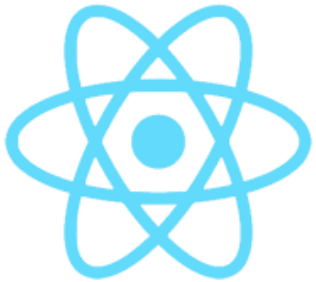
YASH Technologies

IssueTracker-app – Part 01



React

Presented by : Pankaj Sharma

Topics to be covered

- Pure function in React
- this keyword and it’s utility
- About issue-tracker
- Difference between state and props and their use
- Events and Uni Directional Data flow
- How to create and use Stateless functional component in React
- Add issue to a list of issues.
- Delete issue from the list of issues
- Edit issue in the list of issues
- Use of propType

Pure function

Pure function ensures the consistency and predictability because of the below characteristics.

- Pure function always returns the same result given the same arguments.
- Pure function's execution doesn't depend on the state of the application.
- Pure function don't modify the variable outside of their scope.

```
> function sum(x,y){
  return x+y;
  }
< undefined
> sum(10,20);
< 30
> var numbers=[1,2,3,4,5];
< undefined
> numbers.slice(0);
< ▶ (5) [1, 2, 3, 4, 5]
> numbers.slice(0,1);
< ▶ [1]
> numbers.slice(0,2);
< ▶ (2) [1, 2]
> numbers.splice(0,1);
< ▶ [1]
> numbers
< ▶ (4) [2, 3, 4, 5]
```

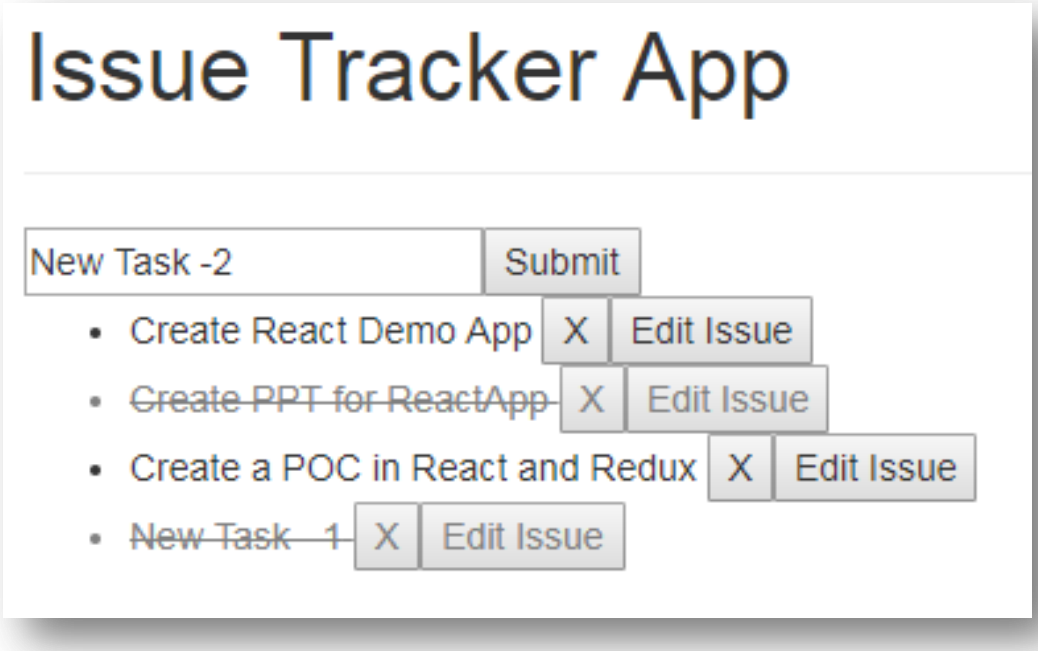
Refrence : https://tylermcginnis.com/building-user-interfaces-with-pure-functions-and-function-composition-in-react-js/

This keyword and it's utility

About issue-tracker

We will create an application which will allow us:

- to add issues
- show the list of issues
- edit the issue in separate edit view
- will give us functionality to toggle the status of the issue.



