Output from React Components

In this lesson, we'll be looking at how to get output from React components.

WE'LL COVER THE FOLLOWING

Output

The first obvious output of a React component is the rendered HTML. Visually, that is what we get. However, because the prop may be anything, including a function, we could also send out data or trigger a process.

In the following example, we have a component that accepts the user's input and sends it out (`<NameField />`).

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './app.js';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

Using lifecycle methods to trigger a process

Very often we need an entry point of our logic. React comes with some handy lifecycle methods that may be used to trigger a process. For example, say we have an external resource that needs to be fetched on a specific page.

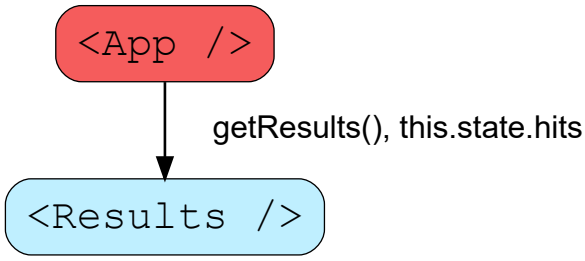
```
import React from 'react';
import Results from 'results';

const API = 'https://hn.algolia.com/api/v1/search?query=';
const DEFAULT_QUERY = 'redux';
```

```
const DEFAULT_QUERY = 'redux';
export default class App extends React.Component {
  constructor() {
    super();
    this.getresults = this.getresults.bind(this);
    this.state = {
      hits: [],
    }
  }
  getresults()
  {
    fetch('https://hn.algolia.com/api/v1/search?query=redux')
      .then(response => response.json())
      .then((json) => {
        this.setState({ hits: json.hits });
     });
  };
render() {
    return <Results hits={this.state.hits} getResults={this.getresults}/>
  }
}
```

The code above renders a loading screen while data is fetched from an API. The `App` component passes the function `getresults()` as a prop to the `Results` component. `App`'s state's `hits` attribute is also passed as a prop to `Results`. `getresults()` is called within the `Results` component when it *mounts* by calling it within the lifecycle method `ComponentDidMount()` on **lines 8-10**. The `getresults()` function updates `App`'s state and when it updates, `App` rerenders thereby updating the props passed to `Results`. This causes the condition within the if-statement to become true and hence rendering the fetched JSON data. Have a look at the slides below for a better understanding.

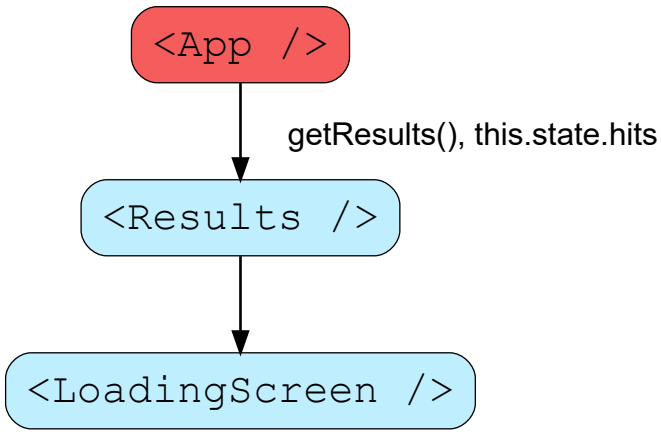
Another variation of this could be building a search-results experience. We have a search page and we enter our criteria there. We click submit and the user goes to `/results` where we have to display the result of the search. Once we land on the results page we render some sort of a loading screen and trigger a request for fetching the results in `componentDidMount` lifecycle hook. When the data comes back we pass it to a `<List>` component.

