Day5(06-June-2020)   React

```
===============================================================
```

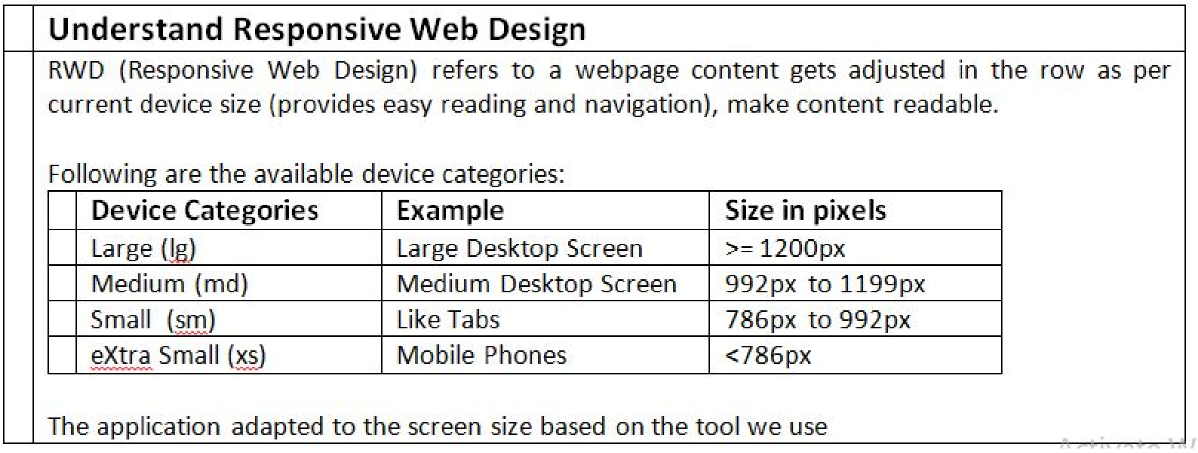
Session Agenda:

Bootstrap concepts
Forms in React
Life Cycle Hooks
Web API
React consume Web API
Higher Order Components
Routing in react

MongoDB Concepts

	RWD: Responsive WebPage Designing
	Refers a webpage content gets adjusted when page size is changed Bootstrap is a collection of files which is used to design RWD pages .NET Application, Java Application, Node.js, Angular, React.js, etc., uses bootstrap for RWD

	Bootstrap contains following files
	CSS file for Styles Javascript for client side handling Jquery for DOM manipulation



Understand Responsive Web Design

RWD (Responsive Web Design) refers to a webpage content gets adjusted in the row as per current device size (provides easy reading and navigation), make content readable.

Following are the available device categories:

Device Categories	Example	Size in pixels
Large (lg)	Large Desktop Screen	>= 1200px
Medium (md)	Medium Desktop Screen	992px to 1199px
Small (sm)	Like Tabs	786px to 992px
eXtra Small (xs)	Mobile Phones	<786px

The application adapted to the screen size based on the tool we use

	Bootstrap class	Purpose
	container	This class refers to area for RWD
	row	This class used to create horizontal one to represent group of columns A container can have more than one row
	col-lg-colsize	Refers to number of column in large screen
	col-xs-offset	Refers to number of offset column
	visible-lg	Make the area visible on particular category

	Device Categories	Size in pixel	Bootstrap class
	Large Devices - Large Desktop Screen	>=1200px	col-lg-*
	Medium Devices - Small Desktop Screen	992px – 1199px	col-md-*
	Small Devices - Like Tabs	786px – 992px	col-sm-*
	eXtra Small Devices - Mobile Phones	<786px	col-xs-*

Understand Bootstrap

Bootstrap is used to create *Responsive Web Pages*.
Bootstrap is a library contains JAVASCRIPT, CSS and JQUERY files.
RWD pages containing contents which are adjusted accordingly on Larger Desktop Screen, Medium Screen, Small screens, Mobile phone screens

History of Bootstrap

Bootstrap is a free and open framework for developing Responsive Web Pages, developed by Mark Otto and Jacob Thronton at Twitter as a framework.
Released on Aug 2011

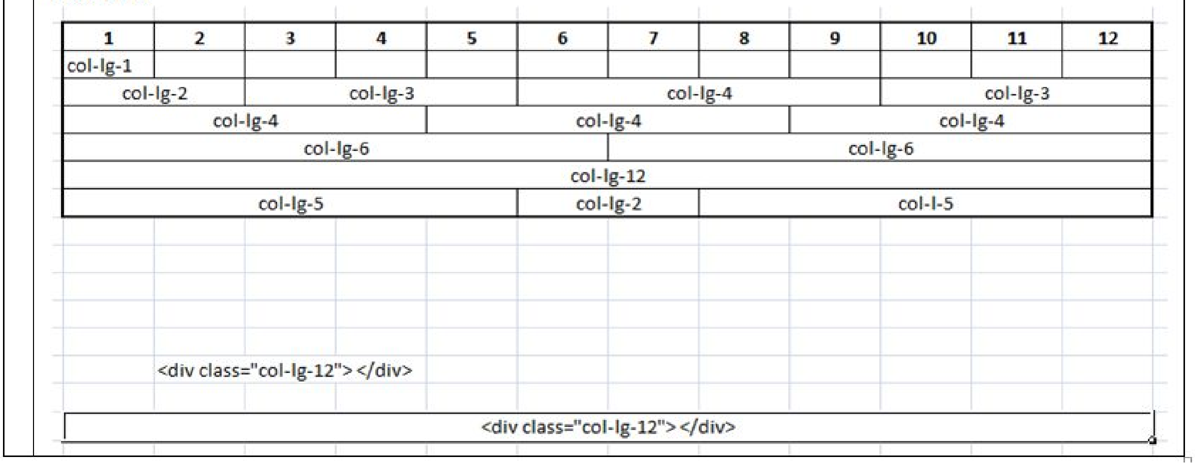
Advantages:

- Mobile First Approach
- Browser Support
- Easy to get started
- Responsive Web Page Design

Bootstrap based on all open standard frameworks like HTML5, CSS3, JQUERY, etc.,
Bootstrap contains HTML and CSS based templates for Text, Form, Button, Navigation and other components.
URL: http://getbootstrap.com
Before Bootstrap we use to have two different web sites for Desktop and Mobile separately.

Bootstrap Grid System

A container in a browser page is categorized in rows and column format
As per the Bootstrap grid system every row is collection of 12 columns
Bootstrap rows are based on column, if the content failed to adjust in the same rows gets carried to next row



1	2	3	4	5	6	7	8	9	10	11	12
col-lg-1											
col-lg-2		col-lg-3			col-lg-4				col-lg-3		
col-lg-4				col-lg-4				col-lg-4			
col-lg-6						col-lg-6					
col-lg-12											
col-lg-5					col-lg-2		col-l-5				

<div class="col-lg-12"> </div>

<div class="col-lg-12"> </div>

RWD Demo: Display different no.of columns in a row based on screen size

<!-- index.html →

```
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
></script>
    <link
rel="stylesheet"href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css
/bootstrap.min.css" />
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></
script>
```

//app.js

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
import EmpDetails from './EmpDetails';
function App() {

  return (<div>

  <div className="container">
  <h1>Root Component</h1>
  <div className="row">
          <div className="visible-lg">
            <h1>Currently Large Screen, You can see 4 columns per
row</h1>
          </div>
          <div className="visible-md">
                  <h1>Currently Medium Screen, You can see 3 columns per
row</h1>
          </div>
          <div className="visible-sm">
                  <h1>Currently Small Screen, You can see 2 columns per
row</h1>
          </div>
          <div className="visible-xs">
            <h1>Currently Extra Small Screen, You can see 1 columns per
row</h1></div>
          </div>

<div className="row">
  <div className="col-lg-3 col-md-4 col-sm-6 col-xs-12">
          <div className="greenBorderClass"> <h3>Div1: First Column </h3>
</div>
  </div>

  <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12">
          <div class="greenBorderClass"> <h3>Div2:Second Column </h3>
</div>
  </div>

  <div className="col-lg-3 col-md-4 col-sm-6 col-xs-12">
          <div className="greenBorderClass"> <h3>Div3: Third Column </h3>
</div>
  </div>

