

Install:

```
npm i -g create-react-app
```

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

```
create-react-app react-ap
```

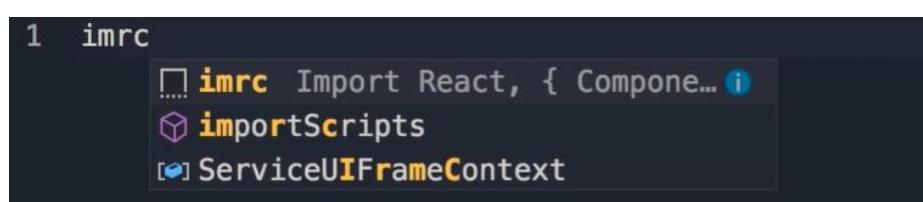
```
npm start
```

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>
))}

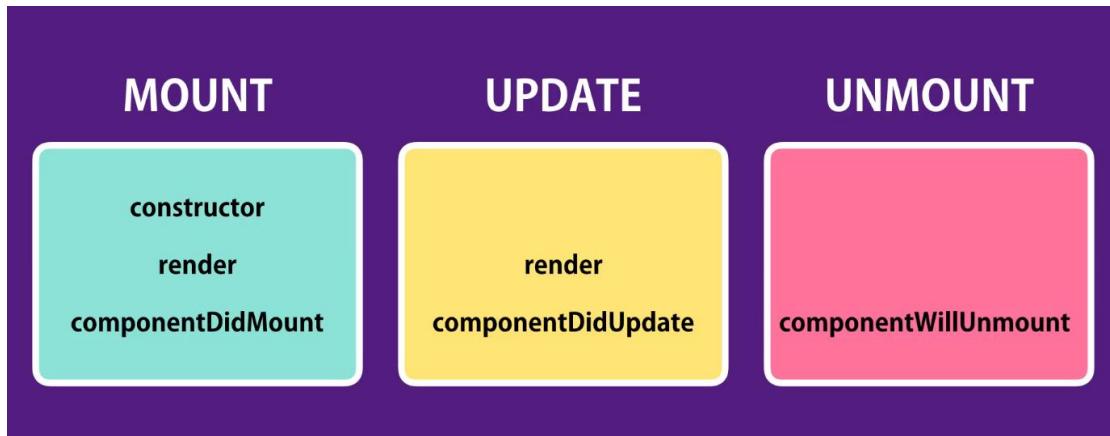
<td>button.btn.btn-danger.btn-sm</td>

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

```
{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>
```

```
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 default NavBar;

```

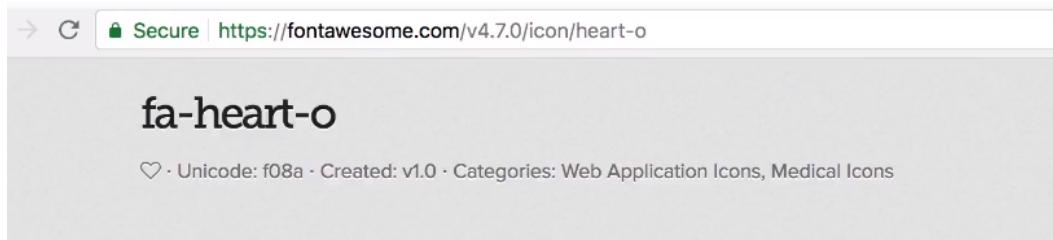
是你不能在无状态功能性组件中使用生命周期钩子

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

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

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

如果有变化，我们可以重新通过Ajax来向服务器取得数据



After you get [up and running](#), you can place Font Awesome icons just about anywhere with the `<i>` tag:

`<i class="fa fa-heart-o" aria-hidden="true"></i>`

Note: to improve [web accessibility](#), we recommend using `aria-hidden="true"` to hide icons used purely for decoration.

