البيت السلام

BAITUSSALAM

BAITUSSALAM TECH PARK

Class Agenda

BAITUSSALAM
—TECH PARK—

Object Oriented Programming, Constructor Function, Class with Typescript, this, new and super keyword

Object Oriented Programming (OOP)

JavaScript is prototype-based procedural language, which means it supports both functional and object-oriented programming.

Object-Oriented Programming (OOP) is a programming paradigm that uses objects to represent real-world entities and manage their interactions. In JavaScript, objects are collections of properties and methods. Properties are values associated with an object, and methods are functions that operate on these properties.

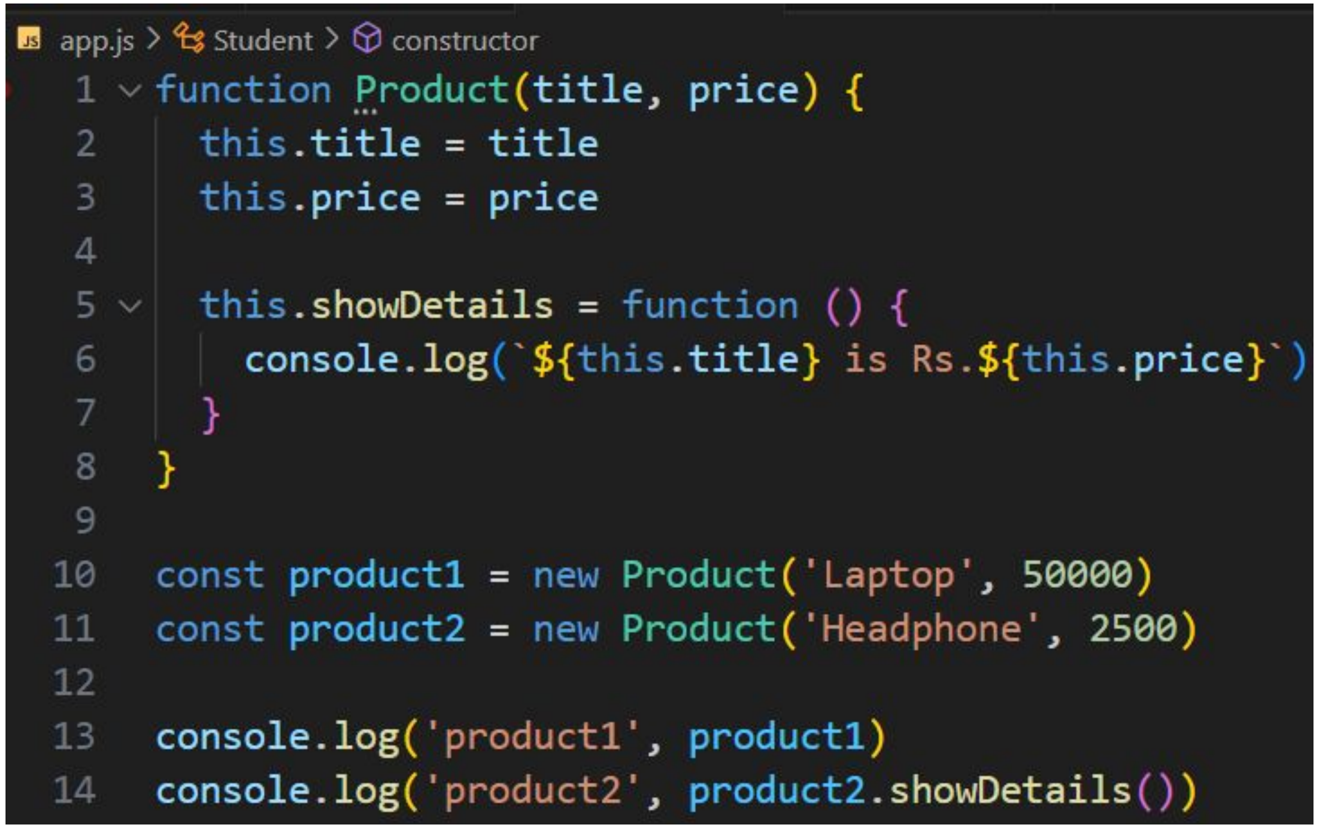
Functional Programming

Functional programming (FP) in JavaScript is all about treating your code as a set of independent, reusable functions that operate on data without side effects.

