

# Summary



**Classes defined**

**Class Members**

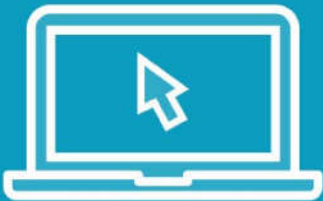
- Constructors
- Properties
- Methods
- Accessors

**Inheritance**

**Abstract Classes**

**Class Expressions**

Demo



Using class expressions

Demo



Creating abstract classes

# Abstract Classes

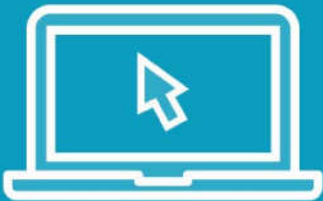
**Created with the “abstract” keyword**

**Base classes that may not be instantiated**

**May contain implementation details**

**Abstract methods are not implemented**

Demo

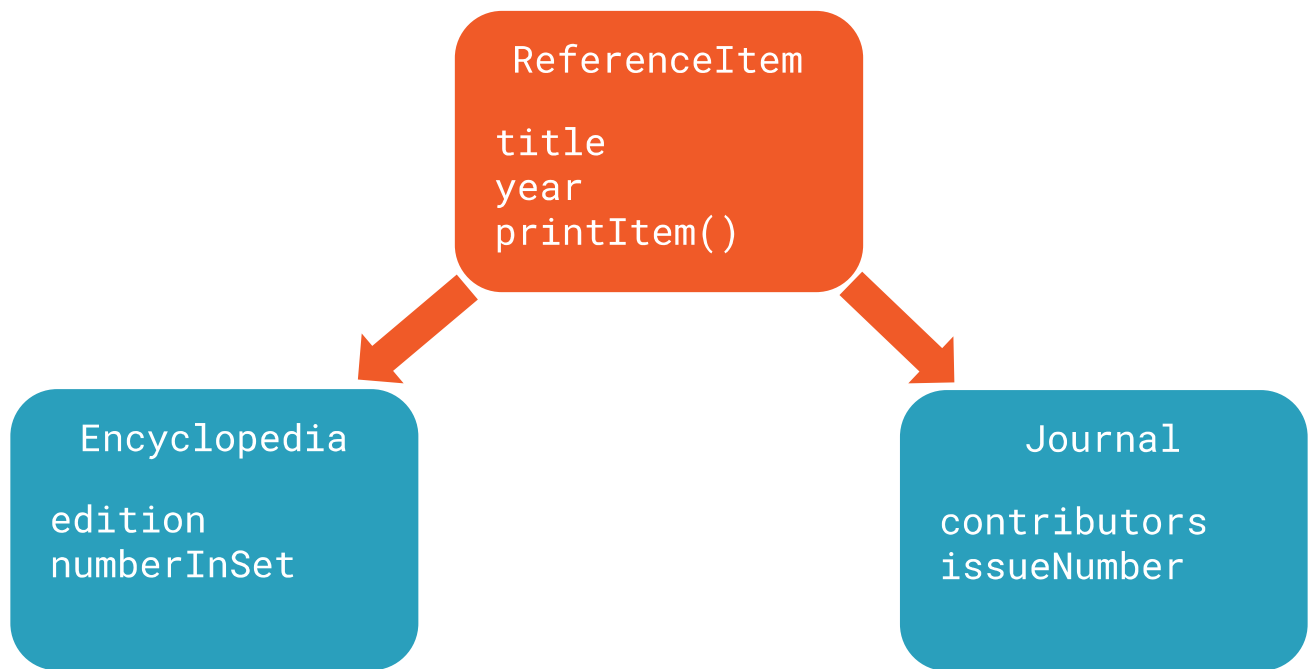


**Defining an inheritance hierarchy**

## Extending Classes with Inheritance

```
class ReferenceItem {  
    title: string;  
    printItem(): void { // print something here }  
}  
↓  
class Journal extends ReferenceItem {  
    constructor() {  
        → super();  
    }  
    → contributors: string[];  
}
```