  <div className="col-lg-3 col-md-4 col-sm-6 col-xs-12">
      <div className="greenBorderClass"> <h3>Div4:Fourth Column </h3>
</div>
  </div>
```

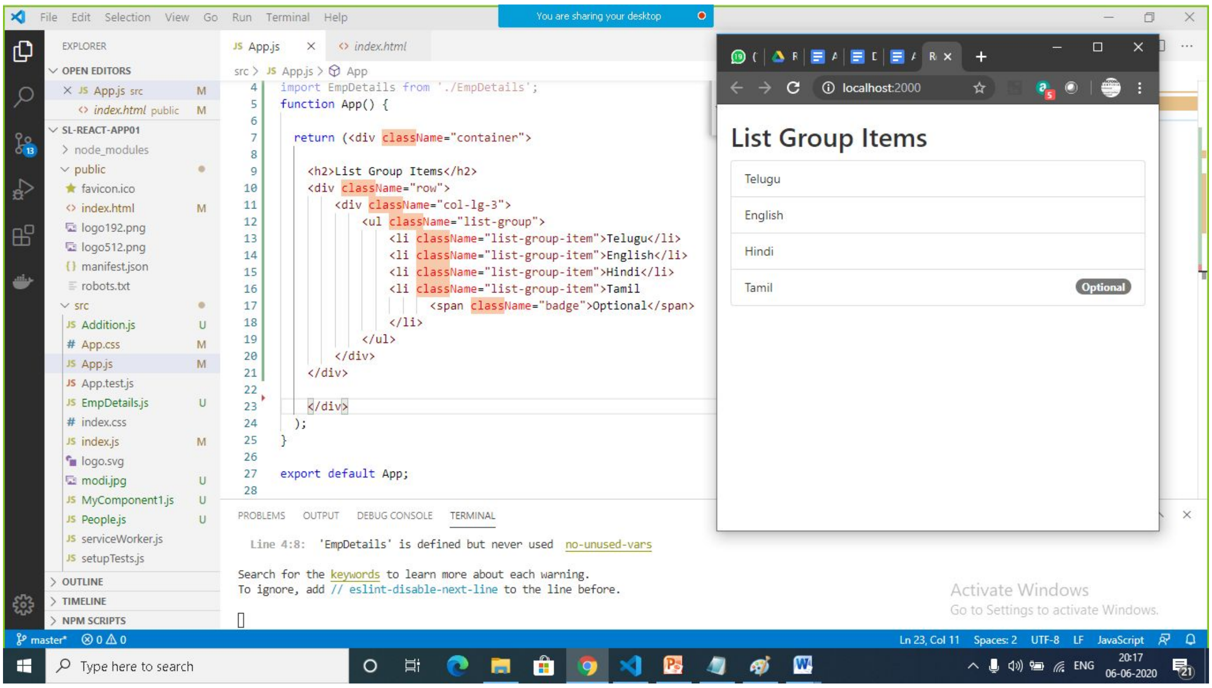
```
</div>
</div>

    </div>
  );
}

export default App;
```

//app.css

```
.greenBorderClass{
  border: 5px  solid green;
}
```

Button class demo

```
function App() {
  return (<div className="container">
  <h1>BootStrap, button class </h1>
  <input type="submit" className="btn" />
  <input type="reset" className="btn" />
  <input type="button" value="click me" className="btn" /> <hr/>

<button className="btn btn-default">Default</button>
<button className="btn btn-primary">Primary</button>
<button className="btn btn-success">Success</button>
<button className="btn btn-info">Info</button>
<button className="btn btn-warning">Warning</button>
<button className="btn btn-danger">Danger</button>
<button className="btn btn-link">Link</button>
    </div>
  );
}
```



```
import EmpDetails from './EmpDetails';
function App() {

  return (<div className="container">

    <h2>List Group Items</h2>
    <div className="row">
        <div className="col-lg-3">
            <ul className="list-group">
                <li className="list-group-item">Telugu</li>
                <li className="list-group-item">English</li>
                <li className="list-group-item">Hindi</li>
                <li className="list-group-item">Tamil
                      <span className="badge">Optional</span>
                </li>
            </ul>
        </div>
    </div>

    </div>
  );
}

export default App;
```

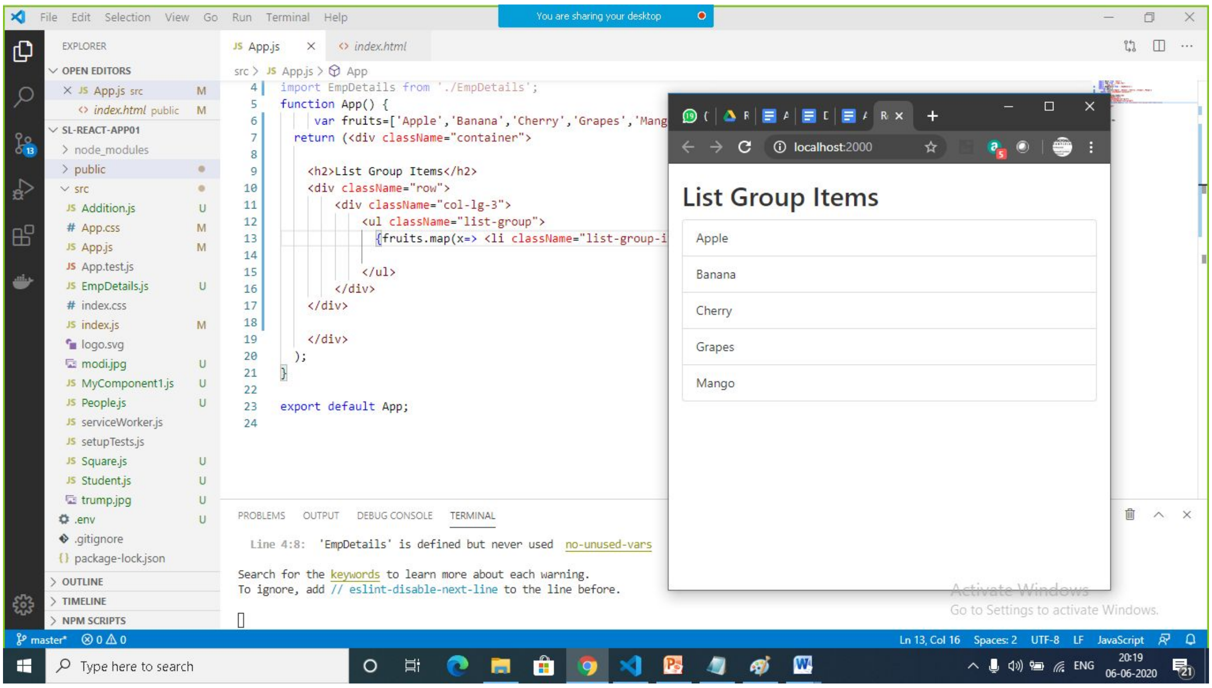
```
  Line 4:8:  'EmpDetails' is defined but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

List Group Items

- Telugu
- English
- Hindi
- Tamil Optional

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
import EmpDetails from './EmpDetails';
function App() {

  return (<div className="container">

    <h2>List Group Items</h2>
    <div className="row">
        <div className="col-lg-3">
            <ul className="list-group">
                <li className="list-group-item">Telugu</li>
                <li className="list-group-item">English</li>
                <li className="list-group-item">Hindi</li>
                <li className="list-group-item">Tamil
                      <span className="badge">Optional</span>
                </li>
            </ul>
        </div>
    </div>


    </div>
  );
}
```

```
export default App;
```



```
import EmpDetails from './EmpDetails';
function App() {
    var fruits=['Apple','Banana','Cherry','Grapes','Mang
  return (<div className="container">

    <h2>List Group Items</h2>
    <div className="row">
        <div className="col-lg-3">
            <ul className="list-group">
              {fruits.map(x=> <li className="list-group-i

