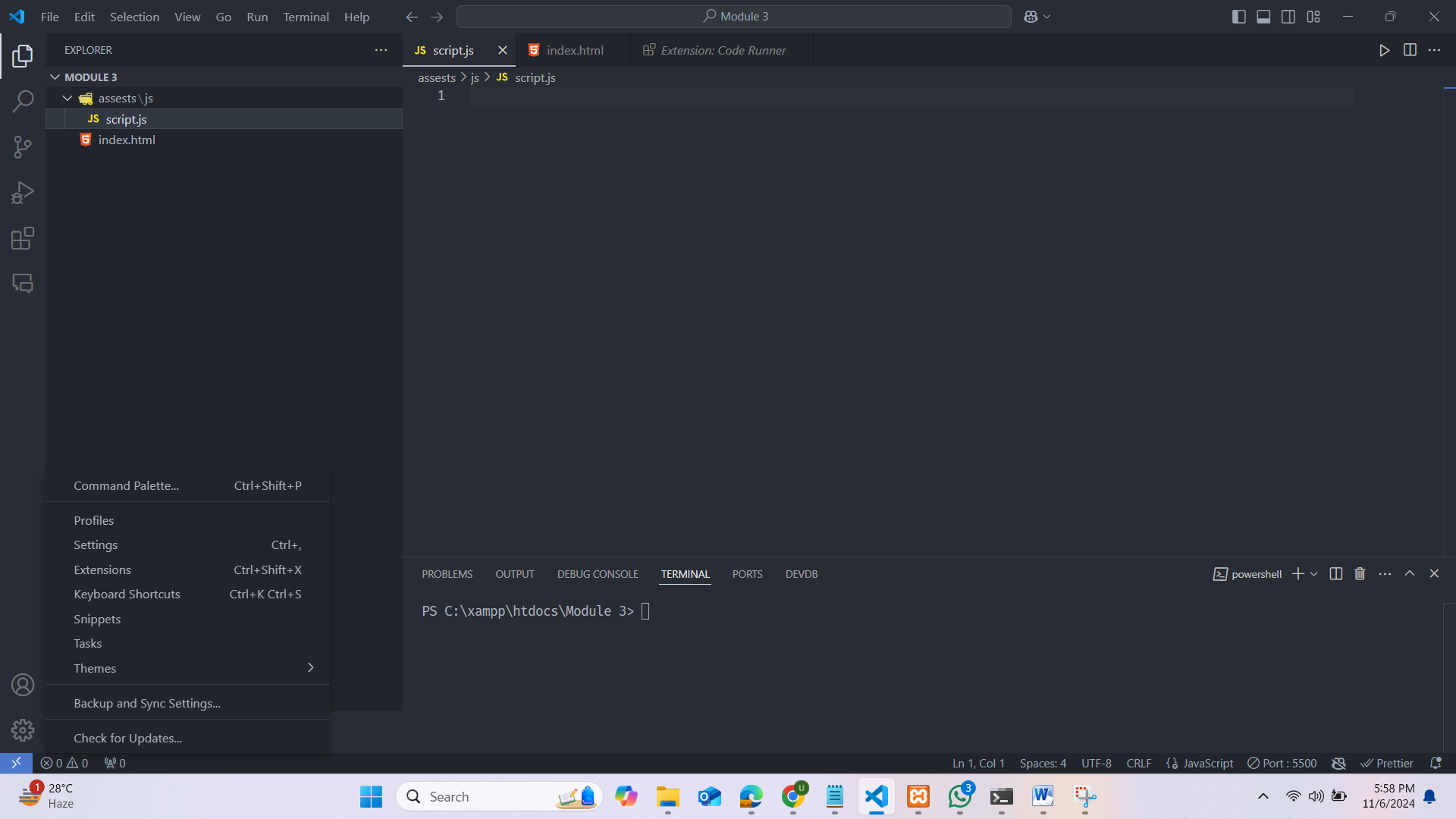
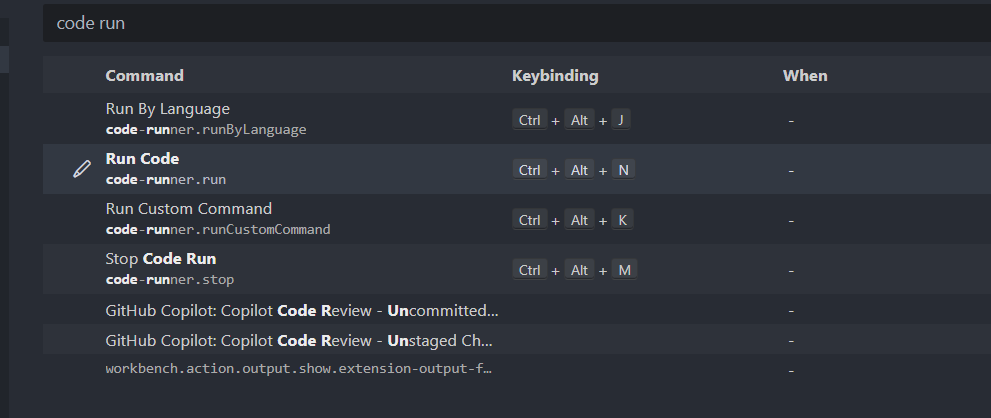
//**Install Code Runner For Running Javascript**

