Install:

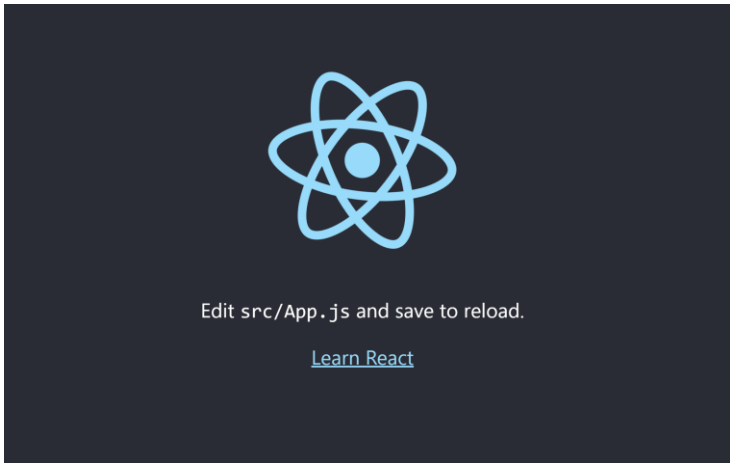
npm i -g create-react-app

```
~ $
~ $sudo npm i -g create-react-app@1.5.2
Password:
```

create-react-app react-ap

npm start

And we get this:



Install bootstrap

```
Demo $cd vidly/
vidly $npm i bootstrap@4.
```

Example:

```
class Counter extends Component {
  render() {
    return (
      <React.Fragment>
        <h1>Hello World</h1>
        <button>Increment</button>
      </React.Fragment>
    );
  }
}
```

Conditional Render

方法一，重新定义一个方法

```
  renderTags() {
    if (this.state.tags.length === 0) return <p>There are no tags!</p>;

    return <ul>{this.state.tags.map(tag => <li key={tag}>{tag}</li>)}</ul>;
  }

  render() {
    return <div>{this.renderTags()}</div>;
  }
}
```

方法二

```
render() {
  return (
    <div>
      {this.state.tags.length === 0 && "Please create a new tag!"}
      {this.renderTags()}
    </div>
  );
}
```

点击事件里传入回调函数：

```
render() {
  return (
    <div>
      <span className={this.getBadgeClasses()}>{this.formatCount()}</span>
      <button
        onClick={() => this.handleIncrement(product)}
        className="btn btn-secondary btn-sm"
      >
        Increment
      </button>
    </div>
  );
}
```

```
{this.state.movies.map(movie => (
  <tr>
    <td>{movie.title}</td>
    <td>{movie.genre.name}</td>
    <td>{movie.numberInStock}</td>
    <td>{movie.dailyRentalRate}</td>
  </tr>
))}
```

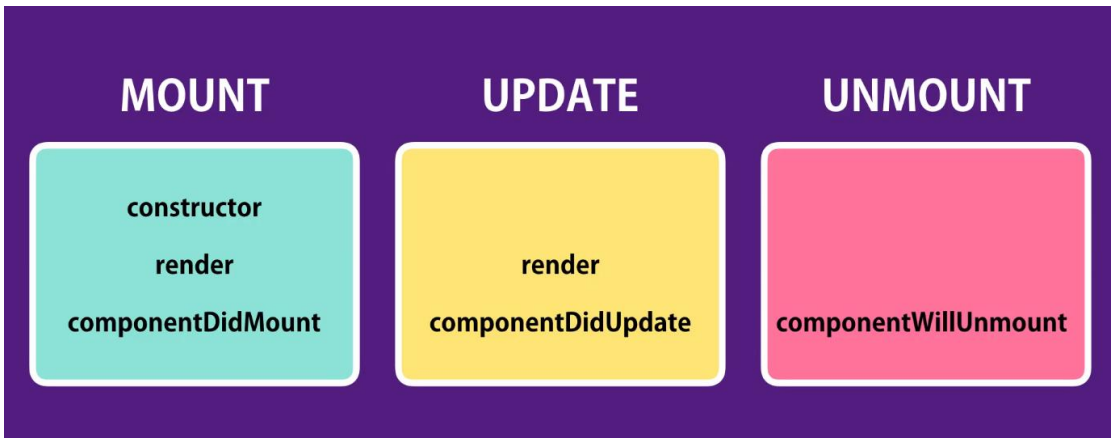
```
<td>button.btn.btn-danger.btn-sm</td>
```

```
<td><button className="btn btn-danger btn-sm">Delete</button></td>
```

```jsx
{this.state.movies.map(movie => (
  <tr key={movie._id}>
    <td>{movie.title}</td>
    <td>{movie.genre.name}</td>
    <td>{movie.numberInStock}</td>
    <td>{movie.dailyRentalRate}</td>
    <td>
      <button
        onClick={() => this.handleDelete(movie)}
        className="btn btn-danger btn-sm"
      >
        Delete
      </button>
    </td>
  </tr>
```

```jsx
render() {
  return (
    <div>
      { this.state.counters.map(counter => <Counter key={counter.id} />)}
    </div>
  );
}
```

The component that owns a piece of the state, should be the one modifying it.



