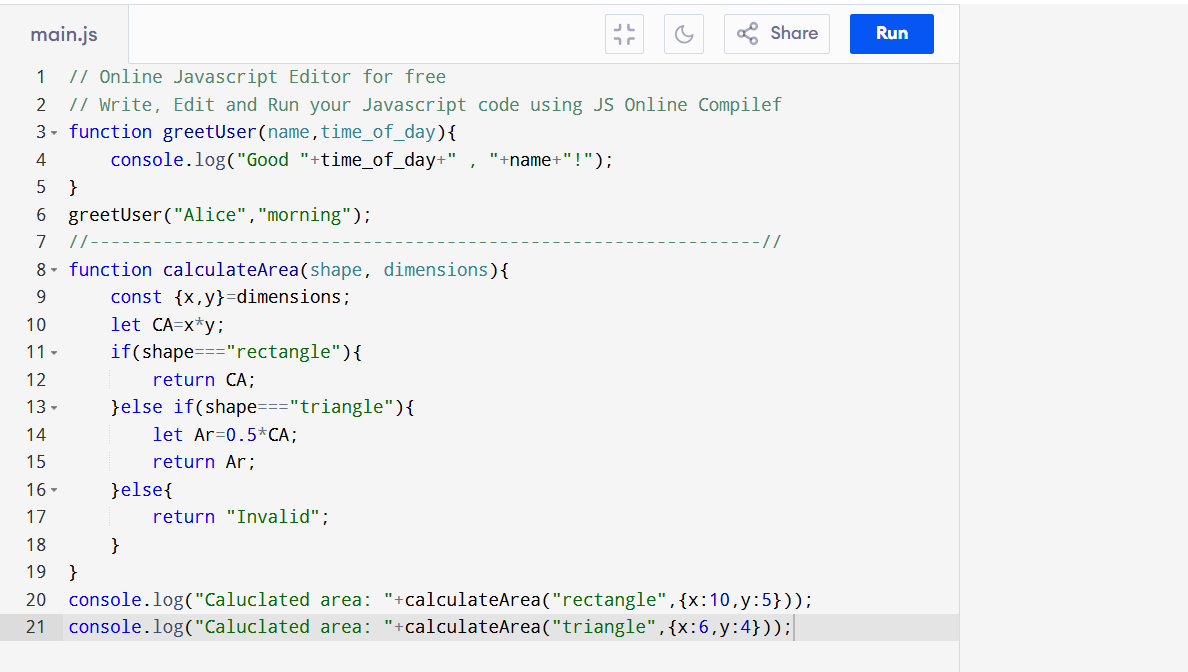
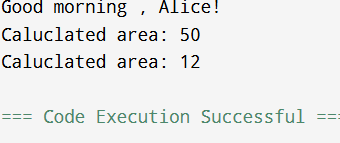
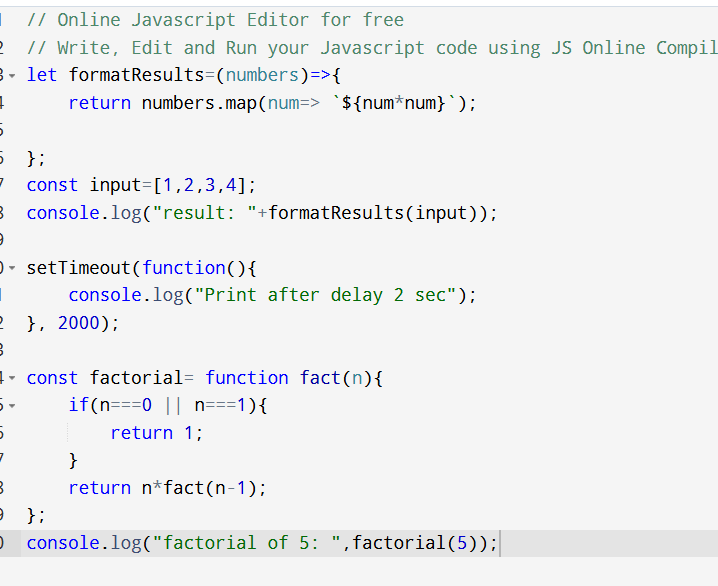
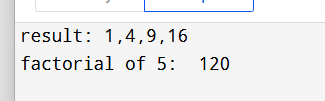
JS Function-objects-3

Function- types-Implementation:









Scope:

let globalVar = "Global Scope";

function testScope() {

let functionVar = "Function Scope";

if (true) {

let blockLet = "Block Scope (let)";

const blockConst = "Block Scope (const)";

var blockVar = "Function Scope (var)";

console.log(blockLet, blockConst, blockVar);

}

console.log(functionVar); // Accessible — declared in function scope

console.log(blockVar); // Accessible — var is function scoped

// console.log(blockLet); // Error — blockLet is block scoped

}

testScope();

console.log(globalVar); // Accessible — global scope

// console.log(functionVar); // Error — not accessible outside the function

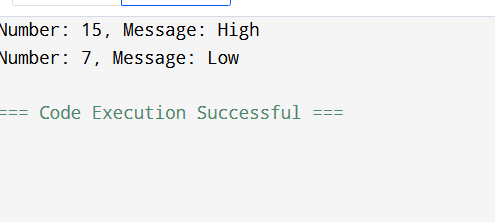
|  |  |
| --- | --- |
| globalVar | - Everywhere (global scope) |

|  |  |
| --- | --- |
| functionVar | -Inside testScope only (function scope) |

|  |  |
| --- | --- |
| blockLet | -Only inside the if block (block scope) |

|  |  |
| --- | --- |
| blockConst | - Only inside the if block (block scope) |

|  |  |
| --- | --- |
| blockVar | - Anywhere inside testScope (function scope via var) |
| Let keyword has a block scope type can be reassign and cannot be redeclared in same scope.  Let const keyword has a block scope and cannot be reassign and cannot be redeclared in the same scope.  Let var keyword is the function scope (ignores block scope) can be reassign and can be redeclared in same scope.  Callback function: |  |



This keyword:



