# FWP
# Semester 2, 2023

## Week 01

Introduction to React

How React works?

Old versus New React

React components

RMIT
UNIVERSITY

# Segment 1: Administrivia

Lectorial

Lab

Assignments

Expectations

Introduction

# Staff

- Official course coordinator
  - Shekhar Kalra (shekhar.kalra@rmit.edu.au)

- Offering course Coordinator and Lecturer:
  - Matt Hayward
    - email: matt.hayward@rmit.edu.au
    - consultation time: *email to set an appointment*

- For a detailed list of other staff members teaching in this course, please look at
  - Course Canvas – Modules- Welcome - Teaching Team *page*

# Why was this course created?

- Feedback from industry who hired RMIT graduates
- Feedback from RMIT graduates
- A gap in the student skill- *full stack development*
- Teach React or Angular so that graduates can pick up other frameworks as well (such as Vue.js, etc.)
- There was always a demand from students to bring back the course
- React is a sought-after skill

# General tone of this course

- **Pure programming course- no theoretical assessments**
- **Medium to *generally* intense**
- **Easy to semi-easy to get pass**
- **Semi-easy to moderate effort to get credit**
- **Need to work hard, research and find answers to get distinction and high distinction**
- **Two assignments worth 70% - related to each other**
- **Work in a group of 2**
- **Use of Github for assignments**
- **Final assessment (time allowed: 2 hours) at the end of semester worth 30%**

# Important advice about assignments 1 and 2

- Assignment 1:
  - based on week 1 till 4/5
  - worth 25%
- Assignment 2: *based on Assignment 1*
  - based on week(s) 5 till 11
  - worth 45%
- DI and HD parts of assignments will require you to research and find out answers
- We will help but you will drive the work
- This has been done on purpose to make you work-ready and prepare for industry

# Pre-reading→ Lectorial→ Lab

- **Pre-reading: always read/watch the recommended pre-reading material before attending the lectorial**
- **Lectorial: will cover the concepts with code examples along with interactive discussion**
- **Lab/PRAC: will cover application of concepts covered during lectorial**
- **Lab/Prac start week 2!**

# How do I get a good grade for this course?

- Attend lectorial and lab
- Practise code each week
- Start working on assessments on time
  - Last minute work can lead to undesirable outcomes
- Try to understand what you code
  - Copying code from various sources without understanding can lead to disastrous outcomes
- Ask questions
  - Keeping quiet and leaving things till the end can lead to unwanted scenarios

# What is this course about?

| ✓ | ✘ |
|---|---|
| React framework | Web design |
| Coding every aspect of the web app | Using drag and drop tools |
| Learning a web framework | Not attending any class or read course material |
| Acquiring a good knowledge of a web framework | Use of frameworks that are not covered in this course |
| Learning by doing things yourself | Reading concepts only and not coding |
| Asking questions and attending consults | Keeping quiet and not participating |

# Few reasons for failing the course among previous cohorts

- *I did not attend any session, I assumed I knew enough to pass*
- *I had coded in React in one of my earlier projects, so I thought how hard could it be*
- *I assumed that I could complete the assessment in this way*
- *Writing code in a framework other than React*
- *I was not sure if I had to follow the assignment specifications exactly*
- *I found React hard and I just gave up*

# Is there a prescribed textbook?

- Prescribed yes but not mandatory recommendation
- No one book has all the topics
- It is not possible to buy multiple textbooks
- However there are some prescribed reference textbooks and tutorials used for this course.
- You will be given handy references of free online references as supplementary material

# Before we start

- **basic html tutorial [https://www.w3schools.com/html/]:**
  - **read sections *HTML Home to Html Lists***
- **basic CSS tutorial [https://www.w3schools.com/css/default.asp]:**
  - **read sections *CSS home to CSS Backgrounds, CSS text to CSS Lists***
- **basic JS tutorial [https://www.w3schools.com/js/DEFAULT.asp]**
- **completed P1/PT (undergraduates); AP (postgraduates)**
- **have a basic knowledge of SQL**