            </ul>
        </div>
    </div>

    </div>
  );
}

export default App;
```

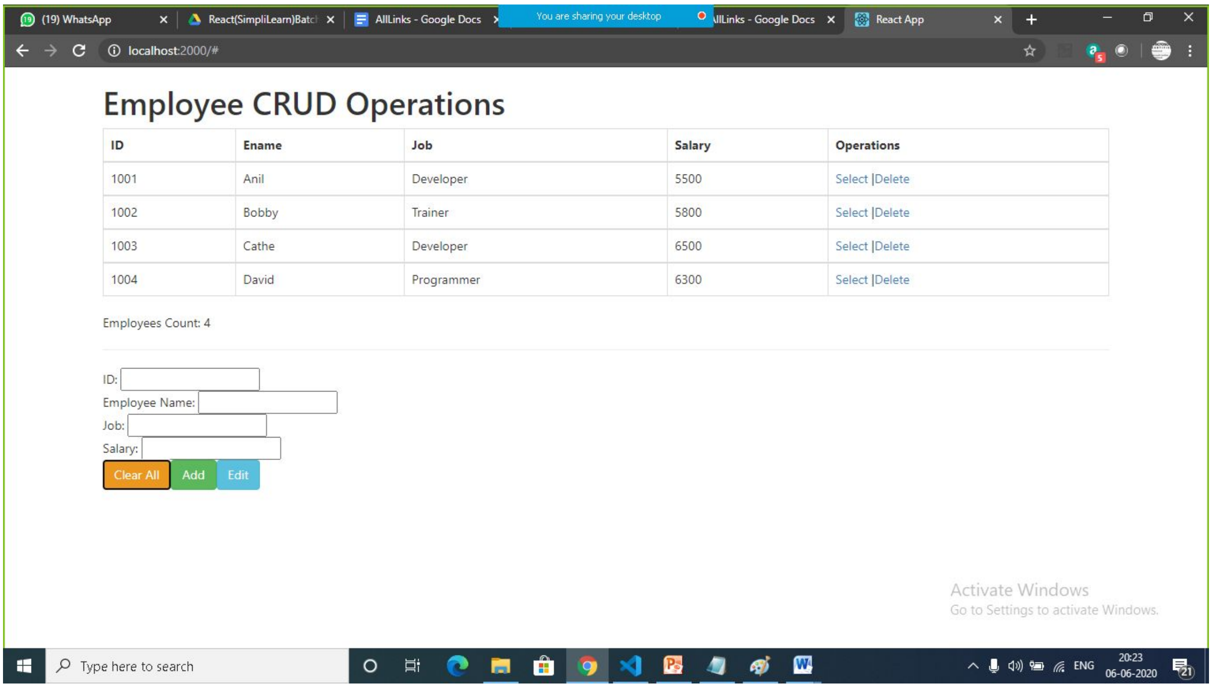
```
 Line 4:8:  'EmpDetails' is defined but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

List Group Items

- Apple
- Banana
- Cherry
- Grapes
- Mango

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
import EmpDetails from './EmpDetails';
function App() {
     var fruits=['Apple','Banana','Cherry','Grapes','Mango']
  return (<div className="container">

    <h2>List Group Items</h2>
    <div className="row">
        <div className="col-lg-3">
            <ul className="list-group">
              {fruits.map(x=> <li className="list-group-item">{x}</li>)}

            </ul>
        </div>
    </div>

    </div>
  );
}
```

```
export default App;
```



Employee CRUD Operations

ID	Ename	Job	Salary	Operations
1001	Anil	Developer	5500	Select |Delete
1002	Bobby	Trainer	5800	Select |Delete
1003	Cathe	Developer	6500	Select |Delete
1004	David	Programmer	6300	Select |Delete

Employees Count: 4

ID:

Employee Name:

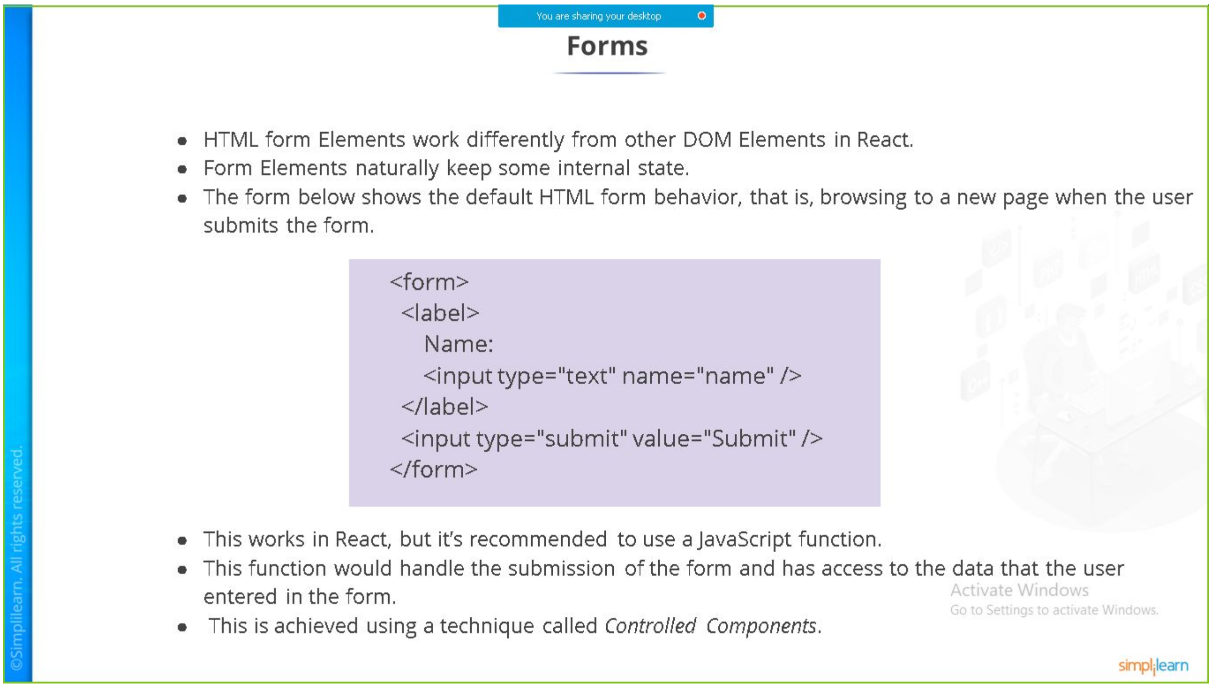
Job:

Salary:

Clear All Add Edit

//empdetails.js

```
import React from 'react';

class EmpDetails extends React.Component{
    state={
        employees:[
            {"id":1001,"ename":"Anil","job":"Developer","salary":5500},
            {"id":1002,"ename":"Bobby","job":"Trainer","salary":5800},
            {"id":1003,"ename":"Cathe","job":"Developer","salary":6500},
            {"id":1004,"ename":"David","job":"Programmer","salary":6300}
        ],
    }
    getRowById(id){
        var e = this.state.employees.find ( x=>x.id==id);
        this.refs.id.value = e.id;
        this.refs.ename.value = e.ename;
        this.refs.job.value = e.job;
        this.refs.salary.value = e.salary;
    }
    deleteRowById(id){
        var index = this.state.employees.findIndex(x=>x.id==id);
        var employees = this.state.employees;
        if ( window.confirm ("Are you sure?") ){
            employees.splice(index,1);
            this.clearAll();
        }
        this.setState({employees:employees});
    }
    addRow(){
        var e  = {
            id:this.refs.id.value,
            ename:this.refs.ename.value,
            job:this.refs.job.value,
            salary:this.refs.salary.value
        };
        var employees = this.state.employees;
        employees.push ( e );
        this.setState({employees:employees});
        this.clearAll();
    }
    editRow(){
        var id = this.refs.id.value;
        var e  = {
            id:this.refs.id.value,
            ename:this.refs.ename.value,
            job:this.refs.job.value,
            salary:this.refs.salary.value
        };
        var index = this.state.employees.findIndex(x=>x.id==id);
        var employees = this.state.employees;
        employees[index]=e;
        this.setState({employees:employees});
    }
    clearAll(){

this.refs.id.value=this.refs.ename.value=this.refs.job.value=this.refs.sa
lary.value="";
    }
    render(){
        return <div>
            <h1>Employee CRUD Operations</h1>
            <table className="table table-hover table-bordered">
                <thead>
                <tr>
                    <th>ID</th> <th>Ename</th>  <th>Job</th>
<th>Salary</th> <th>Operations</th>
                </tr></thead>
                {this.state.employees.map( x =>
                    <tbody><tr>
                        <td>{x.id}</td> <td>{x.ename}</td>
<td>{x.job}</td> <td>{x.salary}</td> <td>
                            <a href="#"
onClick={()=>this.getRowById(x.id)}>Select</a> |
                            <a href="#"
onClick={()=>this.deleteRowById(x.id)}>Delete</a>
                            </td>
                    </tr></tbody>)}
            </table> Employees Count:  {this.state.employees.length}
<hr/>
            ID: <input type="number" ref="id" />    <br/>
            Employee Name: <input type="text" ref="ename" /><br/>
            Job: <input type="text" ref="job" /><br/>
            Salary: <input type="number" ref="salary" /> <br/>
            <button className="btn btn-warning"
onClick={()=>this.clearAll()}>Clear All</button>
            <button className="btn btn-success"
onClick={()=>this.addRow()}>Add </button>
            <button className="btn btn-info"
onClick={()=>this.editRow()}>Edit</button>
        </div>
    }
}
export default EmpDetails;
```

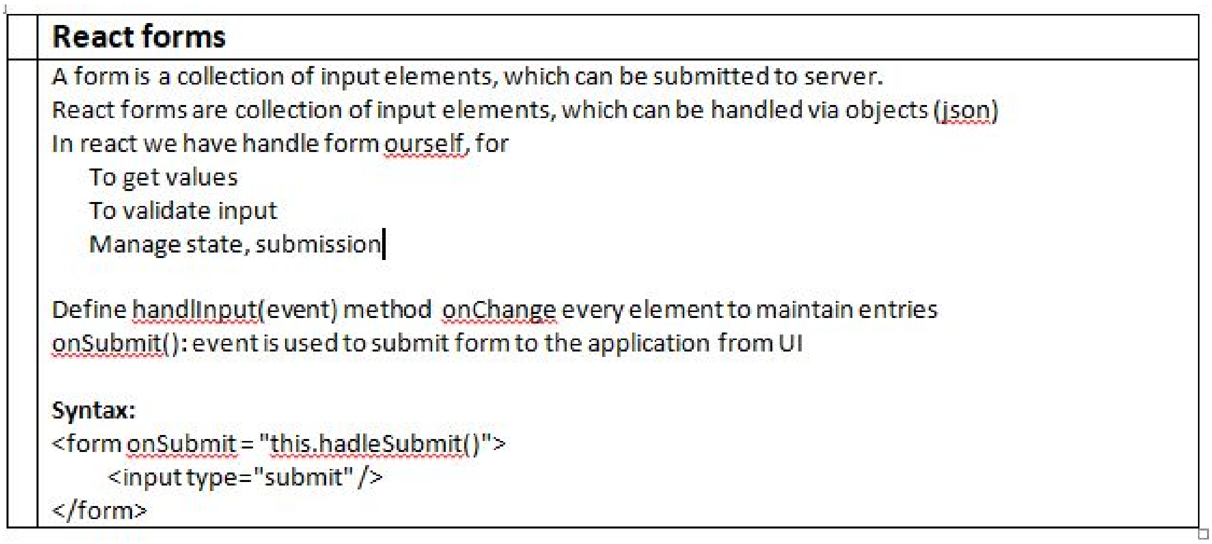

You are sharing your desktop
Activate Windows
Go to Settings to activate Windows.

Forms

- HTML form Elements work differently from other DOM Elements in React.
- Form Elements naturally keep some internal state.
- The form below shows the default HTML form behavior, that is, browsing to a new page when the user submits the form.

```
<form>
  <label>
    Name:
    <input type="text" name="name" />
  </label>
  <input type="submit" value="Submit" />
</form>
```

- This works in React, but it’s recommended to use a JavaScript function.
- This function would handle the submission of the form and has access to the data that the user entered in the form.
- This is achieved using a technique called *Controlled Components*.



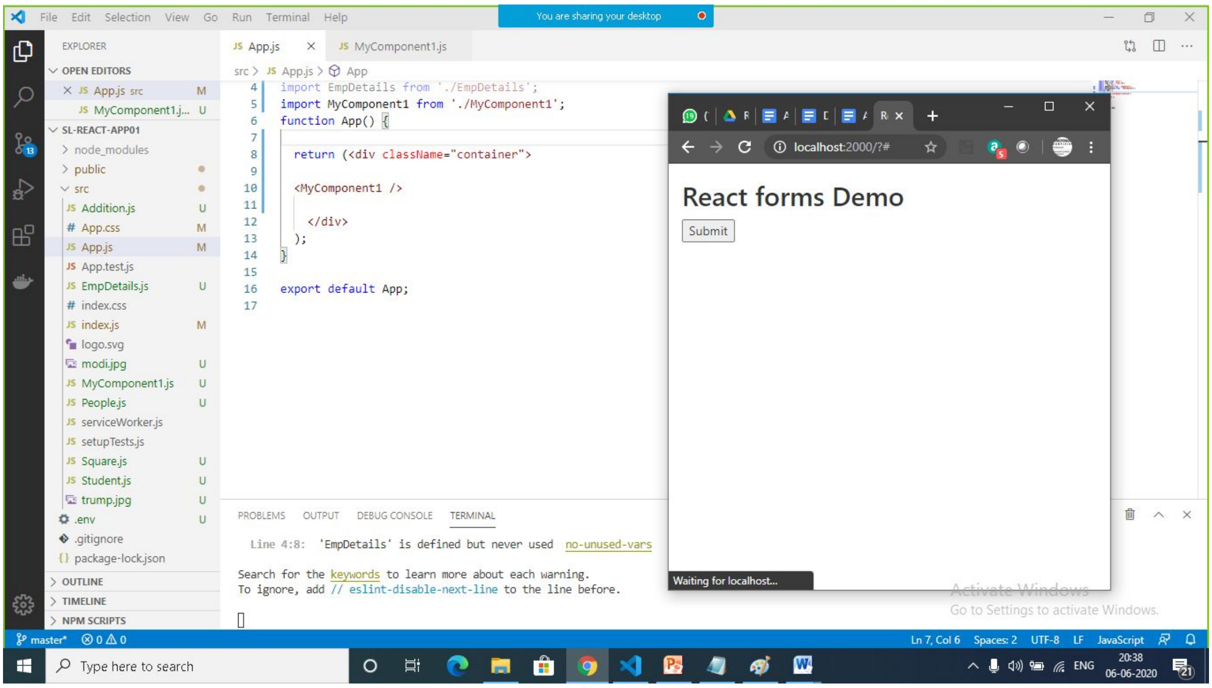
React forms

A form is a collection of input elements, which can be submitted to server.
React forms are collection of input elements, which can be handled via objects (json)
In react we have handle form ourself, for
    To get values
    To validate input
    Manage state, submission

Define handlInput(event) method onChange every element to maintain entries
onSubmit(): event is used to submit form to the application from UI

Syntax:

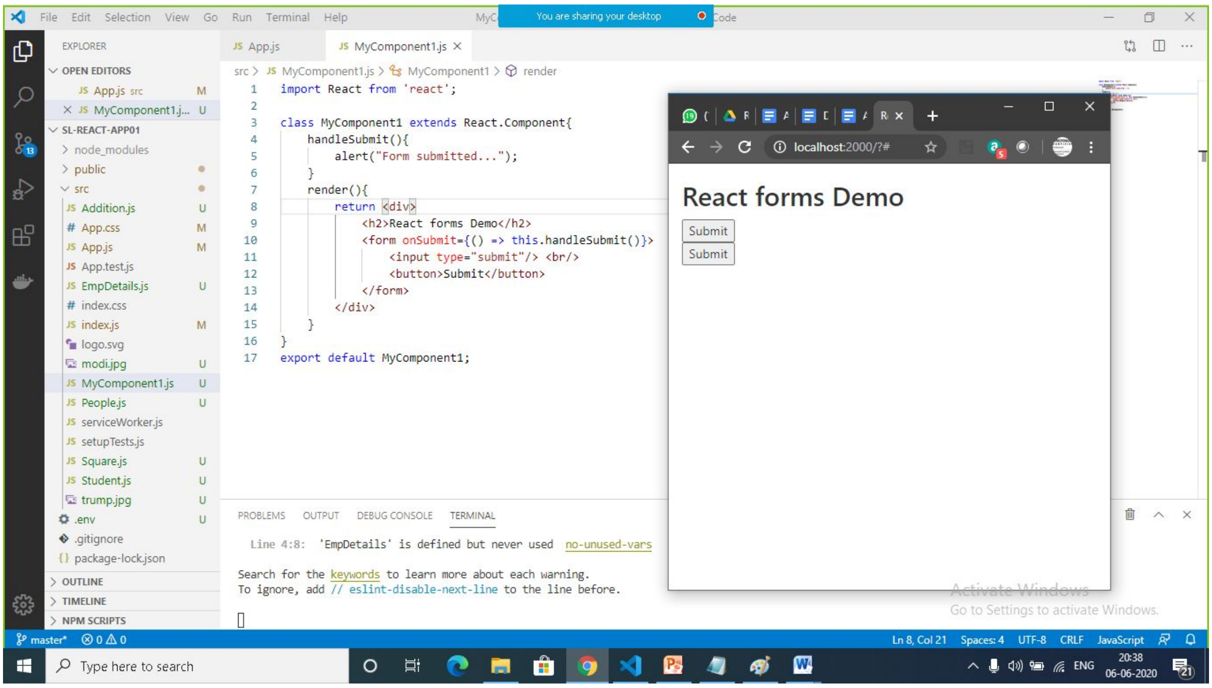
```
<form onSubmit = "this.hadleSubmit()">
      <input type="submit" />
</form>
```



