Loading ...

In this lesson, we'll learn how to add a 'loading' message to our application while results are being fetched.

Displaying a Loading message

Now we get back to the application, where we'll show a loading indicator when a search request submits to the Hacker News API. The request is asynchronous, so you should show your user feedback that something is happening.

Defining the Loading component and state property

Let's define a reusable Loading component in the *src/App.js* file like in the following.

```
const Loading = () =>
  <div>Loading ...</div>
```

Now we need a property to store the loading state. Based on the loading state, we can decide to show the Loading component later.

```
class App extends Component {
  _isMounted = false;

  constructor(props) {
    super(props);

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

    ...
  }

  ...

}
```

Setting the isLoading property

The initial value of that `isLoading` property is false. We don't load anything before the App component is mounted. When the request is made, the loading state is set to true. The request will succeed eventually, and you can set the loading state to false.

```
class App extends Component {

  ...

  setSearchTopStories(result) {
    ...

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

  fetchSearchTopStories(searchTerm, page = 0) {
    this.setState({ isLoading: true });

    axios(`${PATH_BASE}${PATH_SEARCH}?${PARAM_SEARCH}${searchTerm}&${PARAM_PAGE}${page}&${PAR
      .then(result => this._isMounted && this.setSearchTopStories(result.data))
      .catch(error => this._isMounted && this.setState({ error }));
  }

  ...

}
```

When to display the loading message

In the last step, we used the Loading component in the App component. A conditional rendering based on the loading state will decide whether to show a Loading component or the Button component. The latter is the button to fetch more data.

```
class App extends Component {

  ...

  render() {
    const {
      searchTerm,
```

```
      results,
      searchKey,
      error,

      isLoading
    } = this.state;

    ...

    return (
      <div className="page">
        ...
        <div className="interactions">
          { isLoading
            ? <Loading />
            : <Button
                onClick={() => this.fetchSearchTopStories(searchKey, page + 1)}
              >
              More
            </Button>
          }
        </div>
      </div>
    );
  }
}
```

Initially, the Loading component will show when you start your application, because you make a request on `componentDidMount()`. There is no Table component, because the list is empty. When the response returns from the Hacker News API, the result is shown, the loading state is set to false, and the Loading component disappears. Instead, the “More” button to fetch more data appears. Once you fetch more data, the button will disappear again and the Loading component will appear.

The Hacker News App with the Loading message

```
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=';

class App extends Component {

  constructor(props) {
    super(props);

    this.state = {
      results: null,
```

```
    searchKey: '',
    searchTerm: DEFAULT_QUERY,
    error: null,
    isLoading: false,
  };

  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);
}

needsToSearchTopstories(searchTerm) {
  return !this.state.results[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 }
    },
    isLoading: false
  });
}

fetchSearchTopstories(searchTerm, page = 0) {
  this.setState({ isLoading: true });

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

componentDidMount() {
  const { searchTerm } = this.state;
  this.setState({ searchKey: searchTerm });
  this.fetchSearchTopstories(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,
      isLoading
    } = 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">
          { isLoading
            ? <Loading />
            : <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>

const Loading = () =>
  <div>Loading ...</div>

export default App;

export {
  Button,
  Search,
  Table,
};
```

Exercises:

- Use a library such as Font Awesome to show a loading icon instead of the “Loading ...” text in the above!