# Segment 2

**Introduction- *why?***

**React- *what?***

**React- *how?***

**The world of components**

# Why a web framework?

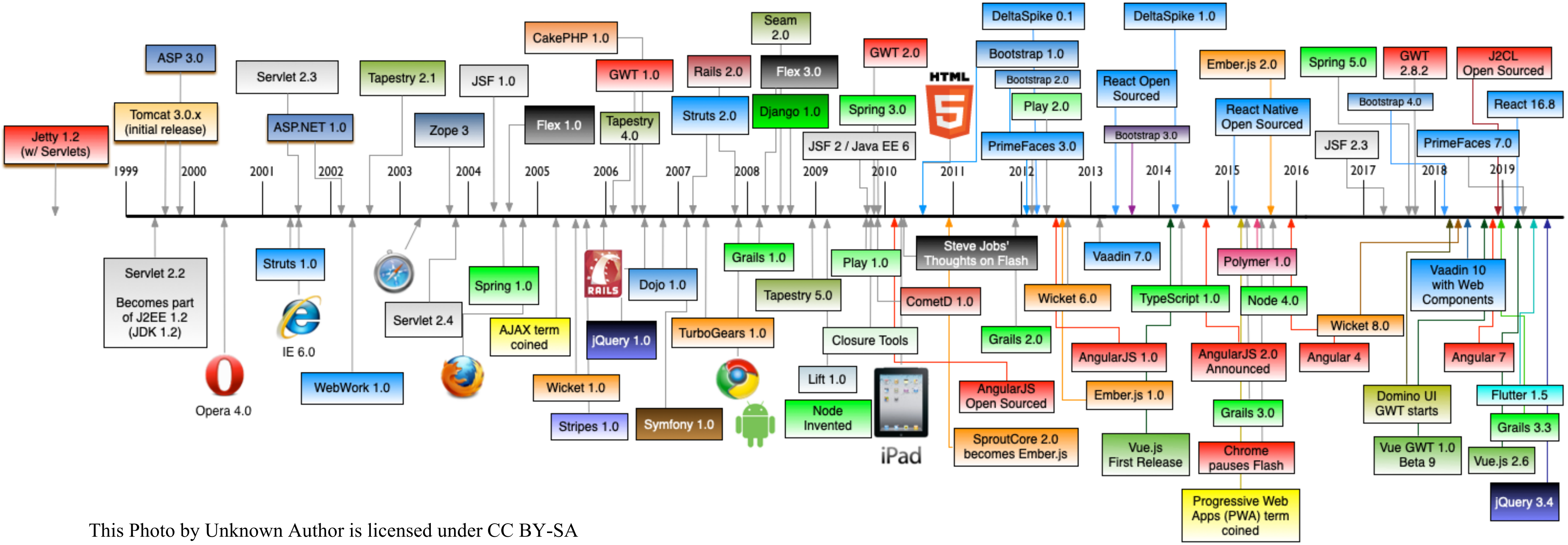
- Surely, one can design a website with just *HTML5, CSS* and *JavaScript*
- Then why learn something new? Besides, *a framework sounds complicated!*
  - Real life/ industry web application projects are complex
  - There is a need for automating repetitive tasks
  - Web is growing and so are the needs, technologies
  - Multiple tiers/stacks are involved

# Why a web framework?

- Frameworks simplify the previously mentioned scenarios
- Abstracts unnecessary details from the user
- It provides templates for the user
- It handles all the logistical aspects of the application development
- In essence, it leads to
  - Accelerated development &
  - Efficient debugging

# Many web frameworks

- ASP.NET, JEE (Java Enterprise Ed.)
- React, Angular, Vue, Django, Ruby on Rail, Laravel, etc.
- Flask, Express.js, jQuery, etc.



Jetty 1.2 (w/ Servlets)
ASP 3.0
Tomcat 3.0.x (initial release)
Servlet 2.3
ASP.NET 1.0
Tapestry 2.1
JSF 1.0
Zope 3
CakePHP 1.0
Flex 1.0
GWT 1.0
Tapestry 4.0
Rails 2.0
Struts 2.0
Seam 2.0
Flex 3.0
Django 1.0
GWT 2.0
Spring 3.0
JSF 2 / Java EE 6
HTML 5
DeltaSpike 0.1
Bootstrap 1.0
Bootstrap 2.0
Play 2.0
PrimeFaces 3.0
DeltaSpike 1.0
React Open Sourced
Bootstrap 3.0
Ember.js 2.0
React Native Open Sourced
Spring 5.0
GWT 2.8.2
Bootstrap 4.0
JSF 2.3
J2CL Open Sourced
React 16.8
PrimeFaces 7.0

1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019

Servlet 2.2 Becomes part of J2EE 1.2 (JDK 1.2)
Opera 4.0
Struts 1.0
IE 6.0
WebWork 1.0
Servlet 2.4
Spring 1.0
AJAX term coined
Wicket 1.0
Stripes 1.0
RAILS
jQuery 1.0
Dojo 1.0
TurboGears 1.0
Symfony 1.0
Grails 1.0
Tapestry 5.0
Play 1.0
Closure Tools
Lift 1.0
Node Invented
iPad
Steve Jobs' Thoughts on Flash
CometD 1.0
AngularJS Open Sourced
SproutCore 2.0 becomes Ember.js
Grails 2.0
Wicket 6.0
Vaadin 7.0
AngularJS 1.0
Ember.js 1.0
Vue.js First Release
TypeScript 1.0
Polymer 1.0
Node 4.0
AngularJS 2.0 Announced
Grails 3.0
Chrome pauses Flash
Progressive Web Apps (PWA) term coined
Angular 4
Wicket 8.0
Vaadin 10 with Web Components
Domino UI GWT starts
Vue GWT 1.0 Beta 9
Angular 7
Flutter 1.5
Grails 3.3
Vue.js 2.6
jQuery 3.4

# Skills for the CSIT industry

- Technical
  - Technology, frameworks, languages, *etc.*
- Graduate attributes
  - EQ, group dynamics, humble disposition, collaboration skills, initiative, creativity, innovation etc.
- *Complete turnoff(s):*
  - *Arrogance / Know-it-all*
  - *Reluctance to learn anything new*
  - *Not open to working alongside others*
  - *Negative attitude*