By understanding first-class functions, higher-order functions, immutability, pure functions, and using tools like map, filter, and reduce, you can write more readable, maintainable, and bug-free code.

Constructor Function

A **constructor function** in JavaScript is a regular function that is used to create and initialize objects. When you use the **new keyword** with a constructor function, it creates a new object, sets the this context within the constructor to that new object, and returns the new object implicitly.



```javascript
function Product(title, price) {
  this.title = title
  this.price = price

  this.showDetails = function () {
    console.log(`${this.title} is Rs.${this.price}`)
  }
}

const product1 = new Product('Laptop', 50000)
const product2 = new Product('Headphone', 2500)

console.log('product1', product1)
console.log('product2', product2.showDetails())
```

Class

In JavaScript, a **class** is like a **blueprint for creating objects**.

A class is not a real world object but we can create objects from a class. It is like an template for an object.

It allows you to define a set of properties and methods, and instantiate (or create) new objects with those properties and methods.

The class keyword is used to declare a class. Here is an example of declaring a Product class:

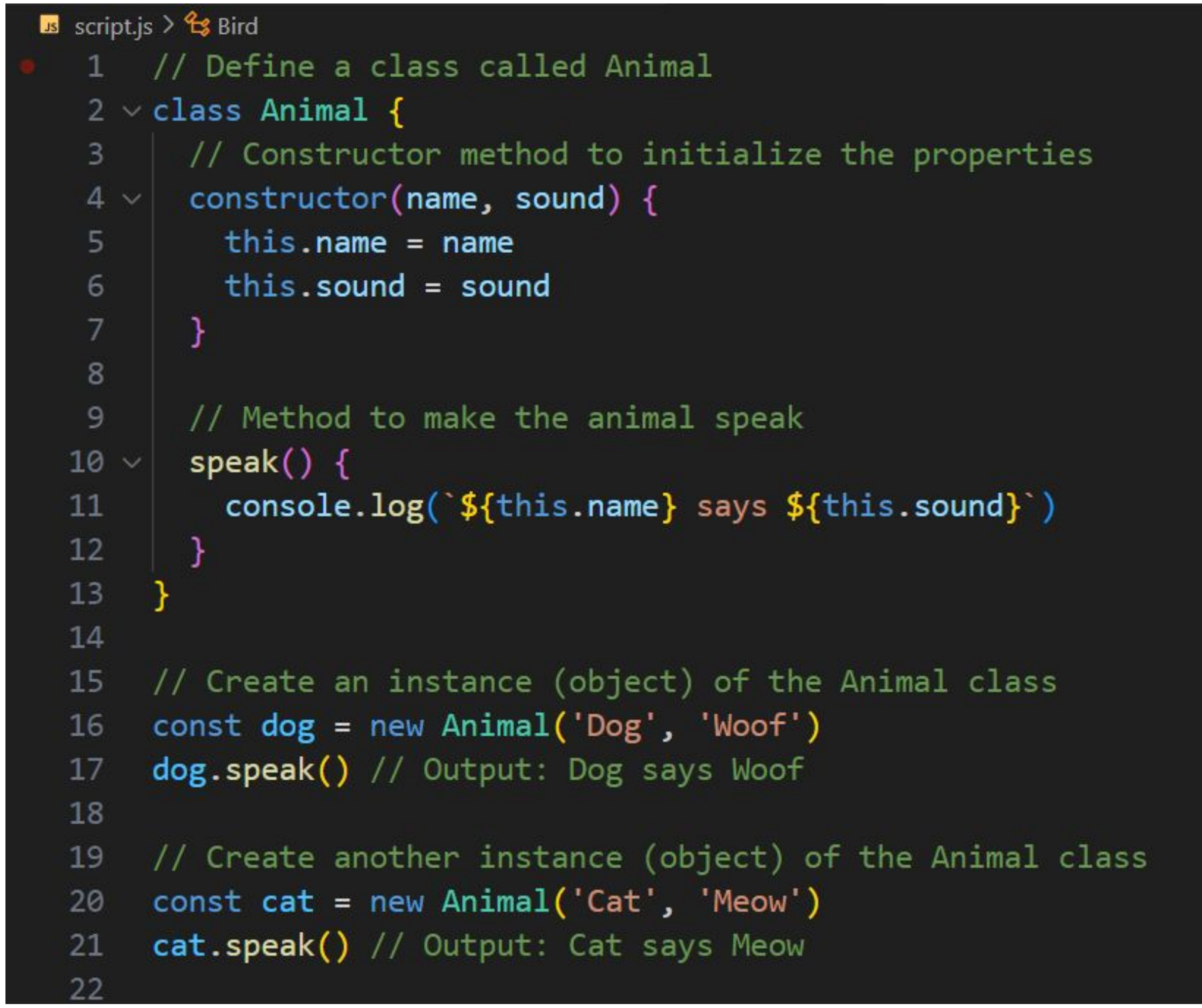
```
class Product {}
```

Constructor Method

Classes have a special constructor method, which is called when a new instance of the class is created.

The **constructor** method is a great place to initialize properties of the class.

