ReactJS Assignment

* OOJS + DOM:

<html>

    <head>

        <title>

            OOJS + DOM

        </title>

    </head>

    <body>

        <div id="root">

            FirstName: <input type="text" id="firstName" /> <br>

            LastName: <input type="text" id="lastName" /> <br>

            Number Of Working days: <input type="text" class="nowd" /> <br>

            Salary: <input type="text" id="salary" /> <br>

            Net Salary: <input type="text" class="salary" /> <br>

        </div>

        <script>

        // create a class => Person (firstName, lastName, age)

        class Person{

            Person(firstName,lastName,age){

                this.firstName=firstName;

                this.lastName=lastName;

                this.age=age;

            }

        }

        // create a class Employee <= Person (numberOfWorkingDays, salary)

        class Employee extends Person{

            constructor(numberOfWorkingDays,salary){

                super(firstName);

                this.numberOfWorkingDays=numberOfWorkingDays;

                this.salary=salary

            }

        }

        // Add a function to Employee class => getSalary() => (salary / 30) \* numberOfWorkingDays

        let x=getSalary(salary,numberOfWorkingDays);

        function getSalary(salary,numberOfWorkingDays){

            return (salary / 30) \* numberOfWorkingDays;

        }

        // create obj <= Employee (Jane, Doe, 25, 25000);

        const Employee = new Employee("Jane","Doe",25,25000);{

            constructor(firstName,lastName,age,salary)

        {

            this.firstName="Jane";

            this.lastName="doe";

            this.age="25";

            this.salary="25000"

        }

    }

        // Assign the properties ir. firstName, lastName, numberOfWorkingDays, salary & output of getSalary(salary class)

        Const firstName=document.getElementById("firstName");

        const lastName=document.getElementById("lastName");

        const numberOfWorkingDays=document.getElementsByClassName("numberOfWorkingDays");

        const salary=document.getElementById("salary");

        const netSalary=document.getElementsByClassName("salary")[0];

        // input => docuement,getElementById('firstName').value = '';

        firstName.value="Jane";

        lastName.value="Doe";

        numberOfWorkingDays.value="25";

        salary.value="25000";

        console.log(obj.getSalary());

        netSalary.value=obj.getSalary().toString();

        </script>

    </body>

</html>

* Class Car with sales and purchases:

<!DOCTYPEhtml>

<htmllang="en">

  <head>

    <metacharset="UTF-8"/>

    <metahttp-equiv="X-UA-Compatible"content="IE=edge"/>

    <metaname="viewport"content="width=device-width, initial-scale=1.0"/>

    <title>Assignment 2</title>

  </head>

  <body>

    <divid="root"></div>

    <script>

      // Create class Sale with properties => name, amount

      classSale {

        constructor(name, amount) {

          this.name = name;

          this.amount = amount;

        }

      }

      // Create class Purchase with properties => name, amount

      classPurchase {

        constructor(name, amount) {

          this.name = name;

          this.amount = amount;

        }

      }

      // Create a Class Company with properties => name, sales(array of Sale), purchases(array of Purchase)

      classCompany {

        // sales & purchases will be private members

        #sales;

        #purchases;

        constructor(name, sales =[], purchases = []) {

          this.name = name;

          this.#sales = sales;

          this.#purchases = purchases;

        }

        getsales() {

          returnthis.#sales;

        }

        setsales(sale) {

          this.#sales.push(sale);

        }

        getpurchases() {

          returnthis.#purchases;

        }

        setpurchases(purchase) {

          this.#purchases.push(purchase);

        }

        // Add a function addSale => will create new objects of Sale and push it to the sales arrays

        addSale(name, amount) {

          constnewSale = newSale(name, amount);

          this.#sales.push(newSale.amount);

          console.log(`${newSale.name} - ${newSale.amount}`);

          //   sales = newSale.amount;

        }

        // Add a function addPurchase => will create new objects of Purchase and push it to the purchaese arrays

        addPurchase(name, amount) {

          constnewPurchase = newPurchase(name, amount);

          this.#purchases.push(newPurchase.amount);

          console.log(`${newPurchase.name} - ${newPurchase.amount}`);

          //   purchases = newPurchase.amount;

        }

        /\*\*

         \* Add a function getProfitLossReport

         \* 1. Calculate Profit using sales

         \* 2. Calculate spends using purchases

         \* 3. Calculate difference between profit and spends

         \* 4. return an object in following format

         \* {

         \* income: sum of sales,

         \* expense: sum of purchases,

         \* profit: income - expense, (if income > expense)

         \* loss: expense - income, (if income < expense)

         \* }

         \*

         \*/

        getProfitLossReport() {

          letincome = this.#sales.reduce(

            (totalIncome, next) => (totalIncome += next)

          );

          console.log(`Income - ${income}`);

          letexpense = this.#purchases.reduce(

            (totalExpense, next) => (totalExpense += next)

          );

          console.log(`expense - ${expense}`);

          letprofit, loss;

          letisProfit = false;

          if (income>expense) {

            profit = income - expense;

            isProfit = true;

            console.log(`Profit - ${isProfit}    Profit - ${profit}`);

            return [profit, isProfit];

          } else {

            loss = expense - income;

            isProfit = false;

            console.log(`Profit - ${isProfit}    loss - ${loss}`);

            return [loss, isProfit];

          }

        }

      }

      // Create a Class MNC by inheriting Company with extra properties =>taxRate, country

      classMNCextendsCompany {

        constructor(name, sales = [], purchases = [], taxRate, country) {

          super(name, (sales = []), (purchases = []));

          this.taxRate = taxRate;

          this.country = country;

        }

        calculateTax() {

          lettaxAmount, profit;

          [profit, isProfit] = getProfitLossReport();

          if (profit>4\_00\_000&&isProfit) {

            taxAmount = profit - 4\_00\_000;

          }

          returntaxAmount \* (this.taxRate / 100);

        }

      }

      // Add a function calculateTax

      // If profit is more than 5,00,000 tax will be applied to amount (profit - 500000)

      // ex. if profit = 600000 &taxRate = 10

      // taxable amount will be 600000 - 500000 = 100000

      // tax = 100000 \* 10/100 = 10000

      // Create an object of MNC and calculate tax with differnet input combination

      // add Sales & purchases to make it realtime

      constbenz = newMNC("benz", [], [], 10, "India");

      console.log(

        `Company Name - ${benz.name}      Country - ${benz.country}`

      );

      console.log("Purchases");

      benz.addPurchase("benz truck", 9500000);

      benz.addPurchase("benz car", 12000000);

       benz.purchases;

      console.log("");

      console.log("");

    console.log("Sales");

      benz.addSale("benz truck", 99000000);

      benz.addSale("benz car", 250000000);

      benz.sales;

      console.log("");

      console.log("");

      console.log("");

      console.log("Profit/Loss Report");

      benz.getProfitLossReport();

      console.log("");

      console.log("");

      console.log("");

      console.log("\*\*\*\*\*\*Tax\*\*\*\*\*\*\*\*");

      console.log(`Tax = ${benz.calculateTax()}`);

    </script>

  </body>

</html>