**i** Looking for more? Check out the [examples](#).  
回到 font-awesome 网站，拷贝下这段代码

SQU

Get a dom  
create a w  
with Squa  
Start your  
today.  
ads via Cart

# Getting Started

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

```
Pagination.propTypes = {
  itemCount: 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");
};




  <li>Link</li>

```

```
<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 [index.js:2178](#) 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>

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

```
▼ {error: Error: child "username" fails because ["username" is not allowed to be empty]. child "password" fails..., value: {...}, then: f, catch: f} ⓘ
  ► catch: f _catch(reject)
  ▼ error: Error: child "username" fails because ["username" is not allowed to be empty]. child
    ► annotate: f (stripColorCodes)
  ▼ details: Array(2)
    ▼ 0:
      ► context: {value: "", invalids: Array(1), key: "username", label: "username"}
        ► message: ""username" is not allowed to be empty"
      ► path: ["username"]
        type: "any.empty"
      ► __proto__: Object
    ▶ 1: {message: ""password" is not allowed to be empty", path: Array(1), type: "any.empt
      length: 2
    ► __proto__: Array(0)
  isJoi: true
  name: "Val"这里的details数组，每个项都有一个message属性
  ► _object: {username: "", password: ""}
  message: "child "username" fails because ["username" is not allowed to be empty]. child
  message: "child "password" fails because ["password" is not allowed to be empty]. child
```

```
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>
  );
}
```

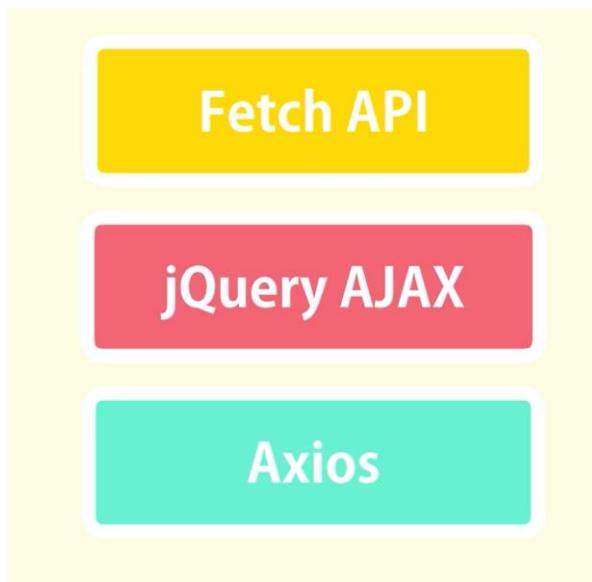
以上三个值用...rest 实现

```
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

```
schema = {
  username: Joi.string()
    .required()
    .email()
    .label("Username"),
  password: Joi.string()
    .required()
    .min(5) | I
    .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 occurred.");
  }
}
```

## 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 occurred.");
  }

  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';
```

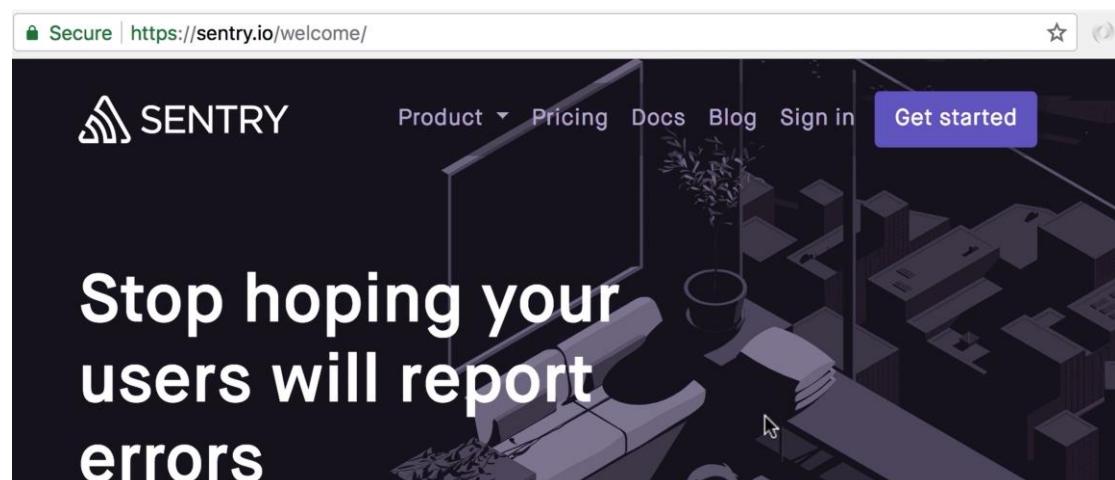
then render a ToastContainer

```
render() {  
  return (  
    <React.Fragment>  
      <ToastContainer />  
      <button className="btn btn-primary" type="button">...</button>  
    </React.Fragment>  
  );  
}
```

HttpService.js

```
import { toast } from 'react-toastify';  
  
toast.error("An unexpected error occurred.");  
  
toast("An unexpected error occurred.");
```

## P148 Log Error



Install

```
http-app $npm i raven-js@3.26.4  
(( ... )) :: loadIdealTree:load
```

Index.js

```
index.js •
1 import React from "react";
2 import ReactDOM from "react-dom";
3 import App from "./App";
4 import registerServiceWorker from "./registerServiceWorker";
5 import Raven from 'raven-js';
6 import "./index.css";
7 import "bootstrap/dist/css/bootstrap.css";
8
9 Raven.config('https://05323d37c9a947eba9daaaab1e6171a9@sentry.io/1249956', {
10   release: '1.0.0',
11   environment: 'development-test',
12 }).install()
```

## HttpService.js

```
index.js | httpService.js x
1 import axios from "axios";
2 import Raven from "raven-js";
3 import { toast } from "react-toastify";
4
5 axios.interceptors.response.use(null, error => {
6   const expectedError =
7     error.response &&
8     error.response.status >= 400 &&
9     error.response.status < 500;
10
11   if (!expectedError) {
12     Raven.captureException(error);
13     toast.error("An unexpected error occurred.");
14   }
15 })
```

## P149 Extracting a Logging Service

```
index.js • httpService.js • logService.js x
1 import Raven from "raven-js";
2
3 function init() {
4   Raven.config("https://05323d37c9a947eba9daaaab1e6171a9@sentry.io/1249956", {
5     release: "1.0.0",
6     environment: "development-test"
7   }).install();
8 }
9
10 function log(error) {
11   Raven.captureException(error);
12 }
13
14 export default {
15   init,
16   log
17 };
```

use it from other file

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

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

## P157 Get Genres from Database

```
genreService.js • fakeGenreService.js
1 import http from './httpService';
2
3 export function getGenres() {
4   return http.get('http://localhost:3900/api/genres');
5 }

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 x
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

```
JS genreService.js | (...) config.json | (...) config.json src
JS genreService.js src/services
1 {
2   "apiUrl": "http://localhost:3900/api"
3 }
```

```
JS genreService.js • (...) config.json
1 import http from './httpService';
2 import config from '../config.json';
3
4 export function getGenres() {
5   return http.get(config.apiUrl + "/genres");
6 }
```

```
JS genreService.js | JS movieService.js x | (...) config.json
1 import http from './httpService';
2 import { apiUrl } from '../config.json';
3
4 const apiEndpoint = apiUrl + "/movies";
5
6 export function getMovies() {
7   return http.get(apiEndpoint);
8 }
9
10 export function deleteMovie(movieId) {
11   return http.delete(apiEndpoint + "/" + movieId);
12 }
```

## 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

The screenshot shows a JWT token being decoded on jwt.io. The token itself is a long string of characters: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1YjYyMGYyNWZmZWRjNzBkZDIwYjEyNTgiLCJuYW1lIjoiTW9zaC1sImVtYWlsIjoidXNlcmFAZG9tYWluLmNvbSIzImlhdcI6MTUzMzE1MzA2MX0..uL5H9aJcq2fXLVLaLyJUx3nrZqbXDhgOutj0mX4OTEE. The interface is divided into two main sections: 'HEADER:' and 'PAYLOAD:'. The HEADER section contains the following JSON: { "alg": "HS256", "typ": "JWT" }. The PAYLOAD section contains the following JSON: { "\_id": "5b620f25ffedc70dd20b1258", "name": "Mosh", "email": "usera@domain.com", "iat": 1533153061 }. A callout box highlights the PAYLOAD section with the text: '第二部分是装载的内容,也就是左边实际携带的数据'.

