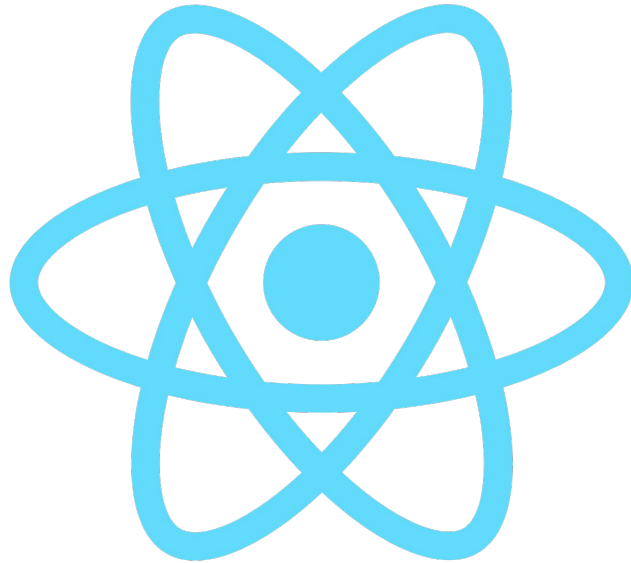


Advanced React.js Training v3



Trainer: Fritz Lim



Website: www.tertiarycourses.com.sg
Email: enquiry@tertiaryinfotech.com

About the Trainer

With more than 10 years of experience teaching at a local polytechnic, Fritz is passionate about imparting knowledge to teens, adults, and children. He believes that an education in technology and in how things work is essential for everyone, so that they can harness and invent the technologies of the future.

He is excited about exploring anything related to computers and IT, with a keen interest in electronics and native cross-platform mobile app development so that our ubiquitous mobile phones can be conveniently used to control and interact with devices wirelessly and over the Internet.

Fritz is ACTA-certified, as well as a registered MOE instructor, and a graduate of the 2018 ConsenSys Blockchain Developer Program, with a Bachelor in Electrical and Electronic Engineering (Computer Specialisation) from Nanyang Technological University.



Let's Know Each Other...

Say a bit about yourself

- Name
- What Industry you are from?
- Do you have any prior knowledge in react?
- Why do you want to learn react?
- What do you expect to learn from this course?

Ground Rules

- Set your mobile phone to silent mode
- Actively participate in the class. No question is stupid.
- Respect each other view
- Exit the class silently if you need to step out for phone call, toilet break

Ground Rules for Virtual Training

- Upon entering, mute your mic and turn on the video. Use a headset if you can
- Use the 'raise hand' function to indicate when you want to speak
- Participant actively. Feel free to ask questions on the chat whenever.
- Facilitators can use breakout rooms for private sessions.



Guidelines for Facilitators

1. Once all the participants are in and introduce themselves
2. Goto gallery mode, take a snapshot of the class photo - makes sure capture the date and time
3. Start the video recording (only for WSQ courses)
4. Continue the class
5. Before the class end on that day, take another snapshot of the class photo - makes sure capture the date and time
6. For NRIC verification, facilitator to create breakout room for individual participant to check (only for WSQ courses)
7. Before the assessment start, take another snapshot of the class photo - makes sure capture the date and time (only for WSQ courses)
8. For Oral Questioning assessment, facilitator to create breakout room for individual participant to OQ (only for WSQ courses)
9. End the video recording and upload to cloud (only for WSQ courses)
10. Assessor to send all the assessment records, assessment plan and photo and video to the staff (only for WSQ courses).

Prerequisite

Learners are assumed to have the following knowledge

- Basic javascript

Agenda

Topic 1 Handling Events

- Event Binding
- Passing Arguments to Event Handlers

Topic 2 Single Page Application Using React Router

- Single Page Application
- Installing React Router
- Set up React Router

Topic 3 Lists and Keys

- Loop a List with Map
- Keys
- Refs

Topic 4 Hooks (Optional)

Google Classroom

- Goto google classroom
<https://classroom.google.com>
- Enter the class code below to join the class on the top right.
- If you cannot access the google classroom, please inform the trainer or staff.

6mrh7ql

Topic 1

Handling Events

Handling Events

Handling events with React elements is very similar to handling events on DOM elements. There are some syntax differences:

- React events are named using camelCase, rather than lowercase.
- With JSX you pass a function as the event handler, rather than a string. Example

```
<button onClick={activateLasers}>  
  Activate Lasers  
</button>
```

- Another difference is that you cannot return false to prevent default behavior in React. You must call `preventDefault` explicitly. Example:

```
function ActionLink() {  
  function handleClick(e) {  
    e.preventDefault();  
    console.log("The link was clicked.");  
  }  
  return (  
    <a href="#" onClick={handleClick}>  
      Click me  
    </a>  
  );  
}
```

Handling Events

- `e` is a synthetic event. React defines these synthetic events according to the W3C spec, so you don't need to worry about cross-browser compatibility. React events do not work exactly the same as native events.
- When using React, you generally don't need to call `addEventListener` to add listeners to a DOM element after it is created. Instead, just provide a listener when the element is initially rendered.
- When you define a component using an ES6 class, a common pattern is for an event handler to be a method on the class.

