QUEATION 1

1)

a variable can be used before it has been declared.

Hoisting is JavaScript's: moving all declarations to the top of the current scope

Variables defined with let and const are hoisted to the top of the block, but not *initialized*.

2)

The super keyword is used to call the constructor of its parent class to access the parent's properties and methods.

3)

The main difference between "var", "let", and "const" is in their scope and assignability. Variables declared with "var" have the function-level scope and can be reassigned, while "let" and "const" variables have the block-level scope and "let" can be reassigned, but "const" cannot.

4)

The rest parameter syntax allows a function to accept an indefinite number of arguments as an array

arrow functions have a shorter syntax compared to regular functions.

syntax compared to regular functions.

```
// Regular function

function regularFunction(a, b) {

  return a + b;

}

// Arrow function

const arrowFunction = (a, b) => a + b;
```

****

Question2: True Or False

1) F
2) F
3) F
4) F
5) F
6) T
7) T
8) T

Question3 MCQ

1) asynchronous, non-blocking, single-threaded language.
2) Encapsulation

Question 4 : What is The Output

1) Error: the fails
   the fails
   the fails
2) Error getFullName is not a function
3) 5

   6

   7
4) Error fradie.colorChange is not a function
5) String
6) 0

   1

   4

   2

3

7) Error: i is not defined (out scope of function)
8) hello world

   10

9) [59.52, 83.7, 93]
10) ['batman', 'bane']

Question 5

1)

2)

```
function printMultiplicationTable(): void {
  const maxNumber = 12;

  for (let i = 1; i <= maxNumber; i++) {
    let row = "";
    for (let j = 1; j <= maxNumber; j++) {
      const result = i * j;
      row += `${i} * ${j} = ${result}\t`;
    }
    console.log(row);
  }
}

// Call the function to print the multiplication table
printMultiplicationTable();
```

3)

```
function getElementsAtOddPositions(list) {
  const result = [];

  for (let i = 1; i < list.length; i += 2) {
    result.push(list[i]);
  }

  return result;
}
```

4)

```
function isPrime(number) {
  // Check if the number is less than 2
  if (number < 2) {
    return false;
  }

  for (let i = 2; i <= Math.sqrt(number); i++) {
    if (number % i === 0) {
      return false;
    }
  }

  return true;
```

6)

```
function countVowels(str) {
  const vowels = ['a', 'e', 'i', 'o', 'u'];
  let count = 0;

  for (let i = 0; i < str.length; i++) {
    if (vowels.indexOf(str[i].toLowerCase()) !== -1) {
      count++;
    }
  }

  return count;
}

const string = 'Hello, World!';
const vowelCount = countVowels(string);
console.log('The number of vowels in the string is: ' + vowelCount);
```

7)

```
class Animal {
  protected name: string;
  protected age: number;

  public set_value(name: string, age: number): void {
    this.name = name;
    this.age = age;
  }
}

class Zebra extends Animal {
  private placeOfOrigin: string;

  constructor(placeOfOrigin: string) {
    super();
    this.placeOfOrigin = placeOfOrigin;
  }

  public getInfo(): string {
    return `Name: ${this.name}, Age: ${this.age}, Place of Origin: ${this.placeOfOrig
in}`;
  }
}

class Dolphin extends Animal {
  private placeOfOrigin: string;

  constructor(placeOfOrigin: string) {
    super();
    this.placeOfOrigin = placeOfOrigin;
  }

  public getInfo(): string {
    return `Name: ${this.name}, Age: ${this.age}, Place of Origin: ${this.placeOfOrig
in}`;
  }
}

// Example usage
const zebra = new Zebra('Africa');
zebra.set_value('Ziggy', 5);
console.log(zebra.getInfo());

const dolphin = new Dolphin('Ocean');
dolphin.set_value('Dolly', 10);
console.log(dolphin.getInfo());
```

9)

```
class myObject {
  constructor(name, message) {
    this.name = name.toString();
    this.message = message.toString();
  }

  getName() {
    return this.name;
  }

  getMessage() {
    return this.message;
  }
}

const obj = new myObject('John', 'Hello, World!');
console.log(obj.getName()); // Output: John
console.log(obj.getMessage()); // Output: Hello, World!
```

10)

```
class Shape {
  protected color: string;
  protected filled: boolean;

  constructor(color: string, filled: boolean) {
    this.color = color;
    this.filled = filled;
  }

  public getColor(): string {
    return this.color;
  }

  public isFilled(): boolean {
    return this.filled;
  }
}

class Circle extends Shape {
  private radius: number;

  constructor(color: string, filled: boolean, radius: number) {
    super(color, filled);
    this.radius = radius;
  }

  public getRadius(): number {
    return this.radius;
  }

  public getArea(): number {
    return Math.PI * this.radius * this.radius;
  }
```

```typescript
}

class Rectangle extends Shape {
  private width: number;
  private height: number;

  constructor(color: string, filled: boolean, width: number, height: number) {
    super(color, filled);
    this.width = width;
    this.height = height;
  }

  public getWidth(): number {
    return this.width;
  }

  public getHeight(): number {
    return this.height;
  }

  public getArea(): number {
    return this.width * this.height;
  }
}

class Square extends Rectangle {
  private side: number;

  constructor(color: string, filled: boolean, side: number) {
    super(color, filled, side, side);
    this.side = side;
  }

  public getSide(): number {
    return this.side;
  }
}

const circle = new Circle('Red', true, 5);
console.log('Circle Color:', circle.getColor());
console.log('Is Circle Filled:', circle.isFilled());
console.log('Circle Radius:', circle.getRadius());
console.log('Circle Area:', circle.getArea());

const rectangle = new Rectangle('Blue', false, 6, 8);
console.log('Rectangle Color:', rectangle.getColor());
console.log('Is Rectangle Filled:', rectangle.isFilled());
console.log('Rectangle Width:', rectangle.getWidth());
console.log('Rectangle Height:', rectangle.getHeight());
console.log('Rectangle Area:', rectangle.getArea());

const square = new Square('Green', true, 5);
console.log('Square Color:', square.getColor());
console.log('Is Square Filled:', square.isFilled());
console.log('Square Side:', square.getSide());
console.log('Square Area:', square.getArea());
```