REACT

MODULE ONE

1. What is React?

React is a JavaScript library created by Facebook for building UI components. Unlike more complete frameworks such as Angular or Vue, React deals only with the view layer. Hence, you'll need additional libraries to handle things such as data flow, routing, authentication etc.
Building a React project involves creating one or more React components that can interact with each other. A React component is simply a JavaScript class that requires the render function to be declared. The render function simply outputs HTML code, which is implemented using either JSX or JavaScript code. A React component may also require additional functions for handling data, actions and lifecyle events.

What is Virtual DOM?

The Virtual DOM is a lightweight, abstract model of the DOM. React uses the render method to create a node tree from React components and updates this tree in response to changes in the data model resulting from actions.
Each time there are changes to the underlying data in a React app, React creates a new Virtual DOM representation of the user interface.

Who uses React?

Today, thousands of companies worldwide are using React, including big names such as Netflix, Slack, Trello and AirBnB. React has become immensely popular, such that a number of apps have been ported to React --- including WhatsApp, Instagram and Dropbox.

Pre-requisite

You need some experience in
1. functional JavaScript
2. object-oriented JavaScript
3. ES6 JavaScript syntax
On your machine, you will need a node.js environment.

Text editor that would be used in this class is atom.

Visit this link to download and install atom https://atom.io/
Packages to be installed on atom
1. linter
2. atom-beautify
3. language-babel
4. platformio-ide-terminal

2. React Setup

Ways to get started with react

1. We can use a package called create-react-app to create a full react project whereby react controls the flow of all the application with the web pack set-up. We will look at this in the next class.
2. React CDN to quickly get up and running with react and it's also good when you want to create widget or update a small section of a web page

CDN LINK (without JSX)

```
<script type="text/javascript" src="https://fb.me/react-0.14.3.js"></script>
<script type="text/javascript" src="https://fb.me/react-dom-
0.14.3.js"></script>
```

Here, without the JSX we will use the React.createClass() function to create a class and the React.createElement() funtion to create elements.

Object is passed into the React.createClass() function like this React.createClass({}), the object must have a render(){return(null)} function. Instead of returning null, a React.createElement() can be returned which can accept three arguements - component, props, content.

Check these links for examples:
 https://codepen.io/opyzyle/pen/wNxPjQ
https://codepen.io/opyzyle/pen/YBjEoL

CDN LINK (with JSX)

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/babel-
standalone/6.23.1/babel.js"></script>
<script crossorigin
src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-
dom.development.js"></script>
```

Here, we will be extending the React.Component class

Check this example: https://codepen.io/opyzyle/pen/YBjYEd

3. What is a Component?

A component is a small reusable chunk of code that is responsible for one job. Usually to render some HTML. We can use multiple JSX in a component. Variables and conditions should be inside render method

React components provide two mechanisms for working with data: properties and state. Dividing data between immutable properties and mutable state more clearly identifies the role of each kind of data, and the component's relationship to it.

JavaScript

```
var components= React.createElement(components, {
      prop1: "some value",
      prop2: "some value"
});
```

JSX

