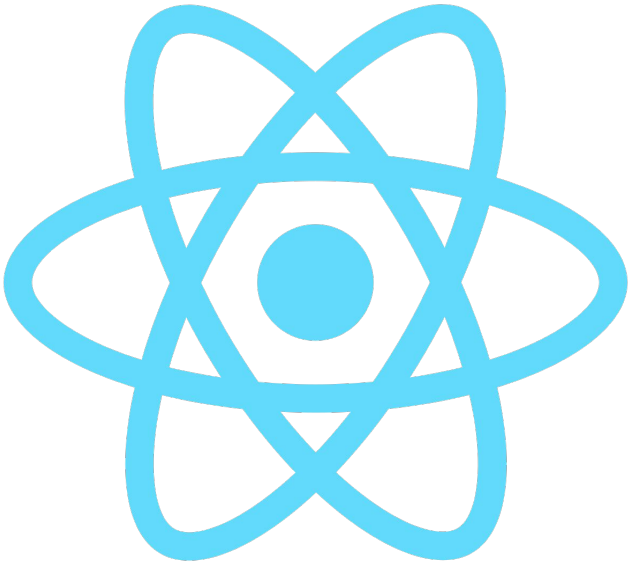
Basic React.js Training v6



Trainer: Fritz Lim

TINFOTECH

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 Get Started on React JS

- Introduction to React JS
- Adding React to a Website
- Create a New React App

Topic 2 Introducing JSX and Rendering Elements

- Introduction to JSX
- Rendering an Element into the DOM
- Updating the Rendered Element
- Securities Concerns

Topic 3 Components and Props

- Introduction to React Components?
- Function and Class Components
- Rendering a Component
- Composing Components
- Import/Export Components
- Passing Data via Props

Agenda

Topic 4 State and Lifecycle

- Adding Local State to a Component
- Component Lifecycle

Google Classroom

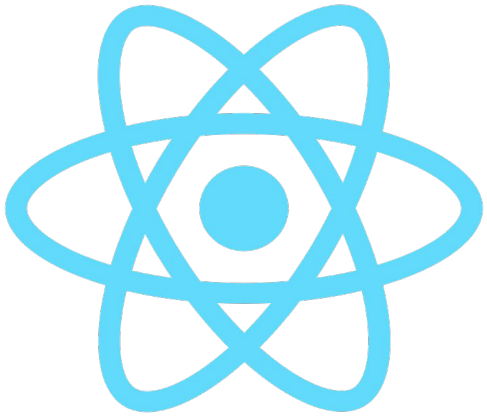
- Goto google classroom https://classroom.google.com
- Enter the class code below to join the class on the top right.
- If you cannot access the google classroom, please inform the trainer or staff.

6mrh7ql

Topic 1
Get Started on React JS

Javascript Frameworks for Web

- Javascript frameworks for Web shift the some of heavy lifting tasks, traditionally done by server scripting language such as PHP, from server to browser
- Currently, there are 3 main Javascript frameworks for web application and UI development
  - Angular
  - React
  - Vue



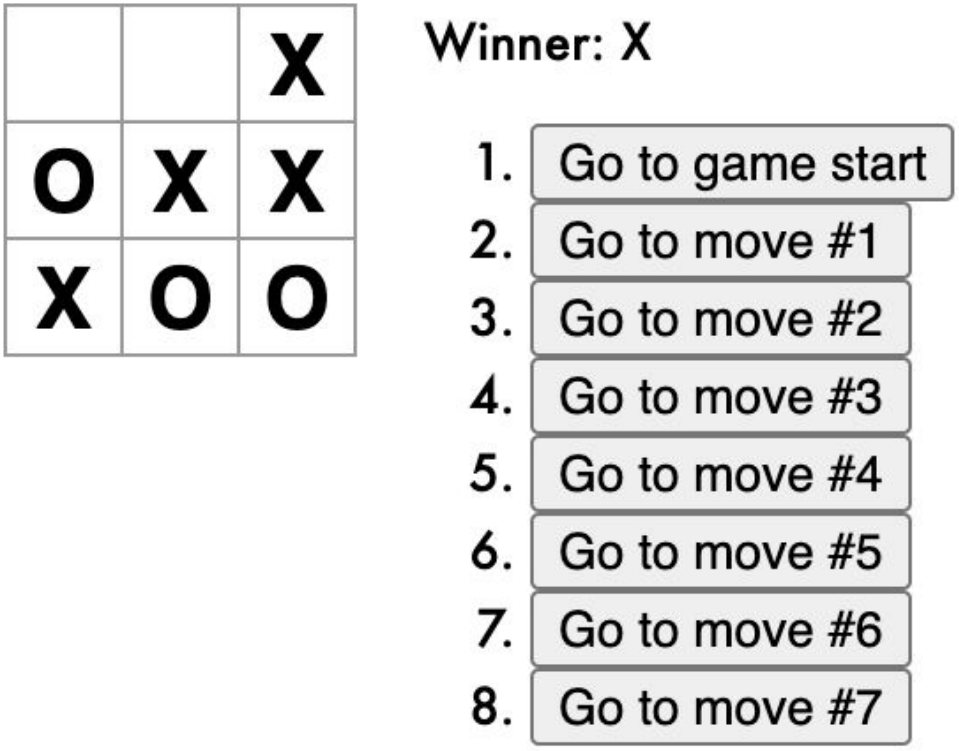
What is React?

