TypeScript objects

What is an Object?

An **object** is a collection of **key-value pairs**.
It contains:

- **Properties (variables)** – e.g., name, age, salary
- **Methods (functions)** – e.g., getDetails(), setDetails()

Objects represent real-world entities like Employee, Student, Product, etc.

👨‍💼 **Example: Employee**

```
let employee = {
  name: "John",
  salary: 50000,
  job: "Engineer",
  getDetails: function () {
    return `${this.name} is a ${this.job} earning ${this.salary}`;
  }
};
```

Accessing properties:

- Dot notation → employee.name
- Bracket notation → employee["name"]

Modifying:

```
employee.job = "Manager";
```

Different Ways to Create Objects in TS/JS

1. Using object type (JS/TS)
2. Inline Type Object (TS)
3. Using type aliases (TS)
4. Using Classes (JS/TS)

1. Using object type (JS/TS)

Basic way without strict typing:

```
let employee: object = {
  name: "John",
  age: 30,
  job: "Engineer"
};
```

But we can't access properties directly unless we define the structure or use any.

2. Inline Type Object (TS)

Here, we define the structure while creating the object.

```
let student: {
  name: string;
  age: number;
  grade: string;
  getSummary: () => string;
} = {
  name: "Scott",
  age: 15,
  grade: "A",
  getSummary: function () {
    return `${this.name} is ${this.age} years old and scored grade ${this.grade}`;
  }
};
```

⚠️ **Limitation**: Need to repeat the type structure for each object.

3. Using type aliases (TS)

Reusable type definitions.

```
type Product = {
  name: string;
  price: number;
  getInfo: () => string;
};
```

Then use it for multiple objects:

```
let book1: Product = { ... };
let book2: Product = { ... };
```

✅ Cleaner and avoids repetition.

Intersection Types:

Combining multiple types:

```
type Candidate = Personal & Contact & {
  getContactInfo: () => string;
};
```

4. Using Classes (JS/TS)

Blueprint for creating multiple objects with same structure and behavior.

```
class Person {
  constructor(public ssn: string, public firstName: string, public lastName: string) {}
  getFullName(): string {
    return `${this.firstName} ${this.lastName}`;
  }
  getDetails(): string {
    return `SSN: ${this.ssn}, Name: ${this.getFullName()}`;
  }
}
```

Create object:

let person1 = new Person("123", "John", "Doe");

✅ Summary Table

APPROACH	TYPESCRIPT SUPPORT	REUSABILITY	RECOMMENDED FOR
OBJECT TYPE	✅ Basic	❌	Small, quick objects
INLINE TYPE	✅ Strong	❌	One-time objects
TYPE ALIASES	✅ ✅	✅	Reusable object types
CLASSES	✅ ✅ ✅	✅ ✅	Object-oriented designs