## 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

```
{user && (
  <React.Fragment>
    <NavLink className="nav-item nav-link" to="/profile"
      {user.name}
    </NavLink>
    <NavLink className="nav-item nav-link" to="/logout">
      Logout
    </NavLink>
  </React.Fragment>
)}
```

我们也要对应的修改路由,保存修改

reload

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

render() {
  return (
    <div>
```

这就是我们说的应用的完全重载

## P177 Logout Funktion

退出登录的时候, 删除 token, 设置应用重载。

```
logout.jsx •
```

```
1 import React, { Component } from 'react';
2
3 class Logout extends Component {
4     componentDidMount() {
5         localStorage.removeItem('token');
6
7         window.location = '/';
8     }
9
10    render() {
11        return null;
12    }
13}
14
15 export default Logout;
```

```
logout.jsx App.js •
```

```
26 }
27
28 render() {
29     return (
30         <React.Fragment>
31             <ToastContainer />
32             <NavBar user={this.state.user} />
33             <main className="container">
34                 <Switch>
35                     <Route path="/register" component={RegisterForm} />
36                     <Route path="/login" component={LoginForm} />
37                     <Route path="/logout" component={Logout} />
38             </Switch>
39         </main>
40     );
41 }
```

## P179 Calling Protected API EndPoint

设置需要进行认证

The screenshot shows the VS Code interface with the Explorer sidebar on the left and a code editor on the right. The Explorer sidebar shows a project structure with a 'config' folder containing 'custom-environment-vari...', 'default.json', and 'test.json'. The code editor displays the 'default.json' file with the following content:

```
1 {  
2   "jwtPrivateKey": "unsecureKey",  
3   "db": "mongodb://localhost/vidly",  
4   "port": "3900",  
5   "requiresAuth": true  
6 }  
7
```

为了能从其他组件获取 token, 进行以下设置

The screenshot shows the VS Code interface with two tabs: 'httpService.js' and 'authService.js'. The code editor displays the 'authService.js' file with the following content:

```
22 try {  
23   const jwt = localStorage.getItem(tokenKey);  
24   return jwtDecode(jwt);  
25 } catch (ex) {  
26   return null;  
27 }  
28 }  
29  
30 export function getJwt() {  
31   return localStorage.getItem(tokenKey);  
32 }  
33
```

A red underline is present under the line 'return localStorage.getItem(tokenKey);' at line 31.

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

The screenshot shows the VS Code interface with two tabs: 'httpService.js' and 'authService.js'. The code editor displays the 'httpService.js' file with the following content:

```
1 import axios from "axios";  
2 import logger from "./logService";  
3 import auth from "./authService";  
4 import { toast } from "react-toastify";  
5  
6 axios.defaults.headers.common["x-auth-token"] = auth.getJwt();
```

A red underline is present under the line 'axios.defaults.headers.common["x-auth-token"] = auth.getJwt();' at line 6.

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)

