16.1 Scope and Hoisting

Block 1

```
function funcA() {
console.log(a);
console.log(foo());
var a = 1;
function foo() {
return 2;
}
}
funcA();
```

The program will output `undefined` and 2.
Since `a` is declared using `var` and `foo()` is a function, they're hoisted by the interpreter when the program starts. But because `a` is passed to `console.log()` before assigning a value to it, the program outputs `undefined`.

Block 2

```
var fullName = 'John Doe';
var obj = {
fullName: 'Colin Ihrig',
prop: {
fullName: 'Aurelio De Rosa',
getFullName: function () {
return this.fullName;
}
}
};
console.log(obj.prop.getFullName());
var test = obj.prop.getFullName;
console.log(test());
```

The program outputs `Aurelio De Rosa` and `John Doe`.
`console.log(obj.prop.getFullName())` outputs the `fullName` of `obj.prop` because the function looks for the closest `fullName` to its scope.

On the other hand, `test` saves a refernce to the function `getFullName`, which if executed, returns the closest `fullName` to it which is the one at the beggining of the program `'John Doe'`.

Block 3

```
function funcB(){
    let a = b = 0;
    a++;
}
funcB();
console.log(typeof a);
console.log(typeof b);
```

The program outputs the type of `a` as `undefined` and `b` asa number, and that because the line `let a = b = 0;` declares `a` as a variable in the function scope with initial value of 0 while implicitly declaring `b` as a global variable with a value of 0.

Block 4

```
function funcC() {
console.log("1");
}
funcC();
function funcC() {
console.log("2");
}
funcC();
```

The program outputs the number 2 twice and that's because when the program runs, the interpreter hoists all function and uses the last declarations of functions with same name.

Block 5

```
function funcD1() {
d = 1;
}
funcD1();
console.log(d);
function funcD2() {
var e = 1;
}
funcD2();
console.log(e);
```

The program outputs the number 1 and thows an error stating that the variable `e` is not defined.

```
Uncaught ReferenceError: e is not defined
    at <anonymous>:10:13
```

That's because the variable `d` is implicitly declared as a global variable while `e` is declared inside the function `funcD2()` and is accessible only inside its scope.
To be able to access variable `e` outside of the function, it should be declared globally.

Block 6

```
function funcE() {
console.log("Value of f in local scope: ", f);
}
console.log("Value of f in global scope: ", f);
var f = 1;
funcE();
```

The program outputs `undefined` and the number 1. That's because when it starts, the variable `f` is hoisted and initialized with `undefined` value. The value changes when the program gets to the line where the value 1 is assigned to it.