# Why React? Why not any other?

- **It is best to learn few frameworks in an efficient manner**
- **This way you can pick up on others- *this is how industry works!***
- **React /Angular / Vue.js are good frameworks to learn**
- **We will focus on React framework in this course**
- **You are expected to write all assessments in React and not in any framework of your choice**
- **You will get ZERO if you went the non-React way!**

# What is *Full stack development*?

- The development of both front end(*client side*) and back end(*server side*) portions of a web application.
- A true full stack developer is proficient in both frontend and backend languages and frameworks, as well as in server, network and setting, hosting environments.
- Technology stacks:
  1. HTML, CSS, Bootstrap, React JS *(front)*
  2. Node.js & Express.js *(middle)*
  3. Database: MySQL *(back)*

# React- introduction

- **Created and maintained by Facebook**
- **Popular in industry**
- **Super flexible**
- **Helps creating SPA(*Single page*) applications**
- **You can also learn React *native* in your free time (building cross platform mobile apps)**
- **JavaScript (JS) or TypeScript (TS) based- these are not hard to learn**
- **Learning React will help you learn/pickup other frameworks**
- **..**

# React with JS in this course

- We will learn how to create React application with JS
- We will also look at TS version from time to time
- Many reasons for doing this
  - Easier for those who have never learned any web framework
  - TS is a superset of JS
  - Many legacy React apps still in JS
  - Appreciate TS more after focussing on JS
  - React with TS more suited for experienced developers- ***you are not expected to use this***

# React – why?

- Single-page applications (SPA) have become increasingly popular with first-generation SPA frameworks like Angular (by Google), etc.
- Although you can create these with vanilla HTML, CSS, JS, but the process becomes cumbersome
- React helps you create SPAs?
- Why SPAs?
  - In the past, websites and web applications were rendered from the server.
  - Modern frameworks shifted the focus from the server to the client
  - Examples: Gmail, Google Maps, Airbnb, Netflix, PayPal, etc.

# Lectorial Exercise

- Why use SPA as compared to traditional websites?
- Provide few examples of popular SPA websites.
- What are some disadvantages of SPA websites?

# React- setup

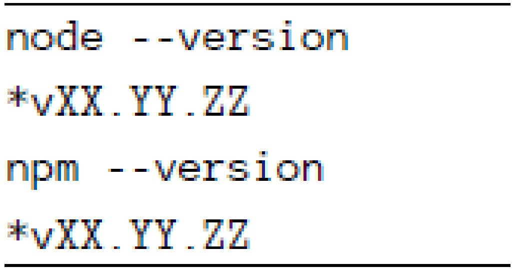
- You will need a laptop/machine to complete this elective course- RMIT labs/myDesktop does not have a React setup (*why?)*
- Download Node.js- latest version – *free to install*
  - [https://nodejs.org/en/download/]
  - If already installed on your machine, update it to the latest version.
- Download Visual Studio code- IDE used for this course – *free to install*
  - [https://code.visualstudio.com/download]

# Lectorial Exercise

- What is Node.js?
- What is it used for?
- Why use it for React?

# Node & NPM

- Installation of Node.js will also setup NPM (Node package Manager)
- You will need both Node and NPM to manage libraries (node packages) that you will need along the way.
- Post installation, check that you have the latest version by running these commands- command prompt (windows), Terminal (Mac, Linux):

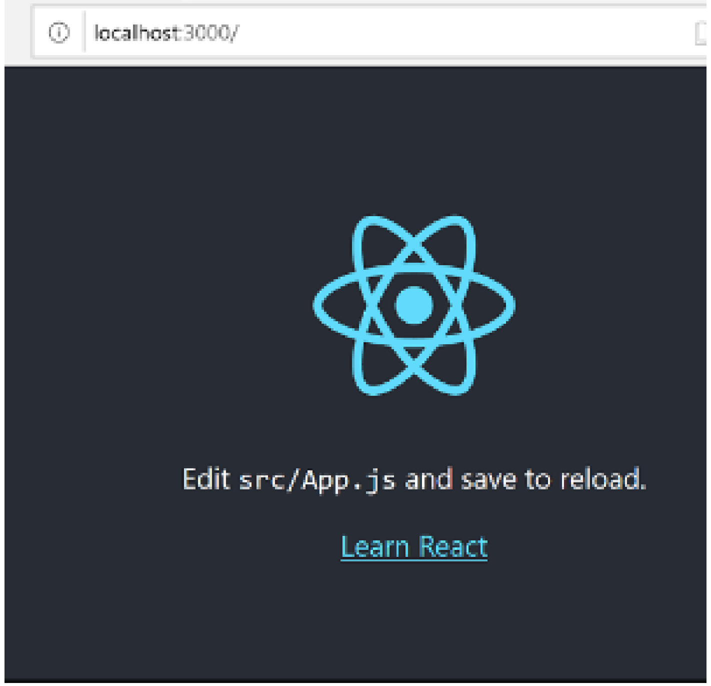


```
node --version
*vXX.YY.ZZ
npm --version
*vXX.YY.ZZ
```