```
var components = <components prop1="some value"  prop2="some value" />
```

What is Props?

Props are read-only and are set by a parent component.

- Information that gets passed from one component to another is known as ‘*props*’.
- A component’s props is an object. It holds information about that component. To see a component’s props object, you use the expression this.props
- We can pass props from component to component. Rendering is the only way for a component to pass props to another component.
- Every component’s props object has a property named children, this.props.children will return everything in between a component’s opening and closing JSX tags
- If nobody passes any text to component, then component display will be blank. It would be better if component could display a default message instead. That’s where defaultProps comes into picture.

4. What is a State?

State is defined within a component and can change during the lifecycle of a component

- Unlike props, a component’s state is not passed in from the outside. A component decides its own state.
- To make a component have state, give the component a state property. This property should be declared inside of a constructor method, like this:
  - *this.state* should be equal to an object.
  - To read a component’s state, use the expression *this.state.name-of-property*.
  - There is a stateful component, and a stateless component. “Stateful” describes any component that has a state property; “Stateless” describes any component that does not.

What is JSX?

Defined by the React Docs as an "extension to JavaScript" or “syntax sugar for calling React.createElement(component, props, ...children))”, JSX is what makes writing your React Components easy.
It doesn't render out to HTML directly, but instead renders to React Classes that are consumed by the Virtual DOM. Eventually, through the mysterious magic of the Virtual DOM, it will make its way to the page and be rendered out to HTML.
JSX is interpreted by a transpiler, such asBabel, and rendered to JavaScript code that the UI Framework (React, in this case) can understand.
https://es6console.com/ convert es6 to es5
Check these example on how state and props are used
JavaScript
https://codepen.io/opyzyle/pen/BMPYem
JSX
https://codepen.io/opyzyle/pen/Odwvzw

5. React Developers Tool

https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en

6. DOM Events

React supports events. This is the summary of the events
Clipboard: onCopy, onCut, onPaste
Focus Events: onFocus, onBlur
Form Events: onChange, onInput, onInvalid, onSubmit
Mouse Events: onClick, onContextMenu, onDoubleClick, onDrag, onDragEnd, onDragEnter, onDragExit, onDragLeave, onDragOver, onDragStart, onDrop, onMouseDown, onMouseEnter, onMouseLeave, onMouseMove, onMouseOut, onMouseOver, onMouseUp
Selection Events: onSelect
Keyboard: onKeyDown, onKeyPress, onKeyUp
Touch: onTouchCancel, onTouchEnd, onTouchMove, onTouchStart, UI, onScroll
Mouse: Wheel, onWheel
Media: onAbort, onCanPlay, onCanPlayThrough, onDurationChange, onEmptied, onEncrypted, onEnded, onError, onLoadedData, onLoadedMetadata, onLoadStart, onPause, onPlay, onPlaying, onProgress, onRateChange, onSeeked, onSeeking, onStalled, onSuspend, onTimeUpdate, onVolumeChange, onWaiting

Image: onLoad, onError
Animation: onAnimationStart, onAnimationEnd, onAnimationIteration, Transition, onTransitionEnd
We will look at how to use some of the common events in react
https://codepen.io/opyzyle/pen/bzOQoB

7. INTRO TO FORMS

Getting to this stage I'm very sure you're not new to form. You have seen in HTML, CSS, JavaScript and now we are about to deal with it in React. We will be looking at how to use form events and changing the states of react.
The example below explains how to play with form in react:
https://codepen.io/opyzyle/pen/ErGMxQ?editors=1111
Do the exercise before checking the result
Create a table of first name and last name in the browser. At every click of the submit button, the first name and last name values should appear in the table.
Solution: https://codepen.io/opyzyle/pen/GzPagd

MODULE
Two

1. Getting Started with create-react-app

To get started with react on the fly, we will use create-react-app tool. We will use our node environment, if you haven't installed nodejs, follow the steps below:
1. visit the node.js website: https://nodejs.org/en/
2. Download the "recommended for users option".
3. Install on your computer.
4. Go to your search bar and search for node.js, launch it by opening as an administrator

Now, you have your node.js environment ready to use.
Next, let's install create-react-app
1. Install or update npm to its latest version: $ npm i -g npm

npm (node package manager) which helps you install packages on your computer.
2. Install a tool: $ npm i -g create-react-app
3. Create a project folder on your computer
4. Navigate to your project's root directory(your project folder) and create a new React project using the tool we just installed: $ create-react-app student-app
5. Inside the directory, run the command, npm start

This might take a while to complete if this is your first time running create-react-app command. A bunch of packages gets installed along the way, which are needed to set up a convenient development environment --- including a web server, compiler and testing tools.

2. Nesting Components

Nesting of components is about rendering a child component in a parent component, this is important in React data flow.
The nested component <NestedComponent/> is imported and nested in the ParentComponent. So, the ParentComponent contains its elements, content and the NestedComponent.
```
import React, {Component} from 'react'
import NestedComponent from './NestedComponent'
class ParentComponent extends Component{
  render(){
    return(
      <div>
        <h1>These are the list of students in my class:</h1>
        <NestedComponent/>
      </div>
    )
  }
}
export default ParentComponent
```

3. Props

We have discussed about props in module 1. We will see how to use it in an app.
Let's update our ParentComponent as thus: pass in two props studentClass and studentSchool to the NestedComponent. These props will now be available in the NestedComponent.

```
class ParentComponent extends Component{
  render(){
    return(
      <div>
        <h1>These are the list of students in my class:</h1>
        <NestedComponent studentClass="Year 2" studentSchool="OAU"/>
      </div>
    )
  }
}
```

So, we can use it in our NestedComponent as thus:

```
import React, {Component} from 'react'
class NestedComponent extends Component{
  render(){
    return(
      <ul>
        <li>Oyekunle Opeyemi {this.props.studentClass}
{this.props.studentSchool}</li>
        <li>Adeferanmi Tolulope {this.props.studentClass}
{this.props.studentSchool}</li>
        <li>Moyin Akinbolwale {this.props.studentClass}
{this.props.studentSchool}</li>
      </ul>
    )
  }
}
export default NestedComponent
```

4. Outputting List

Here, we will dynamically output a list or a row of content instead of hard-coding the data into HTML elements.
Let's update our ParentComponent: Here, we will instantiate a state object, right before the render funtion, with a students array of object comprising name and score.

