ReactJS

Intro:

React är JavaScript bibliotek, för att installera den behöver vi installera **NodeJs** som innehåller **npm (Node Package Manager)** och **npx** (Node Package Execute).

Skapa React projekt:

Vi kan skapa React projekt genom att köra följande kommando i Terminalen:

npx create-react-app myreactappname

Utföra React App:

Cd myreactappname

npm start

React projekt filar:

Under katalogen public hittar vi:

index.html:

HTML-Fil som innehåller en dev med id ”root” där kommer våra komponenter (components) att synas.

Under katalogen src hittar vi många filar, de viktigaste filarna är:

index.js:

innan React 18 varden huvudregel i alla JS filar när vi jobbar med React att importera React bibliotek med följande rad:

```
import React from 'react';
```

och det viktigt att importera ReactDOM i alla JS filar som användercreateRoot() i sidan med följande rad:

```
import ReactDOM from 'react-dom/client';
```

Den filen som agerar med root element.

```
const root =
ReactDOM.createRoot(document.getElementById('
root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

renderar våra komponenter som finns i App.js fil som vi importerat genom den raden:

```
import App from './App';
```

Render-metod React17 : render() är en metod som tillhör ReactDOM och avgör ***vad*** och ***var*** ska visas till exempel :

ReactDOM.render(<h1>Khattab</h1>, Document.getElementById('root'))

App.js:

Innehåller en metod som skapar och returnerar våra HTML element:

```
function App() {
  return (
          <h1>Khattab</h1>
  );
}

export default App;
```

skapa Komponent:

För att skapa komponent behöver vi skapa JS fil och importera react som vanligt i början av filen sen definierar vi en metod (funktion) som har samma JS filen namn och returnerar HTML element men skriven med **JSX:** Return() ska innehålla bara ett element som i sin roll kan innehålla alla elementen, om jag inte behöver ett element för att hålla mina komponenter så kan man använda <>mina komponenter</>,och inline style ska vara style = {{color: ”red”}}.

Och i slutet exportera vi filen som följande:

```
export default App;
```

det ska vara snabbare att skapa en komponent med hjälp av extension **ES7+ React/Redux/React-Native snippets**

då räcker det med att använda snippets rafc(React Arrow Function Component), det skulle räcka med att skriva ra så att jag få den. Den brukar vara mer lättare

att skapa en CSS fil för varje komponent vi skapar, det är vanligt att de finnas i en katalog till exempel Header katalogen ska innehålla Header.js(komponent) och sin CSS fil Header.css, och i så fall börjar vi vår komponentfil med att importera CSS filen så här:

import ’ ./Header.css ’ . Med React 18 behöver vi inte längre importera React.

Eftersom det kommer och våra många komponenter som vi skapar och använder så blir det svårt att importera dem alla i index.js filen och det därför samlar vi dem i components katalog som ska lägga under src katalogen och skapar en ny JS fil index.js till exempel men den ska lägga i components katalog, vi exporterar våra komponenter i index.js så här:

```
export {default as Header} from './Header/Header'
export {default as Footer} from './Footer/Footer'
```

sen importerar vi dem i App.js så här:

```
import { Header, Footer } from './components';
```

Det skulle räcka utan att skriva index.js eftersom den förväntar sig att filen har index som namn , annars behöver vi skriva sökvägen med JS-filen Namn.

JSX-Inline-Style:

```
<h1 style={
        {
            color:”red”,
            fontSize: ”100px”,
```

backgoundColor:”green”

} }>khattab</h1>

Note: I JSX blir CSS Property som består av två ord till exempel font-size ett ord med stor bokstav för den andra ordet(JS variabler).

Note2: I JSX använder vi ordet className i stället för class t ex:

<h1 className=”myClassName”>Khattab</h1>.

Note3: för att skriva JavaScript i JSX skriver vi JS-kod inom { }.

Internal-style:

Deklarera en konstant med våra CSS Property och värde:

Const myStyle = {

color:”red”,

fontSize: ”100px”,

backgoundColor:”green”

}

Return(

<h1 style={myStyle}>kh</h1>

External-style:

Vi skriver CSS i vanlig CSS fil och sen importerar den i index.js:

```
import './CSSFileName.css';
```

Props: vi använder props för att reagera med komponenten som följande:

- I komponentfil ger vi funktion en parameter som t ex ”props” så här :

```
function SocialItems(props){
```

och skickar vi värden när vi ropar komponenten som följande:

```
<SocialItems name = "Khattab"/>
```

Sen använder vi värden i vår komponent:

```
<h1> {props.name}</h1>
```

Eller att använda JSON sätt dvs objekt med key;value som följande:

```
<SocialItems
        socialItemProp =
        {{
          name: 'Khattab',
          job: '.Net Utvecklare',
          pic: 'https://picsum.photos/254',
        }}
      />
```

Och sen kallar dem värden i vår komponent som följande:

```
<h1>
     {props.socialItemProp.name}
</h1>
```

Arrays sätt:

Vi kommer oftast att få våra data i form av Array of objekt då gör vi så här så att få värden och sen skicka tillkomponent:

Här är en Array of objekt:

```
const myArr =

  [

    {id:"1" ,name:"Khattab",job:".NetUtvecklare",pic:"https://picsum.photos/254"},

    {id:"2" ,name:"Maya", job:"Java Utvecklare", pic:"https://picsum.photos/224"},

    {id:"3" ,name:"Per", job:"C++ Utvecklare", pic:"https://picsum.photos/264"},

    {id:"4" ,name:"Jane", job:"Webb Utvecklare", pic:"https://picsum.photos/255"},

  ]
```

Vi mappar den med map() metod som följande:

```
const members = myArr.map(member =><SocialItems
                                    key = {member.id}
                                    name = {member.name}
                                    job = {member.job}
                                     pic = {member.pic}
                                            />);
```

memeber representerar ett objekt som ändras varje loop.

Det är bättre att använda ordet ”key” med unikt värde.

Sen kallar vi våra komponenter genom att kalla objekt members som vi mappat arrayn till med följande kod:

```
  return (
    <ul>
      {members}
    </ul>
```

```
  );
```

Att använda klasser i stället för metoder:

```
import React, { Component } from "react";

class SocialItems extends Component{
    render(){

    return(
            <li>
                <h1>
                    {this.props.name}
                </h1>
                <p>
                    {this.props.job}
                </p>
                <img src ={this.props.pic}
alt="" ></img>
            </li>
    );
}
}
export default SocialItems;
```

- I början importerar vi Component med React.
- Använder nyckordet class i stället för function.
- Nyckelordet extends för att ärva klassen Component.

        - Använder this.props för att nå Property utan att deklarera props som parameter som vi gör med function.

Klassen delar :

- Konstruktören :
    - Här deklarerar vi objektet state som vi använder för att spara värden :

```
constructor(){
        super();
        this.state = { number: 0 }
        this.clickHandler =
this.clickHandler.bind(this);
    }
```

    - Här ansluter vi metoder som kommer att använda ordet (this).
    - För att kunna få ny värde beroende på den gamla i state objekt :

```
   clickHandler(){
        this.setState(prevState => {
            return { number:
prevState.number + 1 }
        })
    }
```

Icons bibliotek: mest kända är Font Awesome, Bootstrap Icons, Remix Icons, och det standard sätt att använda icons i react är med hjälp av react icons på Github, standard vi importerar React-Icons via Terminalen:

```
npm install react-icons -save
```

Sen importerar vi dem Icons vi behöver i projektet :

```
import {FaStar} from 'react-icons/fa'
```

och för att använda den använder vi själv-stängda tags :

```
<h1><FaStar/>Khattab</h1>
```

På det sättet importerar vi icons som svg tag och sökväg som siffror och bokstäver och inte en bild och de gör webbsiten väldigt snabbare.

För att styla icons är det bästa sättet att hålla den mellan span tag och styla spanen JSX-InlineStyle:

```
<span style={{"color":
"greenyellow"}}><FaStar/></span>
```

Eller att ge spanen className och sen styla den med en external CSS fil .

Note : GTmetrix för att undersöka webbsiten efter hostning .

Note 2: importerar Bootstrap via terminalen:

npm install react-bootstrap bootstrap

note 3: importerar Bootstrap i min fil :

```
import'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/js/bootstrap.min.js';
```

Routing:

Först installerar vi biblioteket react-router-dom genom att använda npm så här:

npm install react-router-dom

sen behöver vi i filen App.js importera BrowserRouter men för att använda senare bara Router importerar vi den som Router(as Router), vi behöver importera Route och Routes (switch tidigare), vi importerar vi dem alla så här :

import { BrowserRouter as Router, Route, Routes } from ’ react-router-dom’