Difference between state and props

- State is the place where the data comes from.
- Keep the state as simple as possible.
- You should minimize the number of statefull components. For example if you have ten components that need data from ten state, you should create one container component that will keep the state for all of them.
- Let us see an example of the HeaderComponent with state and render that component somewhere else

Issue-tracker01/src/HeaderComponent.js

```
import React from 'react';
class HeaderComponent extends React.Component{
constructor(){
super();
this.state={
headerText:"Yash Training Management App",
"logo":"YTMS"
}}
render(){
return(
<div>
<h1>{this.state.headerText}</h1>
<h3>{this.state.logo}</h3>
</div>
);
}}
export default HeaderComponent;
```

Issue-tracker01/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import HeaderComponent from
'./HeaderComponent'

ReactDOM.render(<HeaderComponent/>,document
.getElementById("root"))
```

state and props - IssueTracker

Req #1: show one Issue in unordered list.

```
Issue-tracker01/src/index.js
-------------------------------------------
import React from 'react';
import ReactDOM from 'react-dom';
import HeaderComponent from './HeaderComponent';
class IssueListComponent extends React.Component{

constructor(){
super();
this.state={
issue:"Create React Demo App"
}
}
render(){
return(
<ul>
<li>{this.state.issue}</li>
</ul>
)
}
}

ReactDOM.render(<IssueListComponent/>,document.getElementById("root"));
```

state and props - IssueTracker

Req #2: Pass the issue name in child component, and child component will return the list item.

Issue-tracker01/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import HeaderComponent from './HeaderComponent.jsx';
class IssueListComponent extends React.Component{
constructor(){
super();
this.state={
issue:"Create React Demo App!"
}
}
render(){
return(
<ul>
<IssueNameComponent issue={this.state.issue}/>
</ul>
)
}
}
```

Continue. . .

```
class IssueNameComponent extends
React.Component{
render(){
return(
<li>{this.props.issue}</li>
)
}
}

ReactDOM.render(<IssueListComponent/>,documen
t.getElementById("root"));
```

state and props - IssueTracker

Req #3: Create Issue list, iterate it in parent component and pass the issue name to child component. Child component will return the name of the issue. On page issue list should be displayed.

Issue-tracker01/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import HeaderComponent from './HeaderComponent.jsx';
class IssueListComponent extends React.Component{
constructor(){
super();
this.state={
issues:["Create React Demo App","Create PPT for
ReactApp","Create a POC in React and Redux"]
}
}
render(){
return(
<ul>
{
this.state.issues.map(function(issue){
return <IssueNameComponent issue={issue}/>
})
}
</ul>
)
}
}
```

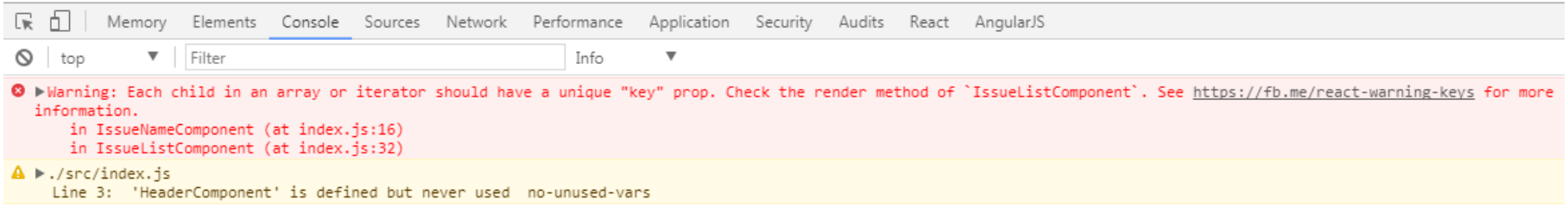
Continue. . .

```
class IssueNameComponent extends
React.Component{
render(){
return(
<li>{this.props.issue}</li>
)
}
}

ReactDOM.render(<IssueListComponent/>,documen
t.getElementById("root"));
```

state and props - IssueTracker

Req #3:
If you refresh the browser, you must get the changes. But there is one issue. Open the browser developer tool and check the console.



This warning is coming because React DOM will check which component need to be rendered based on some unique checking. This warning message can be removed by adding the key attribute while displaying the array element.

```
Issue-tracker01/src/index.js -  IssueListComponent
-------------------------------------------
{
this.state.issues.map(function(issue){
return <IssueNameComponent key={issue} issue={issue}/>
})
}
```

Key will add the uniqueness in record. In real apps, you will replace it with some backend code.

Events and Uni Directional Data Flow - IssueTracker

Event Ref : https://facebook.github.io/react/docs/handling-events.html

Req : Make small change in your application. Your issues list should be of object type with two properties. 1. name and 2. completed. Accordingly make changes in IssueListComponent and IssueNameComponent

Issue-tracker02/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
class IssueListComponent extends React.Component{
constructor(){ super();
this.state={
issues:[
{name:"Create React Demo App",completed:false},
{name:"Create PPT for ReactApp",completed:false},
{name:"Create a POC in React and
Redux",completed:false}
]}}
render(){  return( <ul>
{
this.state.issues.map(function(issue){
return <IssueNameComponent key={issue.name}
issue={issue}/>
})
} </ul>
)}}
```

