# TS Assignment

**What will this code log? Explain your reasoning.**

*type Person = {*

*name: string;*

*sayHi: () => void;*

*};*

*const person: Person = {*

*name: 'Alice',*

*sayHi: function () {*

*console.log(`Hi, my name is ${this.name}.`);*

*}*

*};*

*person.sayHi();*

**What is logged to the console, and why? How would you fix it so it logs "Alice"?**

*type Person = {*

*name: string;*

*greet: () => void;*

*};*

*const person: Person = {*

*name: 'Alice',*

*greet: function () {*

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

*}*

*};*

*const greetFunction = person.greet.bind(person);*

*greetFunction();*

**What will this code log and why?**

*type User = {*

*name: string;*

*logName: () => void;*

*};*

*const user: User = {*

*name: 'Bob',*

*logName: function () {*

*setTimeout(() => {*

*console.log(this.name);*

*}, 100);*

*}*

*};user.logName();*

**You have a User object with a method that logs a welcome message. There's also a "Login" button in your HTML. You want the welcome method to be called when the button is clicked, and the message should use the User object's data.**

*type UserType = {*

*name: string;*

*welcome: () => void;*

*};*

*const User: UserType = {*

*name: 'Alice',*

*welcome: function () {*

*console.log(`Welcome, ${this.name}!`);*

*}*

*};*

*const loginBtn = document.getElementById('loginBtn');*

*if (loginBtn) {*

*loginBtn.addEventListener('click', User.welcome.bind(User));*

*}*

**What will be logged to the console? Explain!**

*let animal: string = 'Cat';*

*function showAnimal(): void {*

*let animal: string = 'Dog';*

*console.log(animal);*

*}*

*showAnimal();*

*console.log(animal);*

**What will this code output? Why?**

*function test(): void {*

*console.log(a);*

*console.log(foo());*

*var a: number = 1;*

*function foo(): number {*

*return 2;*

*}*

*}*

*test();*

**Explain the output of this for loop.**

*for (let i = 0; i < 5; i++) {*

*setTimeout(function() {*

*console.log(i);*

*}, 10);*

*}*

**What will the following code log to the console on the last two lines? Explain why the count variable is not reset.**

*function createCounter(): () => void {*

*let count: number = 0;*

*return function (): void {*

*count++;*

*console.log(count);*

*};*

*}*

*const counter = createCounter();*

*counter();*

*counter();*

**What does this code log? Explain!**

*type MyObjectType = {*

*id: string;*

*createLogger: () => () => void;*

*};*

*const myObject: MyObjectType = {*

*id: 'my-object',*

*createLogger: function () {*

*return () => {*

*console.log(`Logger for ${this.id}`);*

*};*

*},*

*};*

*const logger = myObject.createLogger();*

*logger();*

**Write a function *makeAdder(x)* that takes a number x and returns a new function. The new function should take a number y and return the *sum x + y*. Use a closure to achieve this.**

*function makeAdder(x: number): (y: number) => number {*

*return (y: number): number => {*

*return x + y;*

*};*

*}*

*const add = makeAdder(5);*

*console.log(add(2));*

**Implement a parent Animal and a child Dog relationship in two different ways:**

* Using the Constructor/Prototype pattern.
* Using the ES6 class syntax.

Your solution must satisfy two conditions for both patterns:

* The Animal must have an eat() method that the Dog inherits.
* The Dog must also have its own bark() method.

// Parent constructor

function Animal(this: any, name: string) {

this.name = name;

}

Animal.prototype.eat = function (): void {

console.log(`${this.name} is eating.`);

};

// Child constructor

function Dog(this: any, name: string, breed: string) {

Animal.call(this, name); // Call parent constructor

this.breed = breed;

}

// Inherit from Animal

Dog.prototype = Object.create(Animal.prototype);

Dog.prototype.constructor = Dog;

// Child method

Dog.prototype.bark = function (): void {

console.log(`${this.name} is barking.`);

};

const dog1 = new (Dog as any)('Buddy', 'Labrador');

dog1.eat(); // Buddy is eating.

dog1.bark(); // Buddy is barking.  
  
ES6 class

class Animal {

name: string;

constructor(name: string) {

this.name = name;

}

eat(): void {

console.log(`${this.name} is eating.`);

}

}

class Dog extends Animal {

breed: string;

constructor(name: string, breed: string) {

super(name);

this.breed = breed;

}

bark(): void {

console.log(`${this.name} is barking.`);

}

}

const dog2 = new Dog('Max', 'Beagle');

dog2.eat(); // Max is eating.

dog2.bark(); // Max is barking.

**Design a content feed/post UI (similar to X/Twitter) involving Post, Comment, and User details.[Using either fetch API or axios and DOM APIs]**

Use the **API** from Resources section: <https://jsonplaceholder.typicode.com/>

**Features:**

* Feed View (like X/Twitter)
* Render a list of posts
* Each post should display
  + The user (owner) who created the post
  + The comments related to the post
  + When a user's name is clicked, navigate to their profile page.

**User Profile View:**

* Display the basic details of the user
* Show a photo gallery of the user(**/albums** and **/photos API**)
* Show a list of todos associated with the user**(/todos api)**

**Hint**: *All entities (posts, comments, photos, todos) are interconnected using the userId(users). Use userId to associate and map the relevant data.*

**Let there be two files in the same folder: *index.html* and *index.js*. Update code in index.js so that the counter app works perfectly.**

***index.html***

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8" />*

*<meta name="viewport" content="width=device-width, initial-scale=1.0" />*

*<title>Counter</title>*

*<script src="https://cdn.tailwindcss.com"></script>*

*</head>*

*<body class="bg-gray-100 h-screen flex items-center justify-center font-sans">*

*<div class="bg-white rounded-xl shadow-lg text-center py-10 px-16">*

*<div id="counter-value" class="text-6xl font-bold text-gray-800 mb-8">*

*0*

*</div>*

*<div class="flex justify-center gap-5">*

*<button*

*id="decrement"*

*class="py-4 px-8 text-lg bg-indigo-500 text-white rounded-lg transition duration-200 hover:bg-indigo-600 active:scale-95"*

*>*

*Decrement*

*</button>*

*<button*

*id="increment"*

*class="py-4 px-8 text-lg bg-indigo-500 text-white rounded-lg transition duration-200 hover:bg-indigo-600 active:scale-95"*

*>*

*Increment*

*</button>*

*</div>*

*</div>*

*<script src="./index.js"></script>*

*</body>*

*</html>*

***index.js***

*class Counter {*

*value: number;*

*decrementBtn: HTMLElement | null;*

*incrementBtn: HTMLElement | null;*

*counter: HTMLElement | null;*

*constructor() {*

*this.value = 0;*

*this.decrementBtn = document.getElementById("decrement");*

*this.incrementBtn = document.getElementById("increment");*

*this.counter = document.getElementById("counter-value");*

*if (this.counter) {*

*this.counter.innerText = this.value.toString();*

*}*

*// Bind event listeners*

*this.incrementBtn?.addEventListener("click", () => this.increment());*

*this.decrementBtn?.addEventListener("click", () => this.decrement());*

*}*

*increment(): void {*

*this.value++;*

*if (this.counter) {*

*this.counter.innerText = this.value.toString();*

*}*

*}*

*decrement(): void {*

*this.value--;*

*if (this.counter) {*

*this.counter.innerText = this.value.toString();*

*}*

*}*

*}*

*const counter = new Counter();*