Day 1:

* **MERN/MEAN** – Mongo, Express, React, Node/ Mongo, Express, Angular, Node

**NODE:**

* Nodejs.org
* CLI
  + Node –v

**Types of web app development:**

**Server side client side(modern development)**

* UX: reactJS is used for UX design
* Serverside: use NodeJS

Single page application – html is loaded only once

**NPM tool:**

**~ to maven of java, build tool**

* Node package manager (NPM)
* Command line utility
* Common commands:
  + install
  + uninstall
  + –S
  + –D
  + init
  + list
  + start
  + run
  + build
* every project must have a configuration file -> package.json
  + use **npm init** (will ask questions)
  + or use **npm init** – y (it will not ask q’s and set default as yes)
* <https://getbootstrap.com/>
  + Get bootstrap using npm
  + **npm install bootstrap –S**
    - -S will add the entry to project’s package.json
    - If you want a specific version of bootstrap:
      * Remove existing : **npm uninstall bootstrap**
      * **npm i** [**bootstrap@3.3.7**](mailto:bootstrap@3.3.7) **–S**
  + dependencies can be run time or dev time
  + auto refresh page w your changes – use lite-server
    - need this for dev time: **npm I lite-server -D**
    - D:\ReactTraining\day1project\node\_modules\.bin
    - Use lite-server to tsrt server
  + Npm can be a task runner also
    - Eg start server
    - Run test case etc
    - Two types of tasks:
      * Built in tasks
      * Custom tasks
        + Use **run** to execute custom tasks; eg: npm run lite
        + In package.json
* "scripts": {
* "test": "echo Testing NPM Task Runner",
* "lite": "lite-server"
* *JS is managed by ECMAScript(in short ES) who maintains the standards of javascript*
* *ES5 is reg javascript*
* *Latest version available is ES6*
  + *In react application we will use ES6 and notES5*
  + *Features:*
    - *Block scope variables, use ‘****let’*** *instead of ‘****var’***
    - ***Const***
    - ***Anonymous function***
    - ***Arrow function:***
* Arrow function: ignore 'function' & 'return' keyword instead of return you can use => arrow
  + - ***Object oriented features ~classes***
      * ***Inheritance***
* class Point{
* constructor(x,y){
* this.x = x;
* this.y = y;
* }
* display(){
* console.log("X:", this.x, " Y:",this.y)
* }
* }
* //Instance of point class
* let pt1 = new Point(3,5);
* pt1.display();
* class Point3D extends Point{
* constructor(x,y,z){
* super(x,y)
* this.z = z;
* }
* display(){
* super.display();
* console.log( " Z:",this.z)
* }
* }
* let pt3d = new Point3D(3,2,5);
* pt3d.display();
  + - ***Modules:***
      * *Each file is a module*
      * ***Import*** *or* ***export***

**npm start:** Starts the development server.

**npm run build:** Bundles the app into static files for production.

**npm test:** Starts the test runner.

**npm run eject:** Removes this tool and copies build dependencies, configuration files

and scripts into the app directory. If you do this, you can’t go back!

**HTML tags:**

Can be:

* Built-in
* custom

**REACT:**

* <https://reactjs.org/>
* Used to build user interfaces
* Library to develop component based UI development library and Dom manipulation
* Custom tags are called components which are used in web pages and has UI+Logic
* Components based dev on elements, elements are smallest building block of react apps and defines what we see in screen
* React & react DOM
* React = web component + Dom Manipulation
* Uses **Virtual Dom** instead of actual DOM, thus maintaining application performance

Project Setup:

* Manual setup
* Or readymade setup - CLI (generates boiler plate) *<<we will use this>>*
  + Create react app (**npm i –g create-react-app**) //-g: global mode
  + create-react-app—version

**JSX:** *(java script extension)*

* templating lang ~HTML
* displayed for user

Create component:

* through functions
* through classes

using styles for applications –you can use import

**Child components:**

*Component communication b/w parents and child and vice versa*

DAY 2:

* Component Communication
  + Props & state in react
* Single Page application dev –react router library
* Component lifecycle
* State management w Redux

Props are properties inside a react component which will help pass data from parent component to child component.

* Props are immutable (read only)
* For changing data use **state**
  + Is a js object which helps maintain a mutable data in a component
  + Data members in a class is called state
  + Should be in object notation, KV pairs
  + Use a special attribute “**refs**”, to get value ~ id
  + Parent to child communication: **props**
  + Child to parent comm: **props & state**
  + Communication b/w one component to another: **attributes**

**Single page application:**

React-router-dom (browser based)

React-router-native(for mobile app)

Switch

**Component Life cycle methods**

* Creation of component *(create instance of class)*
* Before rendering
* While rendering
* After render
* Destroy

React handles the life cycle though you can customise

**Mounting**: insert a component into DOM

**Unmounting**: remove a component from DOM

**Methods are:**

*componentWillMount()*

*render()*

*componentDidMount()*

*componentWillUnmount()*

**problem w react:** state management if not possible between sibling elements as communication is between parents to kids and vice versa

**Soln:** *maintain a global state*

* React doesn’t have any global state management process, it only manages local
* To solve this we can use a JS framework – REDUX
  + It’s not for react only but a common framework for any JS

**REDUX**

* State management library
* Only applicable for JS application(like react, js, angular etc
* State is:
  + List of blog posts
  + List of users etc
* Terminologies:
  + **Store**:
    - Central location where al application data is store.
    - Managed as a JS object
    - Should have at least one reducer
  + **Reducer**:
    - JS function
    - Pass current state & action and returns piece of state
    - Decides state for each action
  + **Action**:
    - JS object representing change which is to happen to state
  + **Dispatch**:
    - send action to the state for process
  + **Subscriber**:
    - Listener for change
    - Do something when some change is detected

To run only JS, we need JS run time environment like nodejs

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](https://developers.google.com/v8/). Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, [npm](https://www.npmjs.com/), is the largest ecosystem of open source libraries in the world.

**Run from cmd:**

D:\ReactTraining\reduxproject>node reduxDemo.js

Redux Sample

**NODEMON**: node monitoring process

D:\ReactTraining\reduxproject>node install –g nodemon

**Egghead.io - >go through for details on redux & react and node (VVIMP)**

**DAY 3:**

**CRUD** operations on data – **C**reate, **R**etrieve, **U**pdate, **D**elete

Install libraries redux and react-redux in react project for integrating react w redux

* Wrap component w Provider buy using react-redux library , class Provider
* Connect(function, dispatch\_actions)(Component)

Function: DI sud happen to inject state here

**Day 4:**

* Node JS
* Express JS
* Mongo DB
* Developing REST API