Continue. . .

```
class IssueNameComponent extends
React.Component{
render(){
return(
<li>{this.props.issue.name}</li>
)
}
}

ReactDOM.render(<IssueListComponent/>,documen
t.getElementById("root"));
```

Events and Uni Directional Data Flow - IssueTracker

Req: Now we want to click on list item, and its completed state should be changed. We will also be changing its style. Here our purpose is to handle the event.
Note : child component can not directly change the state. To make any changes in state, child component will notify the parent component.
Steps:
1: very first add one class 'completed' in tag of ***IssueNameComponent*** which will be added dynamically as the user click on <li> item.

```
Issue-tracker02/src/index.js -  IssueListComponent
-------------------------------------------
class IssueNameComponent extends React.Component{
render(){
return(
<li className={this.props.issue.completed ? 'completed' : ''}>
{this.props.issue.name}
</li>
)
}
}
```

Here if issue completed status is true then completed class will be added otherwise nothing will be added.

Events and Uni Directional Data Flow - IssueTracker

2: now create changeStatus() method in **IssueListComponent.** It should be created above render() method.

```
Issue-tracker02/src/index.js -  IssueListComponent
-----------------------------------------------
class IssueListComponent extends React.Component{
. . . .
changeStatus(){
}
. . . .
}
```

- Now in this custom method we want to access this.state.
- Directly we can not access value of this, because its value is changed in custom method.
- To access the value of this, we will have to bind it with changeStatus() method in constructor it self.

```
Issue-tracker02/src/index.js -  IssueListComponent
-----------------------------------------------
class IssueListComponent extends React.Component{
constructor(){
super();
this.changeStatus=this.changeStatus.bind(this);
this.state={
issues:[
. . . .
]
}}
```

Events and Uni Directional Data Flow - IssueTracker

Now when the event will be triggered on <li> element changeStatus() method should be invoked, for that we need to pass clicked element's index to changeStatus() method. We will be passing the event handler to child component using props.
First make changes to your changeStatus() method as below.

```
Issue-tracker02/src/index.js -  IssueListComponent
-------------------------------------------
class IssueListComponent extends React.Component{
. . . .
changeStatus(index){
console.log(this.state.issues[index]);
}
. . . .
}
```

Here changeStatus() method is taking one index and out of clicked issues console.log() will display the information of clicked issue object with its completed status.

Add clickHandler attribute in <IssueNameComponent /> as below.

```
Issue-tracker02/src/index.js -  IssueListComponent
-------------------------------------------
{
this.state.issues.map(function(issue){
return <IssueNameComponent key={issue.name} issue={issue} clickHandler={this.changeStatus}/>
})
}
```

Events and Uni Directional Data Flow - IssueTracker

Let's add onClick() event on <li> which will execute one function which will be calling clickHandler property provided by the parent component, here we will have to pass the index of clicked list element.

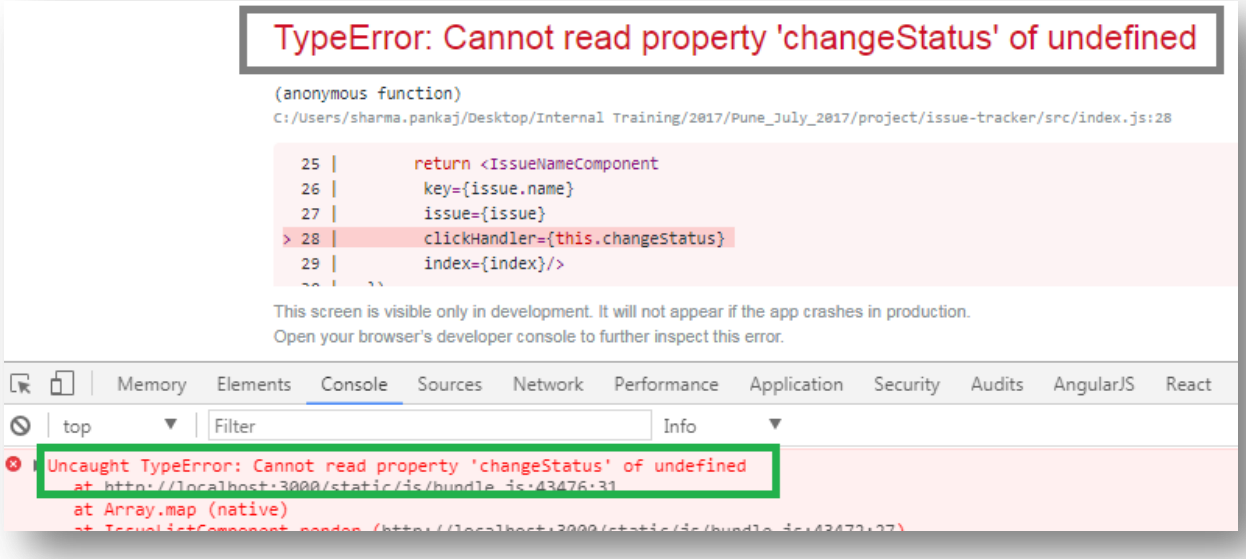
```
Issue-tracker02/src/index.js -  IssueNameComponent
-------------------------------------------
class IssueNameComponent extends React.Component{
render(){
return(
<li onClick={()=>{
this.props.clickHandler(this.props.index)
}} className={this.props.issue.completed ? 'completed' : ''}>
{this.props.issue.name}
</li>
)
}
}
```

