<https://supersimple.dev/react-basics> sitesine gidip oradaki html kodunu alıyoruz.

Ardından live server extension ı kuruyoruz. Sonra aldığımız kodu kopyala yapıştır ile yapıştırıp sağ tık yapıp open with live server diyoruz.

const container = document.querySelector('.js-container');

      ReactDOM.createRoot(container).render('Welcome to SuperSimpleDev React Course');

Üstteki kod ile js-container isimli ilk sınıfı seçeriz ve yazıyı render ederiz.

<script src="https://unpkg.com/supersimpledev/babel.js"></script>

    <script type="text/babel">

İle jsx kodunu js e çeviririz böylece browser anlayabilir.

    <script src="https://unpkg.com/supersimpledev/babel.js"></script>

    <script type="text/babel">

      const button = <button>Hello</button>

      const container = document.querySelector('.js-container');

      ReactDOM.createRoot(container).render(button);

    </script>

İle oluşturduğumuz button u renderlarız.

      const div = <div>

        <button>Hello</button>

        <p>paragraph of text</p>

        </div>;

      const container = document.querySelector('.js-container');

      ReactDOM.createRoot(container).render(div);

İle birden fazla elementi gösterebiliriz ekranda. Div içine yazmaksak sadece 1 elementi gösterebiliriz. Div içine yazarak istediğimiz kadar element gösterebiliriz.

Jsx de tüm elementlerde </> ile kapatma gerekir.

function ChatInput() {

        return (

          <div>

            <input/>

            <button>Send</button>

          </div>

        )

      }

      const div = (

        <div>

          {ChatInput()}

        </div>

      )

      const container = document.querySelector(".js-container");

      ReactDOM.createRoot(container).render(div);

Üstteki kod ile input kutucuğu ve hemen onun yanına send button u oluştururuz.

{ChatInput()} yerine <ChatInput></ChatInput> veya eğer içi boşsa örnekteki gibi <ChatInput/> yazarsak yine aynı sonucu alırız.

      function ChatInput() {

        return (

          <>

            <input/>

            <button>Send</button>

          </>

        )

      }

Burada önceki koddaki divleri sildik ki ekstra div oluşturmasın. Bu divsiz <> yapısına fragment denir.

Aşağıya almak için component div içine almak gerekir.

      function ChatMessage(props){

        const { message, sender } = props;

        if(sender === "robot"){

            return (

              <div>

                <img src="robot.png" width="50"/>

                {message}

              </div>

            );

        }

        return (

          <div>

            {message}

            <img src="user.png" width="50"/>

          </div>

        );

      }

      const app = (

        <>

          <ChatInput/>

          <ChatMessage message="hello Chatbot" sender="user"/>

          <ChatMessage message="Hello!How can I help you?" sender="robot"/>

          <ChatMessage message="Can you get me todays date?" sender="user"/>

          <ChatMessage message="today October 10" sender="robot"/>

        </>

      );

Ile öncelikle mesejı gönderenin bot mu user mı olduğunu ayırt ederek if bloğu ile eğer usersa img in sağ tarafta robotsa sol tarafta görünmesini sağladık

      function ChatMessage(props){

        const { message, sender } = props;

ile props parametresini fonksiyona geçtik ve props sayesinde app içinde aşağıda atadığımız message ve sender attributeslarına props.message veya { message, sender } = props; ile ulaşabildik.

{sender === "robot" && <img src="robot.png" width="50"/>}

Ile if statement ın jsx I halini yazarız. && dan önceki durum doğruysa img src robot olur.

function App() {

        const chatMessages = [

          {

            message: "hello ChatBot",

            sender: "user",

          },

          {

            message: "Hello!How can I help you?",

            sender: "robot",

          },

          {

            message: "Can you get me todays date?",

            sender: "user",

          },

          {

            message: "Today October 10",

            sender: "robot",

          },

        ];

        return (

          <>

            <ChatInput />

            {chatMessages.map((chatMessage) => {

              return (

                <ChatMessage

                  message={chatMessage.message}

                  sender={chatMessage.sender}

                />

              );

            })}

          </>

        );

      }

İle önce chatMessages isimli bir liste oluştururuz. Ardından aşağıda {} içine chatMessage.map içine yazdığımız arrow fonksiyonu ile listenin içindekileri teker teker ekrana yazdırırız.

Her chatMessage listesinin ögesine bir de id key i vermeliyiz ve bu id unique olmalı. Sonra da sender kısmından sonra aşağıda ChatMessage ın içinde key= {chatMessage.id} ile eklenmeli.

function sendMessage() {

          chatMessages.push({

            message: "New Message",

            sender: "user",

            id: crypto.randomUUID(),

          });

        }

        return (

          <>

            <button onClick={sendMessage}>Send Message</button>

            {chatMessages.map((chatMessage) => {

              return (

                <ChatMessage

                  message={chatMessage.message}

                  sender={chatMessage.sender}

                  key={chatMessage.id}

                />

              );

            })}

Üstteki kod ile önce send message ile yazan bir buton oluşturduk. Ardından bu button a onclick isimli eventi ekledik ve event handler olarak da sendMessage fonksiyonunu ekledik. Bu fonksiyon sayesinde butona bastığımızda chatMessages listesine yeni bir eleman ekleyecek.

      function ChatMessages() {

        const array = React.useState([

          {

            message: "hello ChatBot",

            sender: "user",

            id: "id1",

          },

          {

            message: "Hello!How can I help you?",

            sender: "robot",

            id: "id2",

          },

          {

            message: "Can you get me todays date?",

            sender: "user",

            id: "id3",

          },

          {

            message: "Today October 10",

            sender: "robot",

            id: "id4",

          },

        ]);

Ile chatMessages fonksiyonunu state durumuna getiriyoruz. Böylece send message buttonuna tıkladığımızda sabit kalmaz html de yenilenir ve yeni mesajlar da ekranda görünür.

        const chatMessages = array[0];

        const setChatMessages = array[1];

React.useState(list) komutu iki şey oluşturur. Ilki mevcut listeyi alır içine array[0] ile mevcut listeyi seçeriz. Ikinci olarak da bu listeyi ya da state I değiştirmeye yarayan fonksiyonu tutar. Array[1] ile de bu fonksiyonu setChatMessages adlı değişkene atarız. Böylece send button tuşuna bastığımızda sayfadaki html kodunu yenileriz.

function sendMessage() {

          setChatMessages([

            ...chatMessages,

            {

              message: "test",

              sender: "user",

              id: crypto.randomUUID(),

            },

          ]);

        }

sendMessage fonksiyonu button a eklediğimiz fonksiyondu. Bu yeni formatında spread operatörü ile(…) önce array[0] ile aldığımız mevcut listenin bir kopyasını array[1] i atadığımız setChatMessages içine ekleriz sonrasında da yeni nesneyi ekleriz listeye.

function ChatInput() {

        const [inputText, setInputText] = React.useState("");

        function saveInputText(event) {

          setInputText(event.target.value);

        }

        function sendMessage() {

          console.log(inputText);

        }

        return (

          <>

            <input

              placeholder="Send a Message to Chatbot"

              size="30"

              onChange={saveInputText}

            />

            <button onClick={sendMessage}>Send</button>

          </>

        );

      }

Üstteki kod ile text box a yazdığımız metni send butonuna verdiğimiz fonksiyon ile konsola yazdırdık.

**Npx create-vite komutu ile react için gerekli olan dosyaları ve şablonları oluşturur.**

**Arından npm install komutunu gireriz.**

**Npm run dev ile yeni react sitemizi başlatacak.**