second\app.js

```
//1. Rewrite the following code using a ternary operator:
// Assignment 1
let result;
let score;
score >=80 ?  result = "Pass" : result = "Fail";

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

const obj3 = {
    property1: {
      property2: {
        property3: 'Nested Value',
      },
    },
  };

  // With optional chaining
  const valueWithOptionalChaining = obj3.property1?.property2?.property3;

  console.log(valueWithOptionalChaining);

//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.
//Assignment 3

const myobject = { a: 1, b: 2, c: 3 };

for (let key in myobject) {
  console.log(key);
}

const myArray = [1,5,7];

for (let value of myArray) {
  console.log(value);
}

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

function calculateAverage(numbers) {
  return numbers.length > 0 ? numbers.reduce((sum, num) => sum + num, 0) / numbers.length : 0;
}

// Example usage
const numbers1 = [1, 2, 3, 4, 5];
console.log(`Average: ${calculateAverage(numbers1)}`);

```

```
const numbers2 = [10, 20, 30];
console.log(`Average: ${calculateAverage(numbers2)}`);

const numbers3 = [];
console.log(`Average: ${calculateAverage(numbers3)}`);


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

function closure(color) {
  return function () {
    //document.body.style.backgroundColor = `${color}`;
  };
}
//document.getElementById("ora").onclick = closure("orange");
//document.getElementById("yel").onclick = closure("yellow");


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

//Assignment 6

const student = {
  name: 'John',
  age: 20,
  grades: [85, 90, 78, 95, 88],

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

console.log(`Student: ${student.name}`);
console.log(`Age: ${student.age}`);
console.log(`Grades: ${student.grades.join(', ')}`);
console.log(`Average Grade: ${student.calculateAverage().toFixed(2)}`);

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

// Assignment 7

// Shallow
const originalObject = {
  a: 1,
  b: {
  c: 2 }
};

const shallowCopy = { ...originalObject };

```

```
originalObject.a = 10;

console.log(shallowCopy.a);

const deep = { a: 1, b: { c: 2 } };

// Deep copy
const deepCopy = JSON.parse(JSON.stringify(deep));

console.log(deepCopy);

// Ref Copy

let obj ={name: "ali",age : 25} ;
let objCopy = obj
obj.name = "muhammmad"
console.log(objCopy)

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

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

for (let number of numbers) {
  const result = number % 2 === 0 ? 'Even' : 'Odd';
  console.log(`${number} is ${result}`);
}


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

//Assignment 9

// For Loop
for (let i = 0; i < 5; i++) {
  console.log(i);
}

// While Loop
let j = 0;
while (j < 5) {
  console.log(j);
  j++;
}

// Do...While Loop
let k = 0;
do {
  console.log(k);
  k++;
} while (k < 5);
```

```

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

// Assignmet 10

const individuals = [
  { userId: 1, userName: "Alice", userLocation: { userCity: "New York" } },
  { userId: 2, userName: "Bob", userLocation: null },
  { userId: 3, userName: "Charlie" }
];

for (const individual of individuals) {
  const city = individual?.userLocation?.userCity ?? "Unknown City";
  console.log(`${individual.userName}'s city: ${city}`);
}

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

// Assignmet 11

let obj2 = {
  name : "abdullah",
  father : "Rehmatullah"
}
 for( let properties in obj2){
  console.log(properties, obj2[properties])
 }

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

 // Assignmet 12

for (let i = 1; i <= 10; i++) {
  console.log(i);
  if (i === 5) {
    break;
  }
}

for (let i = 1; i <= 5; i++) {
  if (i === 3) {
    continue;
  }
  console.log(i);
}


//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.

 //Assignment 13
```

```
 function calculateTax(income) {
  const taxRate = income <= 50000 ? 0.1 : income <= 100000 ? 0.15 : 0.2;
  return income * taxRate;
}

console.log(`Tax for income $30000: $${calculateTax(30000)}`);
console.log(`Tax for income $75000: $${calculateTax(75000)}`);
console.log(`Tax for income $120000: $${calculateTax(120000)}`);

//14. Create an object car with properties make, model, and a method startEngine that logs a message. Instantiate the object and call the method.

// Assignmet 14

const car = {
  make: 'Toyota',
  model: 'Camry',
  startEngine: function() {
    console.log(`The ${this.make} ${this.model}'s engine is started.`);
  }
};

car.startEngine();

//15. Explain the differences between regular functions and arrow functions in terms of scope, this binding, and their use as methods.

// Assignmet 15

const object1 = {
  name: 'Regular Function',
  regularMethod: function() {
    console.log(`Hello, I am ${this.name}`);
  }
};

//object1.regularMethod();

const object2 = {
  name: 'Arrow Function',
  arrowMethod: () => {
    console.log(`Hello, I am ${this.name}`);
  }
};
object2.arrowMethod();

```