- React.js is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It's used for handling the view layer for web and mobile apps.
- React also allows us to create reusable UI components. React was first created by Jordan Walke, a software engineer working for Facebook.
- React first deployed on Facebook's newsfeed in 2011 and on Instagram.com in 2012.
- React allows developers to create large web applications that can change data, without reloading the page.
- The main purpose of React is to be fast, scalable, and simple. It works only on user interfaces in the application. This corresponds to the view in the MVC template.

Activity: React Tic Tac Toe Game

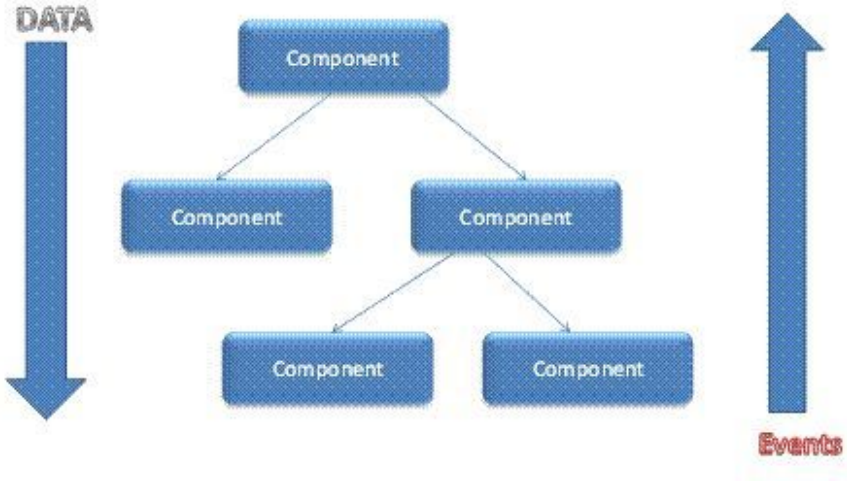
Try out the following Tic-Tac-Toe game written in React

