Foreword

Get a brief look at what we are going to learn in this course!

The Road to learn React teaches you the fundamentals of React. You will build a real-world application along the way in plain React without complicated tooling. Everything from project setup to deployment on a server will be explained for you. The course comes with additional referenced reading material and exercises with each chapter. After taking the course, you will be able to build your own applications in React. The material is kept up to date by me and the community.

In the Road to learn React, I want to offer a foundation before you start to dive into the broader React ecosystem. It has less tooling and less external state management, but a lot of information about React. It explains general concepts, patterns and best practices in a real-world React application.

You will learn to build your own React application. It covers real-world features like pagination, client-side caching and interactions such as searching and sorting. Additionally, you will transition from JavaScript ES5 to JavaScript ES6 along the way. I hope this course captures my enthusiasm for React and JavaScript and helps you to get started with it.

```
import React, { Component } from 'react';
import { sortBy } from 'lodash';
import classNames from 'classnames';

require('./App.css');

const DEFAULT_QUERY = 'redux';
const DEFAULT_PAGE = 0;
const DEFAULT_HPP = '3';

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=';

const SORTS = {
```

```
  NONE: list => list,
  TITLE: list => sortBy(list, 'title'),
  AUTHOR: list => sortBy(list, 'author'),

  COMMENTS: list => sortBy(list, 'num_comments').reverse(),
  POINTS: list => sortBy(list, 'points').reverse(),
};

class App extends Component {

  constructor(props) {
    super(props);

    this.state = {
      results: null,
      searchKey: '',
      searchTerm: DEFAULT_QUERY,
      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) {
    this.setState({ isLoading: true });

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

  componentDidMount() {
```

```
  const { searchTerm } = this.state;
  this.setState({ searchKey: searchTerm });
  this.fetchSearchTopstories(searchTerm, DEFAULT_PAGE);
}

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, DEFAULT_PAGE);
  }

  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,
    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>
        <Table
          list={list}
          onDismiss={this.onDismiss}
        />
        <div className="interactions">
          <ButtonWithLoading
            isLoading={isLoading}
            onClick={() => this.fetchSearchTopstories(searchKey, page + 1)}>
            More
          </ButtonWithLoading>
        </div>
      </div>
    );
  }
}

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

class Table extends Component {

  constructor(props) {
    super(props);

    this.state = {
      sortKey: 'NONE',
      isSortReverse: false,
    };

    this.onSort = this.onSort.bind(this);
  }

  onSort(sortKey) {
    const isSortReverse = this.state.sortKey === sortKey && !this.state.isSortReverse;
    this.setState({ sortKey, isSortReverse });
  }

  render() {
    const {
      list,
      onDismiss
    } = this.props;

    const {
```

```
    sortKey,
    isSortReverse,
  } = this.state;

  const sortedList = SORTS[sortKey](list);
  const reverseSortedList = isSortReverse
    ? sortedList.reverse()
    : sortedList;

  return(
    <div className="table">
      <div className="table-header">
        <span style={{ width: '40%' }}>
          <Sort
            sortKey={'TITLE'}
            onSort={this.onSort}
            activeSortKey={sortKey}
          >
            Title
          </Sort>
        </span>
        <span style={{ width: '30%' }}>
          <Sort
            sortKey={'AUTHOR'}
            onSort={this.onSort}
            activeSortKey={sortKey}
          >
            Author
          </Sort>
        </span>
        <span style={{ width: '10%' }}>
          <Sort
            sortKey={'COMMENTS'}
            onSort={this.onSort}
            activeSortKey={sortKey}
          >
            Comments
          </Sort>
        </span>
        <span style={{ width: '10%' }}>
          <Sort
            sortKey={'POINTS'}
            onSort={this.onSort}
            activeSortKey={sortKey}
          >
            Points
          </Sort>
        </span>
        <span style={{ width: '10%' }}>
          Archive
        </span>
      </div>
      { reverseSortedList.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>

const withLoading = (Component) => ({ isLoading, ...rest }) =>
  isLoading ? <Loading /> : <Component { ...rest } />

const ButtonWithLoading = withLoading(Button);

const Sort = ({
  sortKey,
  activeSortKey,
  onSort,
  children
}) => {
  const sortClass = classNames(
    'button-inline',
    { 'button-active': sortKey === activeSortKey }
  );

  return (
    <Button
      onClick={() => onSort(sortKey)}
      className={sortClass}
    >
      {children}
    </Button>
  );
}

export default App;

export {
  Button,
  Search,
```

```
  Table,
};
```