# Setting up a React project

- **Run this command on command prompt/terminal:**
  - `npx create-react-app my-app`
  - **This will take sometime depending upon various factors- so do not give up!**
- **Once created**
  - `cd my-app`
  - `npm start`

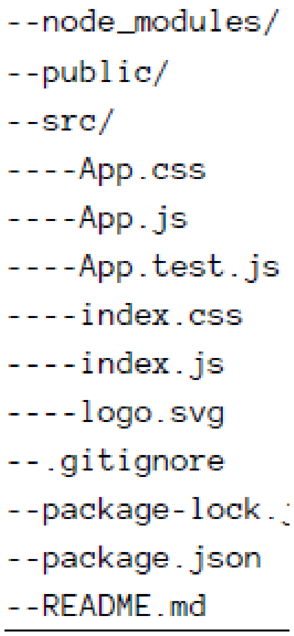
**Example01**



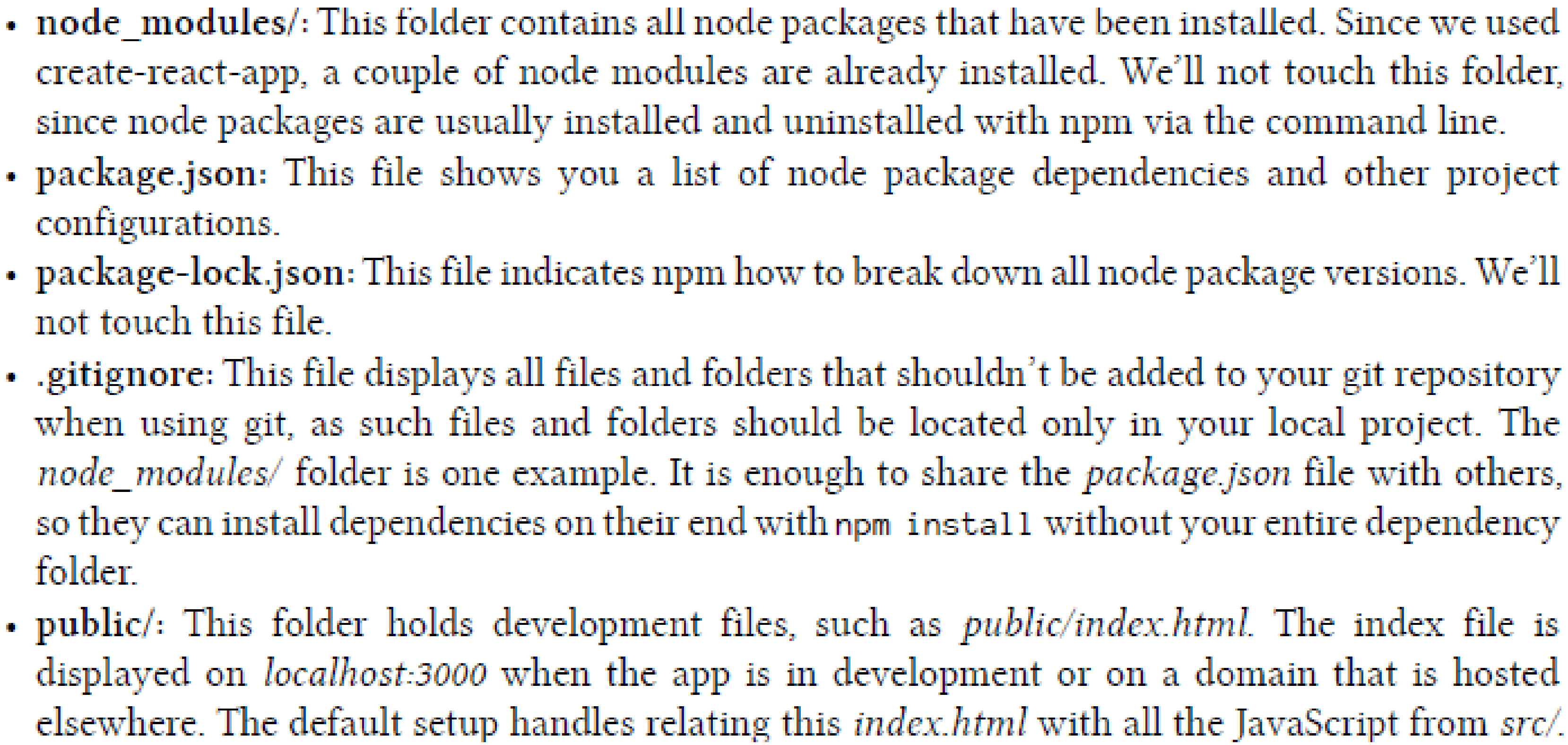
localhost:3000/

Edit `src/App.js` and save to reload.