https://codepen.io/gaearon/pen/gWWZgR



Key Features of React

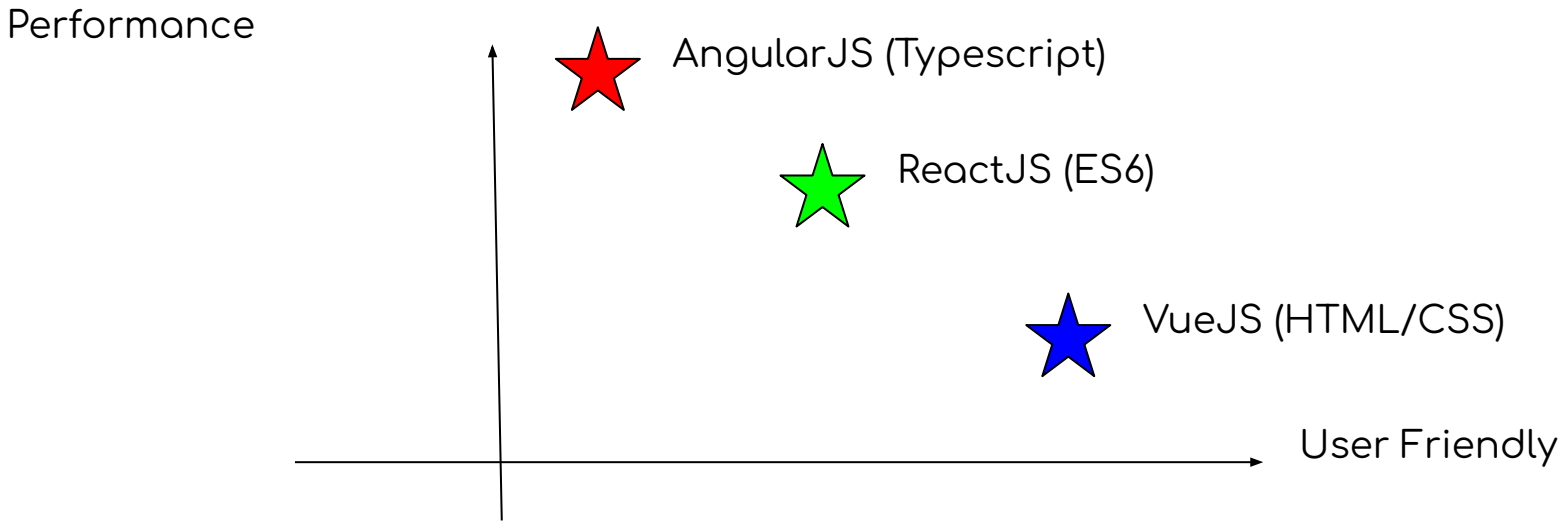
- JSX : In React, instead of using regular JavaScript for templating, it uses JSX. JSX is a simple JavaScript that allows HTML quoting and uses these HTML tag syntax to render subcomponents. HTML syntax is processed into JavaScript calls of React Framework.
- React Native: React has native libraries that were announced by Facebook in 2015, which provides the react architecture to native applications like IOS, Android and UPD.
- Single-Way data flow: In React, a set of immutable values are passed to the components renderer as properties in its HTML tags. The component cannot directly modify any properties but can pass a call back function with the help of which we can do modifications. This complete process is known as "properties flow down; actions flow up"



Why React?

React is preferred for the following reasons:

- Easier to learn than AngularJS but similar performance
- Higher performance than Vue.JS.
- Can do native apps using React Native
- Can do VR content using React VR



Who Use React JS




Two Approaches to Use React

- There are two approaches to use React
  - Add React to a website
    - This is good for small react project since majority of websites don't need to be single-page apps.
    - With a few lines of code and no build tooling, you can add React as a small part of your website using just HTML, CSS and Javascript. You can also make the code less Javascript with JSX babel.
  - Create a React app
    - This is good for bigger react project or single page application.
    - It require node.js, besides HTML, CSS and Javascript.

Add React to a Website

You can add React to a website by inserting the following two React script tags to your HTML

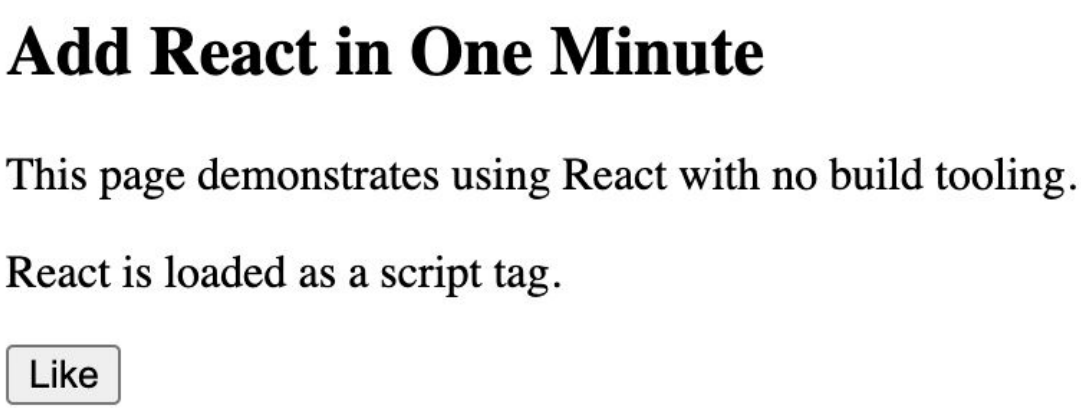
```
<script
src="https://unpkg.com/react@16/umd/react.development.js"></script>

<script
src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
></script>
```

Activity: Add React to a Website

Download the demo code from https://gist.github.com/gaearon/6668a1f6986742109c00a581ce704605/archive/f6c882b6ae18bde42dcf6fdb751aae93495a2275.zip