不可以在 constructor 里面调用 setState

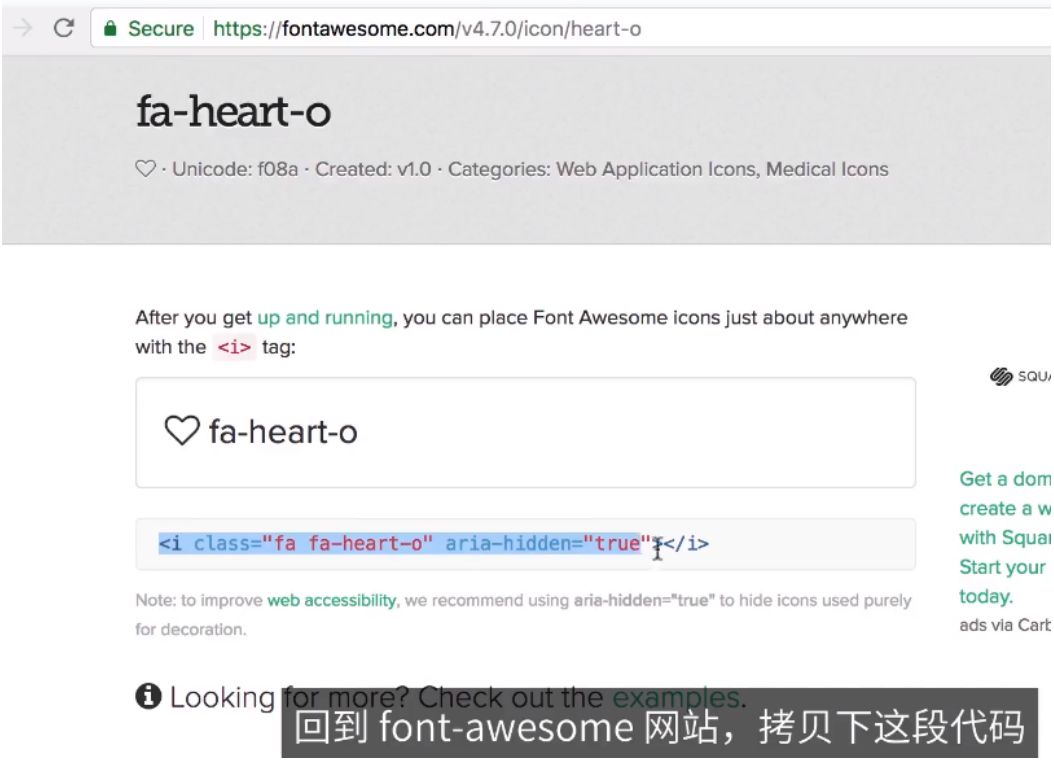
```
constructor() {
  super();
  console.log('App - Constructor');
  this.state = this.props.something;
  this.setState()
}
```

```
// Stateless Functional Component
const NavBar = props => {
  console.log('NavBar - Rendered');

  return (
    <nav className="navbar navbar-light bg-light">
      <a className="navbar-brand">
        Navbar{" "}
        <span className="badge badge-pill badge-secondary">
          {props.totalCounters}
        </span>
      </a>
    </nav>
  );
};

export
```

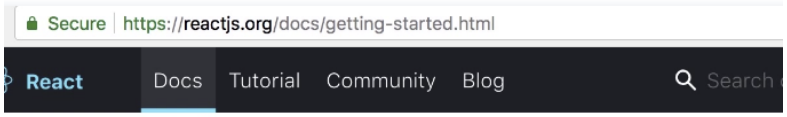
是你不能 在无状态功能性组件中使用生命周期钩子

```
class Counter extends Component {
  componentDidUpdate() {

  }

  render() {
    console.log("Counter - Rendered");

    return (
      <div>
        <span className={this.getBadgeClasses()}>{this.formatCount()}</
        <button
          onClick={() => this.props.onIncrement(this.props.counter)}
          className="btn btn-secondary btn-sm"
        >
          Increment
        </button>
        <button
```

如果有变化，我们可以重新通过Ajax来向服务器取得数据



回到 font-awesome 网站，拷贝下这段代码



Getting Started

his page is an overview of the React documentation and related sources.

```
Pagination.propTypes = {
  itemsCount: PropTypes.number.isRequired,
  pageSize: PropTypes.number.isRequired,
  currentPage: PropTypes.number.isRequired,
  onPageChange: PropTypes.func.isRequired
};
```

```
import queryString from "query-string";

const Posts = ({ match, location }) => {
  const result = queryString.parse(location.search);
```

```
<div className="content">
  <Switch>
    <Route path="/products/:id" component={ProductDetails} />
    <Route
      path="/products"
      render={props => <Products sortBy="newest" {...props} />}
    />
    <Route path="/posts/:year?/:month?" component={Posts} />
    <Route path="/admin" component={Dashboard} />
    <Route path="/" component={Home} />
  </Switch>
</div>
```

```
handleSave = () => {
  // Navigate to /products
  this.props.history.push("/products");
};
```

```
handleSave = () => {
  // Navigate to /products
  this.props.history.replace("/products");
};
```

```
ul>(li>Link[to=''])*
```

```
<ul>
  <li>
    <Link to=""></Link>
  </li>
  <li>
    <Link to=""></Link>
  </li>
</ul>
```

index.js

```
ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
  document.getElementById("root")
);
registerServiceWorker();
```

快捷键：

```
Route[path][component]*4
```

```
<Route path="" component=""></Route>
<Route path="" component=""></Route>
<Route path="" component=""></Route>
<Route path="" component=""></Route>
```

在 React 中使用 DOM 元素

```
username = React.createRef();
```

```
const username = this.username.current.value;
```

```
<input ref={this.username}
```

自动获得焦点

```
componentDidMount() {
  this.username.current.focus();
}
```

也可以直接设置，不需要使用以上 refs

```
<input
  autoFocus
```

P113 Controlled Element

获取表单信息

```
handleChange = e => {
  const account = { ...this.state.account };
  account.username = e.currentTarget.value;
  this.setState({ account });
};
```

```
value={account.username}
onChange={this.handleChange}
id="username"
name="username"
```

P114

如果在 state 里面设置 null 或者 undefined，就是说这个元素不想作为受控元素。

之后又对 username 赋值，就会触发如下错误。因为一赋值又想让这个元素受控了。

```
state = {
  account: { username: null, password: "" }
};
```

```
▶Warning: A component is changing an uncontrolled input of type text to be controlled. Input elements should not switch from uncontrolled to controlled (or vice versa). Decide between using a controlled or uncontrolled input element for the lifetime of the component. More info: https://fb.me/react-controlled-components
```
index.js:2178

P118

```
validate = () => {
  const errors = {};

  const { account } = this.state;
  if (account.username.trim() === '')
    errors.username = 'Username is required.';
  if (account.password.trim() === '')
    errors.password = 'Password is required.';

  return Object.keys(errors).length === 0 ? null : errors;
};
```

```
handleSubmit = e => {
  e.preventDefault();

  const errors = this.validate();
  this.setState({ errors });
  if (errors) return;
```

P119

```
 {error && <div className="alert alert-danger">{error}</div>}
</div>
```

