Client- or Server-side Search

Extending the search functionality to now search on the server side instead of client side.

Now, when you use the Search component with its input field, you will filter the list. That's happening on the client-side, though. We want to use the Hacker News API to search on the server-side. Otherwise, you would only deal with the first API response you got on `componentDidMount()`, the default search term parameter.

You can define an `onSearchSubmit()` method in your App component, which fetches results from the Hacker News API when executing a search in the Search component.

```
class App extends Component {

  constructor(props) {
    super(props);

    this.state = {
      result: null,
      searchTerm: DEFAULT_QUERY,
    };

    this.setSearchTopStories = this.setSearchTopStories.bind(this);
    this.onSearchChange = this.onSearchChange.bind(this);
    this.onSearchSubmit = this.onSearchSubmit.bind(this);
    this.onDismiss = this.onDismiss.bind(this);
  }

  ...

  onSearchSubmit() {
    const { searchTerm } = this.state;
  }

  ...
}
```

The `onSearchSubmit()` method should use the same functionality as the `componentDidMount()` lifecycle method, but this time with a modified search term from the local state and not with the initial default search term. Thus you can extract the functionality as a reusable class method.

```
class App extends Component {

  constructor(props) {
    super(props);

    this.state = {
      result: null,
      searchTerm: DEFAULT_QUERY,
    };

    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);
  }

  ...

  fetchSearchTopStories(searchTerm) {
    fetch(`${PATH_BASE}${PATH_SEARCH}?${PARAM_SEARCH}${searchTerm}`)
      .then(response => response.json())
      .then(result => this.setSearchTopStories(result))
      .catch(error => error);
  }

  componentDidMount() {
    const { searchTerm } = this.state;
    this.fetchSearchTopStories(searchTerm);
  }

  ...

  onSearchSubmit() {
    const { searchTerm } = this.state;
    this.fetchSearchTopStories(searchTerm);
  }

  ...
}
```

Now the Search component has to add an additional button to trigger the search request explicitly. Otherwise, we'd fetch data from the Hacker News API every time the input field changes. We want to do it explicitly in a `onClick()` handler.

As an alternative, you could debounce (delay) the `onChange()` function and spare the button, but it would add more complexity, and we want to keep things simple for now.

First, pass the `onSearchSubmit()` method to your Search component.

```
class App extends Component {

  ...

  render() {
    const { searchTerm, result } = this.state;
    return (
      <div className="page">
        <div className="interactions">
          <Search
            value={searchTerm}
            onChange={this.onSearchChange}
            onSubmit={this.onSearchSubmit}
          >
            Search
          </Search>
        </div>
        { result &&
          <Table
            list={result.hits}
            pattern={searchTerm}
            onDismiss={this.onDismiss}
          />
        }
      </div>
    );
  }
}
```

Second, introduce a button in your Search component. The button has the `type="submit"` and the form uses its `onSubmit` attribute to pass the `onSubmit()` method. You can reuse the `children` property, but this time it will be used as the content of the button.

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

In the Table, you can remove the filter functionality, because there will be no client-side filter (search) anymore. Don’t forget to remove the `isSearched()`

function as well, as it won't be used anymore. Now, the result comes directly from the Hacker News API when the user clicks the "Search" button.

```
class App extends Component {

  ...

  render() {
    const { searchTerm, result } = this.state;
    return (
      <div className="page">
        ...
        { result &&
          <Table
            list={result.hits}
            onDismiss={this.onDismiss}
          />
        }
      </div>
    );
  }
}

...

const Table = ({ list, onDismiss }) =>
  <div className="table">
    {list.map(item =>
      ...
    )}
  </div>
```

Now when you try to search, you will notice the browser reloads. That's a native browser behavior for a submit callback in an HTML form. In React, you will often come across the `preventDefault()` event method to suppress the native browser behavior.

```
onSearchSubmit(event) {
  const { searchTerm } = this.state;
  this.fetchSearchTopStories(searchTerm);
  event.preventDefault();
}
```

You should be able to search different Hacker News stories now. We have interacted with a real API, and there should be no more client-side searches.

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

const DEFAULT_QUERY = 'redux';

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

class App extends Component {

  constructor(props) {
    super(props);

    this.state = {
      result: null,
      searchTerm: DEFAULT_QUERY,
    };

    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);
  }

  setSearchTopstories(result) {
    this.setState({ result });
  }

  fetchSearchTopstories(searchTerm) {
    fetch(`${PATH_BASE}${PATH_SEARCH}?${PARAM_SEARCH}${searchTerm}`)
      .then(response => response.json())
      .then(result => this.setSearchTopstories(result))
        .catch(e => e);
  }

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

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

  onSearchSubmit(event) {
    const { searchTerm } = this.state;
    this.fetchSearchTopstories(searchTerm);
    event.preventDefault();
  }

  onDismiss(id) {
    const isNotId = item => item.objectID !== id;
    const updatedHits = this.state.result.hits.filter(isNotId);
    this.setState({
      result: { ...this.state.result, hits: updatedHits }
    });
  }

  render() {
    const { searchTerm, result } = this.state;
    return (
      <div className="page">
        <div className="interactions">
          <Search
            value={searchTerm}
            onChange={this.onSearchChange}
            onSubmit={this.onSearchSubmit}
          >
            Search
          </Search>
        </div>
        { result &&
          <Table
            list={result.hits}
            onDismiss={this.onDismiss}
          />
        }
      </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;
```

Exercises:

- Experiment with the Hacker News API

Further Reading:

- Read about synthetic events in React