
2022 Spring

CS315

Homework Assignment 1

Scoping in Dart, Javascript, Perl, PHP and Python

Name: Ahmet Faruk

Surname: Ulutaş

Section: 1

ID: 21803717

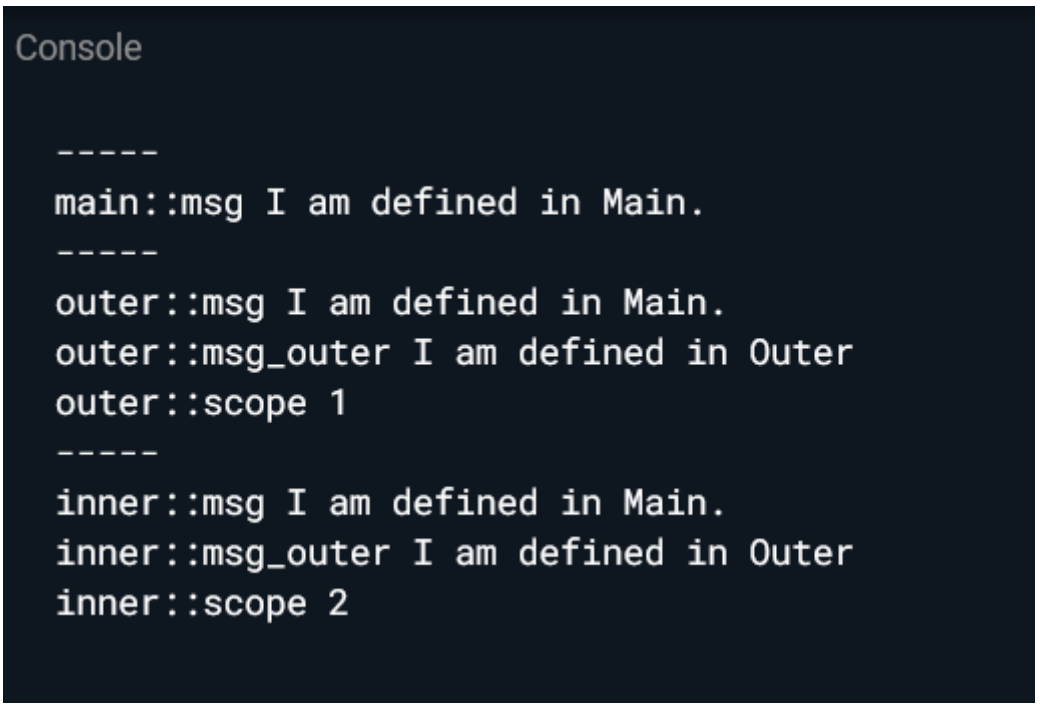
TA Name: Irmak Türköz

# Dart

## Full Code Example:

```dart
// In this main class, scope analysis
// of the Dart language is performed.
main() {
  // Initialize local variable inside Main
  String msg = "I am defined in Main.";
  void outer() {
    // Initialize local variable inside outer
    int scope = 1;
    String msgOuter = "I am defined in Outer";

    void inner() {
      // redefining scope variable found in outer()
      // It will work until it disappears where it is defined.
      scope = 2; // static (lexical) scoping

      print('-----');
      print('inner::msg ' + msg); // prints I am defined in Main.
      print('inner::msg_outer ' + msgOuter); // prints I am defined in Outer
      print('inner::scope $scope'); // prints 2
    }

    // Each part can access variables defined in its own scope and outside scopes.
    print('-----');
    print('outer::msg ' + msg); // prints I am defined in Main.
    print('outer::msg_outer ' + msgOuter); // prints I am defined in Outer
    print('outer::scope $scope'); // prints 1
    inner();
  }

  print('-----');
  print('main::msg ' + msg); // prints I am defined in Main.
  //print('main::msg_outer ' + msg_outer); // Undefined name msg_outer.
  //print('main::scope + $scope'); // Undefined name 'scope'.
  outer();
}
```

**Sample Output For Code:**




Dart language is generally a language based on lexical scope. I focused on this issue in the program I wrote. For the program to run, it must be written inside the main function, just like in C languages. When nested subprograms are defined, the validity of the outer scope continues in every inner scope. That is, variables defined outside work inside unless otherwise specified. However, variables defined inside cannot be accessed externally. Undefined name 'var_name' in Dart if attempting to access a scope that cannot be accessed internally from outside. A similar error is encountered. Therefore, there is no output of the variables of the subprograms among the main outputs in the output part.

