JavaScript Variable Types Complete Guide (var, let, const) - Hindi

---

1️⃣ JavaScript में Variable Types

JavaScript में variables declare करने के तीन main तरीके हैं: 1. var 2. let 3. const

हर एक का **scope, hoisting, reassignability** अलग है।

2️⃣ var

- **Old way** of declaring variables
- Function scoped
- Reassignable and redeclarable
- Hoisting possible (variable undefined but accessible before declaration)

```
function exampleVar() {
  console.log(x); // undefined (hoisted)
  var x = 10;
  x = 20; // ✅ reassign
  var x = 30; // ✅ redeclare allowed
  console.log(x); // 30
}
exampleVar();
```

Scope: Function scope **Best Use:** पुराने projects, avoid in modern JS

3️⃣ let

- **Modern way** for variables
- Block scoped
- Reassignable but not redeclarable in same scope
- Hoisting possible but not initialized (ReferenceError)

```
function exampleLet() {
  // console.log(y); // ❌ ReferenceError
  let y = 10;
  y = 20; // ✅ reassign
  // let y = 30; // ❌ redeclare not allowed
  console.log(y); // 20
}
exampleLet();
```

Scope: Block scope **Best Use:** Dynamic variables, loops, temporary values

---

4️⃣ const

- **Constant variable**
- Block scoped
- Not reassignable, not redeclarable
- Hoisting possible but not initialized (ReferenceError)
- For objects/arrays → modify allowed, reassign not allowed

```
const pi = 3.14;
// pi = 3.1415; ❌ Error

const arr = [1,2,3];
arr.push(4); // ✅ modify allowed
// arr = [5,6]; ❌ Error

const person = {name: 'Sunil'};
person.name = 'Aman'; // ✅ allowed
// person = {name: 'Rakesh'} ❌ Error
```

Scope: Block scope **Best Use:** Fixed values, config, functions, components

---

5️⃣ Scope Comparison

Keyword	Scope	Redeclare	Reassign	Hoisting
var	Function	✅	✅	✅(undefined)
let	Block	❌	✅	✅(ReferenceError)
const	Block	❌	❌	✅(ReferenceError)

Example:

```
if(true){
  var a = 10; // function scope
  let b = 20; // block scope
  const c = 30; // block scope
}
console.log(a); // 10
// console.log(b); // ❌ Error
// console.log(c); // ❌ Error
```

6️⃣ Hoisting Example

```
console.log(x); // undefined
var x = 10;

console.log(y); // ❌ ReferenceError
let y = 20;

console.log(z); // ❌ ReferenceError
const z = 30;
```

- var → hoisted with undefined
- let/const → hoisted but not initialized (TDZ - Temporal Dead Zone)

7️⃣ Project Level Best Practices

1. **Default**: Use `const` for most variables
2. **Dynamic value**: Use `let`
3. **Avoid var**: Unless legacy code
4. **Constants**: Config, API, theme, function references
5. **Loops**: `let` for index, `const` if value fixed per iteration

Example:

```
const API_URL = 'https://api.example.com';
let count = 0;
for(let i=0;i<5;i++){
  const square = i*i;
  console.log(square);
}
```

8️⃣ Summary Table

Keyword	Use Case	Example
var	Old code, function scope	var x = 10; x = 20; var x = 30;
let	Dynamic variable, loops	let y = 10; y = 20;
const	Immutable, fixed values, functions	const PI = 3.14; const arr=[1,2]; arr.push(3);

⬜ **Tip:** Modern JS projects में **mostly const + let** use करें। Var avoid करें।