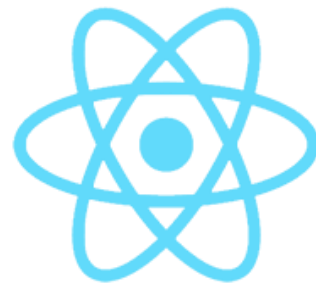




IssueTracker-app – PART 2



React

- 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 propTypes

Req: uptill now we have two components and we have written both these components in one file. We want to organize the custom components in different location.

Keeping all components in one file is not a good approach.

- Create a **component** folder in src
- Create a file named as IssueNameComponent.js
- Copy IssueNameComponent from index.js file and paste in IssueNameComponent.js file
- Import React from 'react' in IssueNameComponent.
- Export IssueNameComponent in IssueNameComponent.jsx file.
- Import IssueNameComponent in index.js file

```
Issue-tracker02/src/component/IssueNameComponent.js
-----
import React from 'react';
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>
    )
  }
}
export default IssueNameComponent;
```

```
Issue-tracker02/src/index.js
-----
import IssueNameComponent from './component/IssueNameComponent.js';
```

We have also used the bootstrap for better UI.

- Copy bootstrap cdn from bootstrap
- Add the link in index.html
- Wrap the root div in container div.

```
Issue-tracker02/public/index.html
-----
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.mi
n.css" >
<div class="container">
<div id="root"></div>
</div>
```

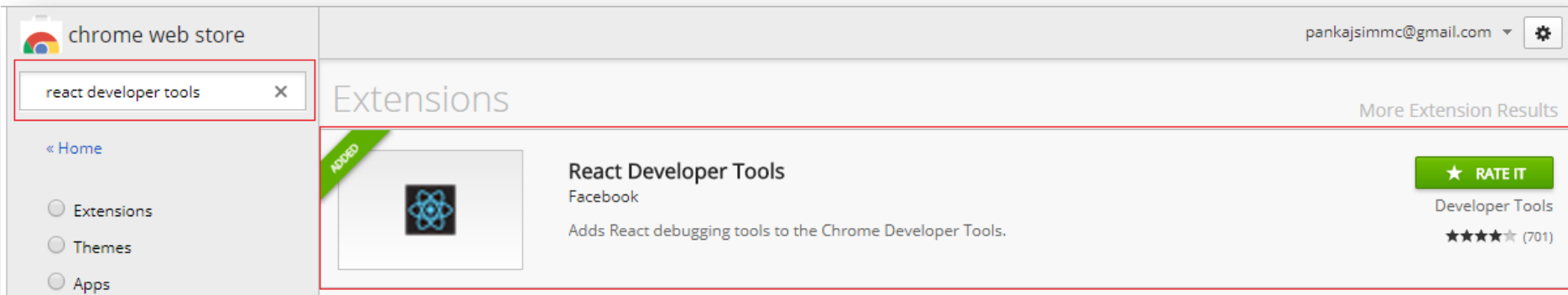
Refresh and check your browser. Check functionality as well.

Add issues to list of issues - IssueTracker

5

At this point we need to add one extension to chrome.

- Open chrome web extension from <https://chrome.google.com/webstore/category/extensions>
- Search for React Developer Tools
- Add the extension in chrome



Req : add IssueForm with text box and a submit button to add new issue.

- Create one IssueForm.js file in component
- Create one stateless IssueForm component with basic `<h1>Issue Form</h1>` return statement and export it.
- Import IssueForm on index.js and call `<IssueForm/>` before the code where Issues are listed.

