* **Virtual DOM**
* **ES6 – ECMAScript 6**
  + Classes
  + Arrow Function
    - Hello = () => { return “Hello There!”;}
    - Hello = () => return “Hello There!”;
  + Variables
  + Array Methods
    - .map() method allows you to run a fxn on each item in the array
    - Const myArray = [‘a’,’b,’,c’];
    - Const myList = myArray.map((i)=><p>{i}</p>)
  + Destucturing
    - Extract only what is needed
    - const vechicles = [“mustang”, “f-150”, “expendition”];
    - old: const car = vehicles[0];
    - const truck = vechicles[1];
    - const suv = vechicles[2];
    - new : const [ car, truck, suv ] = vechicles;
    - the order that variables are declared is important
    - truck cancel out
    - const [car,,suv]=vechicles;
    - destructuring useful when fxn return array : const [add,sub,mul,div] = cal(4,7);
    - obj – obj pass as para only send attributes
    - object no specific order
    - nested obj : {a,b:{ba,bb}}
  + Spread Operator
    - Copy all or part of array or object
    - Used in combination with destructuring
    - Ar=[1,2,3,4,5,6,7]
    - [one,two, …rest]=Ar
    - New property name is new,
    - But old property is overwritten
  + Modules
    - Import and export
    - In-line : export const name = “arjun”
    - At the end: Export {name, age}
    - Default export only one
    - Import time: named exports or default export
    - Named : destructured using { }
  + Ternary Operator
    - <Condition> ? <true> : <false>
* **Render HTML**
  + ReactDOM.render()
  + Two args: html code and Html element
  + Element not mandatory to root, but it Is stnd conv
* **JSX**
  + JavaScript XML
  + No createElement and appendChild
  + JSX: <h1>I love you</h1>
  + JSX: React.createElement(‘h1’, {}, ‘I love you’);
  + Expression : { 5 + 5} | JS exp
  + insert large block of html : html inside parentheses
  + html block in One Top Level Element <></>
  + single tag element must be closed
  + class -> className
  + if outside JSX or use Ternary
* **Components**
  + Reusable, =js fxn but work in isolation
  + Class | Function Components
  + Use the component : <Name />
  + Class Components
  + Function Component
  + Component inside component
* **Class -**  skip for now
* **Props**
  + Properties – arguments in js and attributes in html
  + To pass add attributes < Car brand=”ford”/>
  + Fxn’s (props) {props.brand}
  + Pass data from Component to Component
  + Brand=”name” for string
  + Brand={ carName} for variable / object
  + Props are read only. If change = error
* **Events**
  + onClick instead of onclick
  + onClick={shoot} , not onClick=”shoot()”
  + onClick={arrow fxn} – to pass
  + react event object: event.type
* **Conditionals**
  + && - true-condition && render
* **Lists**
  + List – Garage in car list
  + Cars – including car list
  + Cars – list
* **Forms**
  + HTML – DOM | React – component
  + Control changes by event handler
  + useState Hook to keep track of inputs
  + slight change in textarea and select, it handle a;; input elements in the same way
* **Router**
  + Add react router: npm i –D react-router-dom
  + Src/pages/Home.js|Blogs.js|Contact.js
  + <Router> wrap
  + <Link> set URL and track browsing history
  + <Link to=”/”> instead of <a href=””>
  + <Switch> similar to switch in JS, it render <Router that matches <Link>
* **Memo**
* **CSS Styling**
  + Inline Styling {{}} double curly | camelCase
  + CSS Stylesheets – making new .css
  + CSS Modules – only the imported component can use
* **Sass Styling**
  + CSS pre-prossor
  + Executed on server and send css to browser
  + Npm I sass : install sass
  + Like css with .scss – use var and othe scss fxn
* **Hook**
  + Since –v 16.8
  + Allow fxn comp to have access to state and other React features. Class comp no longer needed
  + Allow us to “Hook” into react features such as state and lifecycle methods
  + useState Hook to keep track of the application state. State refers to application data and properties
  + rule : only called inside React Fxn component
  + rule : only called at top level of a component
  + rule : cannot be conditional
  + not work in class component
  + if you have stateful logic that needs to be reused in several components, build custom Hooks
* **useState**
  + Named export so import {useState}
  + initialize by calling useState in our fxn component
  + accepts an initial state and return 2 values
  + the current state | a fxn that updates the state
  + const [color, setColor] = useState(“”)
  + color is current state | setColor is fxn that update our state
  + useState(“”) is initial state
  + now can use {color} anywhere in the fxn component
  + no direct update like color = “red” but setColor(“red”)
  + if object then call by obj.attributes, and update by setObject ( () => {attr1:””, attr2:””});
  + object : setCar({color: “blue”}) – removes other attributes
  + object all attributes change and only one attributes change
* **useEffect**
* **useContext**
* **useRef**
* **useReducer**
* **useCallback**
* **Custom Hooks**