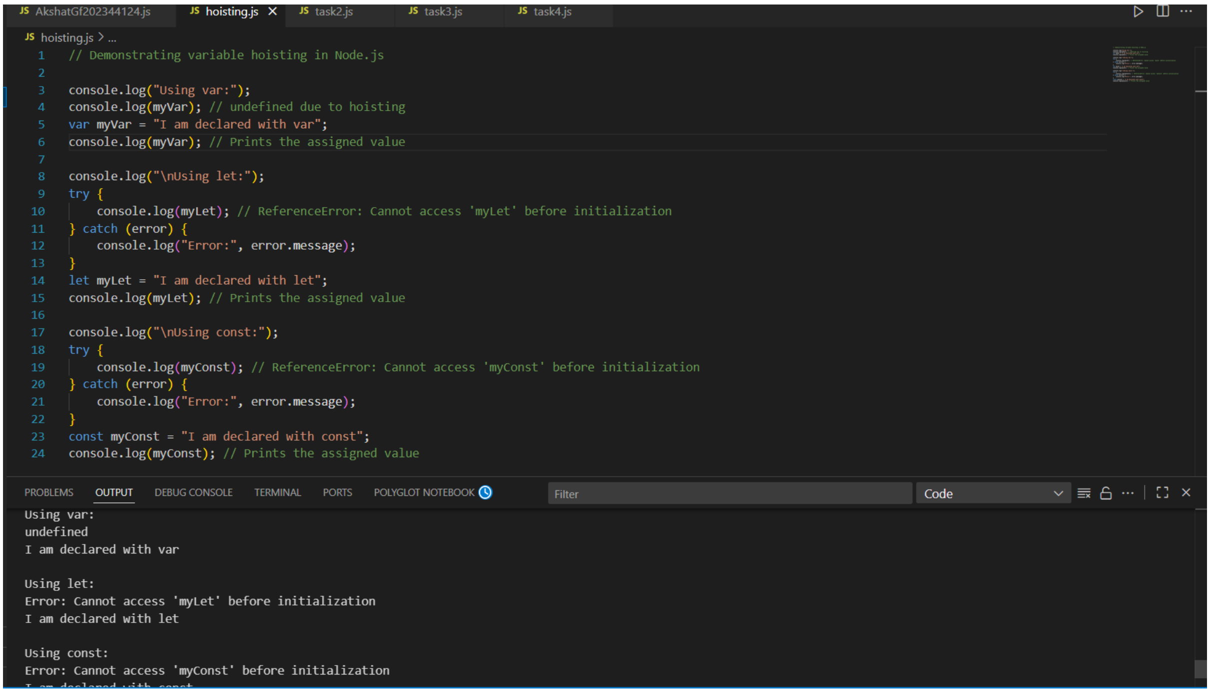
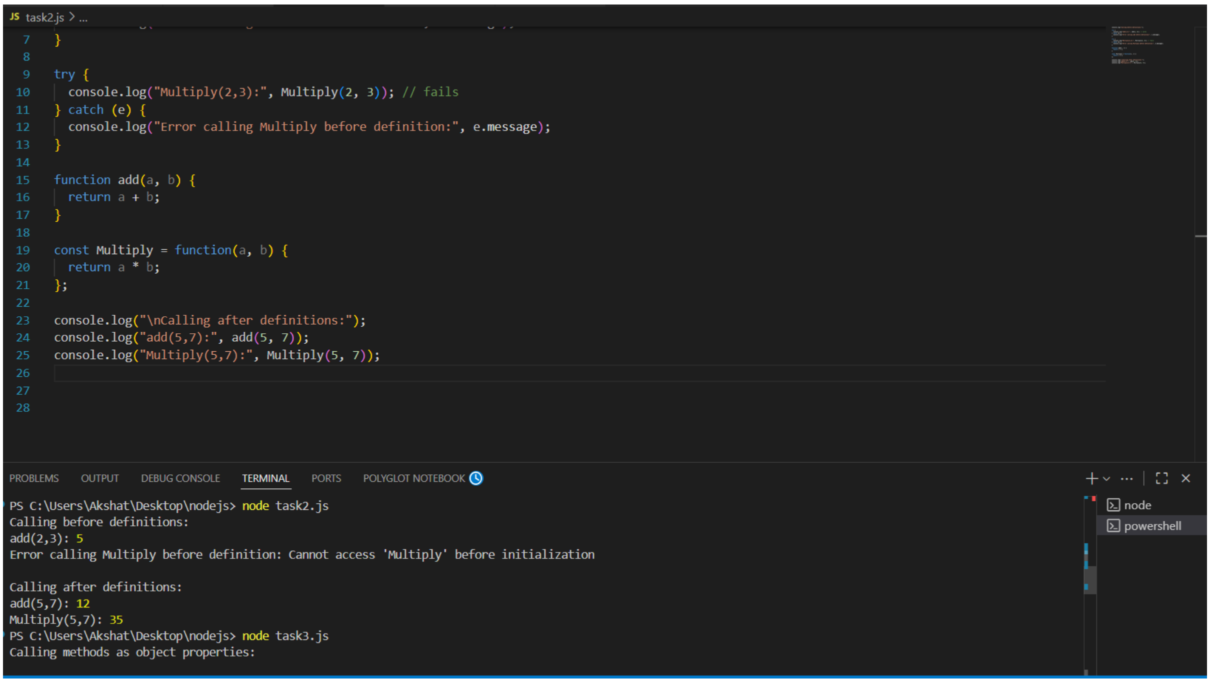


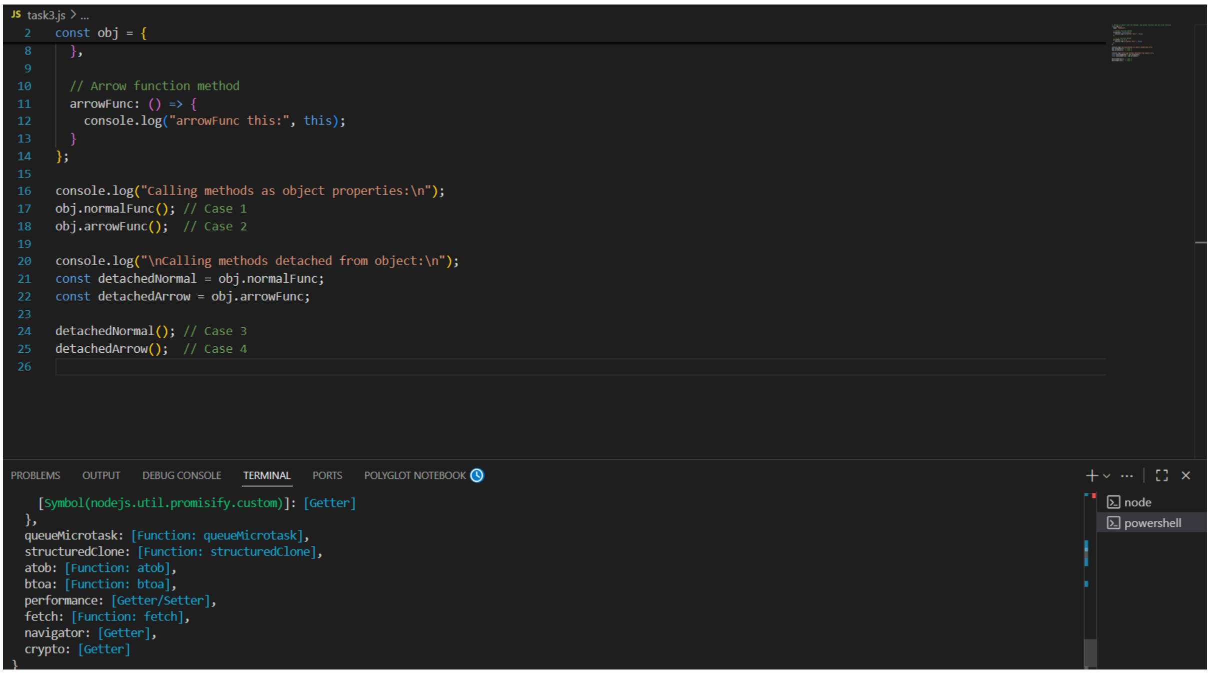
```javascript
// Demonstrating variable hoisting in Node.js

console.log("Using var:");
console.log(myVar); // undefined due to hoisting
var myVar = "I am declared with var";
console.log(myVar); // Prints the assigned value

console.log("\nUsing let:");
try {
    console.log(myLet); // ReferenceError: Cannot access 'myLet' before initialization
} catch (error) {
    console.log("Error:", error.message);
}
let myLet = "I am declared with let";
console.log(myLet); // Prints the assigned value

console.log("\nUsing const:");
try {
    console.log(myConst); // ReferenceError: Cannot access 'myConst' before initialization
} catch (error) {
    console.log("Error:", error.message);
}
const myConst = "I am declared with const";
console.log(myConst); // Prints the assigned value
```

```
Using var:
undefined
I am declared with var

Using let:
Error: Cannot access 'myLet' before initialization
I am declared with let

Using const:
Error: Cannot access 'myConst' before initialization
I am declared with const
```



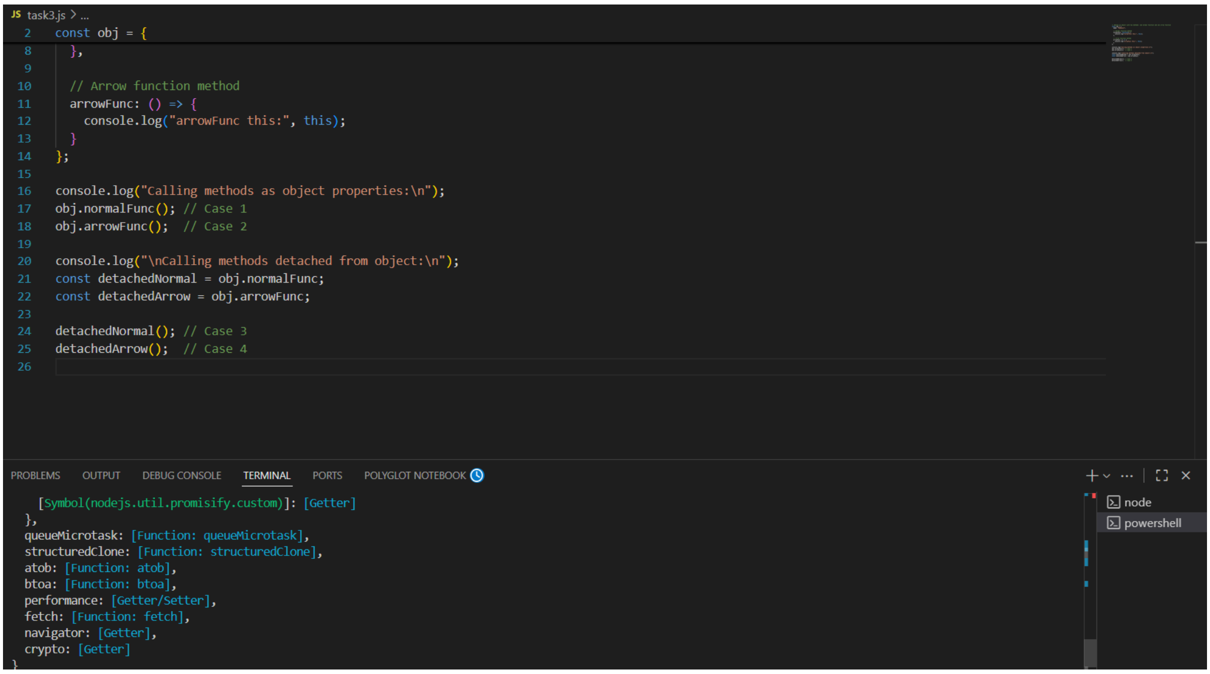
```javascript
}

try {
  console.log("Multiply(2,3):", Multiply(2, 3)); // fails
} catch (e) {
  console.log("Error calling Multiply before definition:", e.message);
}

function add(a, b) {
  return a + b;
}

const Multiply = function(a, b) {
  return a * b;
};

console.log("\nCalling after definitions:");
console.log("add(5,7):", add(5, 7));
console.log("Multiply(5,7):", Multiply(5, 7));
```

```
PS C:\Users\Akshat\Desktop\nodejs> node task2.js
Calling before definitions:
add(2,3): 5
Error calling Multiply before definition: Cannot access 'Multiply' before initialization

Calling after definitions:
add(5,7): 12
Multiply(5,7): 35
PS C:\Users\Akshat\Desktop\nodejs> node task3.js
Calling methods as object properties:
```



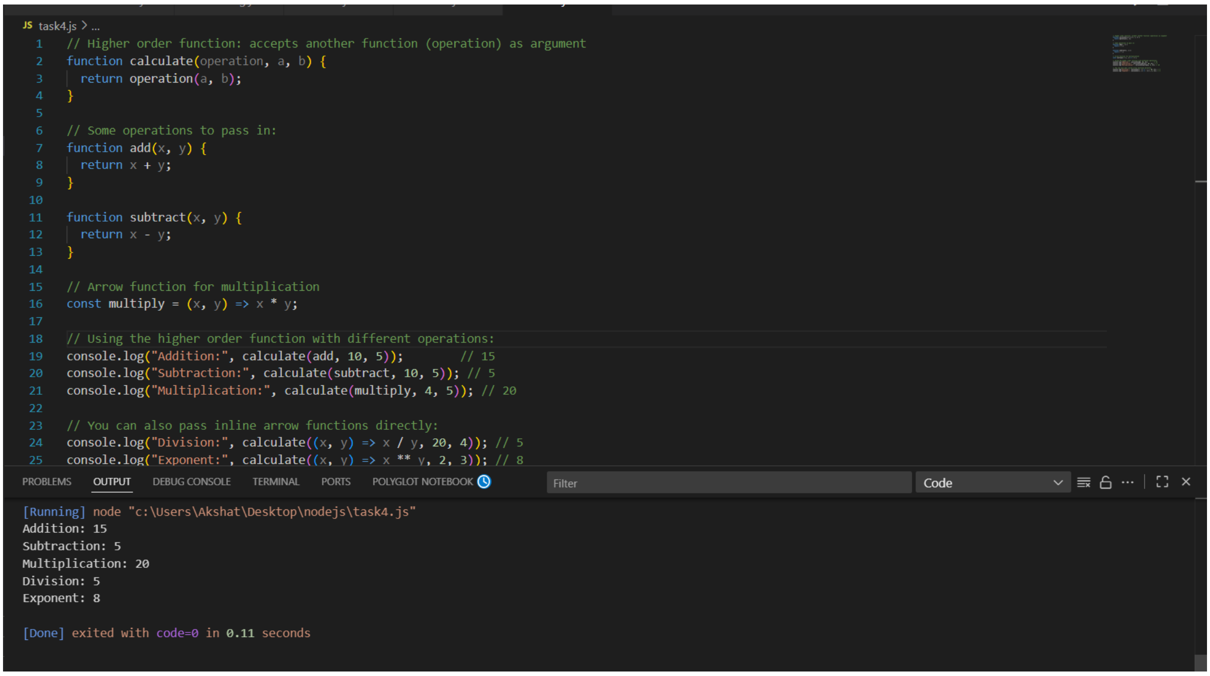
```
const obj = {
  },

  // Arrow function method
  arrowFunc: () => {
    console.log("arrowFunc this:", this);
  }
};

console.log("Calling methods as object properties:\n");
obj.normalFunc(); // Case 1
obj.arrowFunc();  // Case 2

