TypeScript Learning PDF – Days 2, 3 & 4

DAY 2 – Basics & Functions

1. Function with Types

```
function square(n: number): number {
    return n * n;
}
console.log(square(3)); // 9
```

- `number` → input type & return type
- Strong typing prevents errors

---

2. Optional Parameters

```
function greet(name: string, message?: string): void {
    console.log("Name:", name);
    if (message !== undefined) console.log("Message:", message);
}
greet("ashim", "I love you");
```

- `?` marks optional
- TypeScript checks types

---

3. Default Parameters

```
function add(a: number, b: number = 5): number {
    return a + b;
}
console.log(add(3)); // 8
```

---

4. Arrays & Filter

```
const numbers: number[] = [1,2,3,4,5,6];
let even: number[] = numbers.filter(num => num % 2 === 0);
console.log(even); // [2,4,6]
```

---

5. Rest & Spread Operators

```
const nums: number[] = [1,3,5,6,7];
function sum(...nums: number[]): number {
    let total = 0;
    for (let n of nums) total += n;
    return total;
}
console.log(sum(...nums)); // 22
```

DAY 3 – Advanced Types

1. Union Types

```
let id: number | string;
id = 101;
id = "A101";
```

2. Type Aliases

```
type UserID = number | string;
function show(id: UserID) { console.log(id); }
```

3. Literal Types

```
let direction: "up" | "down" | "left" | "right";
direction = "up"; // valid
// direction = "hello"; // ❌ error
```

4. Interfaces & Inheritance

```
interface Rectangle { heigth: number; width: number; }
interface ColoredRectangle extends Rectangle { color: string; }

const coloredRectangle: ColoredRectangle = { heigth:20, width:30, color:"red" };
console.log(coloredRectangle);
```

Challenge 1 – Union Function

```
function formatValue(value: number | string): string {
    return typeof value === "number" ? `Number: ${value}` : `String: ${value}`;
}
console.log(formatValue(10)); // Number: 10
console.log(formatValue("hi")); // String: hi
```

DAY 4 – Generics & Advanced Patterns

1. Generic Function

```
function identity<T>(value: T): T {
    return value;
}
console.log(identity<number>(10));
console.log(identity<string>("ashim"));
```

2. Generic with Arrays

```
function getFirst<T>(arr: T[]): T {
    return arr[0];
}
console.log(getFirst<number>([1,2,3]));
```

3. Generic Constraints

```
interface Person { name: string; }
function printName<T extends Person>(obj: T) {
    console.log(obj.name);
```

```
}
printName({ name: "Ashim", age: 20 });
```

4. Generic Interface

```
interface APIResponse<T> { status: number; data: T; }
const userResponse: APIResponse<string> = { status: 200, data: "Success" };
```

5. Generic Class

```
class Box<T> { value: T; constructor(value: T){ this.value = value; } }
const numberBox = new Box<number>(100);
const stringBox = new Box<string>("hello");
```

6. Generic Arrow Function

```
const add = <T extends number>(a: T, b: T): T => {
    return (a + b) as T;
}
console.log(add(2,3)); // 5
```

7. Utility Types

```
interface User { name: string; age: number; }
const p: Partial<User> = { name: "ashim" };
const user: Readonly<User> = { name: "ashim", age: 20 };
type UserName = Pick<User, "name">;
type UserWithoutAge = Omit<User, "age">;
```

Summary

- **Day 2**: Basics, functions, arrays, rest/spread
- **Day 3**: Union, Literal types, interfaces, inheritance
- **Day 4**: Generics (functions, arrays, interfaces, classes), utility types

Challenge Questions Included

1. Union Function: `formatValue`
2. Generic Arrow: `addGeneric`
3. Generic Array: `getFirst`
4. Interface + inheritance: `ColoredRectangle`