Output

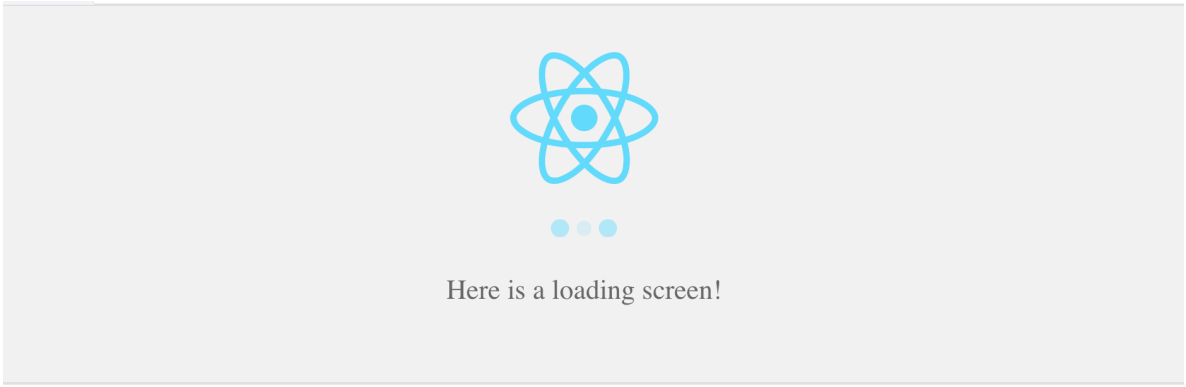


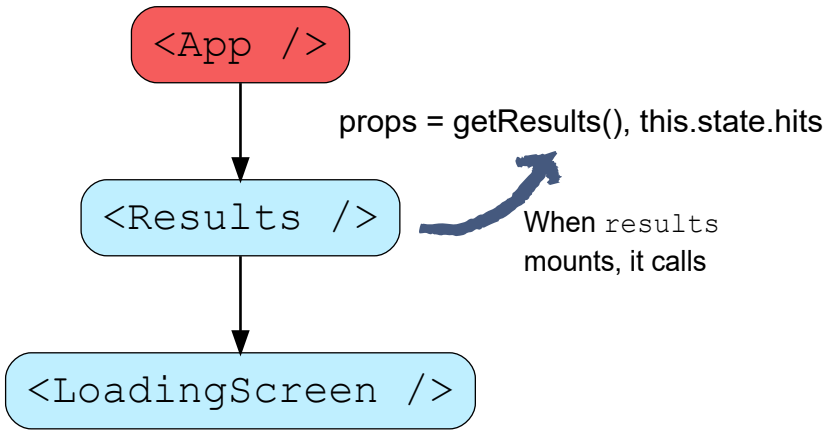
Output

Generating output...