# Example 1- *project structure*



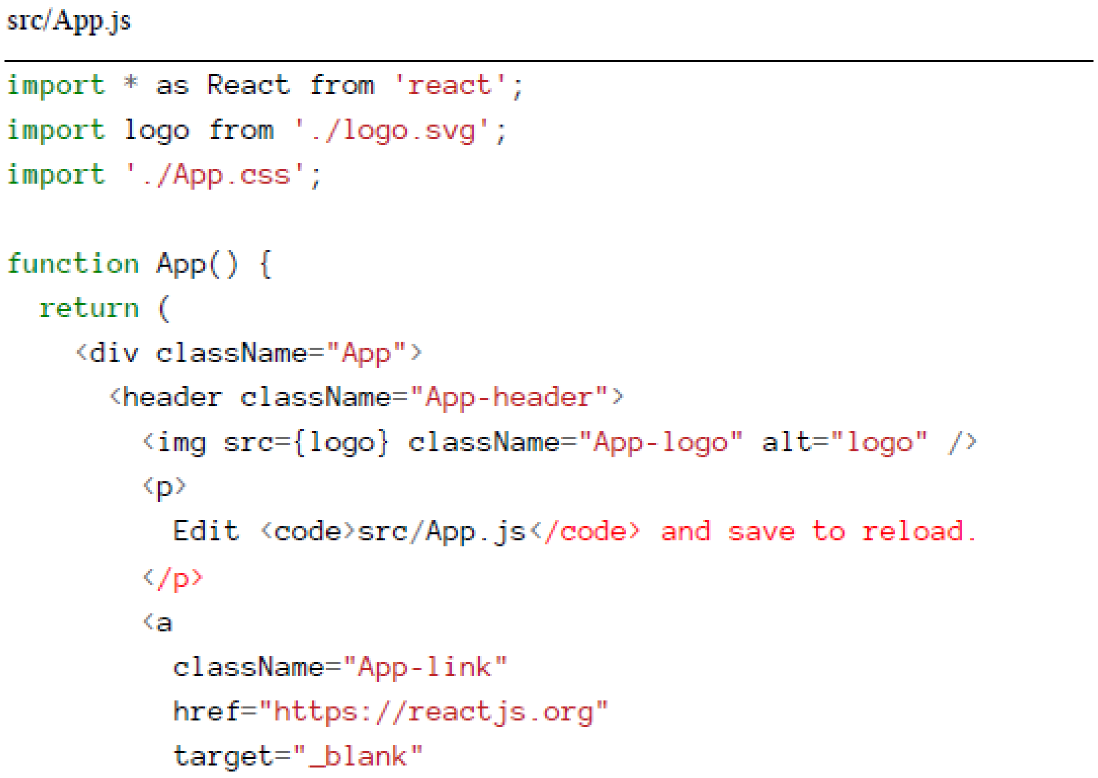
```
--node_modules/
--public/
--src/
----App.css
----App.js
----App.test.js
----index.css
----index.js
----logo.svg
--.gitignore
--package-lock.j
--package.json
--README.md
```



- **node_modules/**: This folder contains all node packages that have been installed. Since we used create-react-app, a couple of node modules are already installed. We'll not touch this folder, since node packages are usually installed and uninstalled with npm via the command line.
- **package.json**: This file shows you a list of node package dependencies and other project configurations.
- **package-lock.json**: This file indicates npm how to break down all node package versions. We'll not touch this file.
- **.gitignore**: This file displays all files and folders that shouldn't be added to your git repository when using git, as such files and folders should be located only in your local project. The *node_modules/* folder is one example. It is enough to share the *package.json* file with others, so they can install dependencies on their end with `npm install` without your entire dependency folder.
- **public/**: This folder holds development files, such as *public/index.html*. The index file is displayed on *localhost:3000* when the app is in development or on a domain that is hosted elsewhere. The default setup handles relating this *index.html* with all the JavaScript from *src/*.

# Component *eh!*

- **Every React application is built on the foundation of React components.**
- **Your first React component which is located in the src/App.js file**



src/App.js

```
import * as React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
```

# Default template of a component

- First, this React component, called the App component, is just a JavaScript function.
- Second, the App component doesn't receive any parameters in its function signature yet.
- And third, the App component returns code that resembles HTML-
  - this new syntax is called JSX, allows you to combine JavaScript and HTML for displaying highly dynamic and interactive content in a browser.