Now as you can see that we have added this.props.index, which should be passed as a props in <IssueNameComponent/>

```
Issue-tracker02/src/index.js -  IssueListComponent
-------------------------------------------
this.state.issues.map(function(issue,index){
return <IssueNameComponent
key={issue.name} issue={issue}  clickHandler={this.changeStatus}  index={index}/>
})
```

Now check your browser, you should get one error.

Events and Uni Directional Data Flow - IssueTracker



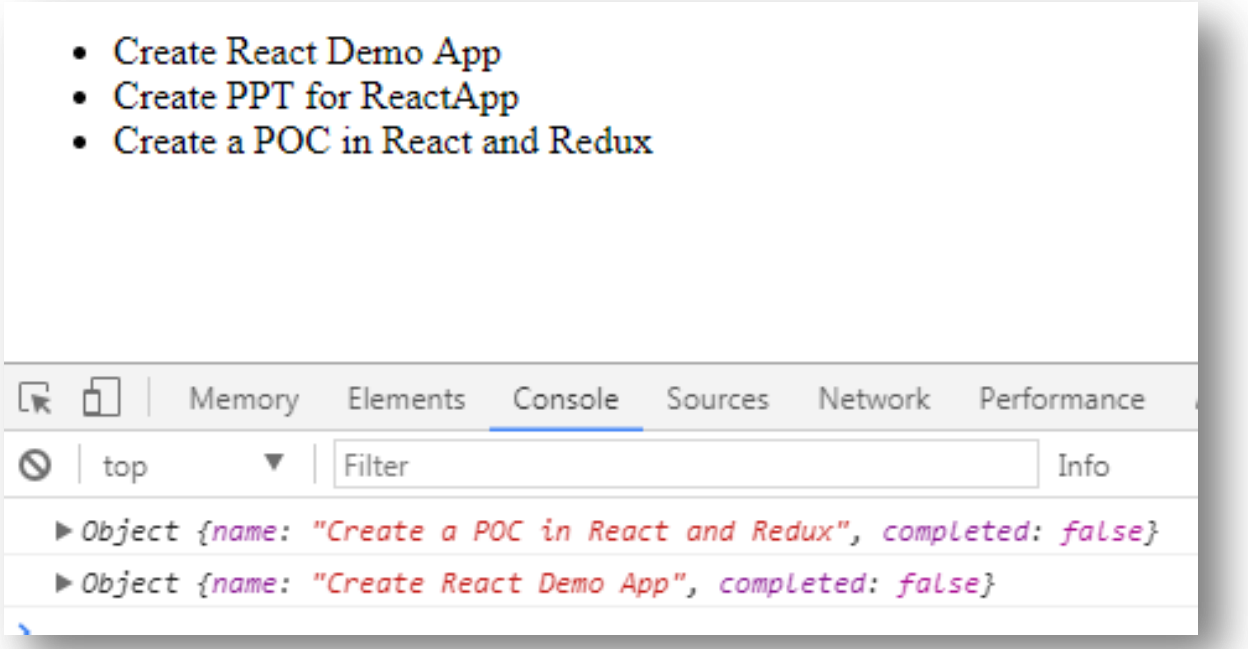
- This is because this.changeStatus is not accessible inside .map(function(issue,index)).
- To make it accessible you will have to use the => function.
- Do the below changes in your code and check the browser.

```
Issue-tracker02/src/index.js -  IssueListComponent
-------------------------------------------
this.state.issues.map((issue,index)=>{
return <IssueNameComponent
key={issue.name} issue={issue}  clickHandler={this.changeStatus}  index={index}/>
})
```

- Now that problem will be resolved.