在前面的有条件渲染组件区块的时候我们讲过

P120 Validate on Change

```
handleChange = ({ currentTarget: input }) => {
  const errors = { ...this.state.errors };
  const errorMessage = this.validateProperty(input);
  if (errorMessage) errors[input.name] = errorMessage;
  else delete errors[input.name];

  const account = { ...this.state.account };
  account[input.name] = input.value;
```

```
validateProperty = ({ name, value }) => {
  if (name === 'username') {
    if (value.trim() === '') return 'Username is required.';
    // ...
  }

}
```

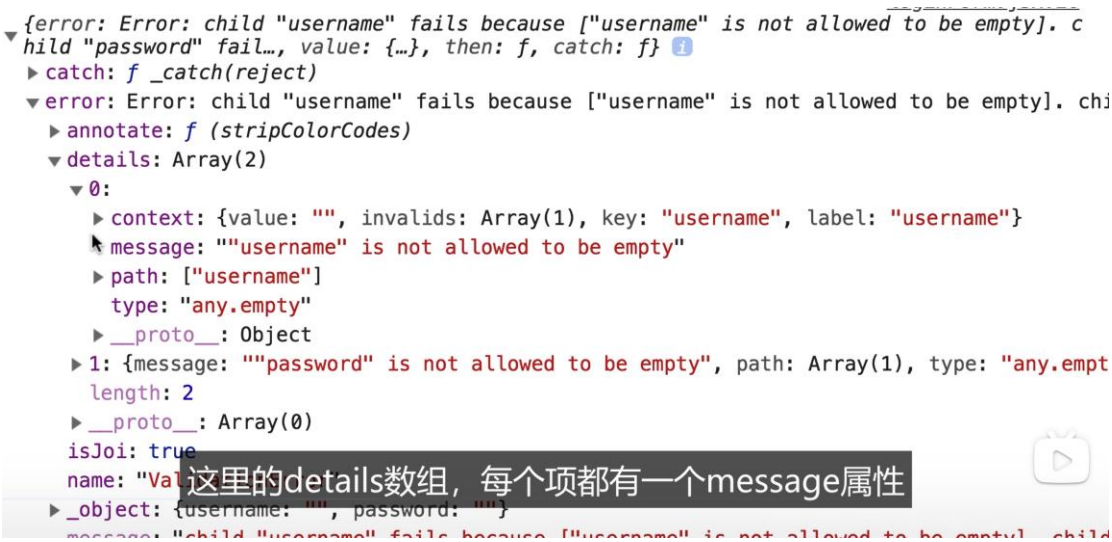
P121 第三方库 JOI

```
schema = {
  username: Joi.string().required(),
  password: Joi.string().required()
};
```

```
validate = () => {
  const result = Joi.validate(this.state.account, this.schema, {
    abortEarly: false
  });
```

P122

需要解析一下内容




```
validate = () => {
  const result = Joi.validate(this.state.account, this.schema, {
    abortEarly: false
  });
  if (!result.error) return null;

  const errors = {};
  for (let item of result.error.details)
    errors[item.path[0]] = item.message;
  return errors;
};
```

解决大小写问题

```
schema = {
  username: Joi.string().required().label('Username'),
  password: Joi.string().required().label('Password')
};
```

P123

```
validateProperty = ({ name, value }) => {
  const obj = { [name]: value };
  const schema = { [name]: this.schema[name] };
  const { error } = Joi.validate(obj, schema);
  return error ? error.details[0].message : null;
};
```

P124

disable the button

```
<button
  disabled={this.validate()}
```

P127

```
const Input = ({ name, label, error, ...rest }) => {
  return (
    <div className="form-group">
      <label htmlFor={name}>{label}</label>
      <i (JSX attribute) value: any
        value={value}
        onChange={onChange}
        type={type}
        name={name}
        id={name}
        className="form-control"
      />
      {error && <div className="alert alert-danger">{error}</div>}
    </div>
```

以上三个值用...rest 实现

```jsx
const Input = ({ name, label, error, ...rest }) => {
  return (
    <div className="form-group">
      <label htmlFor={name}>{label}</label>
      <input {...rest} id={name} className="form-control" />
      {error && <div className="alert alert-danger">{error}</div>}
    </div>
  );
};
```

P128

```jsx
schema = {
  username: Joi.string()
    .required()
    .email()
    .label("Username"),
  password: Joi.string()
    .required()
    .min(5)
    .label("Password"),
  name: Joi.string()
    .required()
    .label("Name")
};
```

发 HTTP 请求到后端，常用的技术：

用 axios 发消息

```
async componentDidMount() {
  // pending > resolved (success) OR rejected (failure)
  const { data: posts } = await axios.get(apiEndpoint);
  this.setState({ posts });
}

handleAdd = async () => {
  const obj = { title: "a", body: "b" };
  const { data: post } = await axios.post(apiEndpoint, obj);
  console.log(post);
};
```

通常的方式是get,为了获得数据

post为了创建数据

put是为了更新数据

delete为了删除数据

P140 更新

我们使用patch更新一个或多个属性

或者使用put更新所有属性

```
handleUpdate = post => {
  post.title = "UPDATED";
  axios.put(apiEndpoint + '/' + post.id, post);
  axios.patch(apiEndpoint + '/' + post.id, { title: post.title });
};
```

```
handleUpdate = async post => {
  post.title = "UPDATED";
  await axios.put(apiEndpoint + "/" + post.id, post);

  const posts = [...this.state.posts];
  const index = posts.indexOf(post);
  posts[index] = { ...post };
  this.setState({ posts });
};
```

P141 delete Data

```
handleDelete = async post => {
  await axios.delete(apiEndpoint + "/" + post.id);

  const posts = this.state.posts.filter(p => p.id !== post.id);
  this.setState({ posts });
};
```

P142 Optimistic Update

```
handleDelete = async post => {
  const originalPosts = this.state.posts;

  const posts = this.state.posts.filter(p => p.id !== post.id);
  this.setState({ posts });

  try {
    await axios.delete(apiEndpoint + "/" + post.id);
  }
  catch (ex) {
    alert('Something failed while deleting a post!');
    this.setState({ posts: originalPosts });
  }
};
```

P143 Expected und unexpected Errors

```
// Expected (404: not found, 400: bad request) - CLIENT ERRORS
//  - Display a specific error message
//
// Unexpected (network down, server down, db down, bug)
//  - Log them
//  - Display a generic and friendly error message
```

```
try {
  await axios.delete(apiEndpoint + "/" + post.id);
} catch (ex) {
  if (ex.response && ex.response.status === 404)
    alert('This post has already been deleted.');
  else {
    console.log('Logging the error', ex);
    alert("An unexpected error occurrred.");
  }
```

