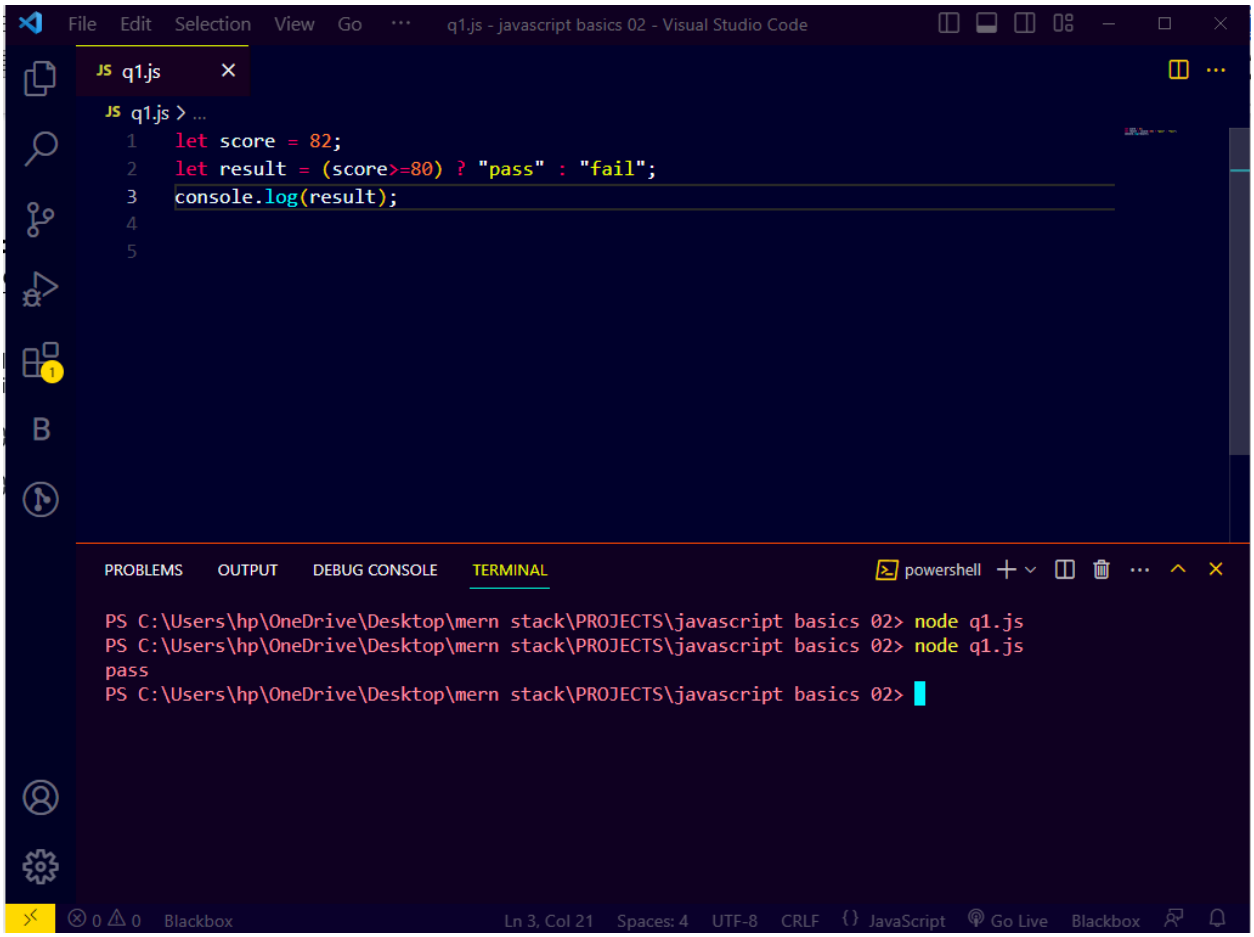
09. Javascript Basics

Name: Ali Sarwar
Course: MERN Stack

1. Rewrite the following code using a ternary operator:

```
let result;
if (score >= 80) {
    result = "Pass";
} else {
    result = "Fail";
}
```



2. How does the optional chaining operator (?.) work, and how can it be used to access nested properties of an object?

The optional chaining operator (?.) was developed to make it easier to access object methods when working with undefined values. By doing this, it is possible to prevent potential errors from happening while attempting to access attributes on a chain of nested objects.

Example:

```
JS q1.js    JS q2.js
JS q2.js > ...
1  // Syntax of optional chaining operator
2  // calling methods
3  const result = object?.method?.();
4
```

3. Compare the for...in loop and the for....of loop in terms of their use cases and the types of values they iterate over.

The for-in loop is used to iterate the properties of an object.
Where as the for-of loop is used to iterate the values of iterable objects.

```js
let arr = [1, 2, 3];

// for-in loop
for (let index in arr) {
    console.log(index); // prints 0, 1, 2
}

// for-of loop
for (let value of arr) {
    console.log(value);
}
```

```
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q3.js
0
1
2
1
2
3
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02>
```

4. Define a function calculateAverage that takes an array of numbers as an argument and returns the average value.

```
JS q4.js > numbersArray
function calculateAverage(numbers) {
    let sum = 0;

    for (let i = 0; i < numbers.length; i++) {
        sum += numbers[i];
    }

    let average = sum / numbers.length;
    return average;
}


let numbersArray = [10, 20, 30, 40, 50];
let avg = calculateAverage(numbersArray);
console.log("Average is:", avg);
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q4.js
Average is: 30
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02>
```