```
JS httpService.js x JS authService.js
Bi-directional Dependencies
12     logger.log(error);
13     toast.error("An unexpected error occurred.");
14 }
15
16 return Promise.reject(error);
17 );
18
19 function setJwt(jwt) {
20     axios.defaults.headers.common["x-auth-token"] = jwt;
21 }
22
23 export default {
24     get: axios.get,
25     post: axios.post,
26     put: axios.put,
27     delete: axios.delete,
28     setJwt
29 };

```

现在回到 auth 服务

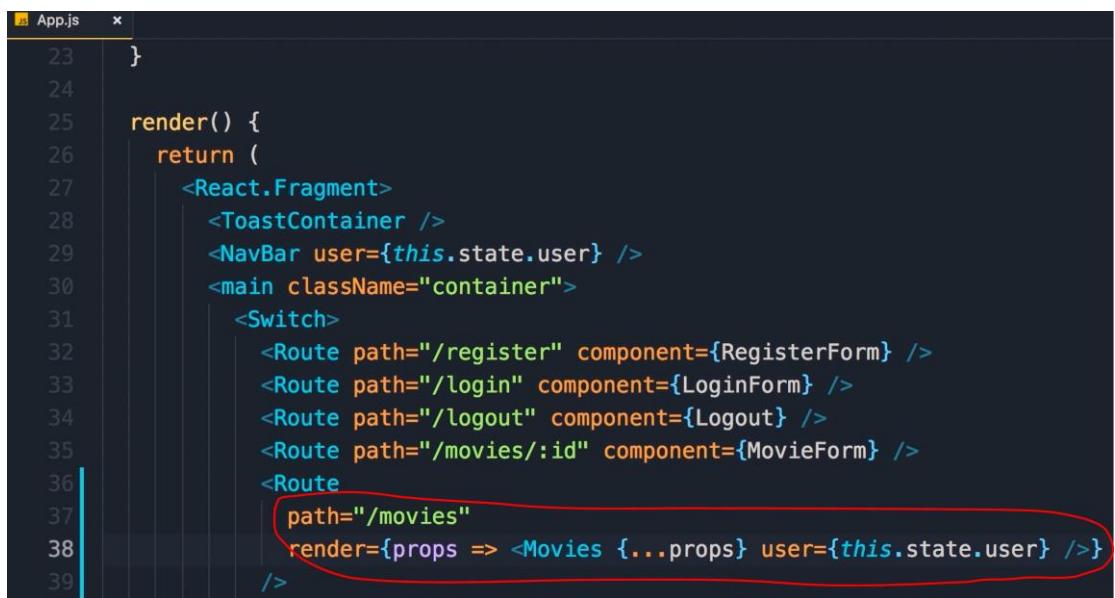
```
JS httpService.js x JS authService.js
Bi-directional Dependencies
1 import jwtDecode from "jwt-decode";
2 import http from "./httpService";
3 import { apiUrl } from "../config.json";
4
5 const apiEndpoint = apiUrl + "/auth";
6 const tokenKey = "token";
7
8 http.setJwt(getJwt());|
```

## 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

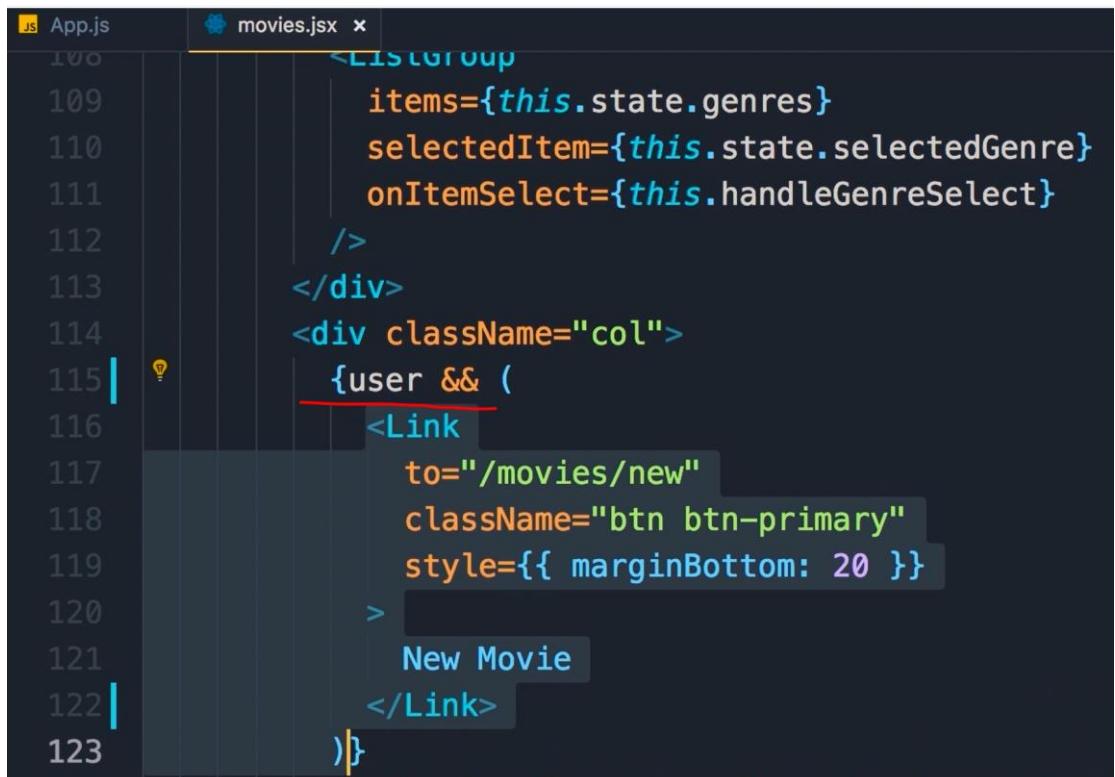
同样也不能给用户编辑和删除电影的权限

## P182 Showing or Hiding Element