P144 Axios Interceptors

```
axios.interceptors.response.use(null, error => {
  const expectedError =
    error.response &&
    error.response.status >= 400 &&
    error.response.status < 500;

  if (!expectedError) {
    console.log("Logging the error", ex);
    alert("An unexpected error occurrred.");
  }

  return Promise.reject(error);
});
```

```
handleDelete = async post => {
  const originalPosts = this.state.posts;

  const posts = this.state.posts.filter(p => p.id !== post.id);
  this.setState({ posts });

  try {
    await axios.delete(apiEndpoint + "/999" + post.id);
  } catch (ex) {
    if (ex.response && ex.response.status === 404)
      alert("This post has already been deleted.");
    this.setState({ posts: originalPosts });
  }
};
```

P147 Use Toastify

App.js

import

```
import { ToastContainer } from 'react-toastify';
```

```
import 'react-toastify/dist/ReactToastify.css';
```

then render a ToastContainer

```
render() {
  return (
    <React.Fragment>
      <ToastContainer />
      <button className="btn btn-prin
```

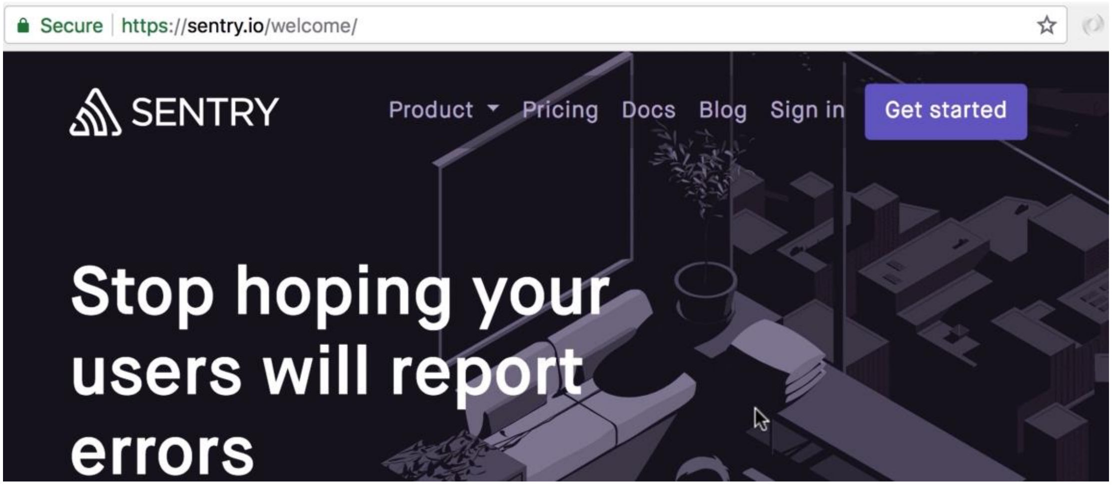
HttpService.js

```
import { toast } from 'react-toastify';
```

```
toast.error("An unexpected error occurrred.");
```

```
toast("An unexpected error occurrred.");
```

P148 Log Error



Install

```
http-app $npm i raven-js@3.26.4
loadIdealTree:loa
```

Index.js

index.js

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App";
import registerServiceWorker from "./registerServiceWorker";
import Raven from 'raven-js';
import "./index.css";
import "bootstrap/dist/css/bootstrap.css";

Raven.config('https://05323d37c9a947eba9daaaab1e6171a9@sentry.io/1249956', {
  release: '1-0-0',
  environment: 'development-test',
}).install()
```

HttpService.js

index.js httpService.js

```
import axios from "axios";
import Raven from "raven-js";
import { toast } from "react-toastify";

axios.interceptors.response.use(null, error => {
  const expectedError =
    error.response &&
    error.response.status >= 400 &&
    error.response.status < 500;

  if (!expectedError) {
    Raven.captureException(error);
    toast.error("An unexpected error occurrred.");
  }
```

P149 Extracting a Logging Service

```javascript
import Raven from "raven-js";

function init() {
  Raven.config("https://05323d37c9a947eba9daaaab1e6171a9@sentry.io/1249956", {
    release: "1-0-0",
    environment: "development-test"
  }).install();
}

function log(error) {
  Raven.captureException(error);
}

export default {
  init,
  log
};
```

use it from other file

```javascript
import logger from './services/logService';
```

```javascript
if (!expectedError) {
  logger.log(error);
  toast.error("An unexpected error occurrred.");
}
```

P157 Get Genres from Database

```javascript
import http from './httpService';

export function getGenres() {
  return http.get('http://localhost:3900/api/genres');
}
```

```javascript
async componentDidMount() {
  const { data } = await getGenres();
  const genres = [{ _id: "", name: "All Genres" }, ...data];

  this.setState({ movies: getMovies(), genres });
}
```

P158 Get Movies from Database

movies.jsx movieService.js ×

```
1  import http from "./httpService";
2
3  const apiEndpoint = "http://localhost:3900/api/movies";
4
5  export function getMovies() {
6    return http.get(apiEndpoint);
7  }
8
9  export function deleteMovie(movieId) {
10   return http.delete(apiEndpoint + "/" + movieId);
11 }
```

```
import { getMovies, deleteMovie } from "../services/movieService";
```

```
const { data: movies } = await getMovies();
this.setState({ movies, genres });
```

```
handleDelete = async movie => {
  const originalMovies = this.state.movies;
  const movies = originalMovies.filter(m => m._id !== movie._id);
  this.setState({ movies });

  try {
    await deleteMovie(movie._id);
  } catch (ex) {
    if (ex.response && ex.response.status === 404)
      toast.error("This movie has already been deleted.");

    this.setState({ movies: originalMovies });
  }
};
```

P159 Extracting a Config

P161 Populating the Form

```
async componentDidMount() {
  const { data: genres } = await getGenres();
  this.setState({ genres });

  const movieId = this.props.match.params.id;
  if (movieId === "new") return;

  try {
    const { data: movie } = await getMovie(movieId);
    this.setState({ data: this.mapToViewModel(movie) });
  } catch (ex) {
    if (ex.response && ex.response.status === 404)
      this.props.history.replace("/not-found");
  }
}
```

P162 Refactoring

