Blocks :—

→ Blocks are defined by "{ }".

→ It is also known as compound statement.

→ Block is used to ~~define~~ combine multiple JS statement into one group.

e.g.
```
{
        var a = 10;
        console.log(a);
}
```

Q. Why do we need to group multiple statement together in a block?

We group this statements together, so that we can use multiple statements in a place where JS excepts only one statement.

Q. What is Block Scope?

Block scope means what all variables & functions we can acess inside this block is known as block scope.

e.g.

```
{
    Var a = 10;
    let b = 20;      } These two variables are
    const C = 30;    } inside block scope
}
```

Note:- ① We cannot access "let" & "Const" outside the ~~Scope~~ block.
That's why let & const are block scoped.

② "Var" is declared in global scope. So it can be used outside the block.

Q. What is shadowing in JS?

If we have same named variable outside the block, so the variable tries to shadow inside variable.

e.g.

```
Var a = 100;
{
 Var a = 10;          → o/p:-   10
 let b = 20;                    20
 Const c = 30;                  30
 Console.log (a);               10
 Console.log (b);
 Console.log (c);
} Console.log (a);    → outside the block
```

→ In case of "let"

```
let b=100;
{
  var a = 10;
  let b=20;
  const c = 30;
  console.log(a);
  console.log(b);
  console.log(c);
}
Console.log(b);
```

seperate memory space

global scope

block scope

O/p:- 10
20
30
100

↳ Similar in case of "const"

Note:- Shadowing works in the same way in "function" as it works in block.

Q. What is illegal shadowing?

We can not "shadow" let using ~~var~~ "var". It is illegal shadowing.

eg.

```
let a=20;
  {
      var a=20;
  }
```

→ Error.

```
var a =20;
{
  let a =20;
}
```

→ It's true. in this shadowin-g take place.

↳ Const is also same.

↳ "Block Scope" also follows "lexical scope".

e.g. ①

```
Const a=20;
{
 const  a=100;
  {
   const a=200;
   console.log (a);
  }
}
```

→ o/p :- 200

②

```
Const  a=20;
 {
  const a=100;
   {
    Const a=200;
   }
  Console.log (a);
}
```

→ o/p :- 100

③

```
Const  a=20;
{ const a=100;
{ Const a = 200;
 }
}
console.log (a);
```

→ o/p :- 20