```
23     }
24
25     render() {
26       return (
27       <React.Fragment>
28         <ToastContainer />
29         <NavBar user={this.state.user} />
30         <main className="container">
31           <Switch>
32             <Route path="/register" component={RegisterForm} />
33             <Route path="/login" component={LoginForm} />
34             <Route path="/logout" component={Logout} />
35             <Route path="/movies/:id" component={MovieForm} />
36             <Route
37               path="/movies"
38               render={props => <Movies {...props} user={this.state.user} />}
39             />
```

现在我们可以依据user对象来显示或隐藏元素了



```
100
101
102
103
104
105
106
107
108
109   <ListGroup
110     items={this.state.genres}
111     selectedItem={this.state.selectedGenre}
112     onItemSelect={this.handleGenreSelect}
113   />
114   </div>
115   <div className="col">
116     {user && (
117       <Link
118         to="/movies/new"
119         className="btn btn-primary"
120         style={{ marginBottom: 20 }}
121       >
122         New Movie
123       </Link>
124     )}
125   </div>
```

## P183 Protecting the Route

```
App.js x
29 <NavBar user={this.state.user} />
30 <main className="container">
31   <Switch>
32     <Route path="/register" component={RegisterForm} />
33     <Route path="/login" component={LoginForm} />
34     <Route path="/logout" component={Logout} />
35     <Route
36       path="/movies/:id"
37       render={props => {
38         if (!user) return <Redirect to="/login" />;
39         return <MovieForm {...props} />;}
40       }
41     />
```

## P184 Extracting Protected Route

```
App.js x protectedRoute.jsx x
1 import React from "react";
2 import { Route, Redirect } from "react-router-dom";
3 import auth from "../../services/authService";
4
5 const ProtectedRoute = ({ path, component: Component, render, ...rest }) => {
6   return (
7     <Route
8       {...rest}
9       render={props => {
10         if (!auth.getCurrentUser()) return <Redirect to="/login" />;
11         return Component ? <Component {...props} /> : render(props);
12       }}
13     />
14   );
15 };
16
17 export default ProtectedRoute;
```

```
App.js x protectedRoute.jsx
35 <Route path="/register" component={RegisterForm} />
36 <Route path="/login" component={LoginForm} />
37 <Route path="/logout" component={Logout} />
38 <ProtectedRoute path="/movies/:id" component={MovieForm} />
```

## P185 Redirecting after Login

Console

```

Object
  ▶ history: {length: 50, action: "REPLACE", location: {...}, createHref: f, push: f, ...}
  ▶ location:
    hash: ""
    key: "7hneuc"
    pathname: "/movies/5b58c8fca5ebf3212ead2f28"
    search: ""
    state: undefined
  ▶ match: Object
  ▶ matchPath: "/movies/:id"
  ▶ routes: Array[2]
  ▶ staticContext: null
  ▶ staticProps: null
  ▶ type: "ProtectedRoute"
  ▶ url: "/movies/5b58c8fca5ebf3212ead2f28"

protectedRoute.jsx
1 import React from "react";
2 import { Route, Redirect } from "react-router-dom";
3 import auth from "../../services/authService";

4
5 const ProtectedRoute = ({ path, component: Component, render, ...rest }) => {
6   return (
7     <Route
8       {...rest}
9       render={props => {
10       console.log(props);
11
12       if (!auth.getCurrentUser()) return <Redirect to={{
13         pathname: '/login',
14         state: { from: props.location }
15       }} />;
16       return Component ? <Component {...props} /> : render(props);
17     }}
18   />

```