```
export function saveMovie(movie) {
  if (movie._id) {
    http.put(apiEndpoint + '/' + movie._id, movie);
  }
}
```

然而我们的RESTful后端不喜欢请求体中有id

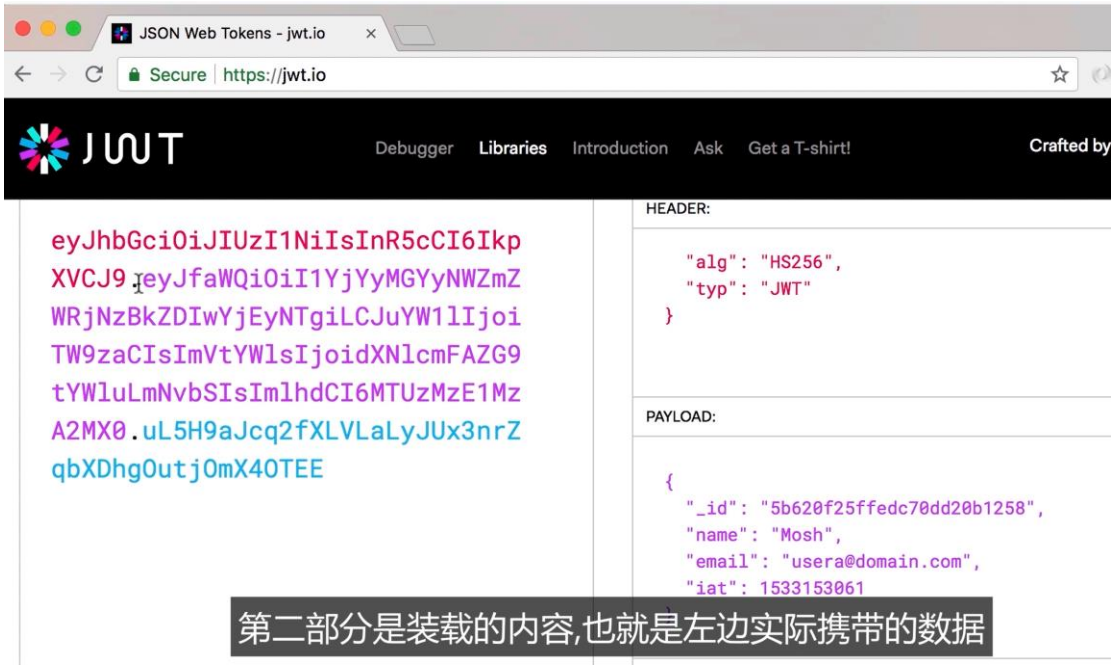
```
export function saveMovie(movie) {
  if (movie._id) {
    const body = { ...movie };
    delete body._id;
    return http.put(apiEndpoint + "/" + movie._id, body);
  }

  return http.post(apiEndpoint, movie);
}
```

Chapter 9: Authentication and Authoritation

保存令牌 token

```
doSubmit = async () => {
  try {
    const { data } = this.state;
    const { data: jwt } = await login(data.username, data.password);
    localStorage.setItem("token", jwt);
    this.props.history.push("/");
  } catch (ex) {
    if (ex.response && ex.response.status === 400) {
      const errors = { ...this.state.errors };
      errors.username = ex.response.data;
      this.setState({ errors });
    }
  }
};
```

JSON Web Token



P175 Getting the User

```
import jwtDecode from 'jwt-decode';
```

App.js

```
15
16  class App extends Component {
17    state = {};
18
19    componentDidMount() {
20      try {
21        const jwt = localStorage.getItem("token");
22        const user = jwtDecode(jwt);
23        this.setState({ user });
24      }
25      catch (ex) {}
26    }
27
28    render() {
29      return (
30        <React.Fragment>
31          <ToastContainer />
32          <NavBar user={this.state.user} />
```

P176 Displaying the User in the NavBar

not registered

```
{!user && (
  <React.Fragment>
    <NavLink className="nav-item nav-link" to="/login">
      Login
    </NavLink>
    <NavLink className="nav-item nav-link" to="/register">
      Register
    </NavLink>
  </React.Fragment>
)}
```

registered

reload

P177 Logout Funktion

退出登录的时候，删除 token，设置应用重载。

logout.jsx

```jsx
import React, { Component } from 'react';

class Logout extends Component {
  componentDidMount() {
    localStorage.removeItem('token');

    window.location = '/';
  }

  render() {
    return null;
  }
}

export default Logout;
```

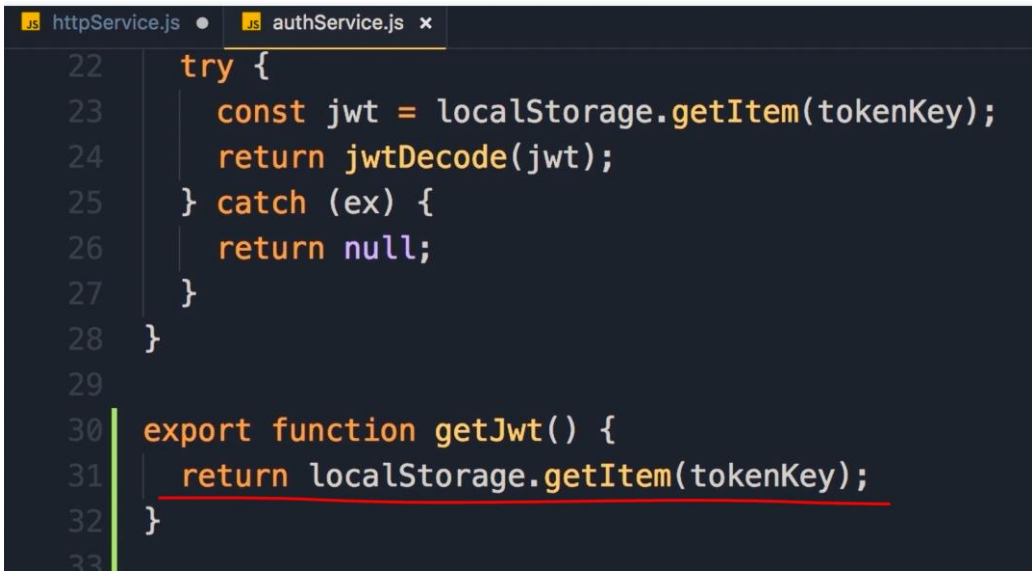
logout.jsx App.js

