotId);

  this.setState({
    results: {
      ...results,
      [searchKey]: { hits: updatedHits, page }
    }
  });
}

render() {
  const {
    searchTerm,
    results,
    searchKey,
    error
  } = this.state;

  const page = (
    results &&
    results[searchKey] &&
    results[searchKey].page
  ) || 0;

  const list = (
    results &&
    results[searchKey] &&
```

```
      results[searchKey].hits
    ) || [];

    return (
      <div className="page">
        <div className="interactions">
          <Search
            value={searchTerm}
            onChange={this.onSearchChange}
            onSubmit={this.onSearchSubmit}
          >
            Search
          </Search>
        </div>
        { error
          ? <div className="interactions">
            <p>Something went wrong.</p>
          </div>
          : <Table
            list={list}
            onDismiss={this.onDismiss}
          />
        }
        <div className="interactions">
          <Button onClick={() => this.fetchSearchTopstories(searchKey, page + 1)}>
            More
          </Button>
        </div>
      </div>
    );
  }
}

const Search = ({
  value,
  onChange,
  onSubmit,
  children
}) =>
  <form onSubmit={onSubmit}>
    <input
      type="text"
      value={value}
      onChange={onChange}
    />
    <button type="submit">
      {children}
    </button>
  </form>

const Table = ({ list, onDismiss }) =>
  <div className="table">
    { list.map(item =>
      <div key={item.objectID} className="table-row">
        <span style={{ width: '40%' }}>
          <a href={item.url}>{item.title}</a>
        </span>
        <span style={{ width: '30%' }}>
          {item.author}
        </span>
        <span style={{ width: '10%' }}>
          {item.num_comments}
```

```
        </span>
        <span style={{ width: '10%' }}>
          {item.points}
        </span>
        <span style={{ width: '10%' }}>
          <Button
            onClick={() => onDismiss(item.objectID)}
            className="button-inline"
          >
            Dismiss
          </Button>
        </span>
      </div>
    )}
  </div>

const Button = ({ onClick, className = '', children }) =>
  <button
    onClick={onClick}
    className={className}
    type="button"
  >
    {children}
  </button>

export default App;
```