How the Function Works:

First Code

```javascript
var x = 1;
a();
b();
console.log(x);

function a() {
  var x = 10;
  console.log(x);
}

function b() {
  var x = 100;
  console.log(x);
}
```

Output:

```
10
100
1
```

Explanation:

1. **Global Execution Context:**

   - `var x = 1;` declares a global variable `x` with an initial value of `1`.

   - Functions `a` and `b` are hoisted (meaning they are moved to the top of the context during the compile phase).

2. **Function `a()` Execution:**

   - When `a()` is called, the function creates its own execution context.

   - Inside the function, the statement `var x = 10;` declares a local variable `x` within the function's scope.

   - The local `x` shadows the global `x`, so `console.log(x);` prints `10`.

3. **Function `b()` Execution:**

   - Similar to `a()`, calling `b()` creates its own execution context.

   - The local variable `x` within `b()` is assigned the value `100`.

   - `console.log(x);` prints `100`.

4. **Global `console.log(x)` Execution:**

   - After both functions are called, `console.log(x);` in the global scope prints `1` because the global variable `x` remains unchanged by the local variables in functions `a()` and `b()`.

Second Code

```javascript
var x = 1;
a();
b();
console.log(x);

function a() {
   x = 10;
   console.log(x);
}

function b() {
   x = 100;
   console.log(x);
}
```

Output:

```
10

100

100
```

Explanation:

1. **Global Execution Context:**

   - `var x = 1;` declares a global variable `x` with an initial value of `1`.

   - Functions `a` and `b` are hoisted.

2. **Function `a()` Execution:**

   - When `a()` is called, it does not create a new local variable `x` using `var`. Instead, it directly assigns `10` to the global variable `x`.

   - `console.log(x);` prints `10`, reflecting the updated global value.

3. **Function `b()` Execution:**

   - Similarly, `b()` directly assigns `100` to the global variable `x`.

   - `console.log(x);` prints `100`, showing the updated global value.

4. **Global `console.log(x)` Execution:**

   - After both functions are called, `console.log(x);` in the global scope prints `100` because the global variable `x` has been updated by the functions `a()` and `b()`.

Summary of Global Execution Context:

- **First Code:** The global variable `x` is not affected by the local variables declared within the functions. The global `x` retains its original value.

- **Second Code:** The global variable `x` is modified by the functions because the assignments inside the functions do not use `var` to declare a new variable, thus modifying the global `x`.