**What React Is (React JS Crash Course):**

React is an open source JavaScript library for building dynamic user interfaces

It is the “V” in MVC (Model–view–controller) and has nothing to do with the back end of one’s application

* The model is responsible for managing the data of the application. It receives user input from the controller.
* The view means the presentation of the model in a particular format.
* The controller responds to the user input and performs interactions on the data model objects. The controller receives the input, optionally validates it and then passes the input to the model.
* Model >> (updates) >> View >> (sees) >> User >> (uses) >> Controller >> (manipulates) >> Model

Maintained by Facebook, Instagram and a community of individual developers

**Some Advantages of React:**

* Design simple declarative (宣言的) views for each state in your application
* Encapsulated components
* Dynamic properties & state
* Virtual DOM (Document Object Model)
  + This allows one to render certain parts of DOM
* Completely independent of the rest of your application
* Can render on the client or server

**Virtual DOM:**

React abstracts away the DOM and creates its own version which is simplified and only includes the things that you need

* Helps to identify which parts have changed
* Determines how to upload the browsers’ DOM more efficiently
* It is much more lightweight/It works faster

**Simple Application:**

Class HelloMessage extends React.Component {

render() {

return <div>Hello {this.props.name}</div>;

// This part is actually JSX (JavaScript Syntax Extension)

}

}

ReactDOM.render(<HelloMessage name="Jane"/>, mountNode);

// We render HelloMessage to mountNode. We can use mountNode in HTML file.

**JSX—JavaScript Syntax Extension:**

A pre-processor step that adds XML syntax to JavaScript

* Looks like XML/HTML (there are some differences)
* Defines a familiar syntax for defining tree structures with attributes
* Is not required but makes things much easier

**What You’ll Learn:**

* How to create a React app
* Create components
* Manage state and properties
* Handle events
* Work with forms and input
* Work with JSX
* Lifecycle methods
* Fetch data from an API

**How to create a React app:**

Other than using NPM, one can also use CDN (Content Delivery Network) to work with React. One just needs to insert some lines in the file “index.html”.

The tool “create-react-app” allows one to start working with React right away

* Enter “npm install -g create-react-app” in the console
* Enter “create-react-app project-manager”, “cd project-manager” and “npm start”

<!-- ./public/index.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1,

shrink-to-fit=no">

<meta name="theme-color" content="#000000">

<!--

manifest.json provides metadata used when your web app is added to the

homescreen on Android. See ttps://developers.google.com/web/fundamentals/

engage-and-retain/web-app-manifest/.

-->

<link rel="manifest" href="%PUBLIC\_URL%/manifest.json">

<link rel="shortcut icon" href="%PUBLIC\_URL%/favicon.ico">

<!--

Notice the use of %PUBLIC\_URL% in the tags above.

It will be replaced with the URL of the `public` folder during the build.

Only files inside the `public` folder can be referenced from the HTML.

Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

work correctly both with client-side routing and a non-root public URL.

Learn how to configure a non-root public URL by running `npm run build`.

-->

<title>React App</title>

</head>

<body>

<noscript>

You need to enable JavaScript to run this app.

</noscript>

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

<!--

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add web fonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the <body> tag.

To begin the development, run `npm start` or `yarn start`.

To create a production bundle, use `npm run build` or `yarn build`.

-->

</body>

</html>

/\* ./public/manifest.json \*/

{

"short\_name": "React App",

"name": "Create React App Sample",

"icons": [

{

"src": "favicon.ico",

"sizes": "64x64 32x32 24x24 16x16",

"type": "image/x-icon"

}

],

"start\_url": "./index.html",

"display": "standalone",

"theme\_color": "#000000",

"background\_color": "#ffffff"

}

/\* ./src/index.js \*/

import React from 'react';

import ReactDOM from 'react-dom';

// import './index.css'; // We do not need this

import App from './App';

import registerServiceWorker from './registerServiceWorker';

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

registerServiceWorker();

/\* ./src/App.js \*/

import React, { Component } from 'react';

// import logo from './logo.svg';

import './App.css';

class App extends Component {

render() {

return (

<div className="App">

My App

</div>

);

}

}

export default App;

/\* ./src/App.css (empty) \*/

**Create components:**

/\* ./src/App.js \*/

import React, { Component } from 'react';

import Projects from './Components/Projects'

import './App.css';

class App extends Component {

render() {

return (

<div className="App">

My App

<Projects test="Hello World"/> // The component is used here

</div>

);

