let, var and const:

1. let has block scope
2. var has global scope
3. const is block scope and can't be updated

typeof:

```
console.log(typeof 42);            // "number"
console.log(typeof 'hello');       // "string"
console.log(typeof true);          // "boolean"
```

Primitive Data Types

1. Number

Represents both integer and floating-point numbers.

Example:

```
let age = 25;
let temperature = 98.6;
```

2. String

Represents sequences of characters.

Example:

```
let name = "Alice";
let greeting = 'Hello, World!';
```

3. Boolean

Represents logical values: true and false.

Example:

```
let isActive = true;
let isVerified = false;
```

4. Undefined

Represents a variable that has been declared but not yet assigned a value.

Example:

```
let uninitialized;
console.log(uninitialized); // undefined
```

5. Null

Represents the intentional absence of any object value.

Example:

```
let emptyValue = null;
```

6. Symbol

7. BigInt (ES2020)

Represents whole numbers larger than the range supported by the Number type.

Example:

```
let bigInt = 1234567890123456789012345678901234567890n;
```

What is typeof(null)?

-> Object. Reasons are -

1. Initial Implementation Bug: The typeof operator was initially designed to return a type tag for the value provided. For objects, the type tag was 0. Due to a bug, the null value was also assigned a type tag of 0, which corresponded to "object". This mistake was carried over and became part of the language.
2. Legacy Compatibility: By the time this was recognized as an issue, there was already a significant amount of JavaScript code in the wild relying on this behavior. Correcting it would have potentially broken existing web pages and applications. Therefore, it was left unchanged to maintain backward compatibility.

For writing objects in js, we use key value pairs combination, like -

```
let o ={
    "name":"Abhrasnata Ray",
```

```
    "job id": 123456
}

console.log(o)



Output:

    { name: "Abhrasnata Ray", "job id": 123456 }
```

Now suppose we want to add more key value pairs, we write like this:

```
o.salary=500000;
console.log(o);

New output:

    { name: "Abhrasnata Ray", "job id": 123456, salary: 500000 }
```