```
import EmpDetails from './EmpDetails';
import MyComponent1 from './MyComponent1';
function App() {

  return (<div className="container">

  <MyComponent1 />

    </div>
  );
}

export default App;
```

```
Line 4:8:  'EmpDetails' is defined but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

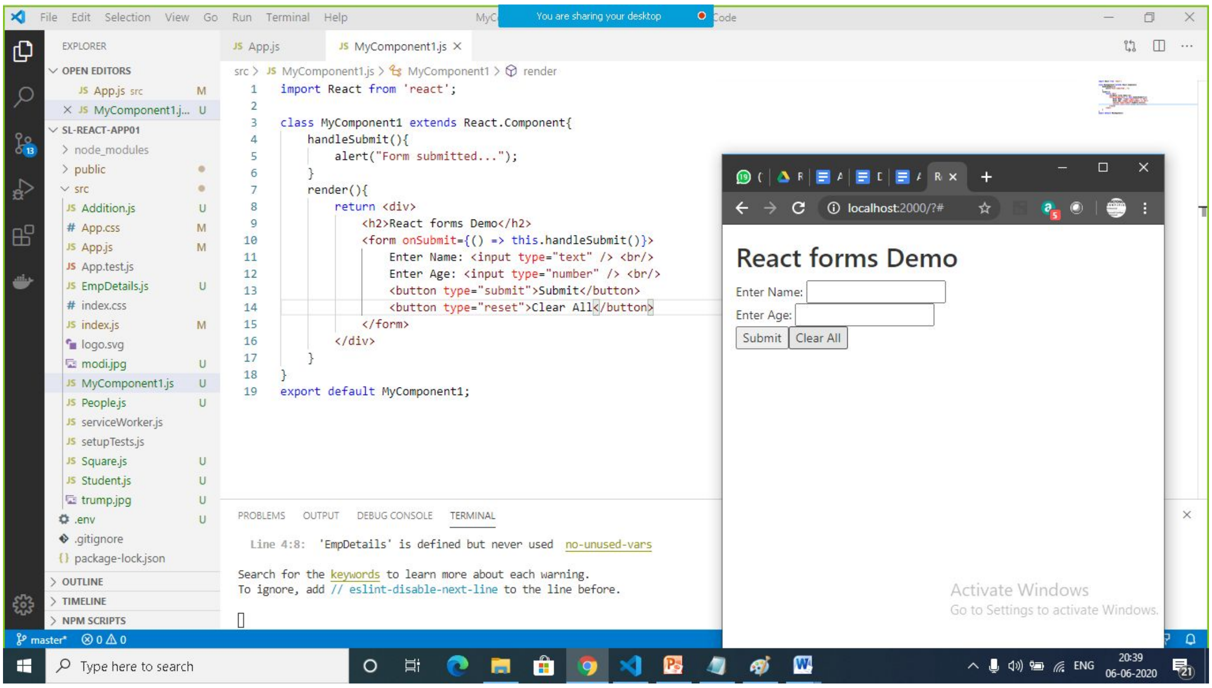
React forms Demo



```
import React from 'react';

class MyComponent1 extends React.Component{
    handleSubmit(){
        alert("Form submitted...");
    }
    render(){
        return <div>
            <h2>React forms Demo</h2>
            <form onSubmit={() => this.handleSubmit()}>
                <input type="submit"/> <br/>
                <button>Submit</button>
            </form>
        </div>
    }
}
export default MyComponent1;
```

```
Line 4:8:  'EmpDetails' is defined but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

React forms Demo



```
import React from 'react';

class MyComponent1 extends React.Component{
    handleSubmit(){
        alert("Form submitted...");
    }
    render(){
        return <div>
            <h2>React forms Demo</h2>
            <form onSubmit={() => this.handleSubmit()}>
                Enter Name: <input type="text" /> <br/>
                Enter Age: <input type="number" /> <br/>
                <button type="submit">Submit</button>
                <button type="reset">Clear All</button>
            </form>
        </div>
    }
}
export default MyComponent1;
```

```
  Line 4:8:  'EmpDetails' is defined but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

React forms Demo

Enter Name:

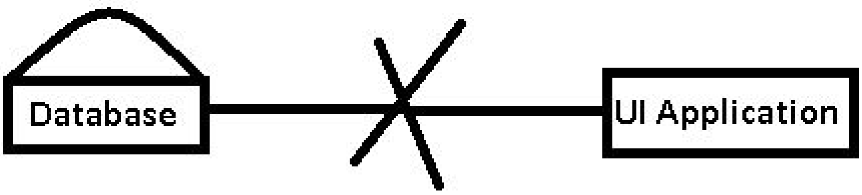
Enter Age:

Submit Clear All

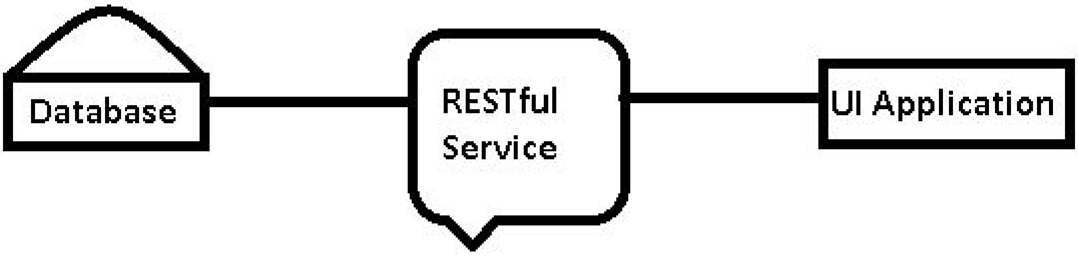
```
import React from 'react';

class MyComponent1 extends React.Component{
    state={ename:null,age:null,gender:null};
    handleSubmit(){
        alert( JSON.stringify( this.state ));
    }
    handleInputChange(event){
        var name = event.target.name;
        var value = event.target.value;
        this.setState({[name]:value});
    }
    render(){
        return <div>
            <h2>React forms Demo</h2>
            <form onSubmit={() => this.handleSubmit()}>
                Enter Name: <input type="text" name="ename"
onChange={(e)=>this.handleInputChange(e)} /> <br/>
                Enter Age: <input type="number" name="age"
onChange={(e)=>this.handleInputChange(e)} /> <br/>
                Enter Gender: <input type="text" name="gender"
onChange={(e)=>this.handleInputChange(e)} />               <br/>
                <button type="submit">Submit</button>
                <button type="reset">Clear All</button>
            </form>
        </div>
    }
}
export default MyComponent1;
```

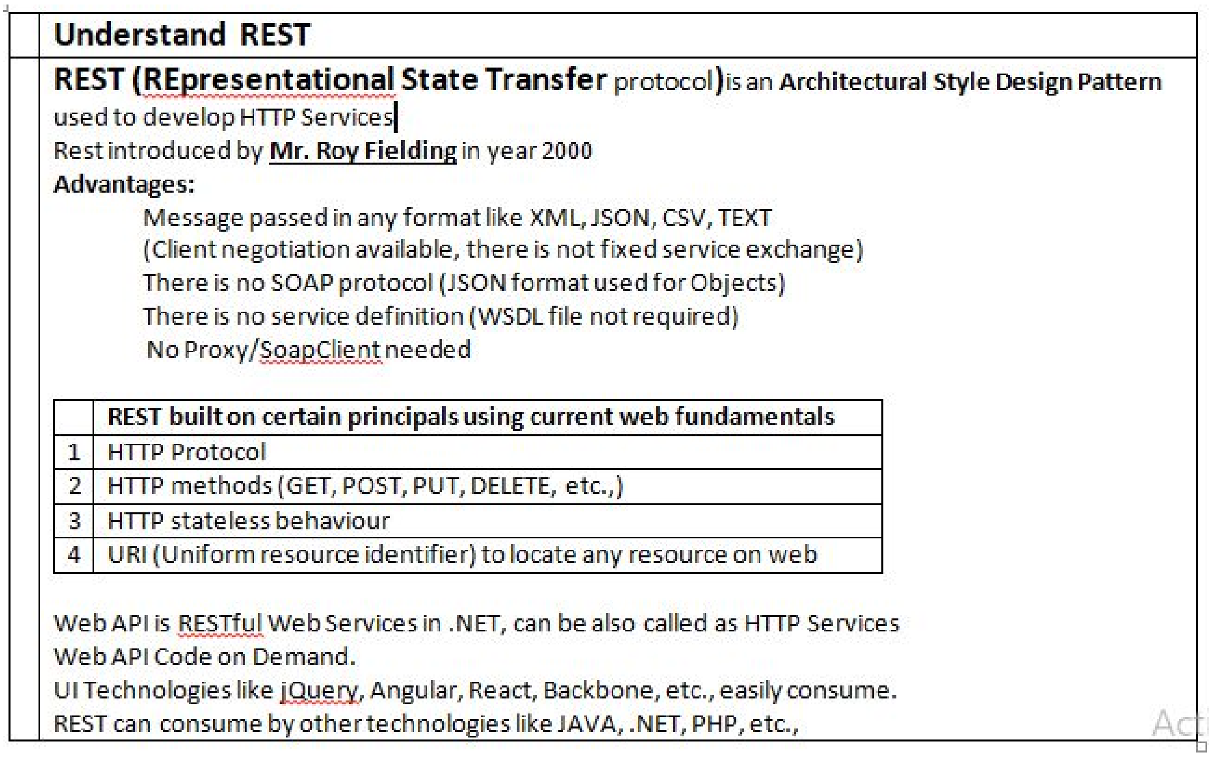
	RESTful Service
	Representational State Transfer It is used to create a service which can be consumed any application developed using any technology REST is also called as Web API In real time UI Application cannot communicate to database, they will communicate to database via RESTful Services Using Node.js, Java, .NET, Python, etc., allow to create REST ful Services Any technology including java, .NET, etc., and Ui Technologies like Angular, React, etc., can communicate with RESTful Services