Here is an example of a class with a constructor method:



```javascript
// Define a class called Animal
class Animal {
  // Constructor method to initialize the properties
  constructor(name, sound) {
    this.name = name
    this.sound = sound
  }

  // Method to make the animal speak
  speak() {
    console.log(`${this.name} says ${this.sound}`)
  }
}

// Create an instance (object) of the Animal class
const dog = new Animal('Dog', 'Woof')
dog.speak() // Output: Dog says Woof

// Create another instance (object) of the Animal class
const cat = new Animal('Cat', 'Meow')
cat.speak() // Output: Cat says Meow

```

this keyword

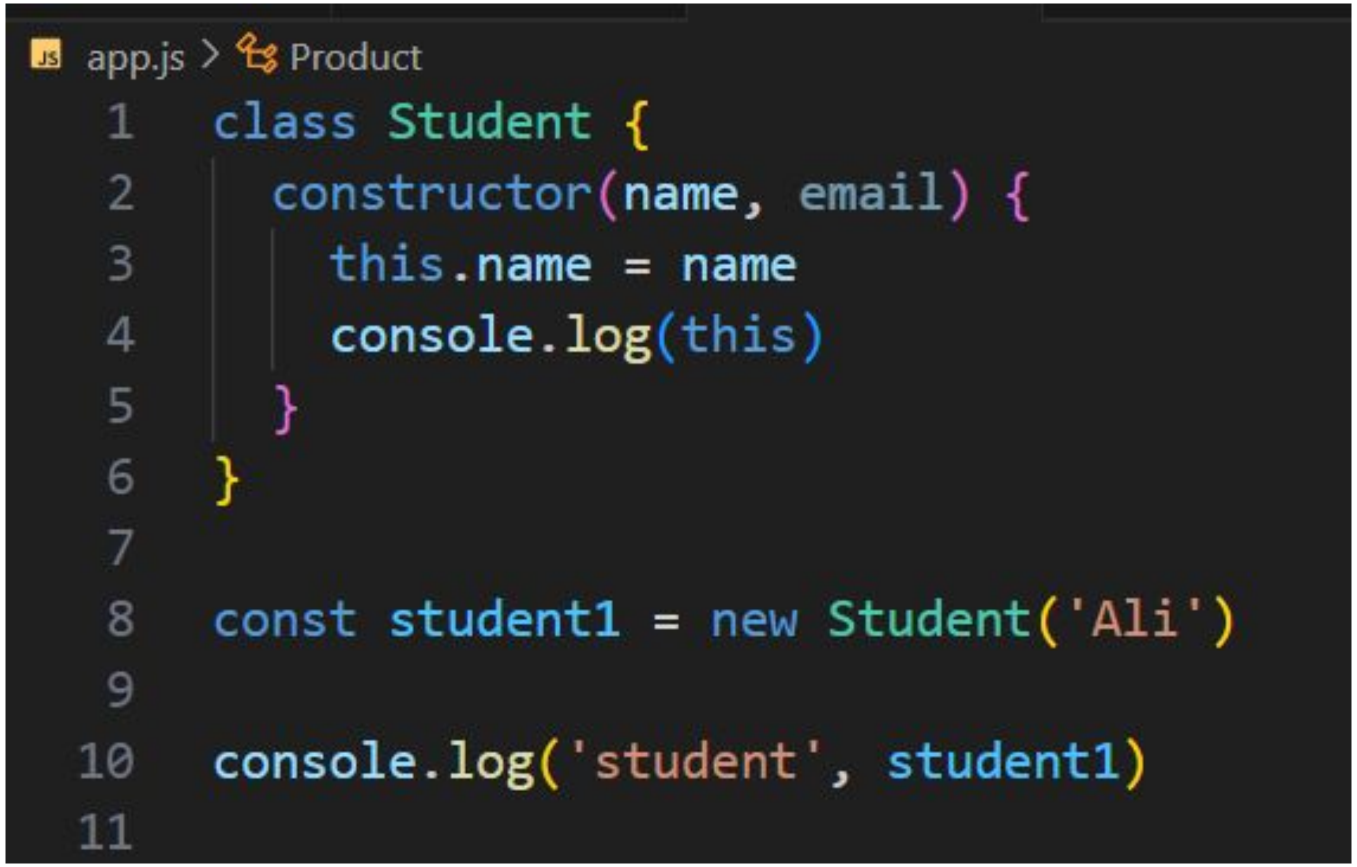
The **this** keyword in JavaScript is used to refer to the current object.

Depending on where this is used

In the case of a **class**, it refers to the instance of the **object being constructed**.

You can use the **this** keyword to set the properties of the object being instantiated.

Here is an example:


app.js > Product

```
class Student {
  constructor(name, email) {
    this.name = name
    console.log(this)
  }
}

const student1 = new Student('Ali')

console.log('student', student1)

```

new keyword

The new keyword in JavaScript is used to create an instance of a user-defined object type or one of the built-in object types that has a constructor function. When you use new, it performs the following actions:

1. Creates a New Object: It creates a new, empty object.
2. Sets the Prototype: It sets the prototype of the new object to the constructor's prototype property.
3. Binds this: It binds this to the new object within the constructor function, allowing you to add properties and methods to the new object.
4. Returns the Object: It returns the new object, unless the constructor function returns a non-primitive value explicitly.

Class Exercise #1

Create a Class:
1. Define a class named **Book**.
2. The constructor method of Book should take two parameters: **title and author**, and assign them to properties of the same name.

Add a Method:
1. Add a method called **getDetails** to the Book class.
2. This method should return a string that includes both the title and author of the book.

Create Instances:
Create two instances of the Book class, each with different title and author values.

Class Exercise #2

Define a Class:
Create a class named **Library**.

Add Methods:

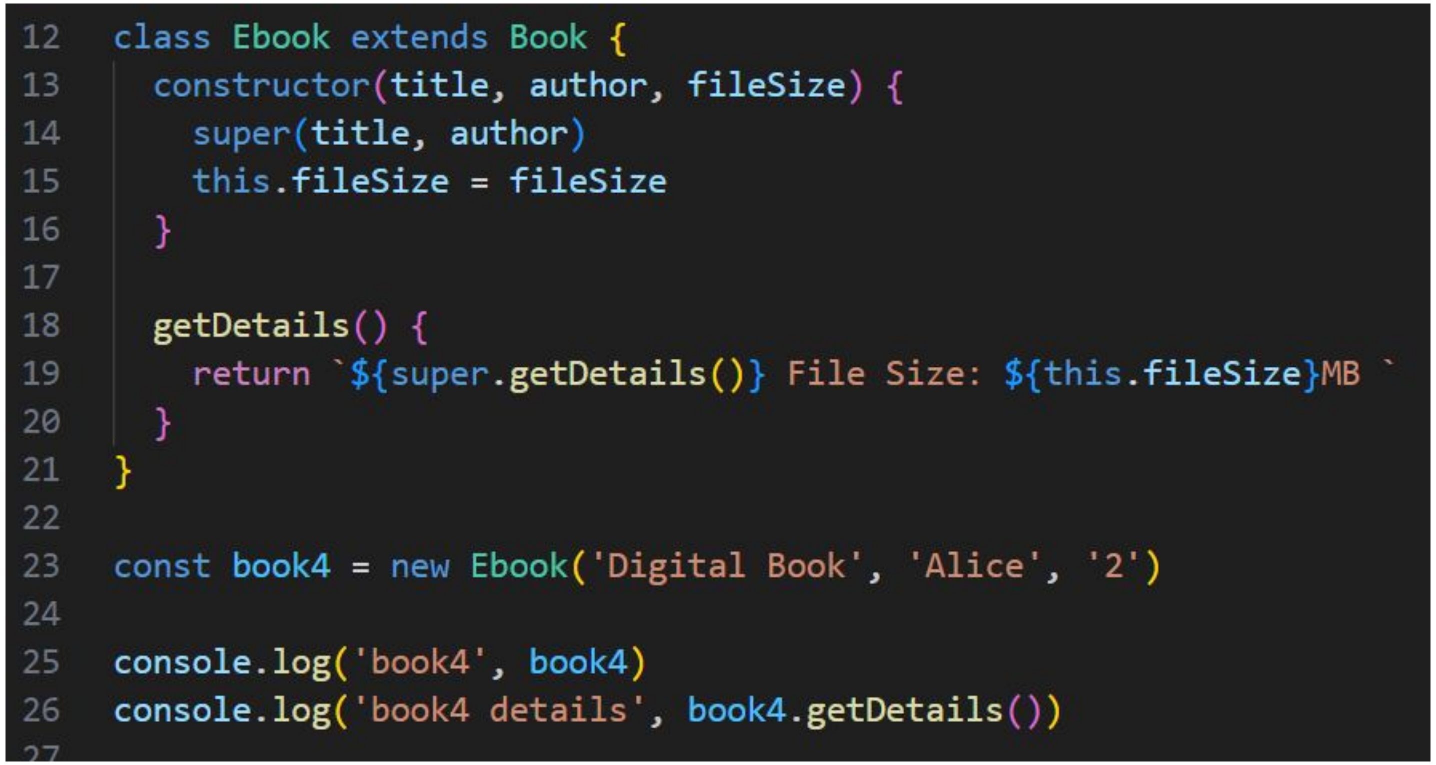
- Add a method called **addBook** that takes a Book object as a parameter and adds it to the library's collection.
- Add a method called **removeBook** that takes a title as a parameter and removes the corresponding book from the library's collection.
- Add a method called **findBook** that takes a title as a parameter and returns the corresponding Book object from the library's collection.
- Add a method called **getBookList** that returns a list of all the books currently in the library.

Class Inheritance

Class inheritance in JavaScript allows one class to inherit the properties and methods of another class.

To create a class inheritance, use the **extends** keyword.

The **super** keyword is used to call the constructor of its parent class to access the parent's properties and methods.