# React JSX

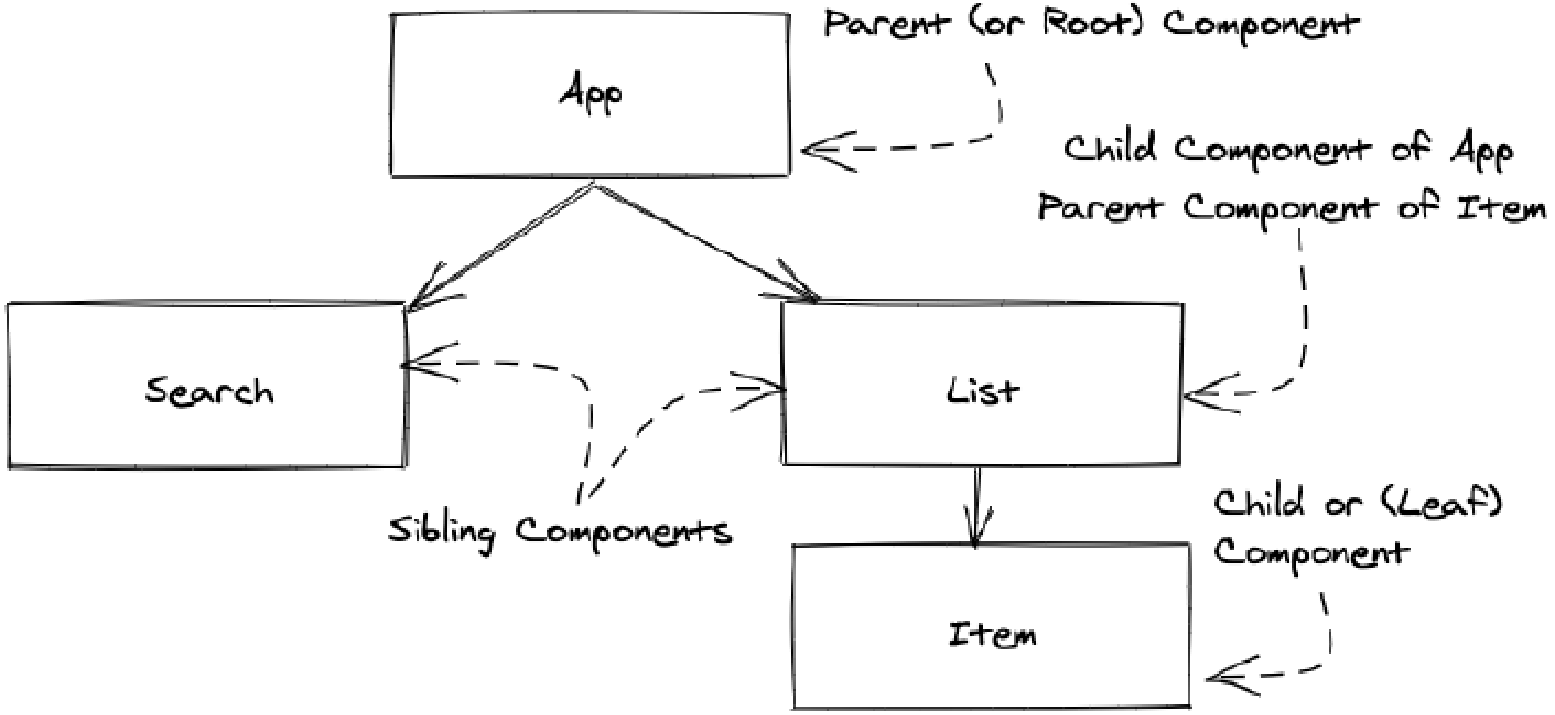
- Until now, we only returned HTML from the App component.
- However, it can also be mixed with JavaScript (JS).
- JSX helps you to combine HTML and JS
- Without any extra templating syntax you will be able to use JavaScript in HTML.
- Every data structure, from JavaScript primitive to a JavaScript function, can be displayed within HTML with the help of JSX
- Let us now make our component a bit more smarter
- Example 2

# Component or Components

- So far we’ve only been using the App component.
- This will never be the case in case of a website which is more complex than a single component
- So instead of making one component larger and more complex, we’ll split one component into multiple components eventually

# Component or Components

- We will now look at more involved example where we will develop multiple components.

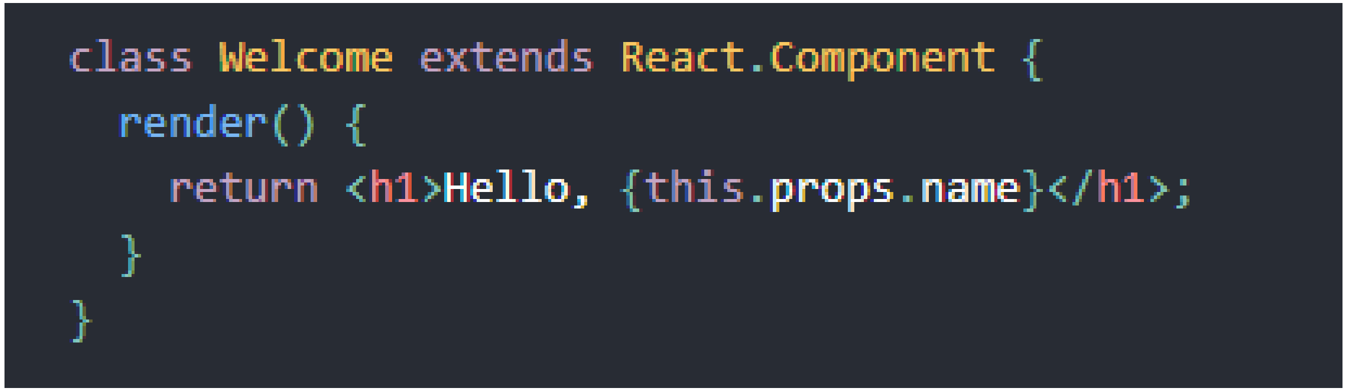


# Component or Components

- React applications have component hierarchies (also called component trees).
- There is usually one uppermost entry point component (e.g. App) that spans a tree of components below it.
- The App is the parent component of the List and Search, so the List and Search are child components of the App component and sibling components to each other.
- Example 3