```
state = {
   students:[
     {'name':'Oyekunle Opeyemi', 'score': 92},
     {'name':'Moyin Akinbolwale', 'score': 52},
     {'name':'Adeferanmi Tolulope', 'score': 32},
  ]
}
```

Let's update create a Students class

```
import React, {Component} from 'react'
class Students extends Component{
  render(){
    return(
      <table>
        <tbody>
        </tbody>
      </table>
    )
  }
}


export default Students
```
Final update on ParentComponent:

Import the Students component and pass props to it. The div element should be returned.
```
import Students from './Students'
      <div>
        <h1>These are the list of students in my class:</h1>
        <NestedComponent studentClass="Year 2" studentSchool="OAU"/>
        <Students students={this.state.students} />
      </div>
```
Final update on Students component

Right inside the render function, iterate through the students array using map function and then return the result.
```
    const students = this.props.students.map((student, key)=>{
      return(
        <tr key={key}>
          <td>Name: {student.name}</td>
          <td>Score: {student.score}</td>
        </tr>
      )
    })
    return(
      <table>
        <tbody>
          {students}
        </tbody>
      </table>
    )
export default Students
```

5. Stateless Components

Stateless components (also known as dumb components) use props to store data, while stateful components (also known as smart components) use state. We don't need to import Component from react to create a stateless component.

Let's update the state in our Parent Component by adding a teacher property that has array of objects as its value.
```
state = {
   students:[
     {'name':'Oyekunle Opeyemi', 'score': 92},
     {'name':'Moyin Akinbolwale', 'score': 52},
     {'name':'Adeferanmi Tolulope', 'score': 32},
   ],
   teachers:[
     {'name':'Oyekunle Opeyemi', 'age': 52},
     {'name':'Moyin Akinbolwale', 'age': 78},
     {'name':'Adeferanmi Tolulope', 'age': 43},
   ]
 }
```

Next, we will create a Teachers component

import React from 'react'

```
const Teachers = ()=>{
    return(
      <div>
        <h3>Teachers</h3>
        <table>
          <tbody>

          </tbody>
        </table>
      </div>

    )
}
export default Teachers
```

Note that in the Teachers component, we did not import react Component, because it is not needed in a stateless component. Also, the render() function is not also available in a stateless component.

We need some data in the Teacher Component, so, we need to pass in props into the Teachers Component from the Parent Component like this:

1. Import Teachers component

import Teachers from './Teachers'

2. Pass props to the Teachers Component

<Teachers teachers = {this.state.teachers} />

Finally, we will update the Teachers component

```
const Teachers = (props)=>{
  const teachers = props.teachers.map((teacher, key)=>{
    return(
      <tr key={key}>
        <td>Name: {teacher.name}</td>
        <td>Age: {teacher.age}</td>
      </tr>
    )
  })
    return(
      <div>
        <h3>Teachers</h3>
        <table>
```

```
          <tbody>
            {teachers}
          </tbody>
        </table>
      </div>

    )
}
```

We cannot access the props the same way we did in Class Component(using this.props). To access the props in stateless component, we will pass it in as argument, so as to make it available in the Teachers component.

6. Conditional Output

All the JavaScript conditional statements are valid in React. Let's explore this simple example:

Here, we will create a Studentsgrades class component to output only the students' data that scores above 50 while FAILED should appear for those that got below 50.

```
import React, {Component} from 'react'
class StudentsGrades extends Component{
  render(){
    return(
      <div>
      <h1>Grades</h1>
      <table>
        <tbody>

        </tbody>
      </table>
      </div>
    )
  }
}


export default StudentsGrades
```

Next, update the ParentComponent

1. Import StudentsGrades

import StudentsGrades from './StudentsGrades'

2. Pass props to the StudentsGrades

```
<StudentsGrades students={this.state.students} />
```

Final update on StudentsGrades component

Put this in the render() function

```
const passedStudents = this.props.students.map((student, key)=>{
      return student.score > 50 ? (
        <tr key={key}>
          <td>Name: {student.name}</td>
          <td>Score: {student.score}</td>
        </tr>
      ) : (
        <tr key={key}>
          <td>Name: {student.name}</td>
          <td>FAILED</td>
        </tr>
      )
    })
```

And this in the tbody element

```
{passedStudents}
```

7. Forms and Functions as Props

This is just a revision/upgrade to what we did in our first meeting. Let's create an AddStudents component and initiate the state like this:

```
import React, {Component} from 'react';
class AddStudents extends Component{
      constructor(){
        super();
        this.state = {score:null, name:null}
        this.handleChange = this.handleChange.bind(this)
        this.handleSubmit = this.handleSubmit.bind(this)
      }
      handleChange = (e)=>{
        this.setState({
            [e.target.id]:e.target.value
        })
      }
      handleSubmit = (e)=>{
        e.preventDefault();
      }
      render(){
        return(
          <form onSubmit={this.handleSubmit}>
            <div>
              <label htmlFor="name">Name</label>
              <input type="text" id="name" onChange={this.handleChange} />
              <label htmlFor="score">Score</label>
              <input type="text" id="score" onChange={this.handleChange} />
              <button type="submit">Submit</button>
            </div>
          </form>
        )
      }
    }


export default AddStudents
```

This is almost 100% the same with what we did previously.

Let's update the Students component by adding an addStudent function that would add new data to the state of the Students component.

```
state = {
    students:this.props.students
```

```
  }
addStudents = (student)=>{
    let students = [...this.state.students, student]
    this.setState({
      students:students
    })
  }
```

So as to pass the function as props, we need to import the AddStudents component.

```
import AddStudents from './AddStudents'
<AddStudents addStudents={this.addStudents} />
```

Back to the AddStudents component, add this.props.addStudents(this.state) to the handleSubmit() function.