There are 2 files in the folder:

- index.html
- like_button.js



Create a React App Project

- Create an react apps allows you to ses up your development environment so that you can use the latest JavaScript features, provides a nice developer experience, and optimizes your app for production.
- You'll need to have Node >= 8.10 and npm >= 5.6 on your machine.
- React app can be used for server side application or mobile apps
- React app is based on node.js. So you need to install node.js prior to create an react app

Installing Node JS

- Download and Install the latest Node JS https://nodejs.org/en/download/current/
- To test if Node JS was installed successfully type the following commands on your Windows CMD Prompt / Mac Terminal

  node -v

- To test if Node Package Manager (NPM) was installed successfully, type

  npm -v

Activity: Create a React App

To create a project, run the following on your command prompt

```
npx create-react-app my-app
cd my-app
npm start
```

It will auto start the react app on http://localhost:3000/

Exploring Project Folders and Files

- node_modules folder contains all the dependencies modules
- public folder contains files used by browser
- src folder contain your react apps files

For the project to build, these files must exist with exact filenames:

- **public/index.html** is the page template;
- **src/index.js** is the JavaScript entry point.

index.js

index.js renders the App Component using the React.DOM.render method

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import registerServiceWorker from './registerServiceWorker';

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

Topic 2
Introducing JSX and Rendering Elements

JSX

- JSX is an HTML-like syntax used by React that extends Javascript
- In the example below, we declare a variable called name and then use it inside JSX by wrapping it in curly braces:

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

What is Babel?

- You can use Babel to convert JSX syntax
- Babel is a toolchain that is mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript in current and older browsers or environments.
- Example

```
// Babel Input: ES2015 arrow function
[1, 2, 3].map((n) => n + 1);

// Babel Output: ES5 equivalent
[1, 2, 3].map(function(n) {
  return n + 1;
});
```

Add Babel to a Website

- The quickest way to try JSX to our React website is to add this Babel <script> tag to your page:

```
<script
src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
```

- You can use JSX in any <script> tag by adding type="text/babel" attribute to it

Activity: Babel

Open up activity2-1-start.html, convert the following script to JSX

```
<body>
    <div id="root"></div>

    <script type="text/javascript">

        ReactDOM.render(
            React.createElement('h1', null, 'Hello World'),
            document.getElementById('root')
        )
    </script>
</body>
```

Embedding Expressions in JSX

In the example below, we declare a variable called name and then use it inside JSX by wrapping it in curly braces

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Embedding Expressions in JSX

In the example below, we embed the result of calling a JavaScript function, formatName(user), into an <h1> element.

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Elements

- Elements are the smallest building blocks of React apps.
- An element describes what you want to see on the screen
- Unlike browser DOM elements, React elements are plain objects, and are cheap to create.
- React DOM takes care of updating the DOM to match the React elements

```
const element = <h1>Hello, world</h1>;
```

Rendering an Element into the DOM

- Let's say there is a root somewhere in your HTML file
- We call this a "root" DOM node because everything inside it will be managed by React DOM.
- Applications built with just React usually have a single root DOM node.
- To render a React element into a root DOM node, pass both to ReactDOM.render()

```
<div id="root"></div>
const element = <h1>Hello, world</h1>;
ReactDOM.render(element, document.getElementById('root'));
```

Updating the Rendered Element

- React elements are immutable. Once you create an element, you can't change its children or attributes. An element is like a single frame in a movie: it represents the UI at a certain point in time.
- The only way to update the UI is to create a new element, and pass it to ReactDOM.render().
- React DOM compares the element and its children to the previous one, and only applies the DOM updates necessary to bring the DOM to the desired state

```
function tick() {
  const element = (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {new Date().toLocaleTimeString()}.</h2>
    </div>
  );
  ReactDOM.render(element, document.getElementById('root'));
}

setInterval(tick, 1000);
```

Cross Site Scripting (XSS)

- Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites.
- XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.
- Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.
- An attacker can use XSS to send a malicious script to an unsuspecting user.
- The end user's browser has no way to know that the script should not be trusted, and will execute the script.