Event Binding

- Event also requires a custom function to change the state value.
- Use `setState` to change the state value

```
handleClick() {  
  this.setState(state => ({  
    isToggleOn: !state.isToggleOn  
  }));  
}
```

- To make `this` work in the callback, this following binding is necessary

```
this.handleClick = this.handleClick.bind(this);
```

Event Handling Example 1

```
class Toggle extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = {isToggleOn: true};  
    this.handleClick = this.handleClick.bind(this);  
  }  
  
  handleClick() {  
    this.setState(state => ({  
      isToggleOn: !state.isToggleOn  
    }));  
  }  
  
  render() {  
    return (  
      <button onClick={this.handleClick}>  
        {this.state.isToggleOn ? 'ON' : 'OFF'}  
      </button>  
    );  
  }  
}  
  
ReactDOM.render(  
  <Toggle />,  
  document.getElementById('root')  
);
```



Event Handling Example 2

```
class Checkbox extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      checked: false
    }
    this.handleClick = this.handleClick.bind(this)
  }
  handleClick() {
    this.setState({
      checked: !this.state.checked
    })
  }
  render() {
    var msg
    if(this.state.checked) {
      msg = "checked"
    } else {
      msg = "not checked"
    }
    return (
      <div>
        <input type="checkbox"
          onChange={this.handleClick}/>
        <p>This box is {msg}</p>
      </div>
    )
  }
}

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



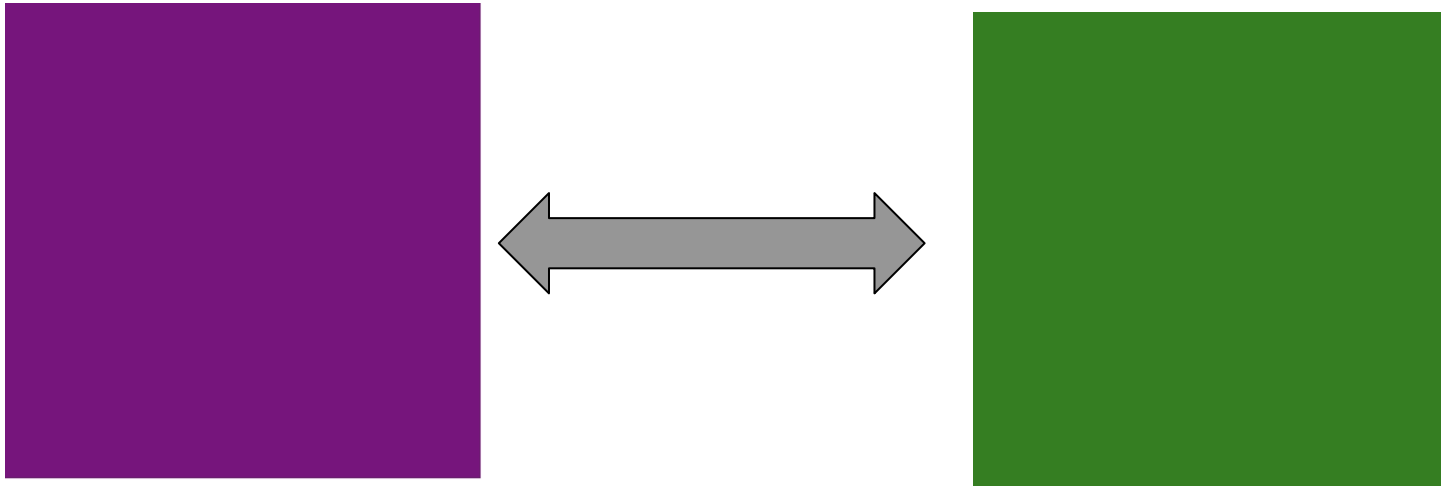
This box is not checked



This box is checked

Ex: Toggle the Box Color

- Open the file activity 5-1-start.html
- Add a click event to toggle the color state from purple to green when user click on the box



Passing Arguments to Event Handlers

Inside a loop, it is common to want to pass an extra parameter to an event handler. For example

