**Exercise 01**

Re-write the following code using TypeScript. Try to be as explicit as possible and add Types to everything you can. When you are done, transpile the TS code to JS code and inspect the JS code.

function University(name, dept) {

this.name = name;

this.dept = dept;

this.graduation = function(year) {

console.log(`Graduating ${this.dept} ${year} students`);

}

}

var miu = new Univeristy("MIU", "Computer Science");

miu.graduation(2019);

 class Univeristy {

    name: string;

    dept: string;

    constructor(name: string, dept: string) {

        this.name = name;

        this.dept = dept;

    }

    graduation(year: number){

        console.log(`Graduating ${this.dept} ${year} students`);

    }

}

var miu = new Univeristy("MIU", "Computer Science");

miu.graduation(2019);

**Exercise 02**

Re-write the following code using TypeScript. Try to be as explicit as possible and add Types to everything you can. When you are done, transpile the TS code to JS code and inspect the JS code.

|  |  |
| --- | --- |
| let bankAccount = {  money: 2000,  deposit(value) {  this.money += value;  }  }; | let myself = {  name: "Asaad",  bankAccount: bankAccount,  hobbies: ["Violin", "Cooking"]  };  myself.bankAccount.deposit(3000);  console.log(myself); |

interface BankAccount{

    money: number;

    deposit(value: number):void;

}

class Myself implements BankAccount{

    money:number = 2000;

    name: string = "Asaad";

    hobbies:[string, string] = ["Violin", "Cooking"];

    deposit(value:number){

        this.money += value;

    }

}

let myself = new Myself();

myself.deposit(3000);

console.log(myself);

**Exercise 03**

Re-write the following code using TypeScript Class syntax. Try to be as explicit as possible and add Types to everything you can. When you are done, transpile the TS code to JS code and inspect the JS code.

function Car(name) {

this.name = name;

this.acceleration = 0;

this.honk = function() {

console.log(` ${this.name} is saying: Toooooooooot!`);

};

this.accelerate = function(speed) {

this.acceleration = this.acceleration + speed;

}

}

var car = new Car("BMW");

car.honk(); // BMW is saying: Toooooooooot!

console.log(car.acceleration); // 0

car.accelerate(60);

console.log(car.acceleration); // 60

|  |  |
| --- | --- |
| "use strict";  class Car {      constructor(name) {          this.acceleration = 0;          this.name = name;      } | honk() {          console.log(` ${this.name} is saying: Toooooooooot!`);      }      accelerate(speed) {          this.acceleration = this.acceleration + speed;      }  } |
| var car = new Car("BMW");  car.honk(); // BMW is saying: Toooooooooot!  console.log(car.acceleration); // 0  car.accelerate(60);  console.log(car.acceleration); // 60 | |

**Exercise 04**

Re-write the following code using TypeScript Class syntax. Try to be as explicit as possible and add Types to everything you can. When you are done, transpile the TS code to JS code and inspect the JS code.

|  |  |
| --- | --- |
| var baseObject = {  width: 0,  length: 0  };  var rectangle = Object.create(baseObject);  rectangle.width = 5;  rectangle.length = 2; | rectangle.calcSize = function() {  return this.width \* this.length;  };  console.log(rectangle.calcSize()); // 10 |

|  |  |
| --- | --- |
| class Base{      \_width: number = 0;      \_length: number = 0;      set width(width: number){          this.\_width = width;      }      set length(length: number){          this.\_length = length;      }      get width(){          return this.\_width;      }      get length(){          return this.\_length;      } | calcSize () {          return this.width \* this.length;      };  }  var rectangle = new Base();  rectangle.width = 5;  rectangle.length = 2;  console.log(rectangle.calcSize()); // 10 |