JSX Prevents Injection Attacks

- By default, React DOM escapes any values embedded in JSX that are not explicitly written in the application code before rendering them.
- Thus it ensures that you can never inject anything that's not explicitly written in your application. Everything is converted to a string before being rendered.
- This helps prevent XSS (cross-site-scripting) attacks.

```
const title = response.potentiallyMaliciousInput;
// This is safe:
const element = <h1>{title}</h1>;
```

Topic 3
Components and Props

Components

- Components let you split the UI into independent, reusable pieces, and think about each piece in isolation.
- Conceptually, components are like JavaScript functions. They accept arbitrary inputs (called “props”) and return React elements describing what should appear on the screen

Props

- You can use props to pass data to components
- props is shorthand for properties

Eg

```
<Hello name="Alfred"/>
```

Function Components

- The simplest way to define a component is to write a JavaScript function
- This function is a valid React component because it accepts a single “props” (which stands for properties) object argument with data and returns a React element

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Class Components

- You can also use an ES6 class to define a component:

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Rendering a Function Component

- When React sees an element representing a user-defined component, it passes JSX attributes and children to this component as a single object. We call this object “props”.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Sara" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Rendering a Function Component

- We call ReactDOM.render() with the <Welcome name="Sara" /> element.
- React calls the Welcome component with {name: 'Sara'} as the props.
- Our Welcome component returns a <h1>Hello, Sara</h1> element as the result.
- React DOM efficiently updates the DOM to match <h1>Hello, Sara</h1>

Rendering a Class Component

Similarly, < .. /> is used to render the class component.

```
class Hello extends React.Component {
        render() {
            return (
                <h1>Hello World</h1>
            )
        }
    }

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

Rendering Component in a Project

You can modify the App.js under the src folder to render a function component.

```
import React from 'react';
import './App.css';

function Welcome(props) {
        return <h1>Hello, {props.name}</h1>;
      }

function App() {
  return (
    <Welcome name="Sara"/>
  );
}

export default App;
```

Export and Import Components

- You can refactor multiple components into external components
- Use export default to create the external components
- Use import to add the external components

Export External Component

In general, one will create one file for one class component in the project, and export the class.

```
import React from 'react';
class Hello extends React.Component {
    render() {
      return (
        <div>
          <h1>Hello World</h1>
        </div>
      );
    }
  }
export default Hello;
```

Import External Components

```
import React from 'react';
import Hello from './Hello';
import Welcome from './Welcome';
```

Activity: Create a Todo Component

Add an external ToDo component to display list of things to do in a project. For example

1. Buy Groceries
2. Buy Lunch

Add CSS to Component

To avoid confusion with class, CSS class is added as className

```
class Hello extends React.Component {
        render() {
            return (
                <div id="fancy">
                    <h1 className="heading">Hello World</h1>
                </div>
                )
            }
        }

ReactDOM.render(
            <Hello/>,
            document.getElementById('react-container')
        )
```

Converting a Function to a Class

You can convert a function component to a class in five steps:

- Create an ES6 class, with the same name, that extends React.Component.
- Add a single empty method to it called render().
- Move the body of the function into the render() method.
- Replace props with this.props in the render() body.
- Delete the remaining empty function declaration.

Stateless Component

A stateless component is a arrow function that return some UI

```
const Hello = () => (
        <div>
            <h1>Hello World</h1>
            <p>Have fun!</p>
        </div>
    )

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

Props in Stateless Function

Using props in stateless function, do not require this

```
const Hello = (props) => (
        <div>
            <h1>Hello {props.name}</h1>
            <p>Learning React Version {props.version}</p>
        </div>

        )

ReactDOM.render(
            <Hello name="Steve" version={16}/>,
            document.getElementById('root')
        )
```

Destructing

We can also use destructuring in ES6 to pass data to a stateless function without using props

```
const Hello = ({name,version}) =>
    <div>
        <h1>Hello {name}</h1>
        <p>Learning React Version {version}</p>
    </div>


ReactDOM.render(
    <Hello name="Alfred" version={16}/>,
    document.getElementById('root')
        )
```

Converting a Function to a Class

- Function Component

```
function Clock(props) {
  return (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {props.date.toLocaleTimeString()}.</h2>
    </div>
  );
}
```

- Class component

```
class Clock extends React.Component {
  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.props.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
```

Render Class Component in a Project

You can also modify the App.js under the src folder to render a class component.

```
import React from 'react';
class Hello extends React.Component {
  render() {
    return (
      <div>
        <h1>Hello World</h1>
      </div>
    )
  }
}
class App extends React.Component {
  render() {
    return (
      <div>
        <Hello/>
      </div>
    );
  }
}
export default App;
```