UI Application cannot communicate with databases



UI Application communicate with database or datasource via RESTful Services





Understand REST

REST (REpresentational State Transfer protocol**)**is an **Architectural Style Design Pattern** used to develop HTTP Services

Rest introduced by **Mr. Roy Fielding** in year 2000

Advantages:

Message passed in any format like XML, JSON, CSV, TEXT
(Client negotiation available, there is not fixed service exchange)
There is no SOAP protocol (JSON format used for Objects)
There is no service definition (WSDL file not required)
No Proxy/SoapClient needed

	REST built on certain principals using current web fundamentals
1	HTTP Protocol
2	HTTP methods (GET, POST, PUT, DELETE, etc.,)
3	HTTP stateless behaviour
4	URI (Uniform resource identifier) to locate any resource on web

Web API is RESTful Web Services in .NET, can be also called as HTTP Services
Web API Code on Demand.
UI Technologies like jQuery, Angular, React, Backbone, etc., easily consume.
REST can consume by other technologies like JAVA, .NET, PHP, etc.,

	To create RESTful Service
	All technologies like Java, .NET, Python, Node.js etc., used to develop RESTful Services Angular, React and other UI Technologies cannot develop RESTful Services, consume RESTful Services

json-server

This module is used to create a RESTful Service for training purpose
This is a fake API development tool available with node.

To install: npm install -g json-server
After installation: json-server CLI available

To create and run Web API: json-server --watch kiranApi1.json

Note: This API is available in the port number 3000

Step1: npm install -g json-server

Step2: json-server --watch kiranApi.json:

Step3: edit kiranapi.json file with source

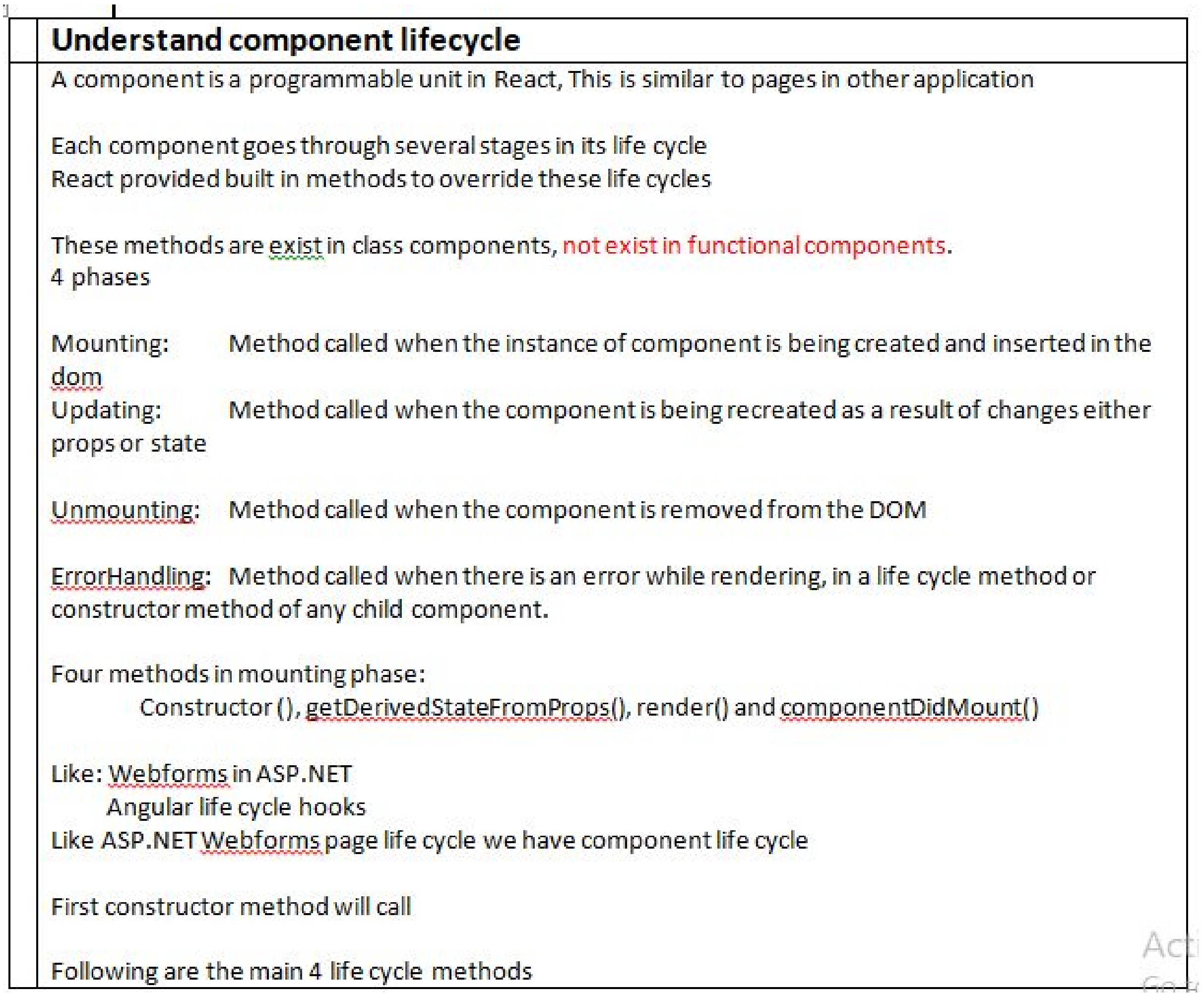
```
{
  "people":[
    {"id":1001,"pname":"Anil","gender":"Male","age":20},
    {"id":1002,"pname":"Madhu","gender":"Female","age":28},
    {"id":1003,"pname":"Ganesh","gender":"Male","age":23},
    {"id":1004,"pname":"Chanti","gender":"Male","age":29},
    {"id":1005,"pname":"Devika","gender":"Female","age":30}
  ],
  "departments":[


  ]
}
```

Operations allowed in Web API

get() :	Refers to fetch data
post() :	Refers to add data
put() :	Refers to edit data
delete():	Refers to delete data

In order to access Web API using react following ways

Javascript and ajax
fetch ()
axios



Understand component lifecycle

A component is a programmable unit in React, This is similar to pages in other application

Each component goes through several stages in its life cycle
React provided built in methods to override these life cycles

These methods are exist in class components, not exist in functional components.
4 phases

Mounting: Method called when the instance of component is being created and inserted in the dom
Updating: Method called when the component is being recreated as a result of changes either props or state

Unmounting: Method called when the component is removed from the DOM

ErrorHandling: Method called when there is an error while rendering, in a life cycle method or constructor method of any child component.

