Scope-Global and Local Scope

In JavaScript, scope refers to the accessibility of variables, functions, and objects in some particular part of your code during runtime. There are two types of scope in JavaScript: global scope and local scope.

Global scope refers to the top-level scope in a JavaScript program. Variables and functions defined in the global scope are accessible from anywhere in the code. For example:

```
var globalVar1 = "global variable";
function globalFunc1() {
    console.log("global function");
}
console.log(globalVar1); // logs "global variable"
globalFunc1(); // logs "global function"
```

Local scope, on the other hand, refers to the scope within a function. Variables and functions defined within a function are only accessible within that function, and are not accessible from the global scope. For example:

```
function myFunc2() {
    var localVar1 = "local variable";
    function localFunc1() {
        console.log("local function");
    }
    console.log(localVar);
    localFunc1();
}
console.log(localVar1); // ReferenceError: localVar is not defined
localFunc1();
```

In this example, the variable localVar1 and the function localFunc1 are only accessible within the myFunc2 function and are not accessible from the global scope.

Rules for Scoping in JS

1. Variables and functions defined in the global scope are accessible from anywhere in the code.
2. Variables and functions defined within a function are only accessible within that function, and are not accessible from the global scope.
3. JavaScript has a mechanism called "variable hoisting" which means that variable and function declarations are moved to the top of the scope. This means that variables and functions can be accessed before they are defined in the code.
4. If a variable is defined with the var keyword within a function, it's accessible within the entire function, including nested functions.
5. If a variable is defined with the let or const keyword within a block, it's only accessible within that block and its nested blocks.
6. If a variable is defined with the same name in the local and global scope, the local variable takes precedence over the global variable.