5. Explain the concept of "closures" in JavaScript and provide an example of their practical use.

   When a function is declared inside of another function in JavaScript, a closure is a key idea that allows access to the variables and scope of the outer (containing) function even after the outer function has completed running.

q1.js q2.js q3.js q4.js q5.js

q5.js > output

```javascript
function outer(a) {

    function inner(b) {
        return a + b;
    }

    return inner;
}

const closure = outer(20);
const output = closure(10);
console.log(output);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q5.js
30
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02>
```

6. Create an object named student with properties name, age, and grades. Add a method calculateAverage that calculates the average of the grades.

```js
const student = {
    name: "John",
    age: 20,
    grades: [85, 92, 78, 90],

    calculateAverage: function() {
        const sum = this.grades.reduce((total, grade) => total + grade, 0);
        const average = sum / this.grades.length;
        return average;
    }
};

console.log("Average Grade:", student.calculateAverage());
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q6.js
Average Grade: 86.25
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02>
```

7. How can you clone an object in JavaScript and also give one example each deep copy, shallow copy, and reference copy

   There are various ways to duplicate an object in JavaScript, each with a different level of depth for copying nested objects or arrays.

   Shallow copy

```
JS q7.js > original > b > c
// Shallow Copy Example
let original = { a: 5, b: { c: 5 } };
let shallowCopy = { ...original };

original.b.c = 10;

console.log(shallowCopy.b.c);
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q7.js
10
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02>
```

Deep copy

```javascript
const original = {
    a: 1,
    b: {
        c: 2,
        d: [3, 4]
    }
};

const deepCopied = deepCopy(original);

original.b.c = 5;
original.b.d[0] = 6;

console.log(deepCopied.b.c);
console.log(deepCopied.b.d[0]);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q7.js
10
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q7.js
5
6
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02>
```

Reference copy

```javascript
// Reference Copy Example
const original = { a: 1, b: { c: 2 } };
const referenceCopy = original;

original.b.c = 3;

console.log(referenceCopy.b.c);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q7.js
10
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q7.js
5
6
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q7.js
3
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02>
```

8. Write a loop that iterates over an array of numbers and logs whether each number is even or odd, using a ternary operator.

JS q1.js JS q2.js JS q3.js JS q4.js JS q5.js

JS q8.js > parity

```js
const numbers = [0,1,2,3,4,5,6,7,8,9,10,11];

for (const num of numbers) {
    const parity = num % 2 === 0 ? 'even' : 'odd';
    console.log(`${num} is ${parity}`);
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
0 is even
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
10 is even
11 is odd
```

9. Describe the differences between the for loop, while loop, and do...while loop in JavaScript. When might you use each?

FOR LOOP

For loop is used when we know the number of iterations.
For(initialization, condition, increment / decrement)

WHILE LOOP

While loop is used when we don’t know the number of iterations

Initilzation

While(condition){

Increment / decrement

}

DO-WHILE LOOP

Do while loop must execute at-least one time.

Initialization

Do{

Increment / decrement

}while(condition)

10. Provide an example of using optional chaining within a loop to access a potentially missing property of an object.

```
JS q10.js > students > name
const students = [
    { name: "Ali", grades: [85, 90, 78] },
    { name: "Umer" },
    { name: "Huzaifa", grades: [92, 88, 95] }
];

for (const student of students) {
    const averageGrade = student.grades?.length
        ? student.grades.reduce((total, grade) => total + grade, 0) / student.grades.length
        : "N/A";

    console.log(`${student.name}: Average Grade - ${averageGrade}`);
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q10.js
Ali: Average Grade - 84.33333333333333
Umer: Average Grade - N/A
Huzaifa: Average Grade - 91.66666666666667
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02>
```

11. Write a for...in loop that iterates over the properties of an object and logs each property name and value.

```js
const person = {
    name: "Ali",
    age: 30,
    occupation: "Engineer"
};

for (const key in person) {
    console.log(`Property: ${key}, Value: ${person[key]}`);
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q11.js
Property: name, Value: Ali
Property: age, Value: 30
Property: occupation, Value: Engineer
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02>
```

12. Explain the use of the break and continue statements within loops. Provide scenarios where each might be used.

BREAK STATEMENT

Break statement is used when a code want to exit the scenario if the condition is found to be true.

```
JS q12.js > ...
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

for (const num of numbers) {
    if (num === 3) {
        console.log("break when 2 is found");
        break;
    }
    console.log(num);
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q12.js
1
2
break when 2 is found
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02>
```

Continue Statement

The continue statement is used to skip the current iteration of a loop and move on to the next iteration.

```javascript
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

for (const num of numbers) {
    if (num % 2 === 0) {
        continue;
    }
    console.log(num);
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q12.js
1
2
break when 2 is found
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q12.js
1
3
5
7
9
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02>
```

13. Write a function calculateTax that calculates and returns the tax amount based on a given income. Use a ternary operator to determine the tax rate.

q13.js > ...

```javascript
function calculateTax(income) {
    const taxRate = income > 50000 ? 0.2 : 0.15;
    const taxAmount = income * taxRate;
    return taxAmount;
}

// Example usage
const income1 = 60000;
const tax1 = calculateTax(income1);
console.log(`Tax amount for $${income1}: $${tax1}`);

const income2 = 30000;
const tax2 = calculateTax(income2);
console.log(`Tax amount for $${income2}: $${tax2}`);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02> node q13.js
Tax amount for $60000: $12000
Tax amount for $30000: $4500
PS C:\Users\hp\OneDrive\Desktop\mern stack\PROJECTS\javascript basics 02>
```