**Question 1 - React component that shows the current time to the user.**

import {useEffect, useState} from 'react';

function Time(){

const [date, setDate] = useState(new Date());

useEffect(() => {

const timer = setInterval(() => setDate(new Date()))

return () => clearInterval(timer);

});

return (

<p>{`${date.toLocaleTimeString()}`}</p>

);

}

export default Time;

------------------------------------------------

var time = null;

function start(){

time = setInterval(

() => {

const d = new Date();

console.log(d.toTimeString());

}

, 1000);

}

function stop(){

clearInterval(time);

}

------------------------------------------------------

**Output of this?**

console.log(1);

function foo() {

console.log(2);

setTimeout(() => console.log(3/1000), 1000)

setTimeout(() => console.log(3/0), 0)

console.log(4);

}

console.log(5);

foo();

console.log(6);

1

5

2

4

6

3/0

3/1000

----------------------------------------------

let num = 1;

const interval = setInterval(

() => {

if(num <=5){

console.log(num);

num += 1;

}else{

console.log('Inside Here')

clearInterval(interval);

}

}

, 1000);

// 1 2 3 4 5 Inside Here

—-------------------------------------------------

const person = {

firstName: 'saranj',

print: function(){

console.log(this.firstName); // saranj // this => object

},

print2: () => {

console.log(this.firstName); // undefined // this => global object = window

}

}

person.print();

person.print2();

—--------------------------------------

***Everything in JavaScript is an object***

function foo(param) {

const name = "Bar"

console.log(name, param);

}

foo.age = 50; // attached a user defined attribute to JavaScript object (i.e. function)

console.log(foo.name); // foo // return name of function (inbuild name method)

console.log(foo.age); // 50 //return user defined attribute

console.log(foo.prototype); // {constructor: f}// to check all the methods and attributes of function foo

foo.name = 12; // no update will happen to inbuild name property

console.log(foo.name); // foo

—------------------------------------------------------

function createUser(name) {

return {

name,

greet: function () {

console.log(`Hello ${this.name}`)

}

}

}

const user1 = createUser("Foo")

const user2 = createUser("Bar")

console.log(user1) // {name: 'Foo', greet: f {}}

console.log(user2) // {name: 'Bar', greet: f {}}

user1.greet(); // hello Foo

user2.greet(); // hello Bar

—---------------------------------------------------

***The Object.create() method creates a new object using the prototype of the given object.***

let Student = {

name: "Lisa",

age: 24,

marks: 78.9,

display() {

console.log("Name:", this.name);

}

};

// create object from Student prototype

let std1 = Object.create(Student);

std1.name = "Sheeran";

std1.display(); // Sheeran

console.log(std1.age) // 24

console.log(std1.marks) // 78.9

function createUser(name) {

const userObj = Object.create(userFunctionStore);

userObj.name = name;

return userObj;

}

const userFunctionStore = {

greet: function() {

console.log(`Hello ${this.name}`)

},

}

const user1 = createUser("Foo")

console.log(user1) //{name: 'Foo'}

console.log(user1.name) // Foo

user1.greet() // Hello Foo

—---------------------------------------

***Exact Center div***

.flexContainer{

display: flex;

align-item: center; // align the items vertically

justify-content: center; // align the items horizontally

}

-------------------------------------------------

**React countdown timer**

import React, { useEffect, useState } from 'react';

function Timer(props) {

const [times, setTime] = useState(props.time);

useEffect(

() => {

const interval = setInterval(

() => {

setTime(prevTime => {

prevTime -= 1;

if (prevTime <= 0) {

clearInterval(interval);

return 0

}

return prevTime;

});

}

, 1000);

return () => clearInterval(interval);

}

, []);

return (

<>

{times? <h1>{times}</h1> : <h1>TimeOut</h1>}

</>

);

}

----------------------------------

import React, {useState, useRef, useEffect} from "react";

import "./App.css";

function App() {

const [counter, setCounter] = useState(0);

const interval = useRef();

useEffect(

() => {

return () => clearInterval(interval.current)

}

, []);

function timer(){

interval.current = setInterval(

() => {

setCounter(prevCount => prevCount + 1);

}

,1000);

}

return (

<>

<p>{counter}</p>

<input type="number" onChange={e => setCounter(Number(e.target.value))} />

<br />

<button onClick={() => timer()}>Start</button>

<button onClick={() => clearInterval(interval.current)}>Stop</button>

<button onClick={() => {

clearInterval(interval.current);

setCounter(0);

}}>Reset</button>

</>

);

}