Issue-tracker03/src/component/IssueForm.js

```
-----  
import React from 'react';  
const IssueForm = () =>{  
  return (  
    <h2>Issue Form</h2>  
  )  
}  
export default IssueForm;
```

Issue-tracker03/src/index.js

```
-----  
. . . . .  
import IssueForm from './component/IssueForm.js';  
import './index.css'  
class IssueListComponent extends React.Component{  
  . . . . .  
  render(){  
    return(  
      <div>  
        <h1>Issue Tracker App</h1>  
        <hr/>  
        <section>  
          <IssueForm/>  
          <ul>  
            . . .  
ReactDOM.render(<IssueListComponent/>,document.getEleme  
ntById("root"));
```

- Open the React Developer Tools
- Click on IssueListComponent, you will find the IssueForm component

The screenshot shows the React Developer Tools interface. The top part displays the 'Issue Tracker App' and 'Issue Form' sections. The 'Issue Form' section contains a list of issues:

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

The bottom part of the screenshot shows the component tree and the React state. The component tree is expanded to show the `<IssueListComponent>` component, which is the parent component. It contains a `<div>` element, which is the child component. The `<div>` element contains a `<h1>` element and a `<section>` element. The `<section>` element contains a `<IssueForm>` component and a `` element. The `` element contains three `<IssueNameComponent>` components.

The React state is shown on the right, indicating an empty object for props and an array of three issues for state.

here in parent component we want a currentIssue state which can be passed to child component, and then child component will be able to update that state.

- Add `currentIssue:` as one more state in the state of `IssueListComponent`.
- Create `updateIssue()` method and bind with this ref same as `changeStatus`.
- `updateIssue()` method will receive one `newValue`.
- `newValue` in `updateIssue()` will modify the `currentIssue`.
- `currentIssue` state and `updateIssue()` method will be passed to `IssueForm` component using props.

Issue-tracker03/src/index.js - IssueListComponent

```
-----  
constructor(){  
  super();  
  this.changeStatus=this.changeStatus.bind(this);  
  this.updateIssue=this.updateIssue.bind(this);  
  this.state={  
    issues:[. . . ],  
    currentIssue:''  
  }  
}
```

```
updateIssue(newValue){  
  this.setState({  
    currentIssue:newValue  
  })  
}
```

Issue-tracker03/src/index.js - IssueListComponent

```
-----  
render(){  
  return(  
    . . . .  
    <IssueForm  
      currentIssue={this.state.currentIssue}  
      updateIssue={this.updateIssue}/>  
    . . . .  
  )  
}
```


- Open the React Developer Tool and observe the changes.

Issue Tracker App

Issue Form

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

The screenshot shows the React Developer Tools interface. The component tree on the left highlights the `<IssueListComponent>` component. The state panel on the right shows the state of the `IssueListComponent`, which includes `currentIssue` (an empty string) and `issues` (an array of 3 objects). Annotations with arrows point from the state panel to the `currentIssue` prop in the `<IssueForm>` component in the tree, and from the `issues` state to the `issues` prop in the `<IssueListComponent>` tree.

```
<IssueListComponent> == $r
  <div>
    <h1>Issue Tracker App</h1>
    <hr />
    <section>
      <IssueForm currentIssue="" updateIssue=bound updateIssue()>...</IssueForm>
      <ul>
        <IssueNameComponent key="Create React Demo App" issue={name: "Create React Demo App"} />
        <IssueNameComponent key="Create PPT for ReactApp" issue={name: "Create PPT for ReactApp"} />
        <IssueNameComponent key="Create a POC in React and Redux" issue={name: "Create a POC in React and Redux"} />
      </ul>
    </section>
  </div>
</IssueListComponent>
```

Props
Empty object

State
`currentIssue: ""`
`issues: Array[3]`

in the state of
`IssueListComponent`
`currentIssue` added.

`currentIssue` and `updateIssue()` is
passed in `<IssueForm />`

- Now we need to create `<form>` with textbox and button control in `IssueForm` component.
- Text box will have `currentIssue` as value, and `updateIssue` as method binding on `onChange` event.
- `currentIssue` and `updateIssue()` method will be accessible using props
- props will be available on `IssueForm` component as an argument.

```
Issue-tracker03/src/component/Issueform.js
```

```
-----  
import React from 'react';  
const IssueForm = (props) =>{  
  return (  
    <div>  
      <form>  
        <input type="text"  
          value={props.currentIssue}  
          onChange={props.updateIssue}  
        />  
      </form>  
    </div>  
  )  
}  
export default IssueForm;
```

- Go on browser
- Open React Developer Tool
- Type something in text box.