```
<button onClick={(e) => this.deleteRow(id, e)}>Delete  
Row</button>
```

The `e` argument representing the React event will be passed as a second argument after the ID.

Topic 2

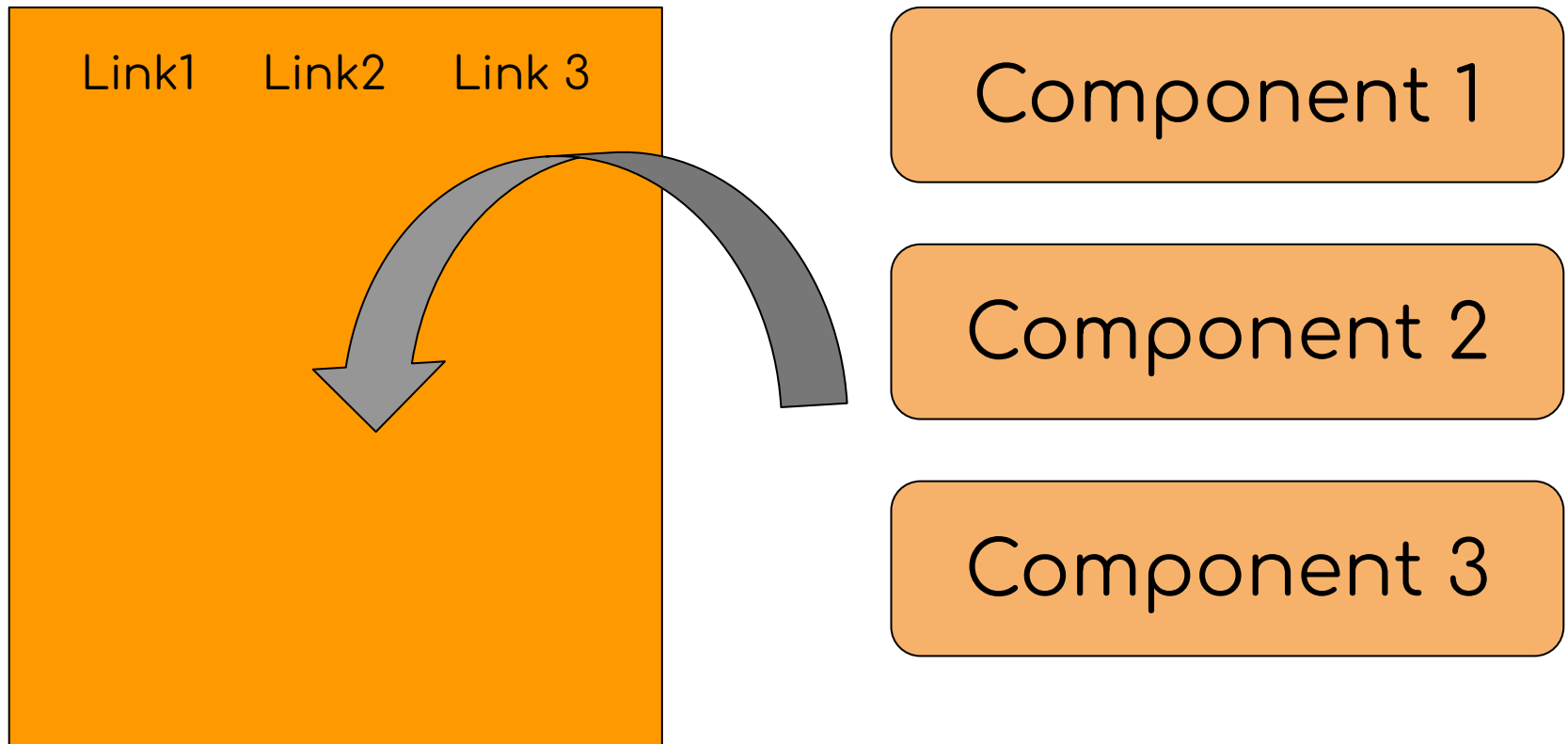
Single Page Application
Using React Router

Single Page Application

- Single Page Application (SPA) make multiple pages (different urls) to render on the same page.
- There is no need to refresh the page every time we make a url request to the server

How SPA Work?

The basic idea of SPA is to swap the component to the content when a url request is made to the server



React Router

- SPA can be achieved using React Router module.
- To install react router, enter the following command to command Prompt or Mac terminal

```
npm install --save react-router-dom
```

Setup Router Steps

Step 1 - Define urls for different links

Step 2 - Add a Route tag and associate the urls to corresponding components. Enclosed the Route tag inside Switch tag.

Step 3 - Add a Router tag to enclosed all the codes

Step 1: Define URL

Create the url for each link using the Link to tag as follows:

```
<li><Link to = '/'>Home</Link></li>
```

```
<li><Link to = '/aboutus'>About Us</Link> </li>
```

Step 2 Route Component

- You can associate the urls to the corresponding components using the Route tag
- The content is placed inside the Switch tag