console.log("\nCalling methods detached from object:\n");
const detachedNormal = obj.normalFunc;
const detachedArrow = obj.arrowFunc;

detachedNormal(); // Case 3
detachedArrow();  // Case 4
```

```
    [Symbol(nodejs.util.promisify.custom)]: [Getter]
  },
  queueMicrotask: [Function: queueMicrotask],
  structuredClone: [Function: structuredClone],
  atob: [Function: atob],
  btoa: [Function: btoa],
  performance: [Getter/Setter],
  fetch: [Function: fetch],
  navigator: [Getter],
  crypto: [Getter]
}
```



```
const obj = {
  },

  // Arrow function method
  arrowFunc: () => {
    console.log("arrowFunc this:", this);
  }
};

console.log("Calling methods as object properties:\n");
obj.normalFunc(); // Case 1
obj.arrowFunc();  // Case 2

console.log("\nCalling methods detached from object:\n");
const detachedNormal = obj.normalFunc;
const detachedArrow = obj.arrowFunc;

detachedNormal(); // Case 3
detachedArrow();  // Case 4
```

```
    [Symbol(nodejs.util.promisify.custom)]: [Getter]
  },
  queueMicrotask: [Function: queueMicrotask],
  structuredClone: [Function: structuredClone],
  atob: [Function: atob],
  btoa: [Function: btoa],
  performance: [Getter/Setter],
  fetch: [Function: fetch],
  navigator: [Getter],
  crypto: [Getter]
}
```



```javascript
// Higher order function: accepts another function (operation) as argument
function calculate(operation, a, b) {
  return operation(a, b);
}

// Some operations to pass in:
function add(x, y) {
  return x + y;
}

function subtract(x, y) {
  return x - y;
}

// Arrow function for multiplication
const multiply = (x, y) => x * y;

// Using the higher order function with different operations:
console.log("Addition:", calculate(add, 10, 5));        // 15
console.log("Subtraction:", calculate(subtract, 10, 5)); // 5
console.log("Multiplication:", calculate(multiply, 4, 5)); // 20

// You can also pass inline arrow functions directly:
console.log("Division:", calculate((x, y) => x / y, 20, 4)); // 5
console.log("Exponent:", calculate((x, y) => x ** y, 2, 3)); // 8
```

```
[Running] node "c:\Users\Akshat\Desktop\nodejs\task4.js"
Addition: 15
Subtraction: 5
Multiplication: 20
Division: 5
Exponent: 8

[Done] exited with code=0 in 0.11 seconds
```