Issue Tracker App

[object Object]

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

The screenshot shows the React DevTools interface. The top bar includes tabs for Memory, Elements, Console, Sources, Network, Performance, Application, and React. The React tab is active, showing the component tree. The selected component is `<IssueListComponent>_...</IssueListComponent>`. The state of this component is displayed on the right, showing `currentIssue` as a `SyntheticEvent{...}` and `issues` as an `Array[3]`. The console on the left shows an error: `Instead of getting the value in CurrentIssue, we are getting SyntheticEvent{...}. this is because in the updateIssue(newValue) method, we are assigning newValue to currentIssue. This is not the value assignment, here newValue is an event. we need to assign the value of targeted source.`

Memory Elements Console Sources Network Performance Application React >>

Highlight Updates Highlight Search

Search (text or /regex/)

`<IssueListComponent>_...</IssueListComponent> == $r`

Props
Empty object

State

- ▶ `currentIssue`: `SyntheticEvent{...}`
- ▶ `issues`: `Array[3]`

Instead of getting the value in CurrentIssue, we are getting SyntheticEvent{...}. this is because in the updateIssue(newValue) method, we are assigning newValue to currentIssue. This is not the value assignment, here newValue is an event. we need to assign the value of targeted source.

- Make below changes in your updateIssue() method, and then try to type something in text box.

```
Issue-tracker03/src/index.js
-----
updateIssue(newValue){
  this.setState({
    currentIssue:newValue.target.value
  })
}
```

Issue Tracker App

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

The screenshot shows a web browser displaying the 'Issue Tracker App'. At the top, there is a text input field containing 'new task'. Below the input field, there is a list of three items: 'Create React Demo App', 'Create PPT for ReactApp', and 'Create a POC in React and Redux'. Below the browser window, the React DevTools component inspector is open, showing the 'React' tab. The component tree on the left shows the selected component as '<IssueListComponent>_</IssueListComponent> == \$r'. The 'Props' section on the right shows an 'Empty object'. The 'State' section on the right shows 'currentIssue: "new task"' and 'issues: Array[3]'. A text box at the bottom of the DevTools window states: 'once you type something in text box. that much will be reflected. this will be verified by observing changes in the currentIssue state while typing.'

- Now we will add the added issue in issues list.
- Create addIssue() method in IssueListComponent

```
Issue-tracker03/src/index.js
-----
addIssue(){
}
```

- Bind the addIssue() method with “this” ref so that addIssue() method can access “this” ref.

```
Issue-tracker03/src/index.js
-----
this.addIssue=this.addIssue.bind(this);
```

- Pass the addIssue() method as a prop to <IssueForm/> tag so that further in IssueForm component that can be available.

```
Issue-tracker03/src/index.js
-----
<IssueForm
currentIssue={this.state.currentIssue}
updateIssue={this.updateIssue}
addIssue={this.addIssue}/>
```

- Add onSubmit event to form in IssueForm component and pass addIssue() method using props to onSubmit event.

```
Issue-tracker03/src/component/IssueForm.js
-----
const IssueForm = (props) =>{
  return (
    <div>
      <form onSubmit={props.addIssue}>
        <input type="text"
          value={props.currentIssue}
          onChange={props.updateIssue}
        />
      </form>
    </div>
  )
}
```

- Create a submit button.
- Note : onSubmit event will be fired if you press Enter key or click the submit button.

```
Issue-tracker03/src/component/IssueForm.js
-----
const IssueForm = (props) =>{
  return (
    <div>
      <form onSubmit={props.addIssue}>
        <input type="text"
          value={props.currentIssue}
          onChange={props.updateIssue}
        />
        <button type="submit">Submit</button>
      </form>
    </div>
  )
}
```