Four methods in mounting phase:
Constructor (), getDerivedStateFromProps(), render() and componentDidMount()

Like: Webforms in ASP.NET
Angular life cycle hooks
Like ASP.NET Webforms page life cycle we have component life cycle

First constructor method will call

Following are the main 4 life cycle methods

	Method	
	constructor	This method will be invoked first in component This is to set default values in state or to read props
	componentWillMount	This event method invoked before rendering the component This is equivalent to pre_init method in ASP.NET webforms

	componentDidMount	This event method invoked after rendering the component This is equivalent to page_init method Load Web API kind of code will be under this event
	componentWillReceiveProps	This event method invoked when component received props
	shouldComponentUpdate	This event method invoked before rendering, after receiving the props
	componentWillUpdate	

```
import React from 'react';

class MyComponent1 extends React.Component{
    state={people:[]};
    componentDidMount(){
        var url = "http://localhost:3000/people";
        fetch(url)
            .then(response=>response.json())
            .then(response => this.setState({people:response}))
    }
    render(){
        var people = this.state.people;
        return <div>
            <h2>People Component</h2>
            <table className="table table-bordered table-hover">
            <thead><tr>
                    <th>ID</th> <th>Name of the Person</th> <th>Gender</th>
<th>Age</th>
                </tr> </thead>
                <tbody>
                {people.map(p => <tr><td>{p.id}</td>
                                <td>{p.pname}</td>
                                <td>{p.gender}</td>
                                <td>{p.age}</td>
                </tr>
                )}
                </tbody>
            </table>
        </div>
    }
}
export default MyComponent1;
```

GET and POST methods

```
import React from 'react';

class MyComponent1 extends React.Component{
    state={people:[]};
    componentDidMount(){
        var url = "http://localhost:3000/people";
        fetch(url)
            .then(response=>response.json())
            .then(response => this.setState({people:response}));
    }

    addPerson(){
        let person = {
            "id": Number(this.refs.id.value),
            "pname":this.refs.pname.value,
            "gender":this.refs.gender.value,
            "age":Number(this.refs.age.value)
        };
        let url = "http://localhost:3000/people";
        fetch(url,{
            method:'POST',
            headers:{'content-type':'application/json'},
            body:JSON.stringify(person)
        })
        .then(response=>response.json())
        .then(() => this.setState({msg:'Row added....'}));
    }
    render(){
        var people = this.state.people;
        return <div>
```

```
            <h2>People Component</h2>
            <table className="table table-bordered table-hover">
            <thead><tr>
                    <th>ID</th> <th>Name of the Person</th> <th>Gender</th> <th>Age</th>
                </tr> </thead>
                <tbody>
                {people.map(p => <tr><td>{p.id}</td>
                                <td>{p.pname}</td>
                                <td>{p.gender}</td>
                                <td>{p.age}</td>
                </tr>
                )}
                </tbody>
            </table> <br/>
            ID: <input type="number" ref="id" /> <br/>
            Person Name: <input type="text" ref="pname" /> <br/>
            Gender: <input type="text" ref="gender" /> <br/>
            Age: <input type="number" ref="age" /> <br/>
            <button onClick={()=>this.addPerson()}>Add Person</button> {this.state.msg}
        </div>
    }
}
export default MyComponent1;
```

CRUD operations on Web API

```
import React from 'react';

class MyComponent1 extends React.Component{
    state={people:[]};
    componentDidMount(){
        var url = "http://localhost:3000/people";
        fetch(url)
            .then(response=>response.json())
            .then(response => this.setState({people:response}));
    }
    componentDidUpdate(){
        var url = "http://localhost:3000/people";
        fetch(url)
            .then(response=>response.json())
            .then(response => this.setState({people:response}));
    }
    addPerson(){
        let person = {
            "id": Number(this.refs.id.value),
            "pname":this.refs.pname.value,
            "gender":this.refs.gender.value,
            "age":Number(this.refs.age.value)
        };
        let url = "http://localhost:3000/people";
        fetch(url,{
            method:'POST',
            headers:{'content-type':'application/json'},
            body:JSON.stringify(person)
        })
        .then(response=>response.json())
        .then(() => this.setState({msg:'Row added....'}));
    }
    editPerson(){
        let person = {
            "id": Number(this.refs.id.value),
            "pname":this.refs.pname.value,
            "gender":this.refs.gender.value,
            "age":Number(this.refs.age.value)
        };
        let url = `http://localhost:3000/people/${person.id}`;
        fetch(url,{
            method:'PUT',
            headers:{'content-type':'application/json'},
            body:JSON.stringify(person)
        })
        .then(response=>response.json())
        .then(() => this.setState({msg:'Row updated....'}));
    }
    deletePerson(){
            let id =Number(this.refs.id.value);
        let url = `http://localhost:3000/people/${id}`;
        fetch(url,{
            method:'DELETE'
        })
        .then(response=>response.json())
        .then(() => this.setState({msg:'Row deleted....'}));
    }
    render(){
        var people = this.state.people;
        return <div>
            <h2>People Component</h2>
            <table className="table table-bordered table-hover">
            <thead><tr>
                    <th>ID</th> <th>Name of the Person</th> <th>Gender</th> <th>Age</th>
                </tr> </thead>
                <tbody>
                {people.map(p => <tr><td>{p.id}</td>
                                <td>{p.pname}</td>
                                <td>{p.gender}</td>
                                <td>{p.age}</td>
                </tr>
                )}
                </tbody>
            </table> <br/>
            ID: <input type="number" ref="id" /> <br/>
            Person Name: <input type="text" ref="pname" /> <br/>
            Gender: <input type="text" ref="gender" /> <br/>
            Age: <input type="number" ref="age" /> <br/>
            <button onClick={()=>this.addPerson()}>Add Person</button> |
            <button onClick={()=>this.editPerson()}>Edit Person</button> |
            <button onClick={()=>this.deletePerson()}>Delete Person</button>
  {this.state.msg}
        </div>
    }
  }
  export default MyComponent1;
```

Axios: library specially designed to access API

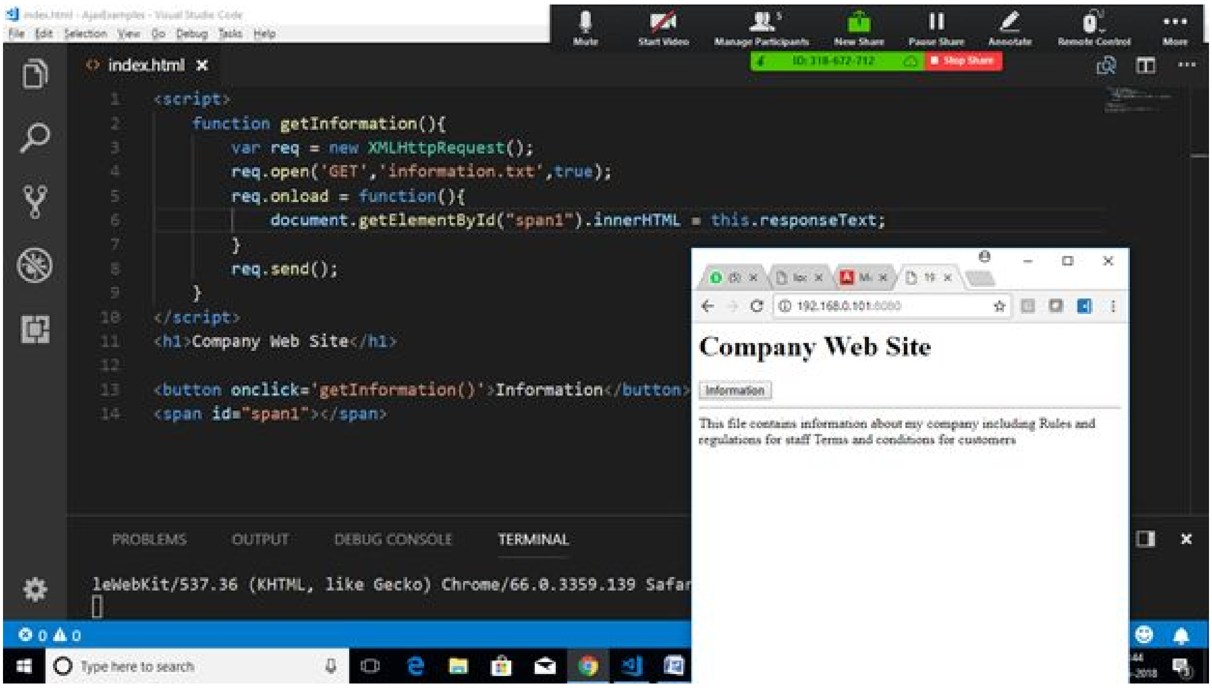
This API is available from normal .js file also

```
npm install axios
```

```
var axios = require(“axios”).default;

var url = http://localhost:3000/deparments;

var deparments = [];

axios.get(url)

                .then(response => console.log ( response.ddata)

                .catch( error => cosole.log ( error );

 var axios = require("axios").default;


var url='http://localhost:3000/employees';

var employees=[];

var error='';
```

```
axios.get(url)

    .then(response=>console.log(response.data))

            .catch(error=>this.error=error);

 var axios = require("axios").default;


var url='http://localhost:3000/employees';

var employees=[];


var getEmployees = async () => {

            await

            axios.get(url)

    .then(response=> employees = response.data )

    .catch(error=>this.error=error);

}

getEmployees()

    .then(()=>console.log(employees));
```