# JavaScript

## Full Code Example:

```
    <script>
      // In this program, scope analysis
      // of the Javascript language is performed.

      // Initialize a global variable.
      var msg = "I am global";

      function changeMsg() {
        // Initialize a local, function-scoped variable. (static scoping)
        var msg = "I am in changeMsg";
        console.log(`Inside of changeMsg function-scoped: ${msg}.`);
      }

      // if variable is not defined; ReferenceError: variable is not defined
      console.log(`Inside of main: ${undefined_msg}.`) // prints undefined
      var undefined_msg = "defined"

      console.log(`Inside of main: ${msg}.`); // prints I am global
      changeMsg() // (function-scoped) print I am in changeMsg

      if (true) {
        // Initialize a block-scoped variable (static scoping)
        var msg = "I am in if condition";
        // prints block-scoped msg; I am in if condition
        console.log(`Inside of if condition block-scoped: ${msg}.`);
      }

      function scope_dyn() {
        var msg = "Dynnamic scoping"

        print_scope_dyn() // prints I am in if condition since there is no global keyword.
        // and there is no dynnamic scoping. It is static scoping.
      }

      function print_scope_dyn() {
        console.log("msg: "+ msg+"\n")
      }

      scope_dyn()
      // If any variable is defined, it cannot be declared as let once again.
      // let msg = "A" // SyntaxError: Identifier 'msg' has already been declared
      // prints last assigned msg I am in if condition
      console.log(`Inside main after block-scoped: ${msg}`)
    </script>
  </head>
</html>
```

**Sample Output For Code:**

```
Inside of main: undefined.                                  Ulutaş_AhmetFaruk_javascript.html:18:15
Inside of main: I am global.                                Ulutaş_AhmetFaruk_javascript.html:21:15
Inside of changeMsg function-scoped: I am in changeMsg.     Ulutaş_AhmetFaruk_javascript.html:14:17
Inside of if condition block-scoped: I am in if condition.  Ulutaş_AhmetFaruk_javascript.html:28:17
msg: I am in if condition                                   Ulutaş_AhmetFaruk_javascript.html:39:17
Inside main after block-scoped: I am in if condition        Ulutaş_AhmetFaruk_javascript.html:46:15
```

In order to test the program in JavaScript, I added the program code that I wrote inside the script tag within the HTML tags. There are three types of variable declarations in JavaScript, var, let and const. Unlimited definitions can be made with var, while a previously defined variable with let cannot be defined in the same scope. When the function and block scope are used, static scoping is done. Also, when a return is requested from a function in which no variable is defined, the function accesses global variables and performs static scoping. When trying to get console output with a variable that has not been defined yet, it gives an error that the variable is not defined.

# Perl

## Full Code Example:

```perl
# In this program, scope analysis
# of the Perl language is performed.

# For a variable to be printed and found, it must be defined before print.
$other_name = "name before print"; # prints name before print
print "$other_name\n";
print "$name\n"; # prints new line (\n)

{
    $name = "name after print";
}

# declaration of global variable
$msg = "This is global msg.";
$count = 1;

# printing global variables
print $count." ".$msg."\n"; # prints 1 This is global msg.

# increment count to 2
$count++;

# block starting
{
    # Where the word my is used, a static variable is created and within its scope it does static scoping instead of a global variable.
    # The variable is defined as local when the keyword my is used.
    my $new_msg = "This is local msg.";

    # global variables can be accessed from different scopes.
    print $count." ".$msg."\n"; # prints 2 This is global msg.

    # increment count to 3
    $count++;

    print $count." ".$new_msg."\n"; # prints 3 This is local msg.

    # increment count to 4
    $count++;
}