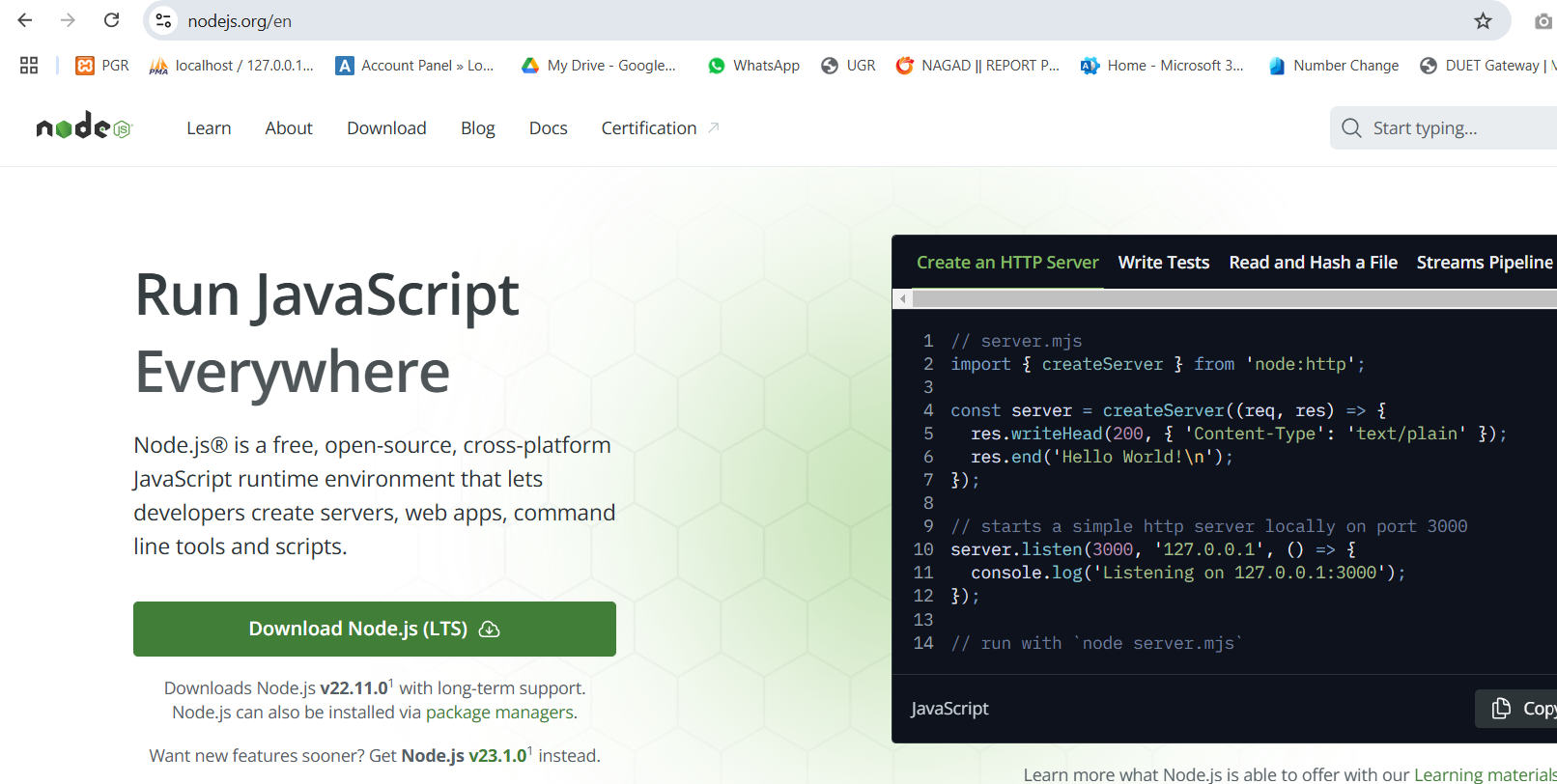






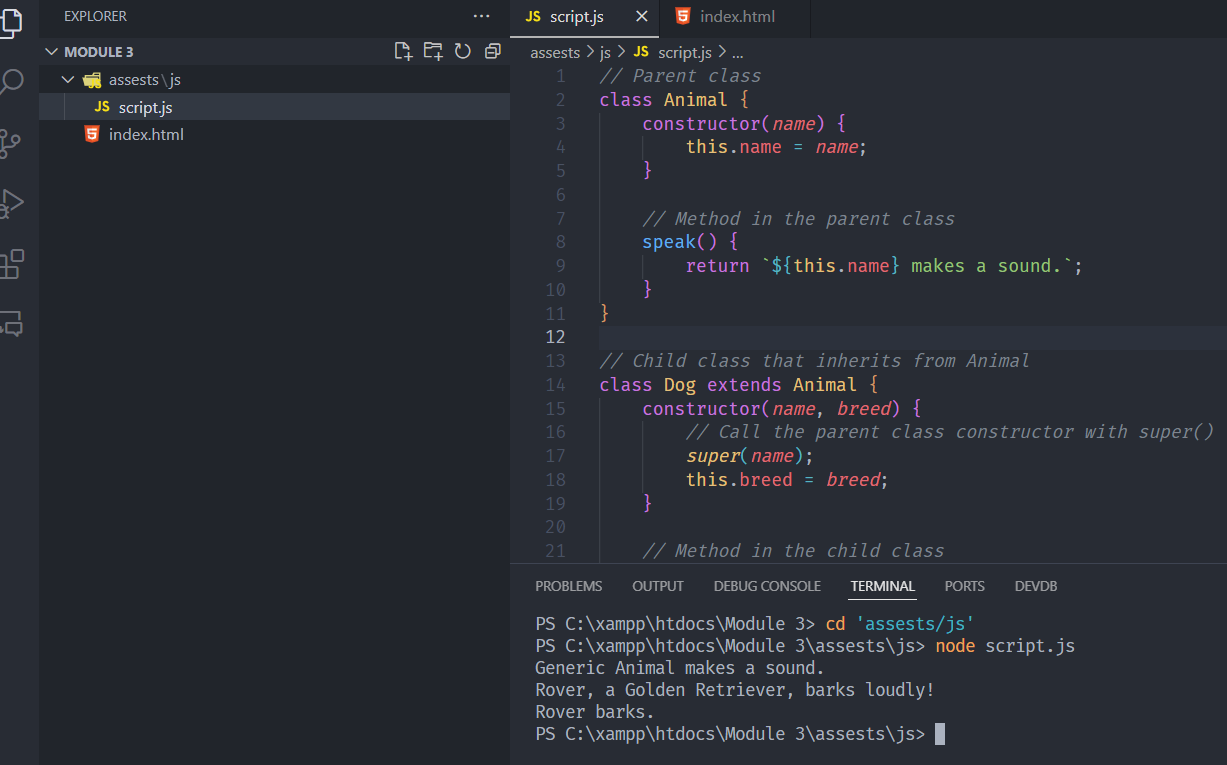
**Code Run Using Node JS**

**১.আমরা node js install করব।**

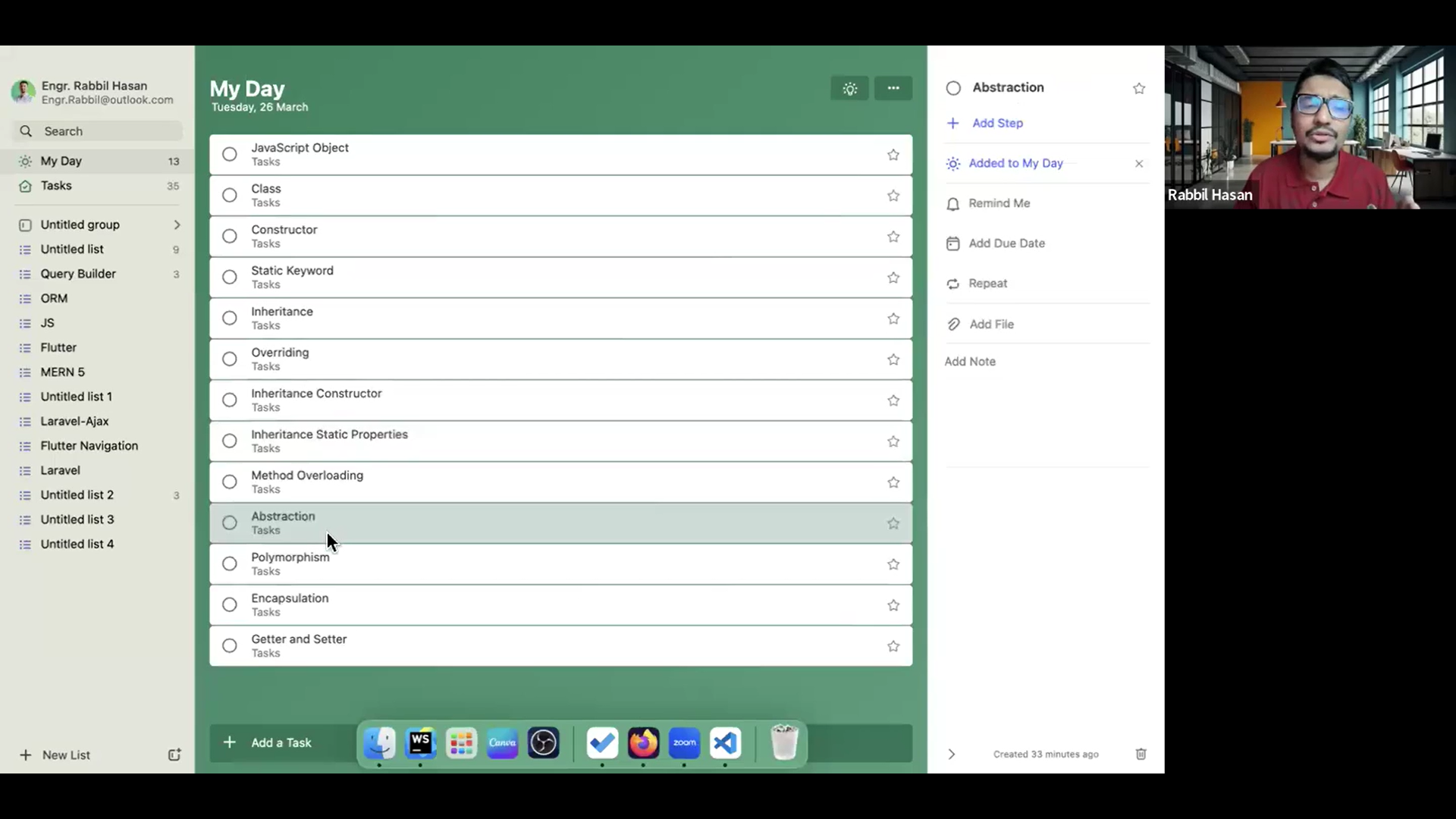
****

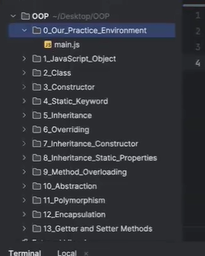
**২.এরপর আমরা নির্দিষ্ট ফোল্ডারে যাব**

**৩.node filename.js দিলে রান হবে।**

****

**Topics**





**1.Object(সরাসরি Building তৈরি করব।)**

(এখানে Object আর Class একই না।)

In JavaScript, an object is a collection of key-value pairs.(const রাখব সবসময়।)

এখানে একটা অবজেক্ট , এখানে ক্লাস create করে আলাদা আলাদা object create করা যাবে না।একটা অবজেক্ট থাকবে তার মধ্যে বিভিন্ন property থাকবে।

Dot(.) দিয়ে অবজেক্টের ভ্যালু access করা যাবে।

const Person = {

    firstName: 'Utsab',

    lastName: 'Roy',

    email: 'utsab@duet.ac.bd',

    fullName: function () {

        return this.firstName + ' ' + this.lastName;

*//return `${this.firstName} ${lastName}`;*

    },

    greet() {

        return `Welcome ${this.firstName}`;

    }

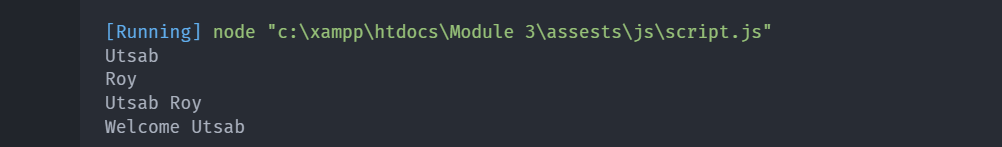
}

console.log(Person.firstName);

console.log(Person.lastName);

console.log(Person.fullName());

console.log(Person.greet());



**আমরা চাইলে যে কোন pre-defined Object/আমাদের Object তৈরি হওয়ার পরেও নতুন মেথড তৈরি করতে পারি:**

const Person = {

    firstName: 'Utsab',

    lastName: 'Roy',

    email: 'utsab@duet.ac.bd',

    fullName: function () {

        return this.firstName + ' ' + this.lastName;

*//return `${this.firstName} ${lastName}`;*

    }

}

Person.allDataJoin = function () {

    return `${this.firstName} ${this.lastName} ${this.email}`;

}

console.log(Person.allDataJoin());

**By Creating Instance Of Object(আগে Object তৈরি হবে না বরং .(ডট) দিয়ে Object এর property add করব।)**

let Person = new Object();

Person.firstName = "utsab";

Person.lastName = "Roy";

Person.email = "utsab@duet.ac.bd";

Person.fullName = function () {

    return this.firstName + " " + this.lastName;

}

console.log(Person);

console.log(Person.fullName());

**Object inside Object**

const university = {

    name: "DUET",

    location: "Gazipur",

*// Nested department object*

    department: {

        name: "Computer Science and Engineering",

        head: "Dr. A. Rahman",

*// Further nested object for courses*

        courses: {

            course1: { title: "Data Structures", code: "CSE201" },

            course2: { title: "Algorithms", code: "CSE202" },

            getCourseList() {

                return `${this.course1.title} (${this.course1.code}),

                          ${this.course2.title} (${this.course2.code})`;

            }

        }

    },

    getDepartmentInfo() {

        return `${this.department.name} - Head: ${this.department.head}`;

    }

};

console.log(university.name);                        *// Outputs: "DUET"*

console.log(university.department.name);             *// Outputs: "Computer Science and Engineering"*

console.log(university.department.courses.course1);  *// Outputs: { title: "Data Structures", code: "CSE201" }*

console.log(university.department.courses.getCourseList());  *// Outputs: "Data Structures (CSE201), Algorithms (CSE202)"*

console.log(university.getDepartmentInfo());         *// Outputs: "Computer Science and Engineering - Head: Dr. A. Rahman"*

**2.CLASS(আগে নকশা তৈরি করব এরপর Building তৈরি করব।)**

You can define a class using the class keyword. Inside the class, you typically have (constructor এর মধ্যে this দিয়ে যা declare করা হবে তাই class variable)

* A **constructor**: A special method used to initialize properties when creating an instance.
* Other **methods**: Functions that define behaviors of the objects created from the class.

class Person {

    constructor(*name*,*email*) {

        this.name = *name*;

        this.email = *email*;

    }

    fullAddress() {

        return `${this.name} ${this.email}`;

     }

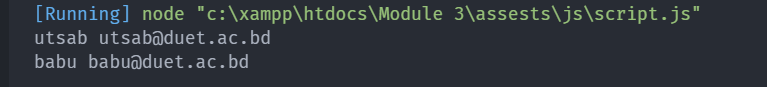
}

let obj1 = new Person('utsab','utsab@duet.ac.bd');

console.log(obj1.fullAddress());

let obj2 = new Person('babu','babu@duet.ac.bd');

console.log(obj2.fullAddress());



**Inheritance**

*// Parent class*

class Animal {

    constructor(*name*) {

        this.name = *name*;

    }

*// Method in the parent class*

    speak() {

        return `${this.name} makes a sound.`;

    }

}

*// Child class that inherits from Animal*

class Dog extends Animal {

    constructor(*name*, *breed*) {

*// Call the parent class constructor with super()*

*super*(*name*);

        this.breed = *breed*;

    }

*// Method in the child class*

    bark() {

        return `${this.name} barks.`;

    }

*// Overriding the parent class's speak method*

    speak() {

        return `${this.name}, a ${this.breed}, barks loudly!`;

    }

}

*// Creating instances*

const genericAnimal = new Animal("Generic Animal");

console.log(genericAnimal.speak());         *// Outputs: "Generic Animal makes a sound."*

const dog = new Dog("Rover", "Golden Retriever");

console.log(dog.speak());                   *// Outputs: "Rover, a Golden Retriever, barks loudly!"*

console.log(dog.bark());                    *// Outputs: "Rover barks."*

Another Example:

class Person {

    constructor(*name*,*age*) {

        this.name = *name*;

        this.age = *age*;

    }

*//no need to mention funtion keywork*

    getName() {

        return this.name;

    }

    getAge() {

        return this.age;

    }

}

class Teacher extends Person {

    constructor(*name*,*age*,*work*) {

*super*(*name*, *age*);

        this.work = *work*;

    }

    getWork() {

        return this.work;

    }

}

let objPerson = new Person("utsab", 10);

console.log(objPerson.getName());

console.log(objPerson.getAge());

let objTeacher = new Teacher("babu", 20, "DUET");

console.log(objTeacher.getName());

console.log(objTeacher.getAge());

console.log(objTeacher.getWork());