}

}

export default App;

/\* ./src/Components.js \*/

import React, { Component } from 'react';

class Projects extends Component {

render() {

return (

<div className="Projects">

My Projects ({this.props.test})

</div>

);

}

}

export default Projects;

**Manage state and properties:**

/\* ./src/App.js \*/

import React, { Component } from 'react';

import Projects from './Components/Projects'

import './App.css';

class App extends Component {

// We want our data to be held in “state”

// Usually, the data in “state” are fetched from APIs or databases

constructor() {

super();

// Here is where one defines the initial state

// You probably want to use a lifecycle method and componentWillMount

// instead of storing actual data here

this.state = {

projects: [

{

title: 'Business Website',

category: 'Web Design'

},

{

title: 'Social App',

category: 'Mobile Development'

},

{

title: 'Ecommerce Shopping Cart',

category: 'Web Development'

}

]

}

}

render() {

return (

<div className="App">

My App

<Projects projects={this.state.projects}/>

</div>

);

}

}

export default App;

/\* ./src/Components/Projects.js \*/

import React, { Component } from 'react';

import ProjectItem from './ProjectItem'

class Projects extends Component {

render() {

// console.log(this.props); // Check out Chrome DevTools' console

let projectItems;

if(this.props.projects){

projectItems = this.props.projects.map(project => {

return (

<ProjectItem key={project.title} project={project}/>

);

});

}

return (

<div className="Projects">

{projectItems}

</div>

);

}

}

export default Projects;

/\* ./src/Components/ProjectItem.js \*/

import React, { Component } from 'react';

class ProjectItem extends Component {

render() {

return (

<li className="Project">

<strong>{this.props.project.title}:</strong> {this.props.project.category}

</li>

// We cannot use the keyword "class" naturally

);

}

}

export default ProjectItem;

To improve the data storage for App.js, …

/\* ./src/App.js \*/

import React, { Component } from 'react';

import Projects from './Components/Projects'

import './App.css';

class App extends Component {

constructor() {

super();

this.state = {

projects: []

}

}

componentWillMount(){

// componentWillMount is a lifecycle method

// Whenever the component is about to render to the DOM for the first time,

// this function will be called

// If you want to fetch data from, for instance, an Ajax call,

// you will want to do it here

// See more lifecycle methods in the documentation

this.setState({projects:[

{

title: 'Business Website',

category: 'Web Design'

},

{

title: 'Social App',

category: 'Mobile Development'

},

{

title: 'Ecommerce Shopping Cart',

category: 'Web Development'

}

]});

}

render() {

return (

<div className="App">

My App

<Projects projects={this.state.projects}/>

</div>

);

}

}

export default App;

**Handle events and work with forms and input:**

/\* ./src/Components/AddProject.js \*/

import React, { Component } from 'react';

class AddProject extends Component {

constructor(){

super();

this.state = {

// We want to set this once the form is submitted

newProject:{}

}

}

static defaultProps = {

categories: ['Web Design', 'Web Development', 'Mobile Development']

}

handleSubmit(e){

// console.log("Submitted");

if(this.refs.title.value === ''){

// We check if the input field is empty

alert('Title is required.');

} else {

// console.log(this.refs.title.value); // "refs" is not a typo

this.setState({newProject:{

title: this.refs.title.value,

category: this.refs.category.value

}}, function(){

// This is a callback function

// console.log(this.state);

// Note that the state is different from the one in App.js

this.props.addProject(this.state.newProject); // We pass the newProject up

});

}

e.preventDefault();

// By default, the form is submitted and the function like

//console.log will not be called

}

// The attribute "ref" helps us to get the value when we submit the form

// ".bind(this)" is necessary in for this.handleSubmit

render() {

let categoryOptions = this.props.categories.map(category => {

return <option key={category} value={category}>{category}</option>

});

return (

<div>

<h3>Add Project</h3>

<form onSubmit={this.handleSubmit.bind(this)}>

<div>

<label>Title</label><br/>

<input type="text" ref="title"/>

</div>

<div>

<label>Category</label><br/>

<select ref="category">

{categoryOptions}

</select>

</div>

<input type="submit" value="Submit"/>

</form>

</div>

);

}

}

export default AddProject;

/\* ./src/App.js \*/

import React, { Component } from 'react';

import AddProject from './Components/AddProject' // New

import Projects from './Components/Projects'

import './App.css';

class App extends Component {

constructor() {

super();

this.state = {

projects: []

