# 16.2 Scope and Hoisting

## Block 1

```
var x = 10;
console.log(x);
if (true) {
var x = 20;
console.log(x);
}
console.log(x);
```

The program will output the values: 10 20 and 20. The `x` is declared and initialzied to 10, then it's changed to 20 in the if-statement because a variable declared with `var` is not restrained to the block scope.

## Block 2

```
var x = 10;
console.log(x);
if (true) {
(function() {
var x = 20;
console.log(x);
})();
}
console.log(x);
```

The program outputs the values 10 20 and 10. `x` is declared with `var`, which means that it only follows the function scope, so when `x` is called inside the function, the program will output 20, otherwise it will output 10.

# Block 3

```
var x = 10;
console.log(x);
function test(){
var x = 20;
console.log(x);
if (x > 10) {
let x = 30;
console.log(x);
}
console.log(x);
}
test();
console.log(x);
```

The program outputs 10, 20, 30, 20, 10. At first, `x` is dclared as a `var` and initialized to 10, then it's called. Then it's changed to 20 and called inside the function `test()` and called. Then, it's declared using `let` and set to 30 and called again. Then, it's called outside of the if-statement which would output 20 because `let` is limited to the block scope and outside of the previous block, it was set to 20. Finally, `x` is called again outside of the function which would output 10 because `var` is limited to the function scope.

# Block 4

```
var x;
x = 10;
function test(){
var x;
if (x > 20) {
x = 50;
}
console.log(x);
}
test();
```

The program outputs `undefined`. `x` is declared as `var` and the value of 10 was assigned to it but, it was called inside the function `test()` which sets it as a `var` again and assigns a value to it only if its ititial value is more than 20, which is not. Then it's called inside the function. This means that the program is trying to call `x` inside the function and that outputs `undefined` because `var` is limited to function scope.

# Block 5

```
function test(){
var x, y;
if (false) {
x = 50;
}
console.log(x);
console.log(y);
y = 100;
console.log(y);
}
test()
```

The program outputs `undefined`, `undefined`, `100`. All variables are declared, initialized and called inside the function. So, the function starts by only declaring them without assigning any values to them. Then, it doesn't assign anything to `x` in the if-statement because the condition for it is false. Then, it calls the variables that still have `undefined` values. In the end, it changes the value of `y` to 100 and calls it for the output.

# Block 6

```
function test(){
foo();
bar();
// Function defiened
// using function declaration
function foo(){
console.log('foo');
}
// Function defined
// using function expression
var bar = function() {
console.log('bar');
}
}
test();
```

The program outputs foo and an error

```
Uncaught TypeError: bar is not a function
    at test (<anonymous>:3:1)
    at <anonymous>:15:1
```

This is because the function `bar()` is called inside the function `test()` and it is decalred using a function expression with `var` . This means that at thhe beginning, `bar` has an undefined value until the program gets to the line where the function is assigned to it and the function is called before that assignment, which means it's undefined at that point