# Component as a class- *OUTDATED!!*

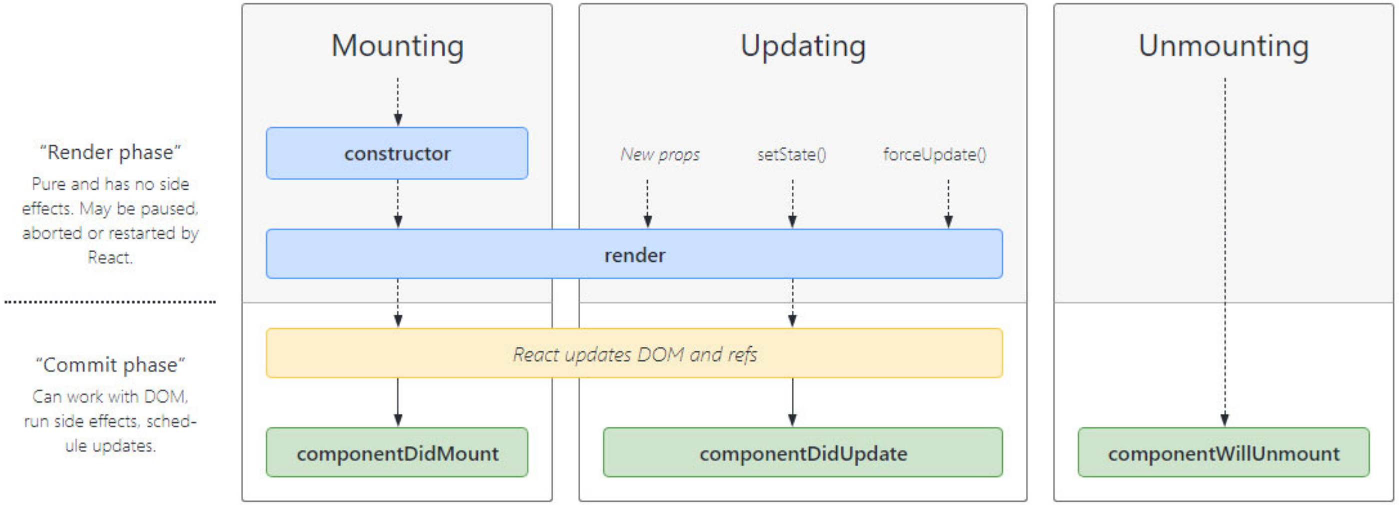
- **React is no more written like that- it is an old way of writing code- <u>WE WILL NOT FOLLOW THIS APPROACH</u>**
- **Developers used to write React component class, by extending from in-built React.Component**
- **The only method you must define in a React.Component subclass is called render()**



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

# Component as a class

- Each component has several “lifecycle methods” that you can override to run code at particular times in the process.
- A very messy way of writing component that made things harder
- Look at the diagram-



“Render phase”
Pure and has no side effects. May be paused, aborted or restarted by React.

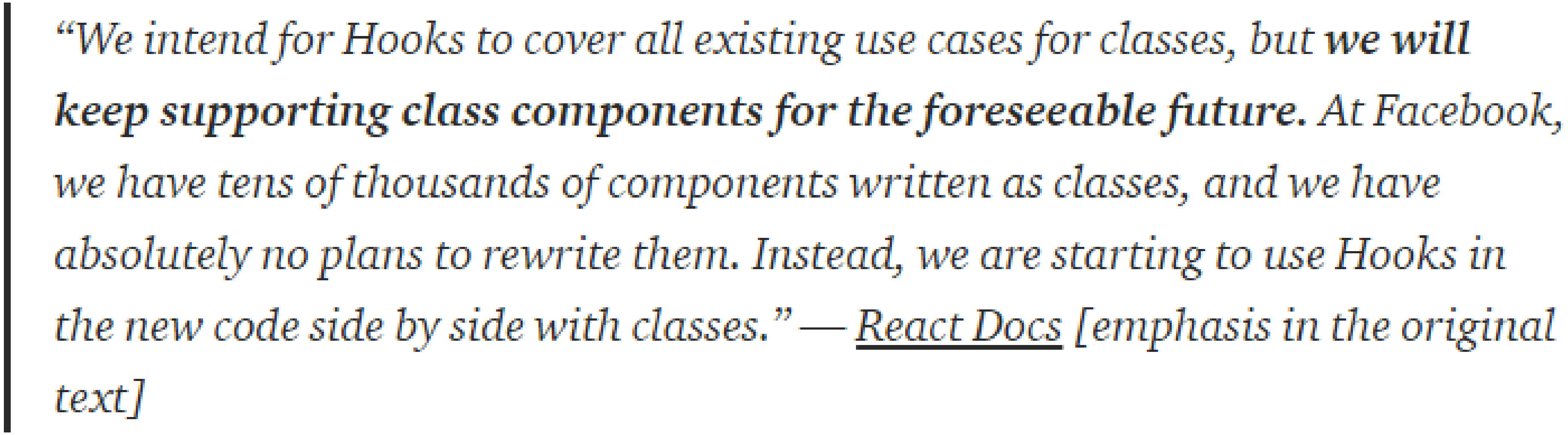
“Commit phase”
Can work with DOM, run side effects, schedule updates.