# Inheritance



Demo



Creating and using classes




# Access Modifiers

**Public**  
**Private**  
**Protected**

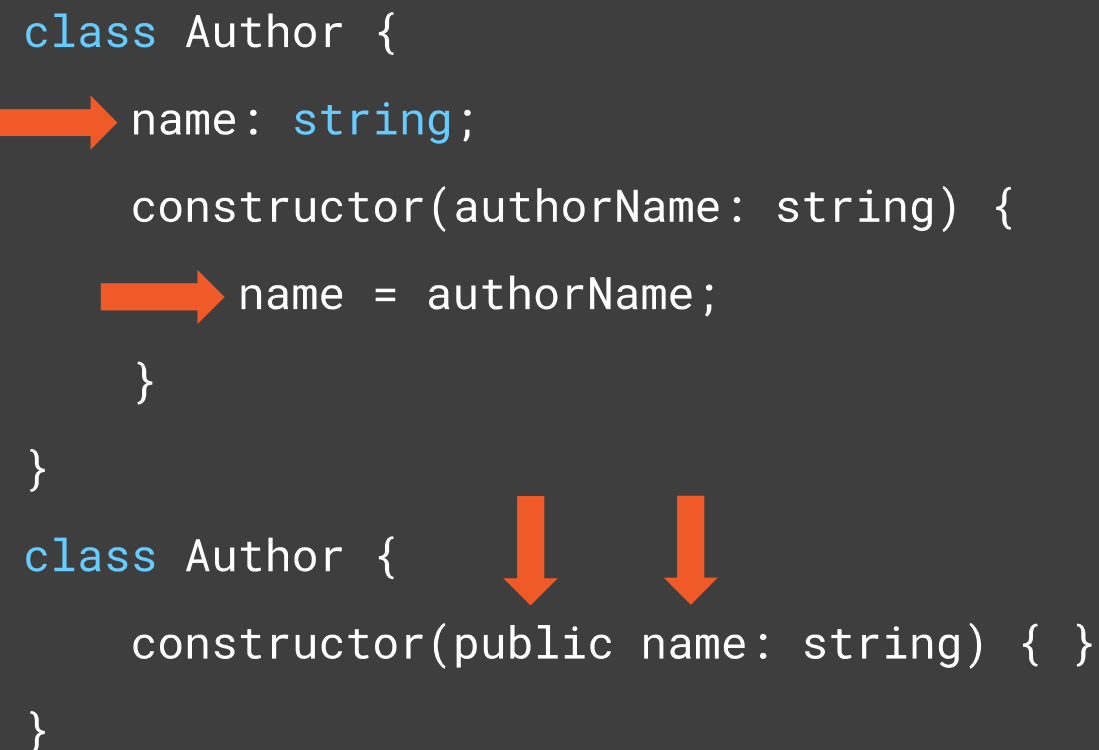
## Static Properties

```
class Library {  
    constructor(public name: string) { }  
    → static description: string = 'A source of knowledge.';  
}  
  
let lib = new Library('New York Public Library');  
let name = lib.name; // available on instances of the class  
let desc = Library.description; // available on the class
```

A diagram illustrating static properties in a TypeScript class. It shows a class named 'Library' with a constructor and a static property 'description'. An orange arrow points down from the title 'Static Properties' to the 'description' property. Another orange arrow points left from the 'description' property to the 'static' keyword. A third orange arrow points up from the 'Library.description' access in the code to the 'description' property. The code is color-coded: 'class' is blue, 'constructor' is grey, 'public' is blue, 'string' is blue, 'static' is blue, 'description' is orange, 'string' is blue, 'A source of knowledge.' is orange, 'let' is blue, 'new' is blue, 'Library' is blue, 'New York Public Library' is orange, 'name' is blue, 'lib.name' is blue, 'available on instances of the class' is green, 'Library.description' is blue, and 'available on the class' is green.

## Parameter Properties

```
class Author {  
  → name: string;  
    constructor(authorName: string) {  
      → name = authorName;  
    }  
}  
  
class Author {  
    constructor(public name: string) { }  
}
```



The diagram illustrates the concept of parameter properties in TypeScript. It shows two versions of a class named `Author`. The first version has a property `name` and a constructor that takes `authorName` and assigns it to `name`. The second version shows the property `name` as a parameter in the constructor signature, indicating that the property is initialized directly within the constructor. Two orange arrows point from the `name` property in the first version to the `public name` parameter in the second version, highlighting the transformation.

# Properties and Methods

```
class ReferenceItem {  
    numberOfPages: number;  
    get editor(): string {  
        // custom getter logic goes here, should return a value  
    }  
    set editor(newEditor: string) {  
        // custom setter logic goes here  
    }  
    printChapterTitle(chapterNum: number): void {  
        // print title here  
    }  
}
```

# Properties and Methods

```
class ReferenceItem {  
    numberOfPages: number;  
    → get editor(): string {  
        // custom getter logic goes here, should return a value  
    }  
    → set editor(newEditor: string) {  
        // custom setter logic goes here  
    }  
    printChapterTitle(chapterNum: number): void {  
        // print title here  
    }  
}
```

```
class ReferenceItem {  
→ constructor(title: string, publisher?: string) {  
    // perform initialization here ↑  
}  
}  
↓  
let encyclopedia = new ReferenceItem('WorldPedia', 'WorldPub');
```

## Constructors

Method named “constructor” – maximum of one per class

Use optional parameters to call different ways

Executed by using the “new” keyword

Sound Familiar?

**Define Types**

**Properties and  
Methods**

**Constructors**

**Access Modifiers**

**Inheritance**

**Abstract Classes**

What is a class?

**Template for creating objects**

**Provides state storage and behavior**

**Encapsulates reusable functionality**



# Overview



**What is a class?**

**Similarity to classes in other languages**

**Class members**

- Constructors
- Properties
- Methods

**Inheritance**

**Abstract classes**

**Class expressions**