```

```perl
# new_msg defined with "my" as local cannot be used outside of the block.
print "This is local message outside of block:".$new_msg."\n"; # not prints message

# declaring static scoping function
sub func {
    # Where the word my is used, a static variable is created and within its scope it does static scoping instead of a global variable.
    # If "my" keyword is used with the same variable name,
    # it replaces the global variable where it is defined.
    my $msg = "This is local func msg with global variable name";
    print $count." ".$msg."\n"; # prints 4 This is local func msg with global variable name
    $count++;
}

# calling the function
func();
print $count." ".$msg."\n"; # prints This is global msg.

# The package is defined using the package keyword
package OurPack;
    # It remains in the package scope defined in the package
    # unless the keyword our is used.
    our $our_msg; # Dynamic scoping is started with the use of our keyword and our_msg variable is used unless otherwise stated.
    $our_msg = "This is our msg";

    # It remains in the package scope defined in the package
    $packMsg;
    $packMsg = "Package's variable";

# The package is defined using the package keyword
package Pack;
    print "Value of our msg from OurPack: ".$our_msg."\n"; # prints This is our msg
    print "Value of main msg: ".$main::msg."\n"; # prints This is global msg.
    print "Value of pack msg from OurPack: ".$packMsg."\n"; # prints new line
```

**Sample Output For Code:**

```
$perl main.pl
name before print

1 This is global msg.
2 This is global msg.
3 This is local msg.
This is local message outside of block:
4 This is local func msg with global variable name
5 This is global msg.
Value of our msg from OurPack: This is our msg
Value of main msg: This is global msg.
Value of pack msg from OurPack:
```

Subprograms can be created in Perl using curly braces. Global variables can be accessed globally using static scoping. If a variable is not wanted to be accessed from outside scopes, it can be made localised by using the my keyword. In addition, there is a keyword called package in the perl language to be able to define static scoping and global variables. Variables defined with the keyword our under the package keyword become accessible from all other scopes without identification. If the keyword Our is not used, access from other packages is not possible.

# PHP

## Full Code Example:

```php
<!DOCTYPE html>
<html>
<body>
<?php
// In this program, the scope of the php language is examined.
/* global scope */
$und_msg = "und msg\n";
$glb_msg = "glb\n";

/*
No global and non-static variables can be accessed from different scopes.
function outer()
{
    $outer_msg = "outer_msg";
    function inner()
    {
        $inner_msg = "inner msg";
        echo "In inner func; outer_msg: $outer_msg, inner_msg: $inner_msg\n"; // PHP Warning:  Undefined variable $outer_msg
    }

    echo "In outer func; outer_msg: $outer_msg, inner_msg: $inner_msg\n"; // PHP Warning:  Undefined variable $inner_msg
    inner();
}
*/

// Before using the global keyword, the glb variable was printed by dynamic scoping
// while it was statically searched and undefined variable was given an error.
function global_scope_test()
{
    // This gives "PHP Warning: Undefined variable $und_msg"
    // echo $und_msg; /* reference to local scope variable */

    global $glb_msg;
    echo $glb_msg; // prints glb since it is defined in func scope as global
    echo $GLOBALS['und_msg']; // prints und msg
}

// Since it is looked at statically here, it is first checked to see
// if there are counters in its scope and that variable is printed.
function static_test()
{
    static $counter = 0;
    echo "Static Test Counter: $counter\n"; // prints counter i.e. 0 then 1
    count_static();
}
```

```php

// Since it is looked at statically here, it is first checked to see
// if there are counters in its scope and that variable is printed.
function count_static()
{
    $counter = 5;
    echo "Static Test Counter: $counter\n"; // prints counter i.e. 0 then 1
}

// It does dynamic scoping because the global keyword is used.
$non_static_counter = 3;
function non_static_test()
{
    global $non_static_counter;
    //$non_static_counter = 0; // Since the variable is defined, it does lexical scoping.
    echo "Non Static Test Counter: $non_static_counter\n"; // prints 0
    $non_static_counter++;
}

/*
function static_variables(){
    static $int = 0; // OK.
    // Static variables run in compile-time so
    static $int = sqrt(121); // PHP Fatal error: Cannot redeclare sqrt()

    echo $int;
}
*/

global_scope_test();
static_test();
static_test();
non_static_test();
non_static_test();
// echo $counter, $non_static_counter\n" // Variables defined inside the function cannot be accessed externally
// outer(); // No global and non-static variables can be accessed from different scopes.
?>
</body>
</html>
```

Output:

```
<!DOCTYPE html>
<html>
<body>
glb
und msg
Static Test Counter: 0
Static Test Counter: 5
Static Test Counter: 0
Static Test Counter: 5
Non Static Test Counter: 3
Non Static Test Counter: 4
</body>
</html>
```

In PHP language, we wrote it in HTML tags so that the program can be tested, as in Js language. In this language, global and non-static variables become inaccessible from other scopes. The global keyword is used for a variable to be visible in another scope. This provides static scoping. Dynamic scoping can be achieved with the static keyword. Also, variables defined in subprograms or functions become inaccessible from outside scopes.

