In TypeScript, classes provide a way to define blueprints for creating objects with similar characteristics and behaviors. They are a fundamental part of object-oriented programming (OOP) in TypeScript. Here's an overview of key concepts related to classes in TypeScript:

Class Declaration:

```typescript
class ClassName {
  // Class members (properties and methods)
}
```

Constructors:

Constructors are special methods used for initializing class instances. They are defined using the `constructor` keyword.

```typescript
class Person {
  constructor(name: string, age: number) {
    this.name = name;
    this.age = age;
  }
}
```

Properties:

Properties are variables that belong to a class. They define the characteristics or attributes of objects created from the class.

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

  constructor(name: string, age: number) {
    this.name = name;
    this.age = age;
  }
}
```

Methods:

Methods are functions defined within a class. They define the behaviors of the objects created from the class.

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

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

  greet() {
    console.log(`Hello, my name is ${this.name} and I am ${this.age} years old.`);
  }
}
```

Inheritance:

Classes can inherit properties and methods from other classes using the `extends` keyword. This allows for code reuse and the creation of class hierarchies.

```typescript
class Student extends Person {
  studentId: number;

  constructor(name: string, age: number, studentId: number) {
    super(name, age);
    this.studentId = studentId;
  }

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

Access Modifiers:

Access modifiers (`public`, `private`, `protected`) control the visibility of class members.

- `public`: Members are accessible from outside the class.
- `private`: Members are accessible only within the class.
- `protected`: Members are accessible within the class and its subclasses.

```typescript
class Car {
  private speed: number;

  constructor(speed: number) {
    this.speed = speed;
  }

  accelerate() {
    this.speed += 10;
  }
}
```

Static Members:

Static members belong to the class itself rather than to individual instances. They are accessed using the class name.

```typescript
class MathUtils {
  static PI: number = 3.14;

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

console.log(MathUtils.circleArea(5)); // Output: 78.5
```

These are the fundamental concepts of classes and OOP in TypeScript. They provide a way to structure and organize code in a more modular and reusable manner.