Classes and Objects TS

Introduction to Classes & Objects

Fundamental of Classes & Objects

Classes & Objects :

Constructor

Classes & Objects: this. keyword

Classes & Objects : Public & Private Access Modifier

Classes & Objects : Protected Access Modifier   Classes & Objects: Optional Properties

Classes & Objects : Parameter Properties

Classes & Objects: Getters & Setters

Classes & Objects: Static Members

Classes & Objects: Abstract Classes given details notes with example in typescript

Here's a detailed note on **Classes and Objects in TypeScript**, covering all the mentioned topics with **clear explanations and examples**.

---

1. Introduction to Classes & Objects

- **Class**: A blueprint to create objects with properties and methods.
- **Object**: An instance of a class.

```ts
class Person {
  name: string;
  age: number;

  greet() {
    console.log(`Hello, my name is ${this.name}`);
  }
}

const person1 = new Person();
person1.name = "Supriya";
person1.age = 26;
person1.greet(); // Output: Hello, my name is Supriya
```

---

2. Fundamentals of Classes & Objects

- A class encapsulates data and behavior.
- Objects are created using the `new` keyword.
- Properties hold data, methods perform actions.

---

3. Constructor

- Used to initialize class members when an object is created.

```ts
class Car {
  brand: string;

  constructor(brand: string) {
    this.brand = brand;
  }

  showBrand() {
    console.log(`Brand: ${this.brand}`);
  }
}

const car1 = new Car("Toyota");
car1.showBrand(); // Output: Brand: Toyota
```

---

4. this Keyword

- Refers to the current instance of the class.

```ts
class Student {
  name: string;

  constructor(name: string) {
    this.name = name;
  }

  sayHello() {
    console.log(`Hi, I am ${this.name}`);
  }
}
```

---

5. Public & Private Access Modifier

- **Public**: accessible anywhere.
- **Private**: only within the class.

```ts
class Account {
  public accountName: string;
  private balance: number;

  constructor(accountName: string, balance: number) {
    this.accountName = accountName;
    this.balance = balance;
  }

  getBalance() {
    return this.balance;
  }
}

const acc = new Account("Supriya", 5000);
// acc.balance = 1000; // Error: balance is private
console.log(acc.getBalance()); // 5000
```

---

6. Protected Access Modifier

- **Protected**: accessible in the class and subclasses.

```ts
class Animal {
  protected type: string = "animal";
}

class Dog extends Animal {
  showType() {
    console.log(`Type: ${this.type}`);
  }
}

const dog = new Dog();
dog.showType(); // Output: Type: animal
```

---

7. Optional Properties

- Use `?` to make a property optional.

```ts
class Employee {
  name: string;
  department?: string;

  constructor(name: string, department?: string) {
    this.name = name;
    this.department = department;
  }
}

const emp1 = new Employee("Supriya");
const emp2 = new Employee("John", "IT");
```

---

8. Parameter Properties

- A shorthand to declare and initialize properties in the constructor.

```ts
class Book {
  constructor(public title: string, private price: number) {}

  getPrice() {
    return this.price;
  }
}

const book = new Book("TypeScript Basics", 299);
console.log(book.title); // TypeScript Basics
console.log(book.getPrice()); // 299
```

---

9. Getters & Setters

- Encapsulation: control access to class properties.

```ts
class User {
  private _age: number = 0;

  get age(): number {
    return this._age;
  }

  set age(value: number) {
    if (value >= 0) {
      this._age = value;
    }
  }
}

const u = new User();
```

```
u.age = 25;
console.log(u.age); // 25
```

---

10. Static Members

- Belong to the class, not instances.

```ts
class MathUtil {
  static PI = 3.14;

  static circleArea(radius: number) {
    return this.PI * radius * radius;
  }
}

console.log(MathUtil.circleArea(5)); // 78.5
```

---

11. Abstract Classes

- Cannot be instantiated.
- Used as a base class.
- May have abstract methods.

```ts
abstract class Shape {
  abstract getArea(): number;

  display(): void {
    console.log("Displaying shape...");
  }
}

class Rectangle extends Shape {
  constructor(private width: number, private height: number) {
    super();
  }

  getArea(): number {
    return this.width * this.height;
  }
}

const rect = new Rectangle(10, 5);
rect.display();
console.log(rect.getArea()); // 50
```

Would you like this compiled into a downloadable PDF or formatted into slides or notes format?

given pdf

Here is your PDF with detailed notes and TypeScript examples on Classes and Objects: