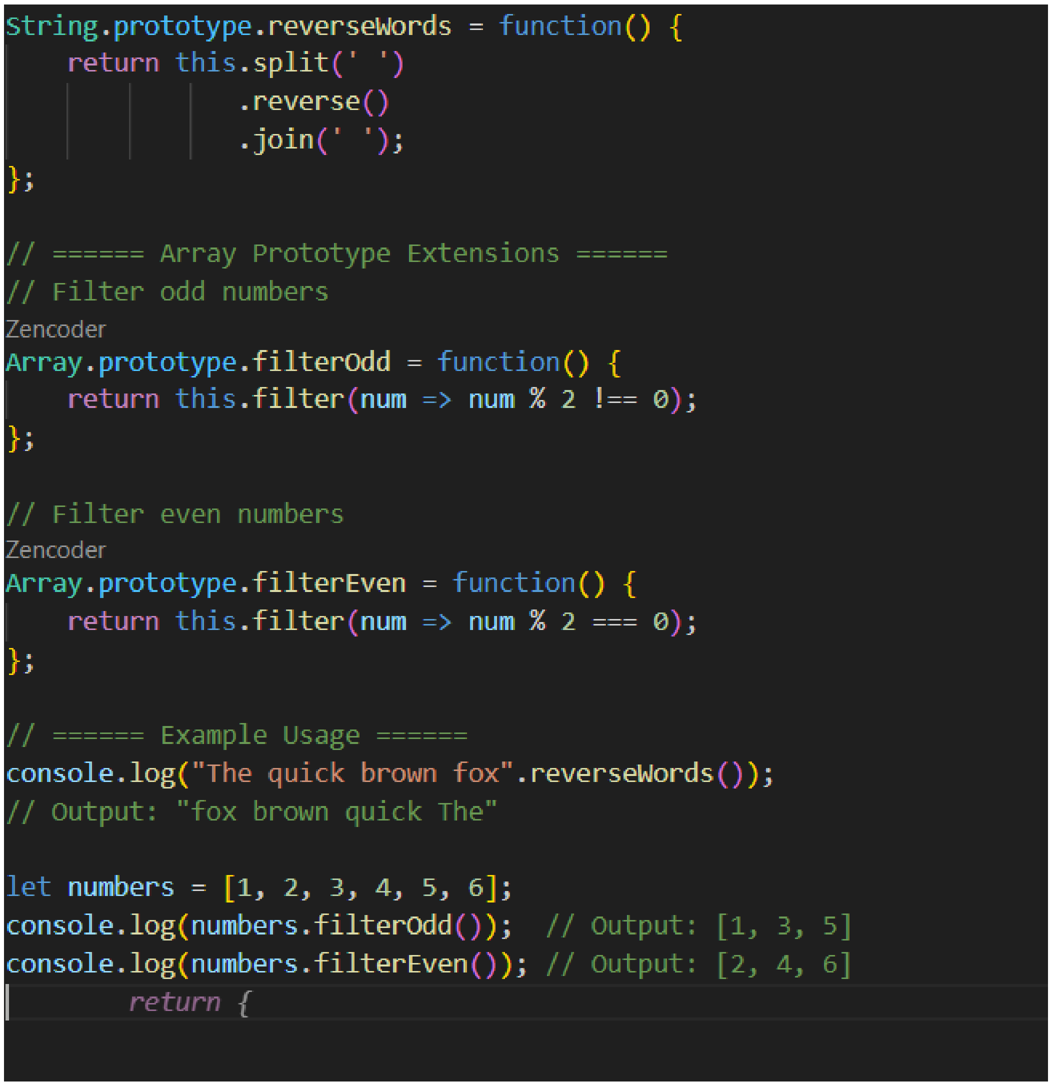
Practical 6

Problem Definition

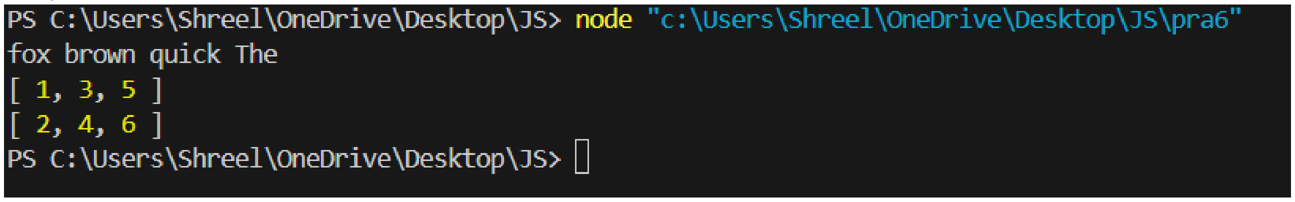
You're optimizing an existing JavaScript application by adding functionality without modifying original constructors. Use the prototype property to add a method to built-in Array or String constructors.

Code



```
String.prototype.reverseWords = function() {
    return this.split(' ')
            .reverse()
            .join(' ');
};

// ====== Array Prototype Extensions ======
// Filter odd numbers
Zencoder
Array.prototype.filterOdd = function() {
    return this.filter(num => num % 2 !== 0);
};

// Filter even numbers
Zencoder
Array.prototype.filterEven = function() {
    return this.filter(num => num % 2 === 0);
};

// ====== Example Usage ======
console.log("The quick brown fox".reverseWords());
// Output: "fox brown quick The"

let numbers = [1, 2, 3, 4, 5, 6];
console.log(numbers.filterOdd());  // Output: [1, 3, 5]
console.log(numbers.filterEven()); // Output: [2, 4, 6]
        return {
```

Output:

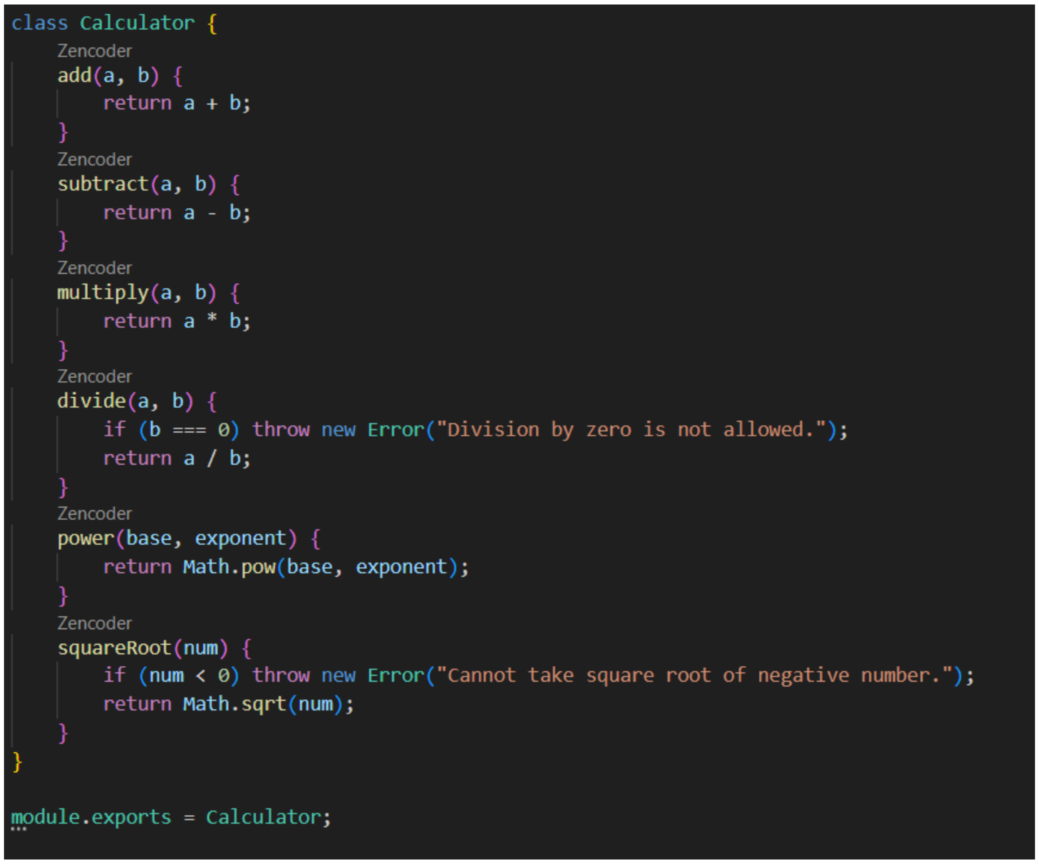


```
PS C:\Users\Shreel\OneDrive\Desktop\JS> node "c:\Users\Shreel\OneDrive\Desktop\JS\pra6"
fox brown quick The
[ 1, 3, 5 ]
[ 2, 4, 6 ]
PS C:\Users\Shreel\OneDrive\Desktop\JS> 
```

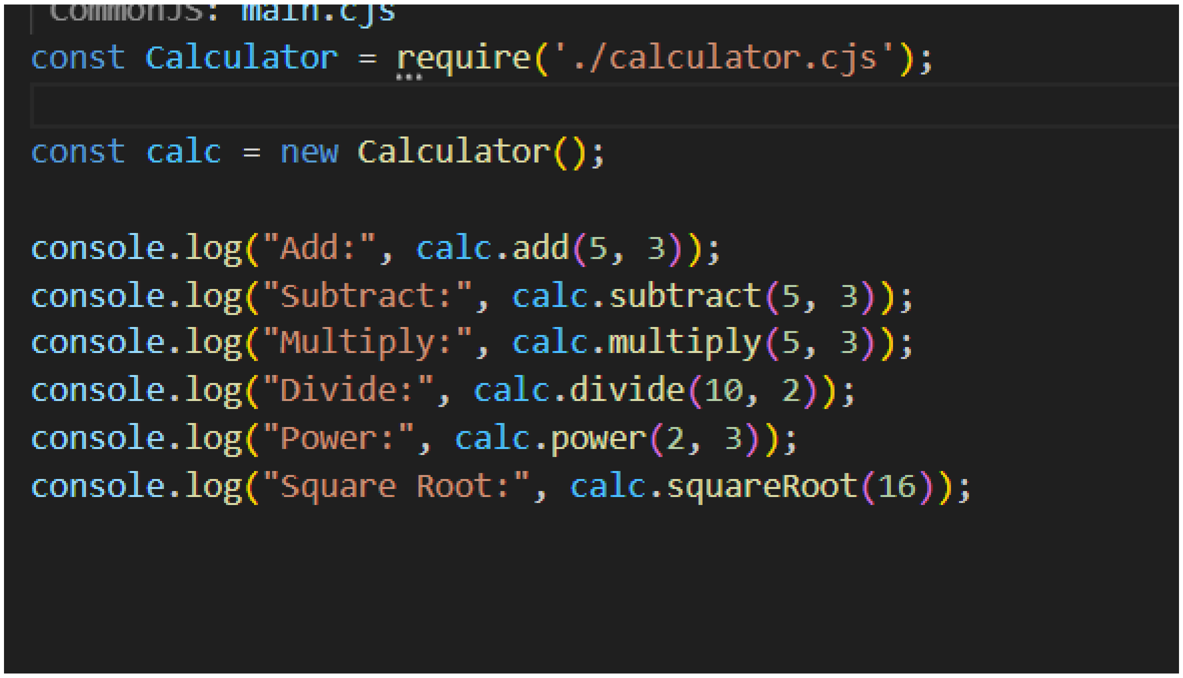
Practical 7

Problem Definition Develop a calculator module that encapsulates arithmetic operations. Export the class and import it into another module for execution. Key Questions to be evaluated during/after Implementation

Code

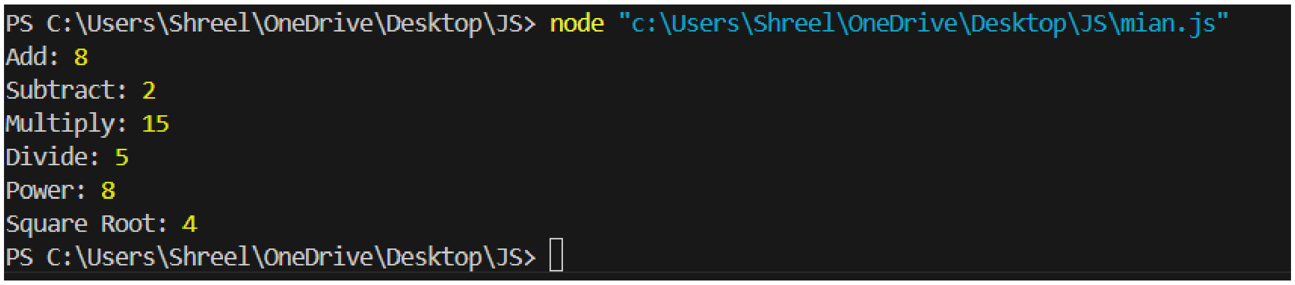


```
class Calculator {
    Zencoder
    add(a, b) {
        return a + b;
    }
    Zencoder
    subtract(a, b) {
        return a - b;
    }
    Zencoder
    multiply(a, b) {
        return a * b;
    }
    Zencoder
    divide(a, b) {
        if (b === 0) throw new Error("Division by zero is not allowed.");
        return a / b;
    }
    Zencoder
    power(base, exponent) {
        return Math.pow(base, exponent);
    }
    Zencoder
    squareRoot(num) {
        if (num < 0) throw new Error("Cannot take square root of negative number.");
        return Math.sqrt(num);
    }
}

module.exports = Calculator;
```



```
CommonJS: main.cjs
const Calculator = require('./calculator.cjs');

const calc = new Calculator();

console.log("Add:", calc.add(5, 3));
console.log("Subtract:", calc.subtract(5, 3));
console.log("Multiply:", calc.multiply(5, 3));
console.log("Divide:", calc.divide(10, 2));
console.log("Power:", calc.power(2, 3));
console.log("Square Root:", calc.squareRoot(16));
```

Output"



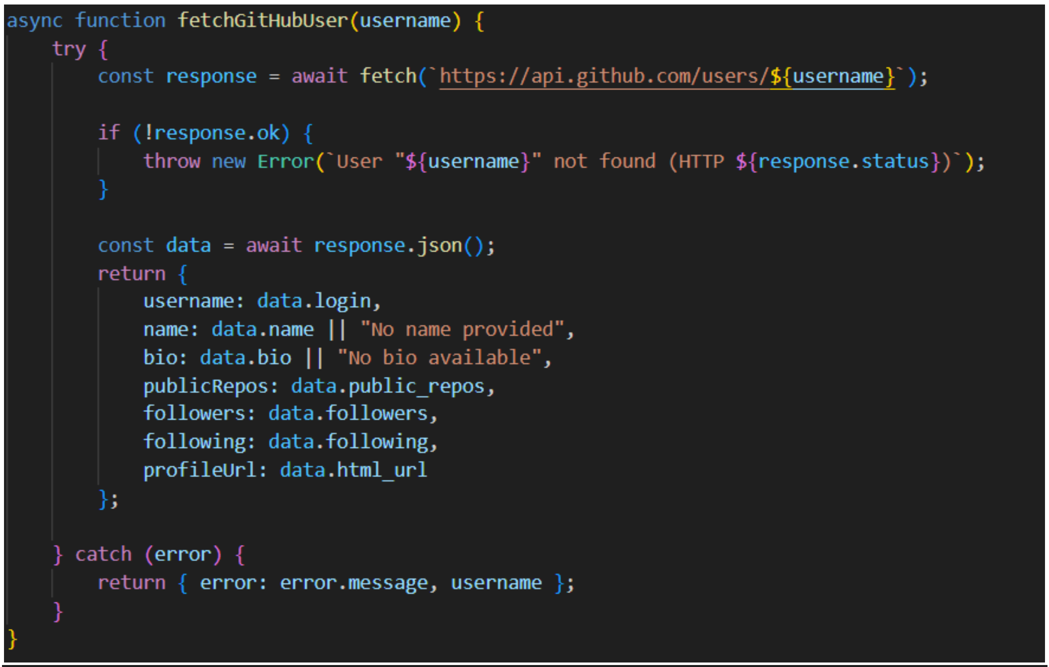
```
PS C:\Users\Shreel\OneDrive\Desktop\JS> node "c:\Users\Shreel\OneDrive\Desktop\JS\mian.js"
Add: 8
Subtract: 2
Multiply: 15
Divide: 5
Power: 8
Square Root: 4
PS C:\Users\Shreel\OneDrive\Desktop\JS>
```

Practical 8

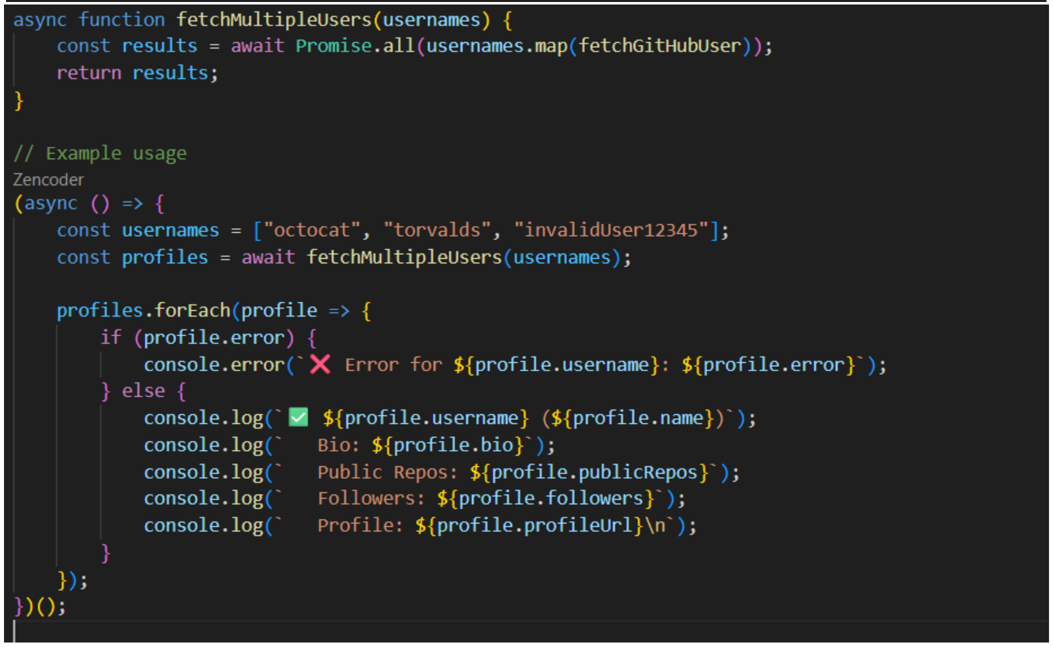
Problem Definition

You are tasked with building a GitHub profile viewer that fetches user data asynchronously using the fetch API. Handle multiple user requests and fail-safe error handling.

Code

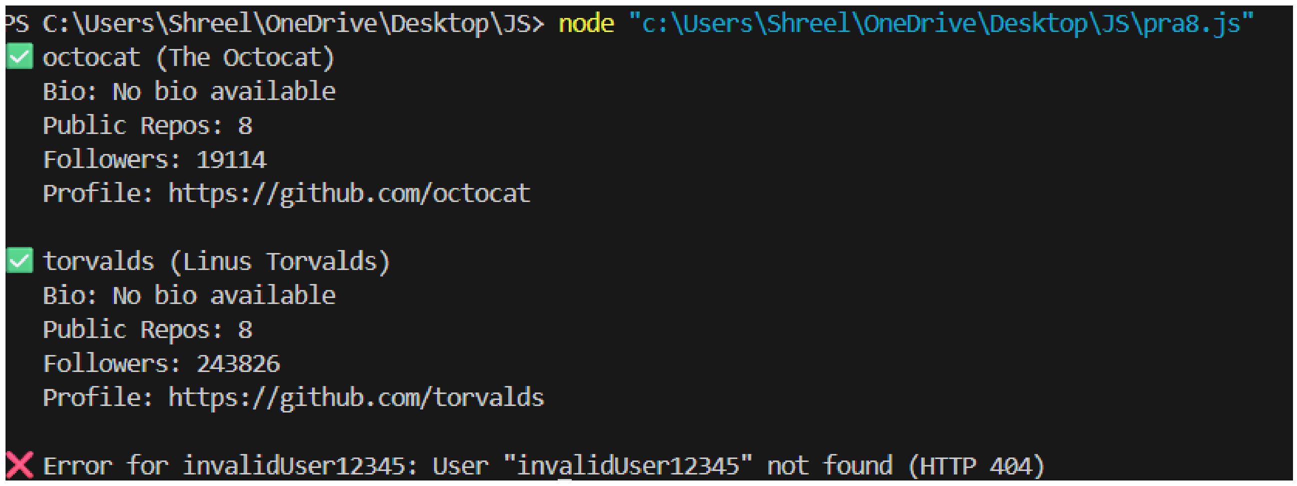


```javascript
async function fetchGitHubUser(username) {
    try {
        const response = await fetch(`https://api.github.com/users/${username}`);

        if (!response.ok) {
            throw new Error(`User "${username}" not found (HTTP ${response.status})`);
        }

        const data = await response.json();
        return {
            username: data.login,
            name: data.name || "No name provided",
            bio: data.bio || "No bio available",
            publicRepos: data.public_repos,
            followers: data.followers,
            following: data.following,
            profileUrl: data.html_url
        };

    } catch (error) {
        return { error: error.message, username };
    }
}
```



```javascript
async function fetchMultipleUsers(usernames) {
    const results = await Promise.all(usernames.map(fetchGitHubUser));
    return results;
}

// Example usage
Zencoder
(async () => {
    const usernames = ["octocat", "torvalds", "invalidUser12345"];
    const profiles = await fetchMultipleUsers(usernames);

    profiles.forEach(profile => {
        if (profile.error) {
            console.error(`❌ Error for ${profile.username}: ${profile.error}`);
        } else {
            console.log(`✅ ${profile.username} (${profile.name})`);
            console.log(`   Bio: ${profile.bio}`);
            console.log(`   Public Repos: ${profile.publicRepos}`);
            console.log(`   Followers: ${profile.followers}`);
            console.log(`   Profile: ${profile.profileUrl}\n`);
        }
    });
})();
```

Output:



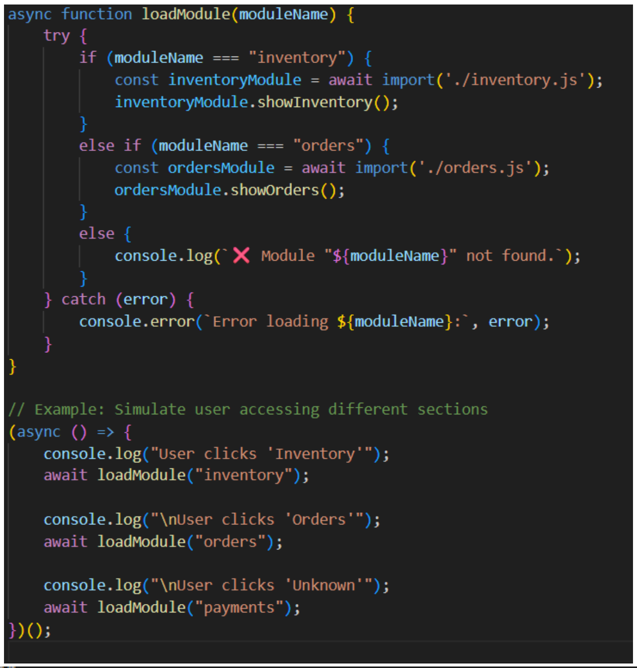
```
PS C:\Users\Shreel\OneDrive\Desktop\JS> node "c:\Users\Shreel\OneDrive\Desktop\JS\pra8.js"
✅ octocat (The Octocat)
   Bio: No bio available
   Public Repos: 8
   Followers: 19114
   Profile: https://github.com/octocat

✅ torvalds (Linus Torvalds)
   Bio: No bio available
   Public Repos: 8
   Followers: 243826
   Profile: https://github.com/torvalds

❌ Error for invalidUser12345: User "invalidUser12345" not found (HTTP 404)
```

Practical 9

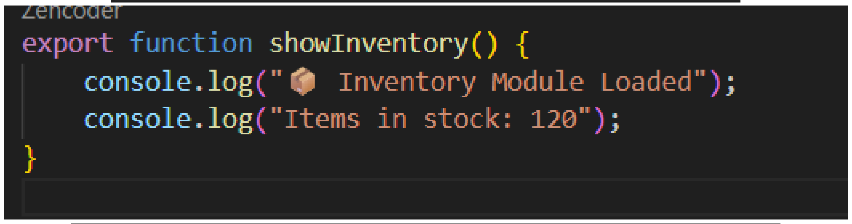
Problem Definition

In an e-commerce dashboard, different modules need to be loaded only when accessed (e.g., inventory vs orders). Use import() to dynamically load modules.

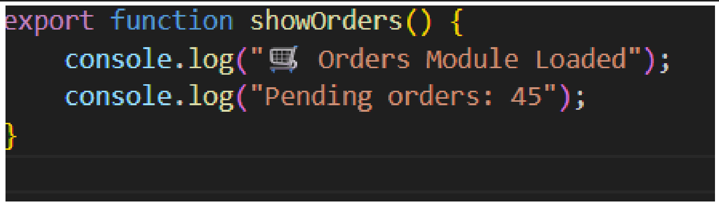
Code



```
async function loadModule(moduleName) {
    try {
        if (moduleName === "inventory") {
            const inventoryModule = await import('./inventory.js');
            inventoryModule.showInventory();
        }
        else if (moduleName === "orders") {
            const ordersModule = await import('./orders.js');
            ordersModule.showOrders();
        }
        else {
            console.log(`❌ Module "${moduleName}" not found.`);
        }
    } catch (error) {
        console.error(`Error loading ${moduleName}:`, error);
    }
}

// Example: Simulate user accessing different sections
(async () => {
    console.log("User clicks 'Inventory'");
    await loadModule("inventory");

    console.log("\nUser clicks 'Orders'");
    await loadModule("orders");

    console.log("\nUser clicks 'Unknown'");
    await loadModule("payments");
})();
```

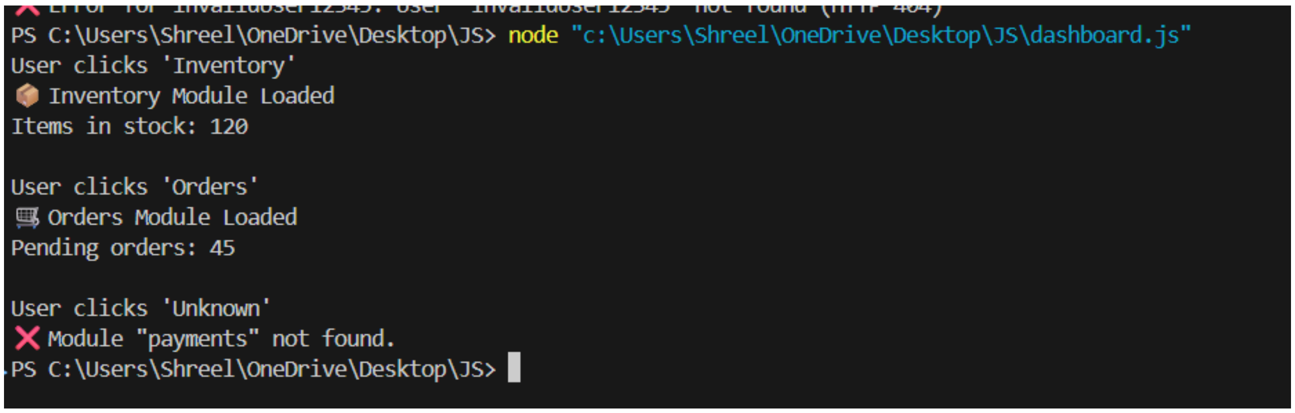


```
export function showInventory() {
    console.log("📦 Inventory Module Loaded");
    console.log("Items in stock: 120");
}
```



```
export function showOrders() {
    console.log("🛒 Orders Module Loaded");
    console.log("Pending orders: 45");
}
```

Output:



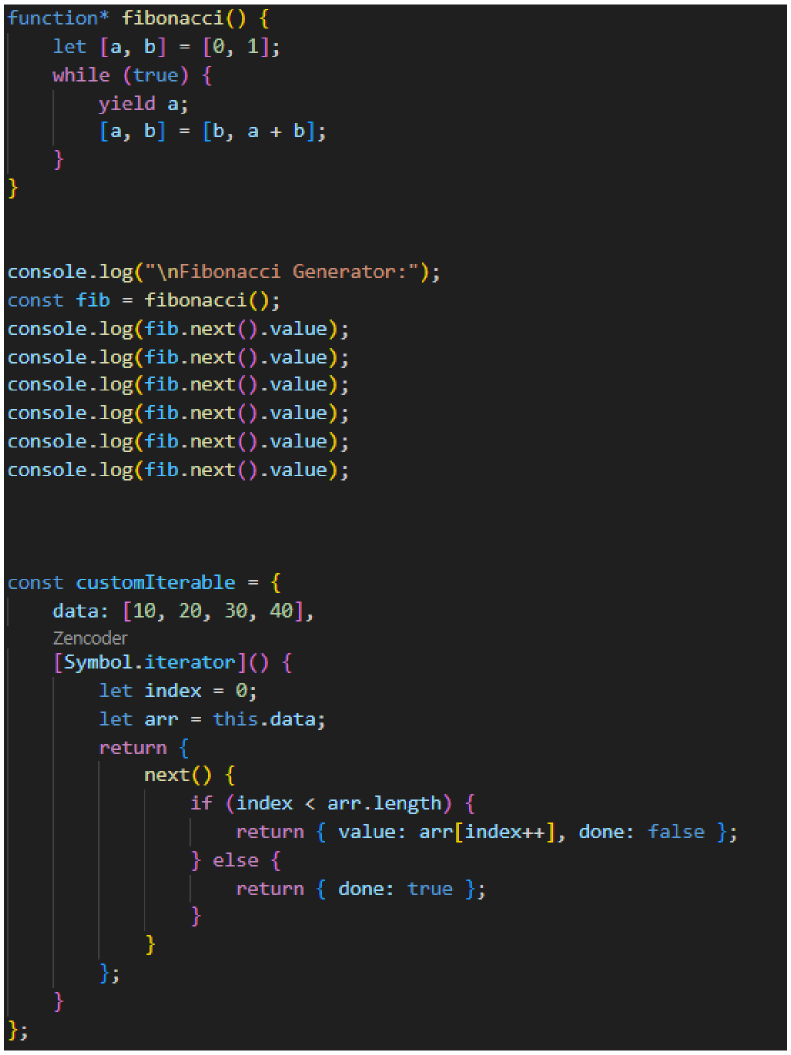
```
❌ Error for invaliduser12345: User "invaliduser12345" not found (HTTP 404)
PS C:\Users\Shreel\OneDrive\Desktop\JS> node "c:\Users\Shreel\OneDrive\Desktop\JS\dashboard.js"
User clicks 'Inventory'
📦 Inventory Module Loaded
Items in stock: 120

User clicks 'Orders'
🛒 Orders Module Loaded
Pending orders: 45

User clicks 'Unknown'
❌ Module "payments" not found.
PS C:\Users\Shreel\OneDrive\Desktop\JS>
```

Practical 10

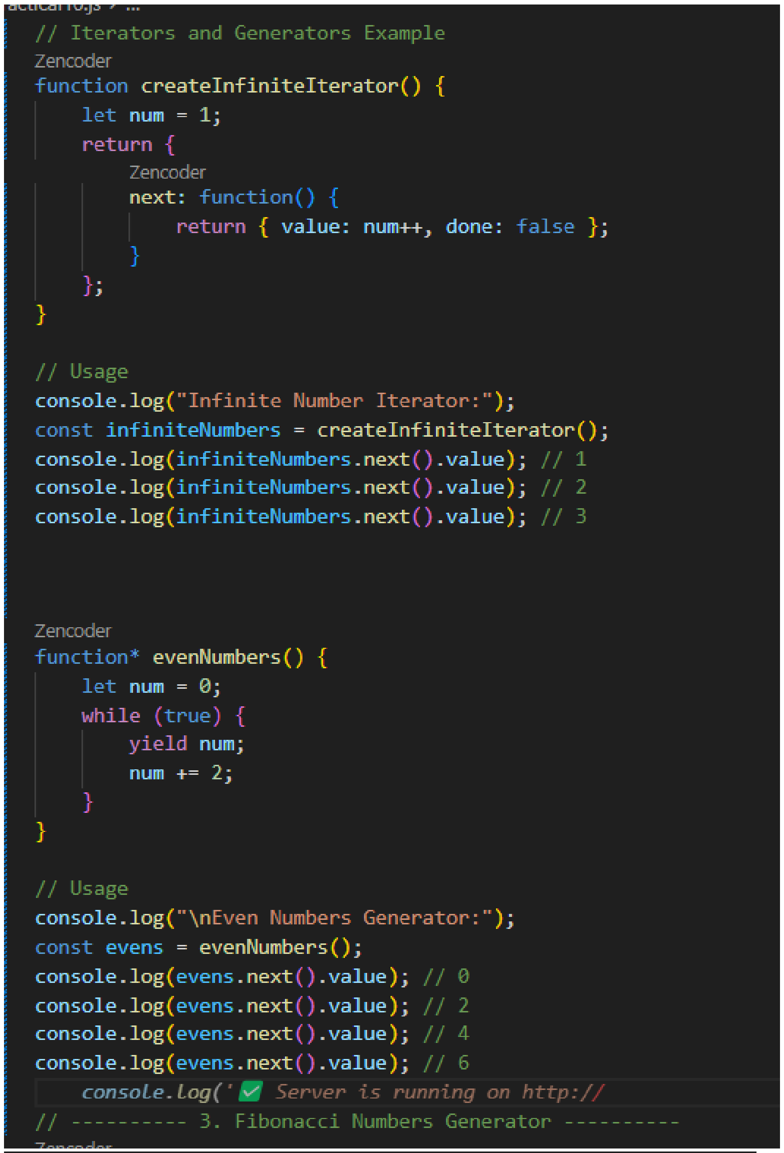
Problem Definition

You are developing a data processing engine. Use an iterator to generate an infinite number sequence and a generator to produce even numbers.

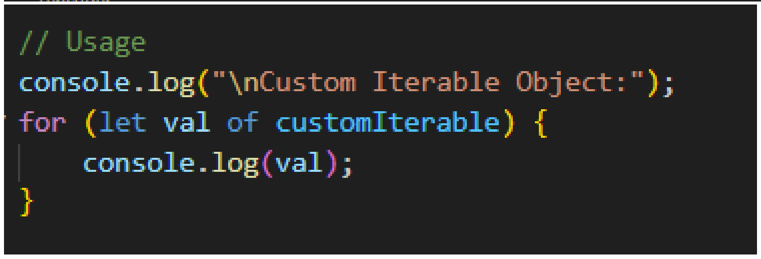
Code



```
function* fibonacci() {
    let [a, b] = [0, 1];
    while (true) {
        yield a;
        [a, b] = [b, a + b];
    }
}


console.log("\nFibonacci Generator:");
const fib = fibonacci();
console.log(fib.next().value);
console.log(fib.next().value);
console.log(fib.next().value);
console.log(fib.next().value);
console.log(fib.next().value);
console.log(fib.next().value);



const customIterable = {
    data: [10, 20, 30, 40],
    Zencoder
    [Symbol.iterator]() {
        let index = 0;
        let arr = this.data;
        return {
            next() {
                if (index < arr.length) {
                    return { value: arr[index++], done: false };
                } else {
                    return { done: true };
                }
            }
        };
    }
};
```

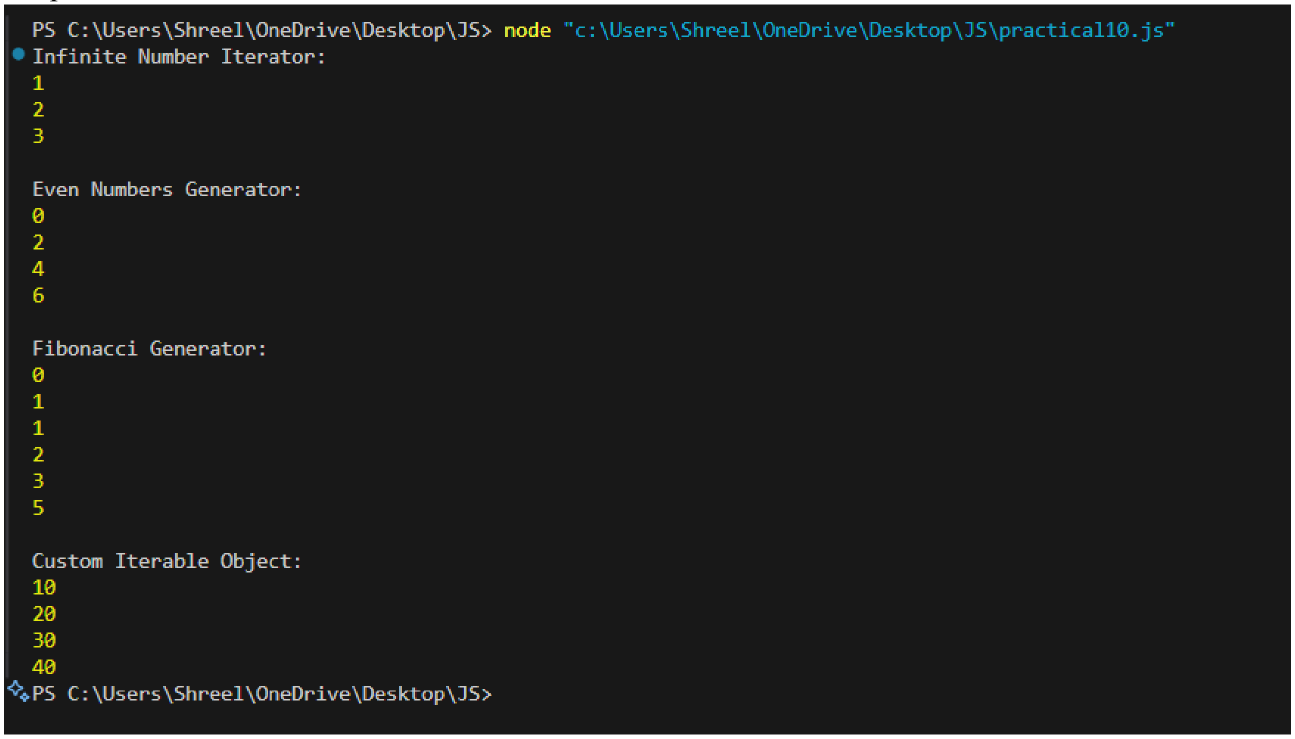


```
// Iterators and Generators Example
Zencoder
function createInfiniteIterator() {
    let num = 1;
    return {
        Zencoder
        next: function() {
            return { value: num++, done: false };
        }
    };
}

// Usage
console.log("Infinite Number Iterator:");
const infiniteNumbers = createInfiniteIterator();
console.log(infiniteNumbers.next().value); // 1
console.log(infiniteNumbers.next().value); // 2
console.log(infiniteNumbers.next().value); // 3



Zencoder
function* evenNumbers() {
    let num = 0;
    while (true) {
        yield num;
        num += 2;
    }
}

// Usage
console.log("\nEven Numbers Generator:");
const evens = evenNumbers();
console.log(evens.next().value); // 0
console.log(evens.next().value); // 2
console.log(evens.next().value); // 4
console.log(evens.next().value); // 6
    console.log('✅ Server is running on http://
// ---------- 3. Fibonacci Numbers Generator ----------
```



```
// Usage
console.log("\nCustom Iterable Object:");
for (let val of customIterable) {
    console.log(val);
}
```

Output:



```
PS C:\Users\Shreel\OneDrive\Desktop\JS> node "c:\Users\Shreel\OneDrive\Desktop\JS\practical10.js"
Infinite Number Iterator:
1
2
3

Even Numbers Generator:
0
2
4
6

Fibonacci Generator:
0
1
1
2
3
5

Custom Iterable Object:
10
20
30
40
PS C:\Users\Shreel\OneDrive\Desktop\JS>
```