# Can I use class components for this course?

- **In this course we will only learn how to write new React i.e. *functional* components**
- **You are NOT allowed OR advised to write class components**
  - **You will get a zero in assessments for using class components**
- **Only legacy projects use class components**
- **So remember**
  - **Class components → ZERO**

# Component- functional

- ***Future of React is functional components!!***
  - ***The goal is to reduce boiler plate..***
- **Facebook has made one thing very clear:**



> *"We intend for Hooks to cover all existing use cases for classes, but **we will keep supporting class components for the foreseeable future.** At Facebook, we have tens of thousands of components written as classes, and we have absolutely no plans to rewrite them. Instead, we are starting to use Hooks in the new code side by side with classes." — React Docs [emphasis in the original text]*

# Lectorial Exercise

- What is DOM?
- Where have you used it?

# Handling events

- Event handling makes it possible for your users to interact with the React app
- As per React documentation- handling events with React elements is similar to handling events on DOM elements except
  - React events are named using camelCase, rather than lowercase.
  - With JSX you pass a function as the event handler, rather than a string
- Assuming you're familiar with how events work in standard HTML and JavaScript, it should be easy for you to learn how to handle events in React.

# Handling events: functional component

- **This is not very different to a class component case**
- **Example04**

# Hooks

## State in functional components

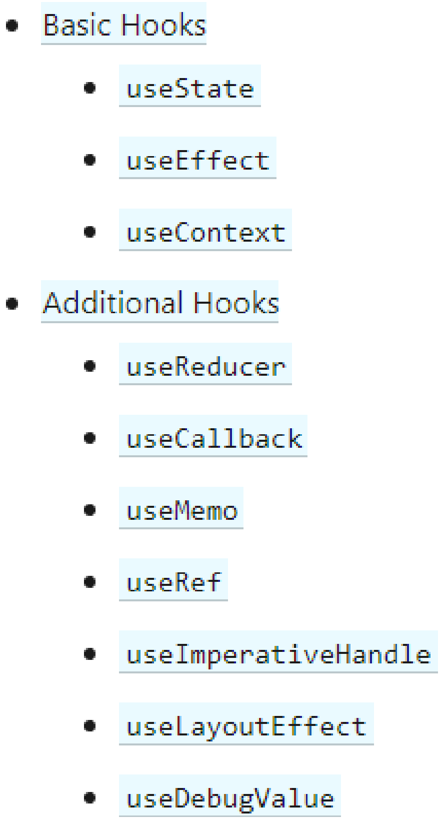
This Photo by Unknown Author is licensed under CC BY-SA-NC

# Lectorial Exercise

- **What is a *state* in a web application?**
- **Why is it needed?**

# State in functional components: *hooks*

- Hooks are a new addition since React 16.8.
- They let you use state and other React features without writing a class.
- They simplify a lot of code writing scenarios in React
- In-built hooks in React
  - https://reactjs.org/docs/hooks-reference.html



- Basic Hooks
  - useState
  - useEffect
  - useContext
- Additional Hooks
  - useReducer
  - useCallback
  - useMemo
  - useRef
  - useImperativeHandle
  - useLayoutEffect
  - useDebugValue

# useState()

- It returns a stateful value

```
const [state, setState] = useState(initialState);
```

- The setState function is used to update the state

```
setState(newState);
```

- Time to look at an example
- Example05

- We will keep revisiting hooks in subsequent weeks- there is a lot to cover

# useState() *nuts & bolts*

1. **Call useState() hook to enable state in a functional component.**

2. **The first argument of the useState(initialValue) is the state's initial value.**

3. **[state, setState] = useState(initialValue) returns an array of 2 items: the state value and a state updater function.**

# useState() *nuts & bolts*

4. **Invoking the state updater function setState(newState) with the new value updates the state.**
   - **Alternatively, you can invoke the state updater with a callback setState(prev => next), which returns the new state based on previous.**

5. **After the state updater is called, React makes sure to re-render the component so that the new state becomes actual.**

# So what is a *hook*?

- **What is a Hook?**
  - A *Hook* is a special function that lets you “hook into” React features. For example, *useState* is a Hook that lets you add React state to function components.
- **When would I use a Hook?**
  - If you write a function component and realize you need to add some state to it, previously you had to convert it to a class. Now you can use a Hook inside the existing function component.

**Reference:**

**https://reactjs.org/docs/hooks-reference.html**

# References

- **Reference:** ***The road to react (2021 edition), by Robin Weiruch; Leanpub***
- **The above will be the prescribed reference textbook for the first few week(s) for this course.**

# Next week

- More on React components
- Components interacting - part 1
- incorporating data
- Styling in React &
- Assignment 1 discussion