■ TypeScript Cheat Sheet (Beginner-Friendly)

■ Basics

```
// Types
let age: number = 25;
let name: string = "Suhas";
let isDone: boolean = false;

// Arrays
let numbers: number[] = [1, 2, 3];
let fruits: Array<string> = ["apple", "banana"];

// Any & Unknown
let data: any = 10;          // avoids checks (■ risky)
let safe: unknown = "hello"; // must check type
```

■ Union & Literal Types

```
let id: string | number;         // can be string OR number
let direction: "up" | "down";    // only specific values
```

■ Functions

```
function add(x: number, y: number): number {
  return x + y;
}

function log(msg: string): void {
  console.log(msg);
}

let multiply = (a: number, b: number): number => a * b;
```

■ Objects & Interfaces

```
interface User {
  id: number;
  name: string;
  isAdmin?: boolean; // optional
}

const user: User = { id: 1, name: "Alice" };
```

■ Type Aliases

```
type ID = string | number;

type Product = {
  id: ID;
  title: string;
};
```

■ Classes

```
class Animal {
  constructor(public name: string) {}
  move(distance: number) {
    console.log(`${this.name} moved ${distance}m`);
  }
}

class Dog extends Animal {
  bark() {
    console.log("Woof!");
  }
}

const dog = new Dog("Buddy");
dog.move(10);
dog.bark();
```

■■ Generics

```
function identity<T>(value: T): T {
  return value;
}

let num = identity<number>(42);
let word = identity("hello"); // inferred
```

■ Enums

```
enum Role {
  User,
  Admin,
  Guest
}

let role: Role = Role.Admin;
```

■ Type Narrowing

```
function printId(id: string | number) {
  if (typeof id === "string") {
    console.log("Uppercase: " + id.toUpperCase());
  } else {
    console.log("ID: " + id);
  }
}
```

■ Type Assertions

```
let input: unknown = "Hello TS";
let length: number = (input as string).length;
```

■ Utility Types (Common)

```
interface User {
  id: number;
  name: string;
  age?: number;
}
type PartialUser = Partial<User>; // all optional
type ReadOnlyUser = Readonly<User>; // cannot change values
type UserName = Pick<User, "name">; // only { name }
type UserId = Omit<User, "age">;    // everything except age
```

■ tsconfig.json (basic)

```
{
  "compilerOptions": {
    "target": "ES6",
    "module": "commonjs",
    "strict": true,
    "esModuleInterop": true
  }
}
```