Fetch data using ajax with javascript



```
<script>
    function getInformation(){
        var req = new XMLHttpRequest();
        req.open('GET','information.txt',true);
        req.onload = function(){
            document.getElementById("span1").innerHTML = this.responseText;
        }
        req.send();
    }
</script>
<h1>Company Web Site</h1>

<button onclick='getInformation()'>Information</button>
<span id="span1"></span>
```

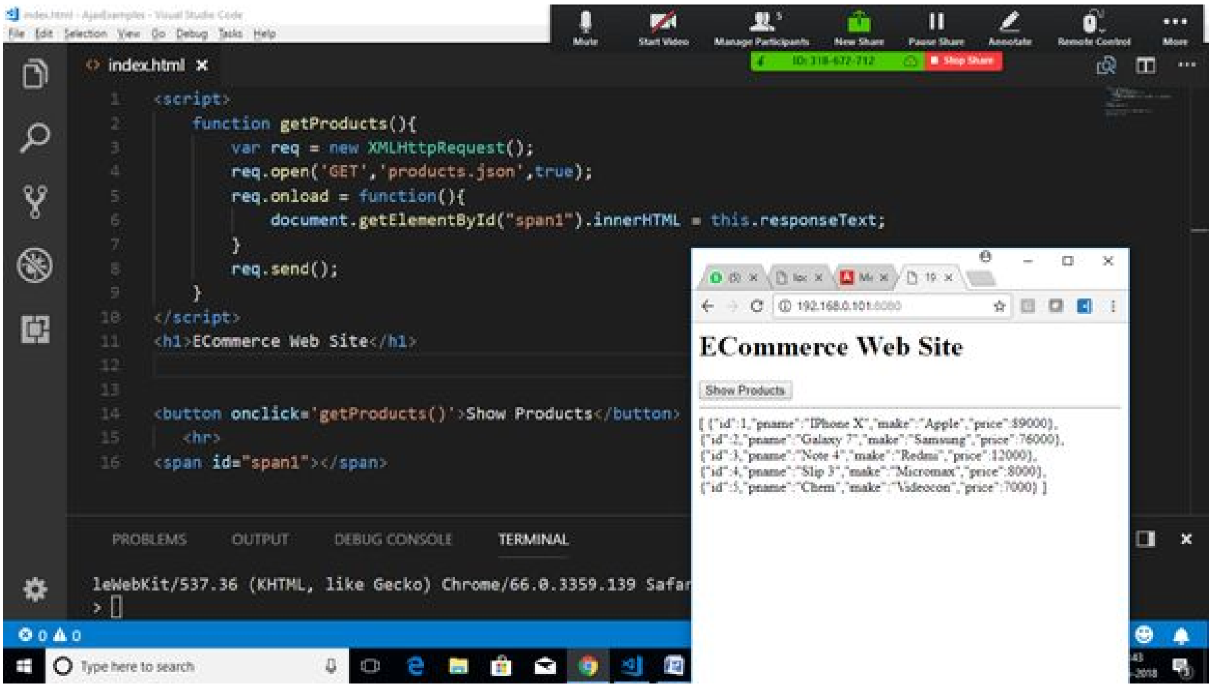
```
leWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.139 Safar
```

192.168.0.101:8080

Company Web Site

Information

This file contains information about my company including Rules and regulations for staff Terms and conditions for customers



```
<script>
    function getProducts(){
        var req = new XMLHttpRequest();
        req.open('GET','products.json',true);
        req.onload = function(){
            document.getElementById("span1").innerHTML = this.responseText;
        }
        req.send();
    }
</script>
<h1>ECommerce Web Site</h1>


<button onclick='getProducts()'>Show Products</button>
    <hr>
<span id="span1"></span>
```

```
leWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.139 Safar
>
```

192.168.0.101:8080

ECommerce Web Site

Show Products

```
[ {"id":1,"pname":"IPhone X","make":"Apple","price":89000},
{"id":2,"pname":"Galaxy 7","make":"Samsung","price":76000},
{"id":3,"pname":"Note 4","make":"Redmi","price":12000},
{"id":4,"pname":"Slip 3","make":"Micromax","price":8000},
{"id":5,"pname":"Chem","make":"Videocon","price":7000} ]
```

.ajax method for jquery

This method is available in jQuery module, used to request to any remote server asynchronously

Following 3 methods can be chained with callbacks

done() => This method invoke callback if the able to connect to remote without any issue

fail() => This method invoke callback when jquery failed to connect to remove

always()=> This method mandate call callback when the request raised to remote server

Jquery reading file content using Ajax

```
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>

<script>

        $(document).ready(function(){

        $("#b1").click(function(){

                var jqxhr = $.ajax( "information1.txt" )

                .done(function() {

                //alert( "success" );

                $("#span1").load('information1.txt');

                })

                .fail(function() {

                //alert( "error" );

                $("#span1").text("File not found");

                })

                .always(function() {

                //alert( "complete" );

                });

        });


        });

</script>

<h1>jQuery Ajax</h1>


<button onclick='getInformation()' id="b1">Get Information </button>     <hr>

<span id="span1"></span>
```

Using Ajax, we can read all type of files like .txt, .json, .html, etc.,

==========

Ajax loading part of the page from the source

	Consume API exposing random joke

```
<head>

<script src="./node_modules/jquery/dist/jquery.min.js"></script>

<script>

        function display(){

        var req = new XMLHttpRequest();

        var url = "http://api.icndb.com/jokes/random";

        req.open('GET',url,true);

        req.onload = function(){

        var response =JSON.parse( this.responseText);

        $("#span1").text(  response.value.joke);

        }

        req.send();

        }

</script>

</head>

<h2>Ajax with javascript, Example-1</h2>


<button id="b1" onclick="display()">a1 file </button>   <hr>

<span id="span1"></span>
```

JSON.parse()

Used to Convert a string into json object

When receiving data from a web server, the data is always a string and need to parse into JSON for programming.

	Parse the data with JSON.parse(), and the data becomes a JavaScript object.

	JSON.stringify()
	Used to Convert a JavaScript object into a string UI Technologies commonly uses JSON is to exchange data to/from a web server. When sending data to a remote server, the data has to be a string and done using JSON.stringify()

Understand Promises

Asynchronous refers to submission of request asynchronously

Promise is a an object contains resolve and reject parameters, contains two sections under asynchronous call which will execute based on conditions

ES6 provided Javascript promise object

The Promise type is a callback used to initialize a promise. This callback passed two arguments resolve and reject.

resolve is used to resolve the promise with a value or result of another promise.

reject is used to reject the promise with a provided reason or error.

Syntax:

```
var promise = new Promise(function(resolve,reject){

        let value = true;

        if ( value )

        resolve("The value is true");

        else

        resolve("The value is false");

});



promise.then( x=> console.log(x), x=>console.log(x));
```

https://davids-restaurant.herokuapp.com/menu_items.json

Jquery ajax promise

==============

Suggestions for other page

```
<div id="div1">

        Enter your Employee ID

</div>

<div id="div2">

        Enter your Name

</div>

<div id="div3">

        Enter your Job

</div>

<div id="div4">

        Enter your Salary

</div>
```

//index.html

```
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>

<script>

        $(document).ready(function(){

        $("#t1").focus(function(){

                $("#div1").load("suggestions.html #div1");

        });
        $("#t1").blur(function(){

                $("#div1").html('');

        });

        });

</script>

<table border="1">

        <tr>

        <td>ID</td>

        <td><input type="text" id="t1"></td>

        <td><div id="div1"></div></td>

        </tr>

        <tr>

        <td>EName</td>

        <td><input type="text"></td>

        <td><div id="div2"></div></td>

        </tr>

        <tr>

        <td>Job</td>

        <td><input type="text"></td>

        <td><div id="div3"></div></td>

        </tr>

        <tr>

        <td>Salary</td>

        <td><input type="text"></td>

        <td><div id="div4"></div></td>

        </tr>

</table>
```

.getJSON()		=>	This is an ajax method used to fetch the response in json format

http://date.jsontest.com/