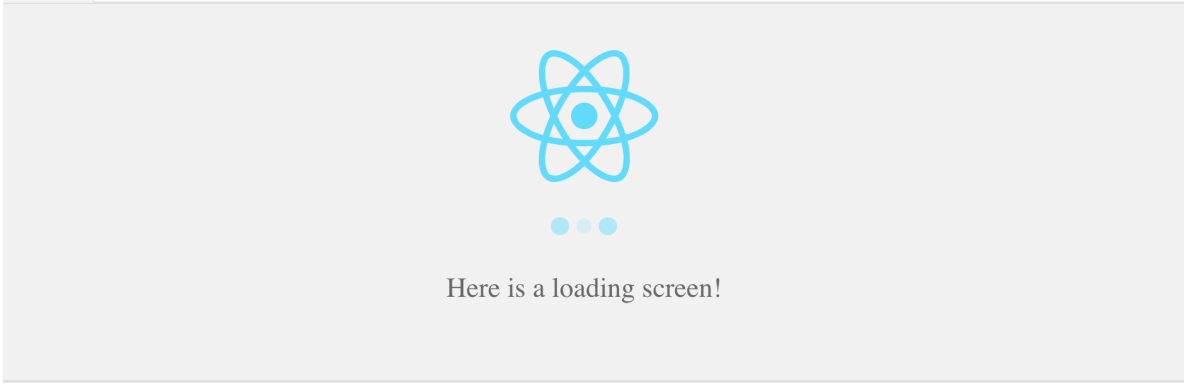

<App />
getResults(), this.state.hits
<Results />
<LoadingScreen />

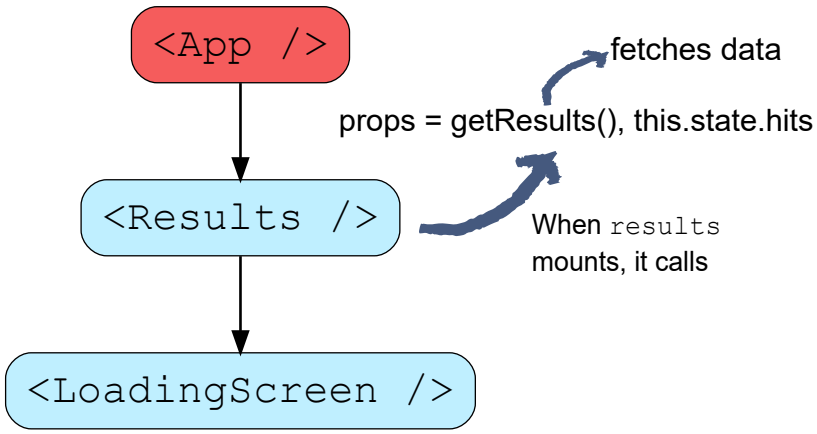
Output


Here is a loading screen!

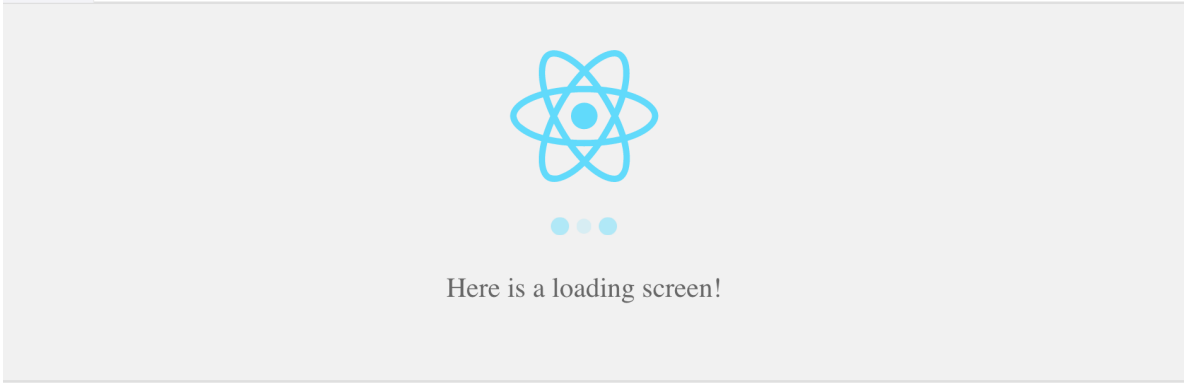

<App />
props = getResults(), this.state.hits
<Results />
When results
mounts, it calls
<LoadingScreen />

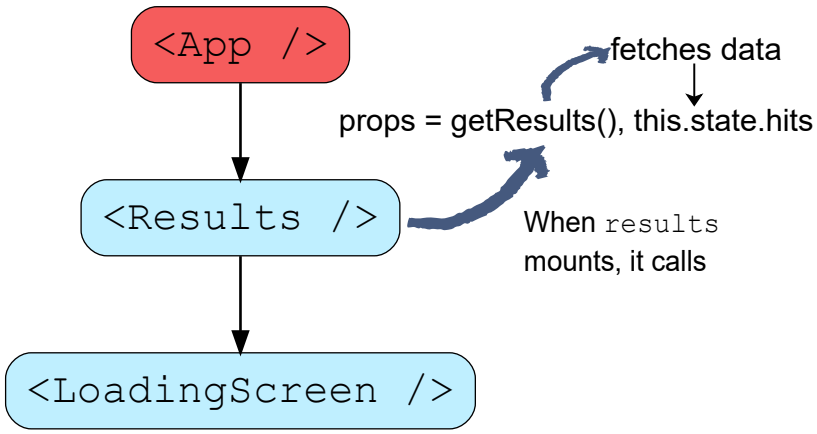
Output


Here is a loading screen!