```

protectedRoute.jsx
doSubmit = async () => {
  try {
    const { data } = this.state;
    await auth.login(data.username, data.password);
  }
  const { state } = this.props.location;
  window.location = state ? state.from.pathname : "/";
}

loginForm.jsx

```

## P186 Exercise

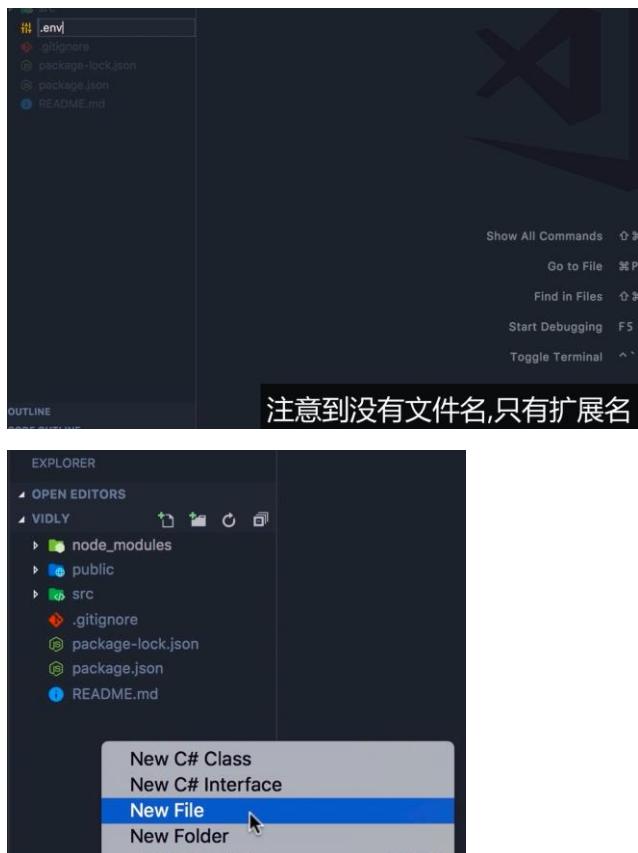
## P187 Hiding the delete Column

```
moviesTable.jsx •
33         </button>
34     )
35   };
36
37   constructor() {
38     super();
39     const user = auth.getCurrentUser();
40     if (user && user.isAdmin) this.columns.push(this.deleteColumn);
41   }
42
```

## P188 Deployment

## P189 Adding Environment 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  
environment variables.

All variables are `key and values` pairs.

All the keys start with `REACT_APP_`

`process` is current process, `env` represents all the environment  
variables.

Everytime we change the environment files, we need to restart the  
Application.

```
# .env      .env.development    index.js
1 import React from "react";
2 import ReactDOM from "react-dom";
3 import { BrowserRouter } from "react"
4 import "./index.css";
5 import App from "./App";
6 import registerServiceWorker from ".";
7 import "bootstrap/dist/css/bootstrap
8 import "font-awesome/css/font-awesom
9
10 console.log(process.env);
```

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 MongoDB in the Cloud

**we use MLab to deploy our MongoDB 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.

```
! .gitignore ●  
1 node_modules/
```

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 heroku.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
```

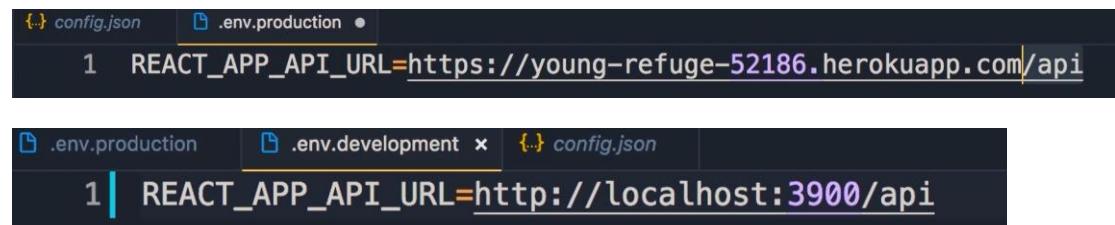
## P195 Viewing logs

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

## P196 Setting Environment Variables in Heroku

```
vidly-api-node $ heroku config:set vidly_db=mongodb://vidlyuser:1234@ds012538.mlab.com:1  
2538/vidly user password
```

## 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 x {...} config.json  
1 REACT_APP_API_URL=http://localhost:3900/api
```