Events and Uni Directional Data Flow - IssueTracker

- Check on chrome browser.
- Open the browser console.
- Click on any list item.
- In console you should get which list item is clicked.



Events and Uni Directional Data Flow - IssueTracker

Uptill now we are able to get the clicked object.
Req : now we want to change the completed status of clicked list. We will have to implement the below logic in changeState() method.

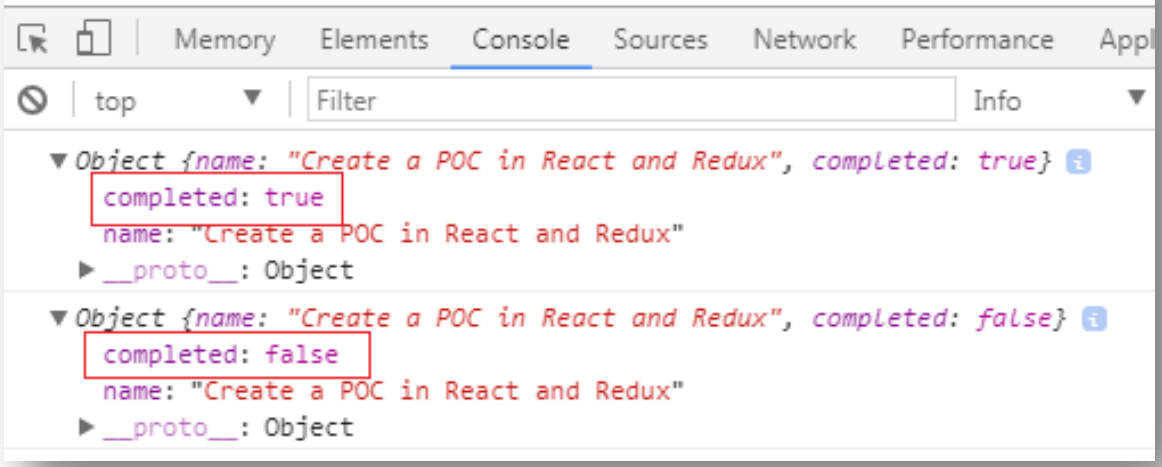
- Copy all issues in separate issues variable.
- Get the clicked issue from issues.
- Change the completed status of clicked issue
- Change the complete list of issues by setState

```
Issue-tracker02/src/index.js -  IssueListComponent
-------------------------------------------
changeStatus(index){
var issues=this.state.issues;
var issue=issues[index];
issue.completed=!issue.completed;
this.setState({
issues:issues
})
console.log(this.state.issues[index])
}
```

Open the browser, open browser console and now click on any list item twice, and check the completed status in browser.
We can remove console.log() from production code, as this is only for testing.

Events and Uni Directional Data Flow - IssueTracker

- Create React Demo App
- Create PPT for ReactApp
- Create a POC in React and Redux



Events and Uni Directional Data Flow - IssueTracker

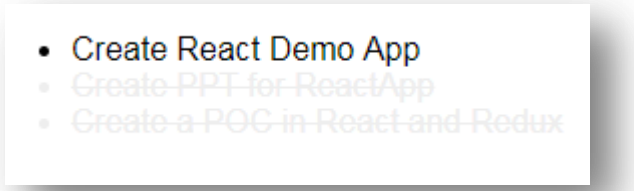
Req: now let us add some CSS to see the completed issues effect.

Add style for completed class in index.css
Import index.css in index.js file.

```
Issue-tracker02/src/index.js
-----------------------------------------
import './index.css'
```

```
Issue-tracker02/src/index.css
-----------------------------------------
.completed{
text-decoration: line-through;
color:#eee;
}
```

Now check on browser.



Thank You!

Email: info@yash.com
Web: www.yash.com

© YASH Technologies, 1996-2013. All rights reserved.

The information in this document is based on certain assumptions and as such is subject to change. No part of this document may be reproduced, stored or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of YASH Technologies Inc. This document makes reference to trademarks that may be owned by others. The use of such trademarks herein is not as assertion of ownership of such trademarks by YASH and is not intended to represent or imply the existence of an association between YASH and the lawful owners of such trademarks.

Presented by : Pankaj Sharma