}

}

componentWillMount(){

this.setState({projects:[

{

title: 'Business Website',

category: 'Web Design'

},

{

title: 'Social App',

category: 'Mobile Development'

},

{

title: 'Ecommerce Shopping Cart',

category: 'Web Development'

}

]});

}

handleAddProject(project){

let projects = this.state.projects;

projects.push(project);

this.setState({projects: projects});

// State in React is immutable

}

render() {

return (

<div className="App">

<AddProject addProject={this.handleAddProject.bind(this)}/>

<Projects projects={this.state.projects}/>

</div>

);

}

}

export default App;

/\* ./src/Components/Projects.js \*/

import React, { Component } from 'react';

import ProjectItem from './ProjectItem'

class Projects extends Component {

render() {

let projectItems;

if(this.props.projects){

projectItems = this.props.projects.map(project => {

return (

<ProjectItem key={project.title} project={project}/>

);

});

}

// "<h3>Latest Projects</h3>" is added so that

// the content will not scrunch together"

return (

<div className="Projects">

<h3>Latest Projects</h3>

{projectItems}

</div>

);

}

}

export default Projects;

When one reloads the page, the data is gone. React is for user interfaces. If we want the data to persist, we can save the data using API or local storage (check out MongoDB).

We want to have some unique IDs

* Type “npm install -S uuid” in the console

/\* ./src/App.js \*/

import React, { Component } from 'react';

import uuid from 'uuid'; // New

import AddProject from './Components/AddProject';

import Projects from './Components/Projects';

import './App.css';

class App extends Component {

/\* ... \*/

componentWillMount(){

this.setState({projects:[

{

id: uuid.v4(), // “uuid.v4()” generates a unique ID

title: 'Business Website',

category: 'Web Design'

},

{

id: uuid.v4(),

title: 'Social App',

category: 'Mobile Development'

},

{

id: uuid.v4(),

title: 'Ecommerce Shopping Cart',

category: 'Web Development'

}

]});

}

/\* ... \*/

}

export default App;

/\* ./src/Components/AddProject.js \*/

import React, { Component } from 'react';

import uuid from 'uuid'; // New

class AddProject extends Component {

/\* ... \*/

handleSubmit(e){

if(this.refs.title.value === ''){

alert('Title is required.');

} else {

this.setState({newProject:{

id: uuid.v4(),

title: this.refs.title.value,

category: this.refs.category.value

}}, function(){

this.props.addProject(this.state.newProject);

});

}

e.preventDefault();

}

/\* ... \*/

}

export default AddProject;

We want to be able to delete the items

/\* ./src/Components/ProjectItem.js \*/

import React, { Component } from 'react';

class ProjectItem extends Component {

deleteProject(id){ // New

this.props.onDelete(id);

}

render() {

return (

<li className="Project">

<strong>{this.props.project.title}: </strong>

{this.props.project.category} <a href="#"

onClick={this.deleteProject.bind(this, this.props.project.id)}>X</a>

</li>

);

}

}

export default ProjectItem;

/\* ./src/Components/Projects.js \*/

import React, { Component } from 'react';

import ProjectItem from './ProjectItem'

class Projects extends Component {

deleteProject(id){ // New

this.props.onDelete(id);

}

render() {

let projectItems;

if(this.props.projects){

projectItems = this.props.projects.map(project => {

return (

<ProjectItem key={project.title} onDelete={this.deleteProject.bind(this)}

project={project}/>

);

});

}

return (

<div className="Projects">

<h3>Latest Projects</h3>

{projectItems}

</div>

);

}

}

export default Projects;

/\* ./src/App.js \*/

import React, { Component } from 'react';

import uuid from 'uuid'; // New

import AddProject from './Components/AddProject';

import Projects from './Components/Projects';

import './App.css';

class App extends Component {

constructor() {

super();

this.state = {

projects: []

}

}

componentWillMount(){

this.setState({projects:[

{

id: uuid.v4(),

title: 'Business Website',

category: 'Web Design'

},

{

id: uuid.v4(),

title: 'Social App',

category: 'Mobile Development'

},

{

id: uuid.v4(),

title: 'Ecommerce Shopping Cart',

category: 'Web Development'

}

]});

}

handleAddProject(project){

let projects = this.state.projects;

projects.push(project);

this.setState({projects: projects});