```jsx
}

render() {
  return (
    <React.Fragment>
      <ToastContainer />
      <NavBar user={this.state.user} />
      <main className="container">
        <Switch>
          <Route path="/register" component={RegisterForm} />
          <Route path="/login" component={LoginForm} />
          <Route path="/logout" component={Logout} />
```

P179 Calling Protected API EndPoint

设置需要进行认证

为了能从其他组件获取 token, 进行以下设置



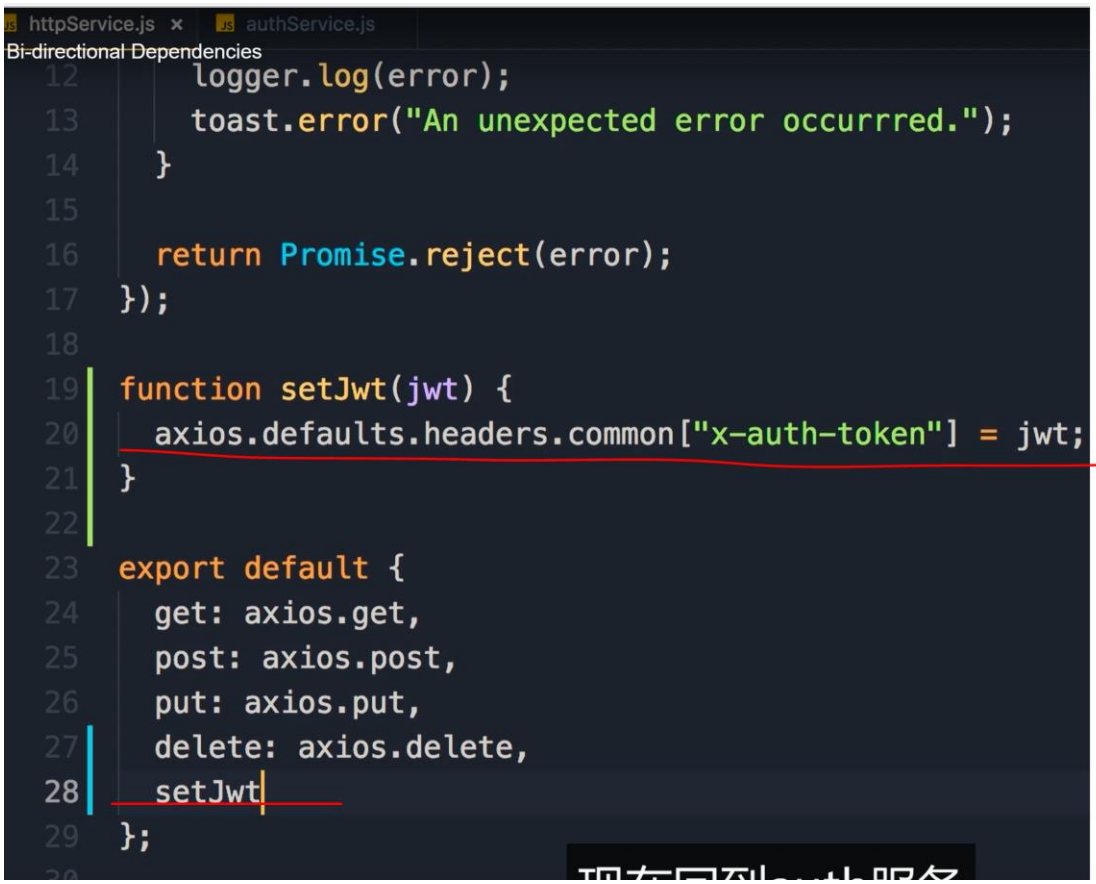
在 HttpService 组件，发 Axios 请求的时候带上 TOKEN

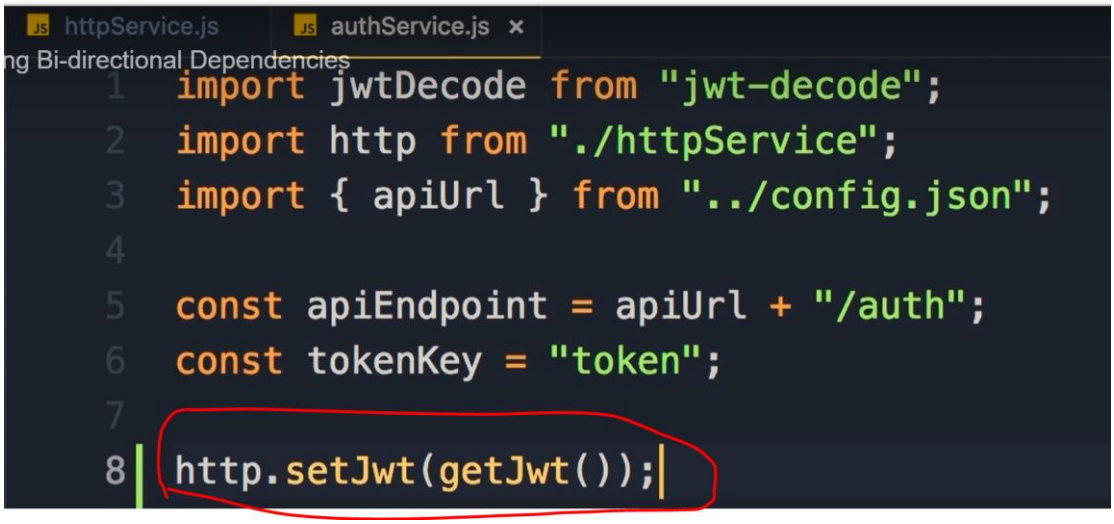
就好像我们和Axios说,无论发送什么给服务器,都带上这个头部

P180 双向依赖问题解决

我们的程序可以运行,但是我们也创造了一个很常见且很危险的设计错误

那就是bi-directional dependencies(双向依赖)





P181 Authoritation

比如,这一页我们就想在用户未登录时不显示新增电影按钮

同样也不能给用户编辑和删除电影的权限

P182 Showing or Hiding Element

App.js