```
<Switch>
```

```
<Route exact path="/" component = {Home}/>
```

```
<Route exact path="/aboutus" component =  
{AboutUs}/>
```

```
</Switch>
```


Step 3 Router

Add a Router tag to enclosed the Link to and Switch tags

```
<Router>
  <div>
    <li><Link to = '/'>Home</Link></li>
    ....
    <Switch>
      <Route exact path="/" component = {Home}/>
      ...
    </Switch>
  </div>
</Router>
```

React Router Code Example

```
import React, { Component } from 'react';
import { BrowserRouter as Router, Switch, Route, Link } from
'react-router-dom';
import Home from './Home';
import AboutUs from './AboutUs';
class App extends Component {
  render() {
    return (
      <Router>
        <div>
          <li><Link to = '/'>Home</Link></li>
          <li><Link to = '/aboutus'>About Us</Link> </li> <hr/>
          <Switch>
            <Route exact path='/' component = {Home}/>
            <Route exact path='/aboutus' component = {AboutUs}/>
          </Switch>
        </div>
      </Router>
    );
  }
}
export default App;
```

Activity: React Router

Add one more contact us link and setup the router

Topic 3

Lists and Keys

Javascript Map

- In Javascript, we use the map() function to take an array of numbers and double their values.
- We assign the new array returned by map() to the variable doubled and log it:

```
const numbers = [1, 2, 3, 4, 5];  
const doubled = numbers.map((number) => number *  
2);  
console.log(doubled);
```

React Map

- In Javascript, we use the map() function to take an array of numbers and double their values.
- We assign the new array returned by map() to the variable doubled and log it:

```
const numbers = [1, 2, 3, 4, 5];  
const listItems = numbers.map((number) =>  
  <li>{number}</li>  
);
```

```
ReactDOM.render(  
  <ul>{listItems}</ul>,  
  document.getElementById('root')  
);
```

Keys

- A “key” is a special string attribute you need to include when creating lists of element
- Keys help React identify which items have changed, are added, or are removed.
- Keys should be given to the elements inside the array to give the elements a stable identity:

```
function NumberList(props) {  
  const numbers = props.numbers;  
  const listItems = numbers.map((number) =>  
    <li key={number.toString()}>  
      {number}  
    </li>  
  );  
  return (  
    <ul>{listItems}</ul>  
  );  
}  
const numbers = [1, 2, 3, 4, 5];  
ReactDOM.render(  
  <NumberList numbers={numbers} />,  
  document.getElementById('root')  
);
```

Loop for To Do List

For this to do list demo, you can loop through the array using map function

```
render() {  
  const itemList = this.state.todoList.map((item)=>  
    <li>{item}</li>  
  );  
  return (  
    <div>  
      <h1>To Do List:</h1>  
      <ol>  
        {itemList}  
      </ol>  
    </div>  
  )  
}
```


Refactor and Render Todo

- Refactor the Todo component to the Input function (Todo) and the list function (TodoList)
- Use props to pass the task lists to TodoList component

```
import React, { Component } from 'react';
import Todo from './Todo';
import TodoList from './TodoList';

class App extends Component {
  constructor() {
    super();
    this.state = {
      todoList: [
        "Buy Groceries",
        "Buy Lunch",
        "Buy Dinner"
      ]
    }
    this.addTodoList = this.addTodoList.bind(this)
  }

  addTodoList(todo) {
    this.setState({
      todoList: this.state.todoList.concat([todo])
    });
  }
}
```

```
render() {
  return (
    <div>
      <h1>Add Task</h1>
      <Todo addFunction={this.addTodoList}/>
      <h1>To Do List</h1>
      <TodoList listItems={this.state.todoList}/>
    </div>
  );
}

export default App;
```

Activity: Hobby List

- Create a hobby list react project
- Create a hobby list component
- Create a display hobby list component

Refs

- Refs provide a way to get input from DOM
- The ES6 syntax is as follows

```
<input ref={input => this.hobbyInput = input}/>
```

To Read the Input

```
addHobby() {  
  let hobby = this.hobbyInput.value;  
  this.setState({  
    hobby: this.state.hobby.concat([hobby])  
  })  
}
```

Activity: Ref

- Create a input box to read the new todo item
- Display the list of todo items

Summary Q&A



**Practice
Makes
Perfect**

Course Feedback


<https://goo.gl/R2eumq>





INFOTECH **CERTIFICATE** *of* **ACCOMPLISHMENT**

*You will receive a digital certificate in your email
after the completion of the class
If you did not receive the digital certificate,
please send your request to
enquiry@tertiaryinfotech.com*



Thank You!

Fritz Lim

Tel: 91836486

fritzlim@gmail.com