- Now add a `console.log('-----addIssue triggered-----');` statement in `addIssue()` method and click on submit or either press Enter, and see whether `addIssue()` method triggered or not.

```
Issue-tracker03/src/index.js
-----
addIssue(){
  console.log('-----Add Issue Triggered-----');
}
```

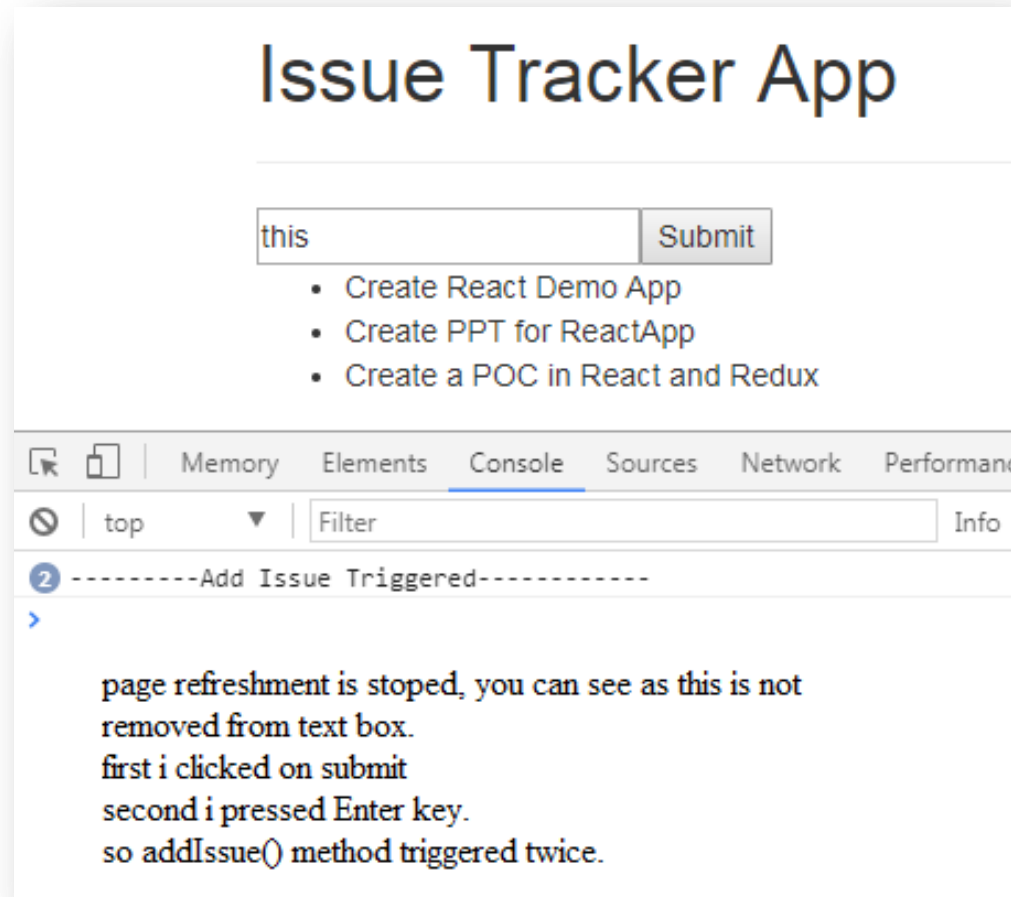
Check the console, nothing will happen. This is because `addIssue()` method should have a ref of event triggered. We need one argument in `addIssue()` method.

```
Issue-tracker03/src/index.js
-----
addIssue(event){
  console.log('-----Add Issue Triggered-----');
}
```



Write something in text box and click on submit. Still nothing `addIssue()` seems not to be triggered. But actually it is triggered, now the issue is that page is refreshed. We need to stop the page refresh. Make below changes in your `addIssue()` method.

```
Issue-tracker03/src/index.js
-----
addIssue(event){
  event.preventDefault();
  console.log('-----Add Issue Triggered-----');
}
```


- Now check the browser.