Composing Components

- We can create an App component that renders many times, for example

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
function App() {
  return (
    <div>
      <Welcome name="Sara" />
      <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>
  );
}
ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

Activity: Rendering Component

- Create two components, Hello and Welcome
- Render the two components on the browser
- Render the two components on a project

Hello World

Welcome to React JS Training

Topic 4
State and Lifecycle

State

- State is similar to props, but it is private and fully controlled by the component.
- State is initialized by constructor in the component Eg

```
constructor(props) {
        super(props),
        this.state ={
            id: 10,
            checked: false
        }
  }
```

Adding Local State to a Class

- Replace this.props.date with this.state.date in the render() method
- Add a class constructor that assigns the initial this.state:
- Pass props to the base constructor
- Remove the date prop from the <Clock /> element

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }
  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
ReactDOM.render(
  <Clock />,
  document.getElementById('root')
);
```

Activity: State

Convert the following component with props to state

```
class Hello extends React.Component {
    render() {
        return (
            <div>
                <h1>Hello {this.props.name}</h1>
                <p>Learning React Version {this.props.version}</p>
            </div>
            )
        }
    }

ReactDOM.render(
        <Hello name="Alfred" version={16}/>,
        document.getElementById('root')
        )
```

Adding Lifecycle Methods to a Class

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  componentDidMount() {
    this.timerID = setInterval(
      () => this.tick(),
      1000
    );
  }

  componentWillUnmount() {
    clearInterval(this.timerID);
  }

  tick() {
    this.setState({
      date: new Date()
    });
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is
{this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}

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

Adding Local State to a Class

- When <Clock /> is passed to ReactDOM.render(), React calls the constructor of the Clock component. Since Clock needs to display the current time, it initializes this.state with an object including the current time. We will later update this state.
- React then calls the Clock component's render() method. This is how React learns what should be displayed on the screen. React then updates the DOM to match the Clock's render output.
- When the Clock output is inserted in the DOM, React calls the componentDidMount() lifecycle method. Inside it, the Clock component asks the browser to set up a timer to call the component's tick() method once a second.
- Every second the browser calls the tick() method. Inside it, the Clock component schedules a UI update by calling setState() with an object containing the current time. Thanks to the setState() call, React knows the state has changed, and calls the render() method again to learn what should be on the screen. This time, this.state.date in the render() method will be different, and so the render output will include the updated time. React updates the DOM accordingly.
- If the Clock component is ever removed from the DOM, React calls the componentWillUnmount() lifecycle method so the timer is stopped.

Using State Correctly

- Do not modify state directly. Always use setState().
- The only place where you can assign this.state is the constructor. For example

```
// Wrong
this.state.comment = 'Hello';
// Correct
this.setState({comment: 'Hello'});
```

- State updates may be asynchronous. Because this.props and this.state may be updated asynchronously, you should not rely on their values for calculating the next state.
- To fix it, use a second form of setState() that accepts a function rather than an object. That function will receive the previous state as the first argument, and the props at the time the update is applied as the second argument:

```
// Wrong
this.setState({
  counter: this.state.counter + this.props.increment,
});
// Correct
this.setState(function(state, props) {
  return {
    counter: state.counter + props.increment
  };
});
```

Add State to Component in Project

You can also modify the App.js under the src folder to add state to a component in a project.

```
class App extends Component {
      constructor() {
            super();
            this.state = {
                        Name: "Steve",
                        Country: "Singapore"
                  }}

render() {
   return (
     <div >
            <h1>Name : {this.state.Name}</h1>
            <h1>Country: {this.state.Country}</h1>
     </div>
   );
 }}
```

Activity: Add State in Project

Add state to ToDo component to display a list of things to do, for example:

1. Buy Groceries
2. Buy Lunch

Hint: State

```
import React, { Component } from 'react';
class Todo extends Component {
        constructor() {
                super();
                this.state = {
                        todoList: [
                                "Buy Groceries",
                                "Buy Lunch"]
                        }
                }
                render() {
                        return (
                        <div>
                                <ol>
                                        <li>{this.state.todoList[0]}</li>
                                        <li>{this.state.todoList[1]}</li>
                                </ol>
                        </div>
                                )
                        }
                }
export default Todo;
```

Summary Q&A

Course Feedback

https://goo.gl/R2eumq

Thank You!

Fritz Lim
Tel: 91836486
fritzlim@gmail.com