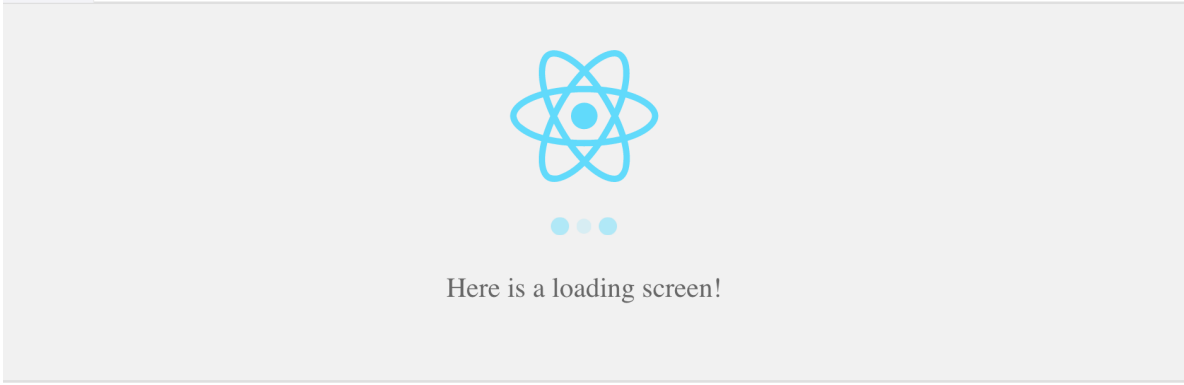

<App />
fetches data
props = getResults(), this.state.hits
<Results />
When results
mounts, it calls
<LoadingScreen />

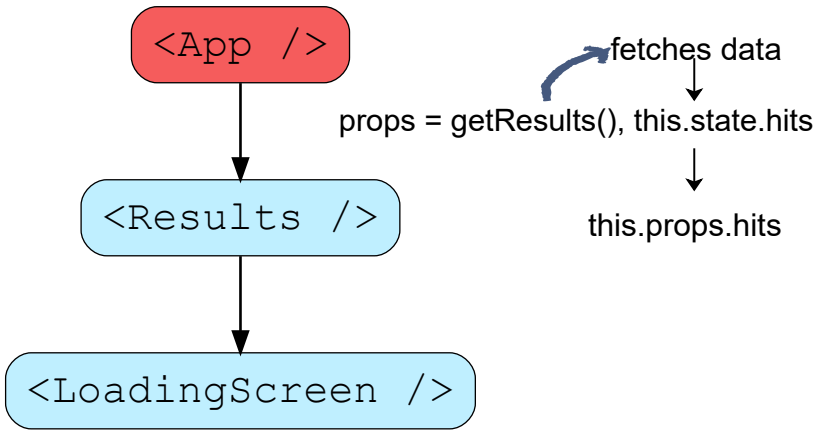
Output


Here is a loading screen!


<App />
fetches data
props = getResults(), this.state.hits
<Results />
When results
mounts, it calls
<LoadingScreen />

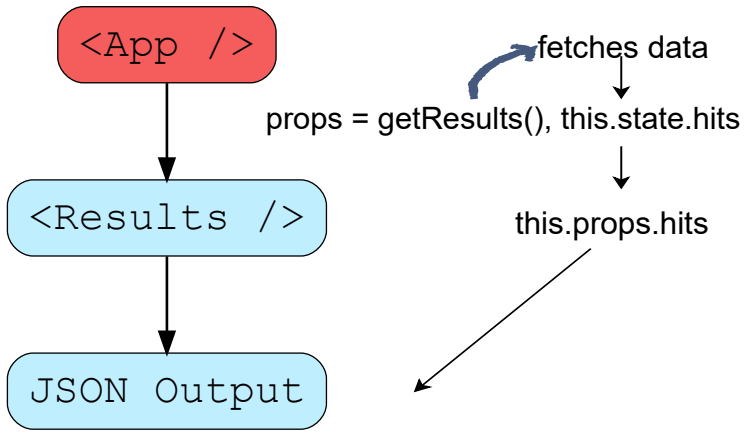
Output


Here is a loading screen!



Output

Here is a loading screen!



Output

ld Yourself a Redux

)s://zapier.com/engineering/how-to-build-redux/

ngs to learn in React before using Redux

)s://www.robinwieruch.de/learn-react-before-using-redux/

)w HN: ORY Editor – A rich editor for the browser, built with React and Redux

)s://github.com/ory/editor?branch=master

)oundCloud client in React and Redux

):://www.robinwieruch.de/the-soundcloud-client-in-react-redux/

Now that we know how to get output from React components, lets do a few exercises in the next lesson!