Req : We want to push the currentIssue in Issues array and would like to make currentIssue in default state.

```
addIssue(event){  
  event.preventDefault();  
  // console.log('-----Add Issue Triggered-----');  
  let issues=this.state.issues;  copying issues array into local issues variable.  
  let currentIssue=this.state.currentIssue;  copying currentIssue in local currentIssue variable.  
  


issues.push({  
  name:currentIssue,  
  completed:false  
})

 create new object with currentIssue and completed status by default as false and push that object in local issues array  
  


this.setState({  
  issues:issues,  
  currentIssue:''  
})

 assign local issues array into issues state of IssueListComponent. and put the currentIssue in default state.  
  
}
```

Now check your application, you should be able to add the new task and text box will be empty. You will be able to add task either by pressing Enter key or by pressing submit Button.

Req: we want to create a delete button with each issue, and when user click on delete button. the corresponding issue must be deleted.

- Create deleteIssue() method in index.js file and bind with “this” reference.

```
Issue-tracker04/src/index.js
-----
. . .
this.deleteIssue=this.deleteIssue.bind(this);
. . .
deleteIssue(){
}
```

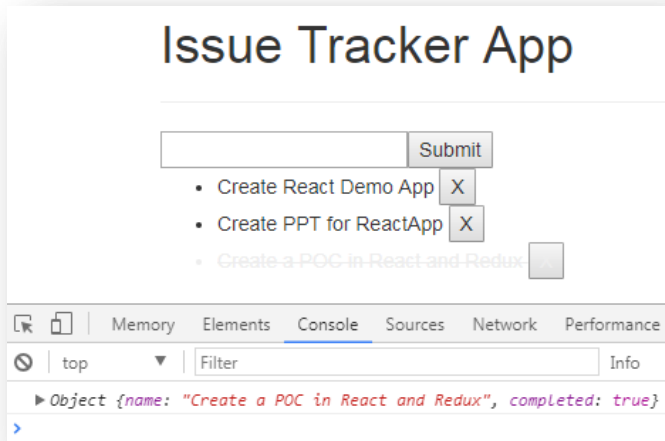
- We need to pass deleteIssue() to <IssueNameComponent> tag.

```
Issue-tracker04/src/index.js
-----
return <IssueNameComponent
key={issue.name}
issue={issue}
clickHandler={this.changeStatus}
deleteIssue={this.deleteIssue}
index={index}/>
})
```

- Create a delete button in IssueNameComponent and apply onClick event on button which will trigger the deleteIssue() method.

```
Issue-tracker04/src/component/IssueNameComponent.js
-----
<li onClick={()=>{
  props.clickHandler(props.index)
}} className={props.issue.completed ? 'completed' : ''}>
  {props.issue.name}&nbsp;  
  <button onClick={props.deleteIssue}>X</button>
</li>
```

- Now if you go on browser you will get the changes. Now click on Delete button.
- By clicking on delete button list item will be changed.
- Here we need to first work on clicked source.



- Which element is clicked for delete request, for that we need to apply the same technique as we have done for which element is clicked for changing the status.

```
Issue-tracker04/src/component/IssueNameComponent.js
-----
<li onClick={()=>{
  props.clickHandler(props.index)
}} className={props.issue.completed ? 'completed' : ''}>
  {props.issue.name}&nbsp;
  <button
    onClick={()=>{
      props.deleteIssue(props.index)
    }}>X</button>
</li>
```

- Now to varify the clicked item, add below code in deleteIssue() method.

```
Issue-tracker04/src/component/IssueNameComponent.js
-----
<li onClick={()=>{
  props.clickHandler(props.index)
}} className={props.issue.completed ? 'completed' : ''}>
  {props.issue.name}&nbsp;
  <button
    onClick={()=>{
      props.deleteIssue(props.index)
    }}>X</button>
</li>
```

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