```
}

render() {
  return (
    <React.Fragment>
      <ToastContainer />
      <NavBar user={this.state.user} />
      <main className="container">
        <Switch>
          <Route path="/register" component={RegisterForm} />
          <Route path="/login" component={LoginForm} />
          <Route path="/logout" component={Logout} />
          <Route path="/movies/:id" component={MovieForm} />
          <Route
            path="/movies"
            render={props => <Movies {...props} user={this.state.user} />}
          />
```

现在我们可以依据user对象来显示或隐藏元素了

App.js movies.jsx

```
            items={this.state.genres}
            selectedItem={this.state.selectedGenre}
            onItemSelect={this.handleGenreSelect}
          />
        </div>
        <div className="col">
          {user && (
            <Link
              to="/movies/new"
              className="btn btn-primary"
              style={{ marginBottom: 20 }}
            >
              New Movie
            </Link>
          )}
```

P183 Protecting the Route

App.js

```
<NavBar user={this.state.user} />
<main className="container">
  <Switch>
    <Route path="/register" component={RegisterForm} />
    <Route path="/login" component={LoginForm} />
    <Route path="/logout" component={Logout} />
    <Route
      path="/movies/:id"
      render={props => {
        if (!user) return <Redirect to="/login" />;
        return <MovieForm {...props} />;
      }}
    />
```

P184 Extracting Protected Route

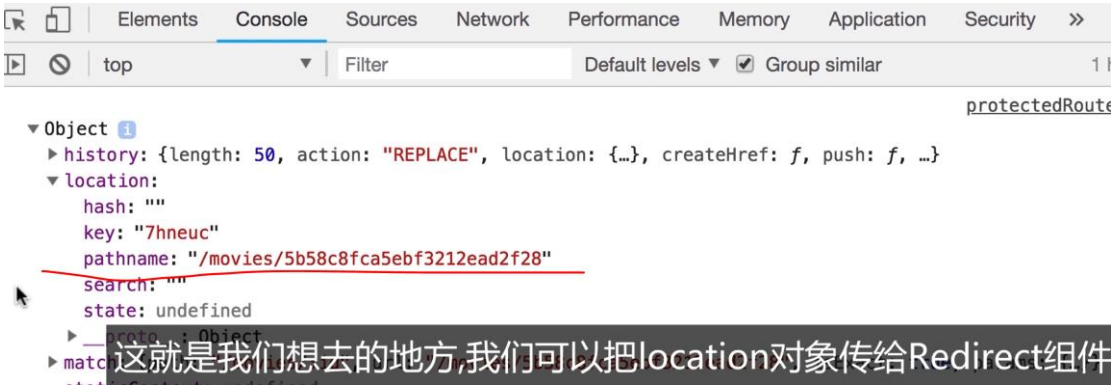
App.js protectedRoute.jsx

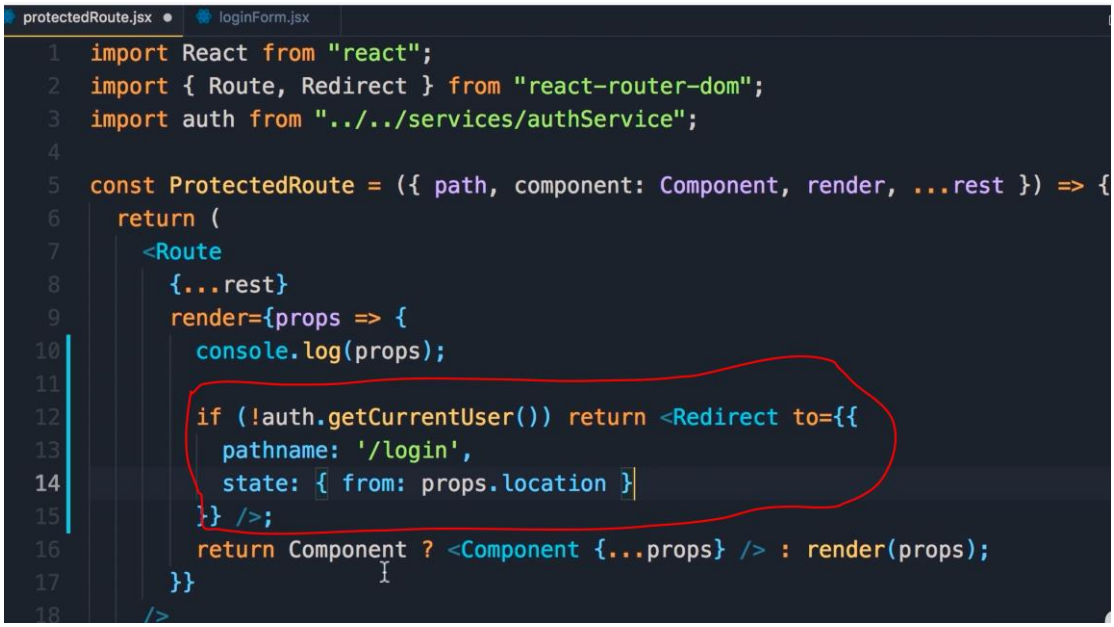
```
import React from "react";
import { Route, Redirect } from "react-router-dom";
import auth from "../../services/authService";

const ProtectedRoute = ({ path, component: Component, render, ...rest }) => {
  return (
    <Route
      {...rest}
      render={props => {
        if (!auth.getCurrentUser()) return <Redirect to="/login" />;
        return Component ? <Component {...props} /> : render(props);
      }}
    />
  );
};

export default ProtectedRoute;
```

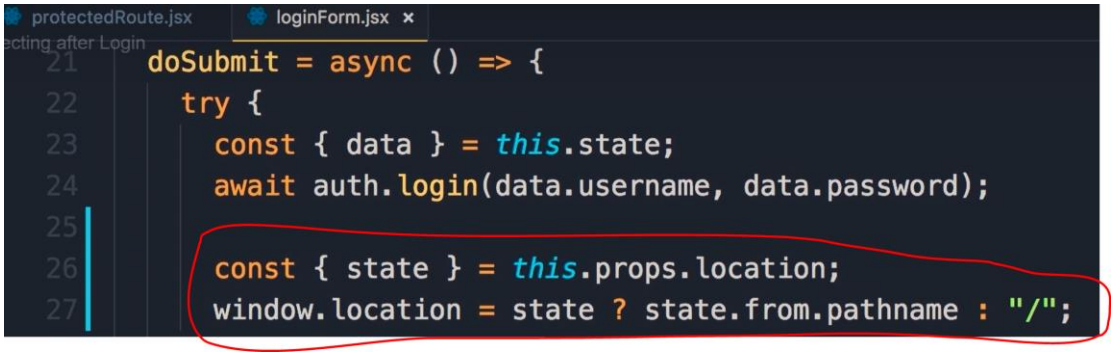
App.js protectedRoute.jsx

