```
1  // Question 3
2  //for in loop
3  const person = { name: "Elon
   Musk", age: 50 };
4
5  for (const key in person) {
6    console.log(key, person[key]);
7  }
8  // for off loop
9  const numbers = [1, 2, 3, 4];
10
11 for (const num of numbers) {
12   console.log(num);
13 }
```

Run

```
name Elon Mu
age 50
1
2
3
4
```

```
// Question 2
const user = {
  name: "Tom Cruise",
  address: {
    street: "47",
    city: "Karachii",
    zip: "12345"
  }
};
const city = user.address &&
user.address.city;
const cityOptionalChaining =
user?.address?.city;
console.log(city);
console.log(cityOptionalChaining);
```

Run

```
Karachii
Karachii
```

```
// Question 4

const nmbrs = [1, 2, 3, 4, 5];
const average =
calculateAverage(nmbrs);
function calculateAverage(nmbrs) {
  if (nmbrs.length === 0) {
    return 0;
  }
let sum = 0;
for (let i = 0; i < nmbrs.length;
i++) {
    sum += nmbrs[i];
  }
  return sum / nmbrs.length;
}
console.log(average);
```

Run

```
3
```

```
// Question 5

function outerFunction(x) {
  function innerFunction(y) {
    return x + y;
  }
  return innerFunction;
  }
const closureExample =
outerFunction(10);
const result = closureExample(5);
console.log(result);
//Explain
//JavaScript closure is a feature
that allows inner functions to
access the outer scope of a
function. Closure helps in binding
a function to its outer boundary
and is created automatically
whenever a function is created. A
block is also treated as a scope
since ES6.
```

```
15
```

```
// Question 6
const student = {
  name: "Salman",
  age: 17,
  grades: [20, 80, 90, 92, 88],
  calculateAverage: function () {
    if (this.grades.length === 0) {
      return 0;
    }
    let sum = 0;
for(let i =0;i<this.grades.length;
i++) {
      sum += this.grades[i];
    }
    return sum /this.grades.length;
  }
};
const averageGrade =
student.calculateAverage();
console.log(` ${student.name}
${averageGrade}`);
```

Run

```
Salman 74
```

```
3  //shallow Copy
4  const originalObject1 = {
5    name: 'Salman',
6    age: 18,
7    address: { city: "Karachi i",
   zip: '12345' }
8  };
9  const shallowCopy = {
   ...originalObject1};
10 shallowCopy.address.city =
   'NewCity';
11 console.log(originalObject1.address
   .city);
12
13 //Deep Copy
14 const originalObject2 = {
15   name: 'Salman',
16   age: 18,
17   address: { city: 'Karachii',
   zip: '12345' }
18 };
19 const deepCopy = JSON.parse(
20 JSON.stringify(originalObject2));
21 deepCopy.address.city = 'NewCity';
22 console.log(originalObject2.address
   .city);
```

```
24 //Reference Copy
25 const originalObject3 = {
26   name: 'John',
27   age: 25,
28   address: { city: 'Examplevil
   zip: '12345' }
29 };
30
31 const referenceCopy =
   originalObject3;
32 referenceCopy.address.city =
   'NewCity';
33 console.log(originalObject3.ad
   .zip);
```

Run 359m

```
NewCity
Karachii
12345
```

```
1  //Question 8
2  const numbers = [1, 2, 3, 4, 5];
3
4  for (let i = 0; i < numbers.length;
   i++) {
5    const result = (numbers[i] % 2 ===
    0) ? 'Even' : 'Odd';
6    console.log(`${numbers[i]} is
   ${result}`);
7  }
```

Run

```
1 is Odd
2 is Even
3 is Odd
4 is Even
5 is Odd
```

```
1  //Question 10
2  const users = [
3    { id: 1, name: 'Salman',
   address: { city: 'Karachi' } },
4    { id: 2, name: 'Faizan' },
5    { id: 3, name: 'Ali', address: {
   city: 'Lahore' } }
6  ];
7
8  for (const user of users) {
9  const cityName =user?.address?.city
    || 'Unknown';
10   console.log(`${user.name}'s city
   is ${cityName}`);
11 }
```

Run

```
Salman's city is Ka
Faizan's city is U
Ali's city is Laho
```

```
1  //Question 11
2  const myObject = {
3    name: 'Salman',
4    age: 17,
5    city: 'Karachi'
6  };
7
8  for (const key in myObject) {
9    if (myObject.hasOwnProperty(key))
    {
10     console.log(`${key}:
   ${myObject[key]}`);
11   }
12 }
```

Run

```
name: Salman
age: 17
city: Karachi
```

Question:Describe the differences betwe the for loop, while loop, and do...while lo JavaScript. When might you use each?

Use a for loop when you know the loop should execute n time while loop for reading a file into a variable. Use a while loop w asking for user input. Use a while loop when the increment va nonstandard.

```javascript
//Question 12
//Break Statement in loop
//The break statement is used to terminate the execution of a loop prematurely.
for (let i = 0; i < 10; i++) {
  if (i === 5) {
    console.log("Breaking loop at i = 5");
    break;
  }
  console.log(i);
}
// Continue Statement
//The continue statement is used to skip the current iteration and move to the next one in a loop.
for (let i = 0; i < 5; i++) {
  if (i === 2) {
    console.log("Skipping iteration at i = 2");
    continue;
  }
  console.log(i);
}
```

Run 392m

```
0
1
2
3
4
Breaking loop at i = 5
0
1
Skipping iteration at i =
3
4
```

```javascript
//Question 13
function calculateTax(income) {
  const taxRate = income <= 50000 ? 0.1 : income <= 100000 ? 0.15 : 0.2;
  const taxAmount = income * taxRate;
  return taxAmount;
}

// Example usage:
const income1 = 40000;
const tax1 = calculateTax(income1);
console.log(`Tax for ${income1}: $${tax1}`);
const income2 = 80000;
const tax2 = calculateTax(income2);
console.log(`Tax for ${income2}: $${tax2}`);
const income3 = 120000;
const tax3 = calculateTax(income3);
console.log(`Tax for ${income3}: $${tax3}`);
```

Run 326

```
Tax for 40000: $4000
Tax for 80000: $12000
Tax for 120000: $24000
```

```
1  //Question 14
2  const car = {
3    make: 'Toyota',
4    model: 'Camry',
5    startEngine: function() {
6      console.log('Engine started. Ready to go!');
7    }
8  };
9  car.startEngine();
```

Run

```
Engine started. Read
```

Question: Explain the differences betwee regular functions and arrow functions in terms of scope, this binding, and their us methods. More details

Comparison: Use arrow functions for concise, simple functi don’t require their own this binding or other special feature regular functions when you need more control over this , w defining methods within classes, or when working with constructor functions.

```
1  //Question 1
2  let score = 70;
3  let result = (score >= 80) ? "Pass"
   : "Fail";
4  console.log(result);
```

Run 292ms

```
Fail
```