}

handleDeleteProject(id){

let projects = this.state.projects;

let index = projects.findIndex(x => x.id === id); // findIndex is built-in

projects.splice(index, 1);

this.setState({projects: projects});

}

render() {

return (

<div className="App">

<AddProject addProject={this.handleAddProject.bind(this)}/>

<Projects projects={this.state.projects}

onDelete={this.handleDeleteProject.bind(this)}/>

</div>

);

}

}

export default App;

To check the type of props, enter “npm install -S prop-types” in the console

/\* ./src/Components/Projects.js \*/

import PropTypes from 'prop-types'; // New

import React, { Component } from 'react';

import ProjectItem from './ProjectItem';

class Projects extends Component {

/\* ... \*/

}

export default Projects;

/\* ./src/Components/ProjectItem.js \*/

import PropTypes from 'prop-types'; // New

import React, { Component } from 'react';

class ProjectItem extends Component {

/\* ... \*/

}

ProjectItem.propTypes = {

project: PropTypes.object,

onDelete: PropTypes.func

}

export default ProjectItem;

/\* ./src/App.js \*/

import PropTypes from 'prop-types'; // New

import React, { Component } from 'react';

import uuid from 'uuid';

import AddProject from './Components/AddProject';

import Projects from './Components/Projects';

import './App.css';

class App extends Component {

/\* ... \*/

}

AddProject.propTypes = {

category: PropTypes.array,

addProject: PropTypes.func

}

export default App;

**Fetch data from an API:**

/\* ./src/App.js \*/

import $ from 'jquery';

import PropTypes from 'prop-types';

import React, { Component } from 'react';

import uuid from 'uuid';

import AddProject from './Components/AddProject';

import Projects from './Components/Projects';

import Todos from './Components/Todos';

import './App.css';

class App extends Component {

constructor() {

super();

this.state = {

projects: [],

todos: []

}

}

getTodos(){ // New

// We want to make some request here

$.ajax({

url: 'https://jsonplaceholder.typicode.com/todos', // Free online material

dataType: 'json',

cache: false,

success: function(data){

this.setState({todos: data}, function(){

console.log(this.state);

});

}.bind(this),

error: function(xhr, status, err){

console.log(err);

}

});

}

getProjects(){ // New

this.setState({projects:[

{

id: uuid.v4(),

title: 'Business Website',

category: 'Web Design'

},

{

id: uuid.v4(),

title: 'Social App',

category: 'Mobile Development'

},

{

id: uuid.v4(),

title: 'Ecommerce Shopping Cart',

category: 'Web Development'

}

]});

}

componentWillMount(){

// The default data is migrated

this.getProjects();

this.getTodos();

}

componentDidMount(){

this.getTodos();

}

handleAddProject(project){

let projects = this.state.projects;

projects.push(project);

this.setState({projects: projects});

}

handleDeleteProject(id){

let projects = this.state.projects;

let index = projects.findIndex(x => x.id === id);

projects.splice(index, 1);

this.setState({projects: projects});

}

render() {

return (

<div className="App">

<AddProject addProject={this.handleAddProject.bind(this)}/>

<Projects projects={this.state.projects}

onDelete={this.handleDeleteProject.bind(this)}/>

<hr/>

<Todos todos={this.state.todos}/>

</div>

);

}

}

AddProject.propTypes = {

category: PropTypes.array,

addProject: PropTypes.func

}

export default App;

/\* ./src/Components/Todos.js \*/

import PropTypes from 'prop-types'; // New

import React, { Component } from 'react';

import TodoItem from './TodoItem';

class Todos extends Component {

render() {

let todoItems;

if(this.props.todos){

todoItems = this.props.todos.map(todo => {

return (

<TodoItem key={todo.title} todo={todo}/>

);

});

}

return (

<div className="Todos">

<h3>TodoList</h3>

{todoItems}

</div>

);

}

}

Todos.propTypes = {

todos: PropTypes.array,

}

export default Todos;

/\* ./src/Components/TodoItem.js \*/

import PropTypes from 'prop-types';

import React, { Component } from 'react';

class TodoItem extends Component {

deleteProject(id){

this.props.onDelete(id);

}

render() {

return (

<li className="Project">

{this.props.todo.title}

</li>

);

}

}

TodoItem.propTypes = {

todo: PropTypes.object,

}

export default TodoItem;

Other than “get” request, we can also have “post” requests or submit data to external databases through API and so on.