Answer the following

1. **Explain the difference between var, let and const keywords.**

Ans –

Feature	**var**	**let**	**const**
Scope	Function-scoped	Block-scoped	Block-scoped
Redeclaration	Allowed	Not allowed in the same scope	Not allowed in the same scope
Reassignment	Allowed	Allowed	Not allowed after initialization
Hoisting	Hoisted and initialized as undefined	Hoisted but not initialized	Hoisted but not initialized
Usage	Older syntax, less predictable behavior	Modern and preferred for variables	Preferred for constants

Example

1. **Scope**:
   - var is limited to the function it is defined in or the global scope if outside any function.
   - let and const are limited to the block (enclosed by {}) they are defined in.

```
function example() {
  if (true) {
    var x = 10;   // Function-scoped
```

```
        let y = 20;   // Block-scoped
        const z = 30; // Block-scoped
    }
    console.log(x); // Output: 10
    // console.log(y); // Error: y is not defined
    // console.log(z); // Error: z is not defined
}
example();
```

2. **Redeclaration**:
    - var allows you to declare the same variable multiple times.
    - let and const throw an error if a variable is redeclared in the same scope.

```
var x = 5;
var x = 10; // No error
let y = 20;
// let y = 30; // Error: Identifier 'y' has already been declared
const z = 40;
// const z = 50; // Error: Identifier 'z' has already been declared
```

3. **Reassignment**:
    - Variables declared with var and let can be reassigned.
    - const variables cannot be reassigned after initialization.

```
var a = 10;
a = 20; // Allowed
let b = 30;
b = 40; // Allowed

const c = 50;
// c = 60; // Error: Assignment to constant variable
```

2. Mention the different datatypes with an example for each.

Ans –

Different Data Types in JavaScript

JavaScript has **7 primitive types** and **1 non-primitive type (object)**.

1. Number

Represents both integer and floating-point numbers.
Example:

```
let num = 42;

console.log(typeof num); // Output: number
```

2. String

Sequence of characters enclosed in single ('), double ("), or backticks (`).
Example:

```
let str = "Hello, World!";

console.log(typeof str); // Output: string
```

3. Boolean

Represents either true or false.
Example:

```
let isActive = true;

console.log(typeof isActive); // Output: boolean
```

4. Undefined

A variable declared but not assigned any value.
Example:

```
let undef;

console.log(typeof undef); // Output: undefined
```

5. Null

Represents an intentional absence of value.
Example:

```
let emptyValue = null;
console.log(typeof emptyValue); // Output: object (a known quirk in JavaScript)
```

6. Symbol

Used to create unique and immutable values, typically for object properties.
Example:

```
let sym = Symbol('unique');
console.log(typeof sym); // Output: symbol
```

7. BigInt

Used for very large integers beyond the range of the Number type.
Example:

```
let bigIntValue = 1234567890123456789012345678901234567890n;
console.log(typeof bigIntValue); // Output: bigint
```

8. Object

A collection of key-value pairs, including arrays and functions.
Example:

```
let obj = { name: "Alice", age: 25 };
console.log(typeof obj); // Output: object

let arr = [1, 2, 3];
console.log(typeof arr); // Output: object
```