any value.

But let & const its not possible
let can't be re-declaracred in same scope
const must be intialized when declared.

```
// const x = 5;   let y = 6;

z = x + y;
```

Const vs var vs let

const & var are used to declare the type of variables.

- const is used when variable is not going to change.
- var is used when variable can be changed. (old way). Hoisted with undefin
- Let is used is if variable is not constant (modern appareh)

! Don't use var if you can use let

JavaScript Datatypes

A variable can Interprete according to the data without declaring the data type.

var name; →def
var name="shumaila"; →def
var name; → now don't contain any ve

e.g;

```
<script>
    let carName = "Volvo";
    document.getElementById("demo").
    innerHTML = carName;
</script>
```

String

"Hello", 'World'

Number

42, 3.14, -5

Boolean

true, false

Undefined

A variable declared but not assigned

Null

intentional "" → empty value

Symbol

unique value

BigInt

1234567890123456789012345678901n

```
let str = "Hello";
let num = 42;
let bool = true;
let undef;
let empty = null;
let syb = Symble("id");
let big = 12345678901234567890;

console.log(typeof str); // String
console.log(typeof num); // number
console.log(typeof bool); //boolean
```

```
console.log (typeof udef); //undefine
console.log (typeof empty); //object (bug)

console.log (typeof syb); //symbol
console.log (typeof big); bigint
```

Object & Arrays

Object store data in key: value pairs

```
let object = {nam: "Shumaila",
              age: 21}
```

⇒ uses name to access element

```
let arr = [1, 2, 3, "hello"]
```

⇒ uses index number to access element

Hoisting → var & let

Hoisting means declaring at the top of scope then using the varible without intializing.

→ var can be hoisted but shows undefined value

→ let can be hoisted but stay in Temporeal dead zone (TDZ) until declared. using before can cause error.

```
var a;                          let b;
:                               :
:                               :
:                               :
:                               :
console.log(a);                 console.log(b);
//undefined                     //Reference Error
var a=10;                       let b=20;
```

Undefine vs null

Undefined → variable is declared but not defined

null → explicity set to not.

```
let x;
let y = null;

console.log(x); // undefined
console.log(y); // null
```

Why prefer const?

- Prevent accident reassignment
- Code is more predictable
- Use const by default and let when reassignment needed.