**Python**

**Full Code Example:**

```python
# This program examines scoping in python.
msg = "global scope msg"

# Since no global variable is defined in the static scope function,
# it is looking for the msg variable in its scope.
def static_scope():
    msg = "local scope msg" # defining local variable
    print(msg) # prints local scope msg
    def inner_local_scope():
        print("from inner", msg) # prints from inner local scope msg
    inner_local_scope()

# In the dynamic scope function, since msg is defined with the global keyword
# inside the inner function, the print statement looks at the global variable.
def dynammic_scope():
    msg = "local scope msg" # defining local variable
    print(msg) # prints local scope msg
    def inner_glb_scope():
        global msg
        print("from inner", msg) # prints from inner local scope msg
    inner_glb_scope()

glb_msg = "global message" # defining global variable
def global_scope():
    print(glb_msg) # prints global message

# create global varibale in local scope by using global keyword
def global_keyword():
    global glb_msg
    glb_msg = "This is new global message from local"

static_scope()
dynammic_scope()
global_scope()
global_keyword()
print("In main:", glb_msg) # prints In main: This is new global message from local
```

**Sample Output For Code:**

```
local scope msg
from inner local scope msg
local scope msg
from inner global scope msg
global message
In main: This is new global message from local
>
```

Python is one of the easiest languages to understand. It does dynamic scoping unless a variable with a similar signature is defined. Static scoping can be achieved using the global keyword. I have included these in my program with a few examples.

# My Learning Strategy

I started homework. In the Assignment, she wants me to research the dynamic and static scopes of 5 different programming languages, Dart, Javascript, Perl, PHP and Python, to write sample programs related to the research, and to test and analyse the outputs. I thought the assignment was pretty easy. First of all, I repeated the subject of scope that we covered in the lesson. Then I read the coverage summary for each of these 5 different languages on the internet. After completing this, I examined the syntaxes of the languages in order. They have very similar syntaxes, and the general logic remains mostly unchanged when it comes to scope. I've seen that some languages have special keywords in their syntax that are given to variables so that they can be scoped. It was enough to just look at the syntax of the languages on the information we saw in the lesson. However, I reviewed the scope binding analyses for each language in summary form on the internet. I then wrote sample scripts for each language. While writing the sample scripts, I tried to review possible scenarios and tested the scenarios I determined in other languages. It was a little troublesome to write different examples for each language. In general, I have studied calling a global variable in nested functions and calling variables defined in static scope. In addition to this, I also considered possible different scope cases. It was quite similar as we encountered questions such as js, Perl, and python in the midterm exam. I think the assignment is quite instructive. Because when you study each language, you review the syntax related to the subject. You also need to debug the code to understand its scope. The fact that the Dart language works in a completely lexical scope felt different. Being able to do dynamic scope differently in Perl, that is, defining the packages and our keyword is a different approach. Among them, python was my favourite. It's a streamlined language. As with others undefined etc. no problems. If we do not count some cases, everything can pass to different scopes as it is. In addition to these, I added HTML tags to PHP and js files and wrote comments for all cases. It made me think again about this part and helped me grasp the subject. Then I uploaded the programs I wrote to the Dijkstra server using FileZilla. Then I connected to the Dijkstra with putty and tested the programs one by one. A lot of people in the class were talking about creating and testing the public HTML file. However, when such topics are searched on the internet, they appear directly, so I used internet searches at this point. While testing in Dijkstra I noticed that I had set the HTML tags incorrectly. Then I checked and rearranged it with its original form on the internet. While rechecking my codes before uploading, I noticed some mistakes in the comments. I have completed my homework by correcting the errors and omissions in the comments.