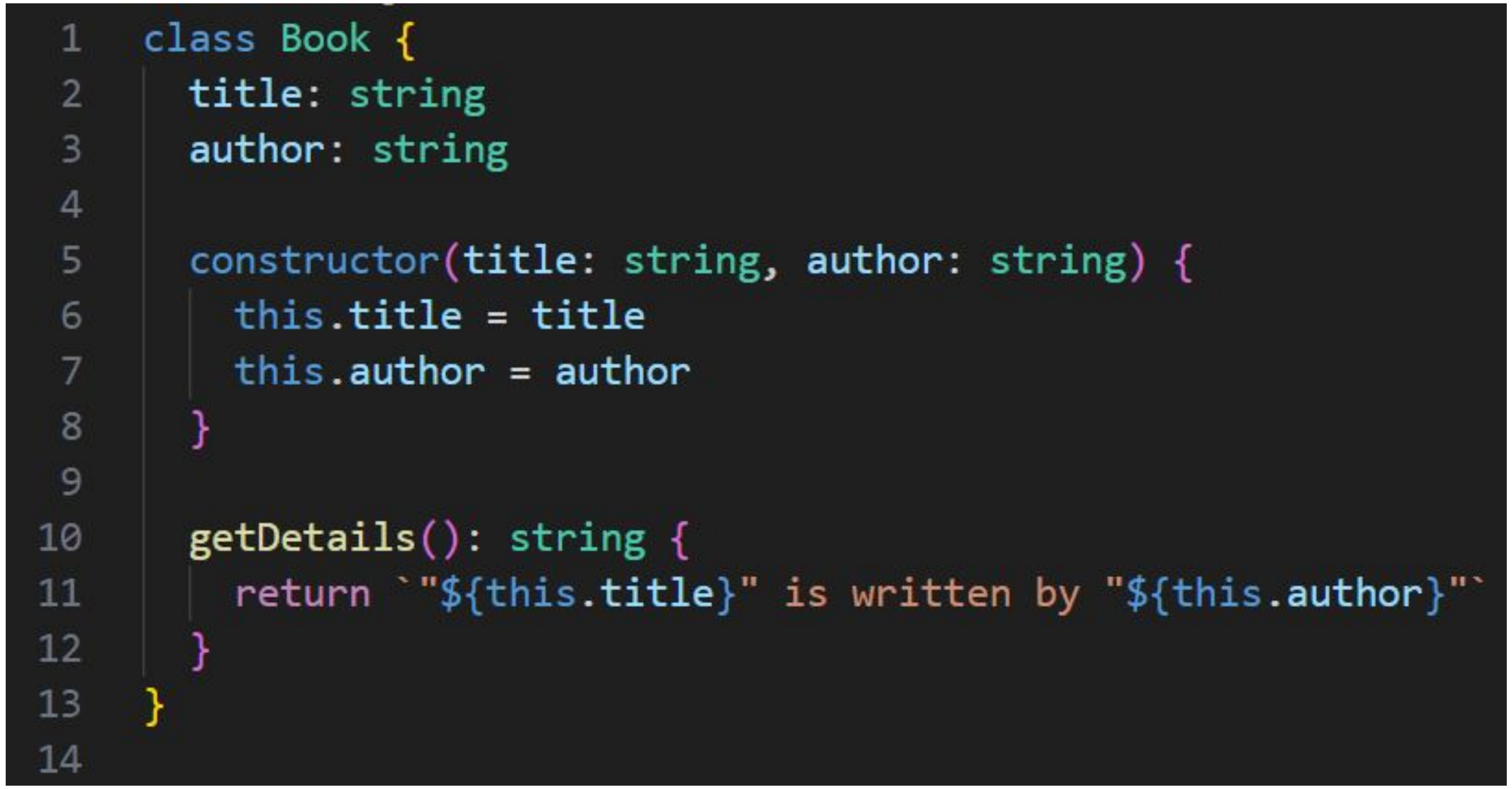
```javascript
class Ebook extends Book {
  constructor(title, author, fileSize) {
    super(title, author)
    this.fileSize = fileSize
  }

  getDetails() {
    return `${super.getDetails()} File Size: ${this.fileSize}MB `
  }
}

const book4 = new Ebook('Digital Book', 'Alice', '2')

console.log('book4', book4)
console.log('book4 details', book4.getDetails())
```

Classes with Typescript



```
class Book {
  title: string
  author: string

  constructor(title: string, author: string) {
    this.title = title
    this.author = author
  }

  getDetails(): string {
    return `"${this.title}" is written by "${this.author}"`
  }
}

```

The End

BAITUSSALAM
—TECH PARK—