```
<Route path="/register" component={RegisterForm} />
<Route path="/login" component={LoginForm} />
<Route path="/logout" component={Logout} />
<ProtectedRoute path="/movies/:id" component={MovieForm} />
```

P185 Redirecting after Login







P186 Exercise

P187 Hiding the delete Column

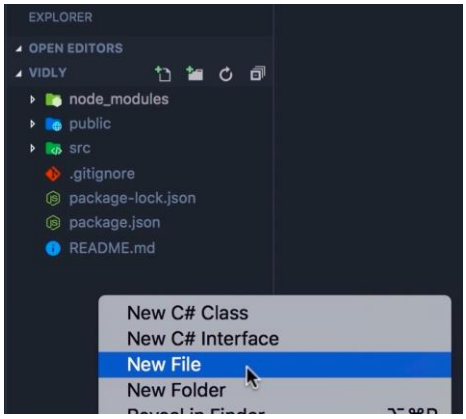
moviesTable.jsx

```
      </button>
    )
  };

  constructor() {
    super();
    const user = auth.getCurrentUser();
    if (user && user.isAdmin) this.columns.push(this.deleteColumn);
  }
```

P188 Deployment

P189 Adding Enviroment Variables

Add a file to the root



我创建一个新文件.env.development
这个文件保存所有开发环境的变量
所有的变量都是键和值的配对
所有的键都以REACT_APP_开头

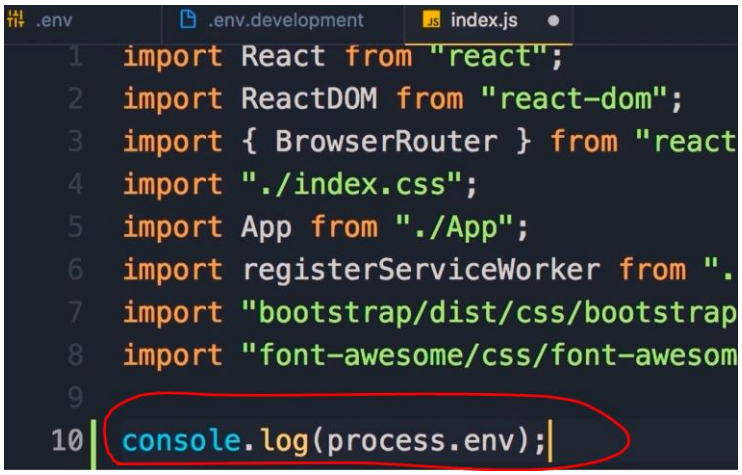
Create a file named .env.development, this file saves all the enviroment variables.

All variables are key and values pairs.

All the keys start with REACT_APP_

process is current process, env represents all the enviroment variables.

Everytime we change the enviroment files, we need to restart the Appication.



The Expressions that references to the enviroment variable, are replaced with the actual value of that the enviroment variable. ↓↓

```
console.log("SUPERMAN", process.env.REACT_APP_NAME);
```

P190 Building for Production

Creating an optimized production build:

```
vidly $
vidly $npm run build
```

```
The build folder is ready to be deployed.
You may serve it with a static server:

  npm install -g serve
  serve -s build
```

P191 Getting Started with Heroku

First, create a Heroku account.

Then, install Heroku CLI

If you are behind a firewall, that requires the use of proxy to connect extra HTTP services.

```
vidly $
vidly $export HTTP_PROXY=http://proxy.server.com:1234
```

P192 MangoDB in the Cloud

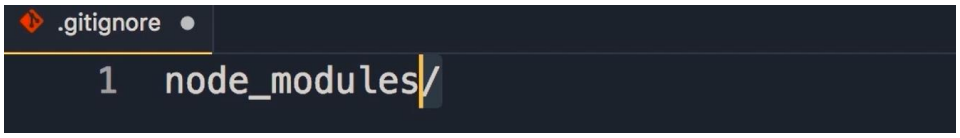
we use MLab to deploy our MangoDB database in the Cloud.

P193 Adding the code to a Git Repository

为了在heroku上发布应用,我们要先把代码添加到Git的仓库

First download Git.

Tell Git what files and folders should be ignored.



Commit the code to Git repository.

```
vidly-api-node $
vidly-api-node $git init
Initialized empty Git repository in /Users/mosh
fake/vidly-api-node/.git/
vidly-api-node $
vidly-api-node $git add .
vidly-api-node $git commit -m "Initial commit"
```

P194 Deploying to Heroku

Let Heroku create a name for us, use heroku create

The name is young-refuge-52186

The Address(the blue font) of our Application in herokku.com, our backend should hosted to this address.Our frontend should send request to this address/api/...

```
vidly-api-node $
vidly-api-node $heroku create
 ▸    heroku-cli: update available from 6.13.0 to 6
Creating app... done, ⬢ young-refuge-52186
https://young-refuge-52186.herokuapp.com/ | https:/
```

Use the local repository to update the remote repository.

```
vidly-api-node $git push heroku master
```

```
vidly-api-node $
vidly-api-node $heroku open
```

P195 Viewing logs

```
vidly-api-node $heroku logs
```

P196 Setting Enviroment Variables in Heroku

```
vidly-api-node $heroku config:set vidly_db=mongodb://vidlyuser:1234@ds012538.mlab.com:12538/vidly
```

P197 Preparing the Front-end for Deployment

config.json .env.production

```
1  REACT_APP_API_URL=https://young-refuge-52186.herokuapp.com/api
```

.env.production .env.development config.json

```
1  REACT_APP_API_URL=http://localhost:3900/api
```