Çağatay Şafak

Professor: Halil Altay Güvenir

CS315 - Programming Languages

April 12th, 2022

Scoping in Dart, JavaScript, Perl, PHP, and Python

In this homework, I am asked to observe the concepts of static and dynamic scoping by analyzing the differences between the programming languages Dart, JavaScript, Perl, PHP, and Python.

Scoping: Static vs. Dynamic

In *lexical* scoping, the interpreter searches the function definition in the local function first. Afterwards, it searches in the scope where that function was defined, then in the scope where that function was defined, and so forth. The word 'lexical' means something related to words or vocabulary of a language, so here is the context. The values of the variables in the called scope remain the same, even if it has been called from different scopes.

In *dynamic* scoping, the compiler searches in the local function first, just like in the lexical scoping. However, when there appears a failure in searching the definition in the local function, then the compiler goes on searching in the function or scope, which has called the function of interest. When there appears a failure again, it searches in the function that called that function, and it goes on. The word "Dynamic" means a system or a process, which is open for constant changes, so here is the context. The values of the variables in the called scope can be changed depending on the scope where it is called.

Scoping in Dart

Dart is a lexically scoped language by design. This means that the Dart compiler follows the pattern starting from the most inner scope to the outer ones. In Dart language, a variable in a scope is called statically from the other scopes.

For instance, here is an example of a bunch of code that I wrote in order to explain the topic in detail.

```
String globalVariable = 'global\n';

void func_1()
{
  String outerVariable = 'outer\n';
  print(globalVariable);
  print(outerVariable);

  func_2()
  {
    String innerVariable = 'inner\n';
    print(globalVariable);
    print(outerVariable);

    func_3()
    {
      print(globalVariable);
      print(outerVariable);
      print(innerVariable);
    }

    func_3();
  }
  /* print(innerVariable); */ // this line gives an error.

  func_2();
}

void main() => func_1();
```

On the line which is written in red, an error would appear, since `innerVariable` is defined in a scope which is inner than the scope it is called.

Scoping in JavaScript

Firstly, one may find it useful to know that JavaScript commands do not give errors, they are meant to run even in the toughest possible situation. Although defining variables and functions are very flexible in JavaScript, dynamic scoping is not included in the language. Here follows an example of static scoping in JavaScript.

```
var name = "cagatay";

function rename() {
    var name = "caca";
    console.log(name);
}

console.log(name);
rename();
console.log(name);
```

The output will be “cagatay, caca, cagatay”. When we use **`let`** instead of **`var`**, then it changes. Here is an example.

```
var ta = "irmak";

if (isGrading)
{
    var ta = "turkoz";
    console.log(`name1 = ${ta}`); // prints name1 = turkoz
}

console.log(`name2 = ${ta}`); // prints name2 = turkoz

___________________________________________________________

let ta = "irmak";

if (isGrading)
{
    let ta = "turkoz";
    console.log(`name1 = ${ta}`); // prints name1 = irmak
}

console.log(`name2 = ${ta}`); // prints name2 = turkoz
```

Scoping in Perl

In the Perl language, dynamic and static scoping are much more clear than the other languages, since the variable definitions play an important role in scoping. Here is an example of dynamic and static scoping in Perl.

```
$a = 10;              # global
sub foo
{
      return $a;
}

sub staticScope
{
      my $a = 99;
      return foo();
}

print staticScope(); # a is returned as '10'
$b = 100;             # global

sub bar
{
      return $b;
}

sub dynamicScope
{
      local $b = 999;
      return bar();
}

print dynamicScope(); # a is returned as '999'
```

Perl Keywords

Obviously, the keyword '**`local`**' refers to dynamic scoping and the keyword '**`my`**' refers to dynamic scoping. The value of a does not change due to where it is called, however, the value of b changes.

Scoping in PHP

PHP does also allow only lexical scoping. Different from the previous ones, PHP scopes cannot see the variables, or functions, defined on the higher scopes. Here is an example.

```
<?php
$globalVariable = 666;

function test_1()
{
    echo $globalVariable;   // gives an error.

    $innerVariable = 999;

    function test_2()
    {
        echo $innerVariable;     // gives an error.
    }
    test_2();
}

test_1();
?>
```

For the lines written in red, the code gives ‘Undefined variable error’, since the **`globalVariable`** and **`innerVariable`** are defined in outer (global for this example) scopes.

In the PHP language, the compiler can see the global variables by accessing a special **`GLOBALS`** array.

```
<?php
function main()
{
    $city = "Ankara";

    echo 'global city: ' . $GLOBALS["city"] . "\n";
    echo 'local city: ' . $city;
}

$city = "Tekirdag";
```

```
main();
?>
```

This program will be compiled and run perfectly, and output that Ankara is a local city and Tekirdag is a global city.

Scoping in Python

The version of the language to be discussed is Python-3. In Python, there is both dynamic and lexical scoping, just as in Perl. There are four kinds of variable scopes in the language, built-in, global scope, enclosing scope, and local scope. Here is an example of local scope in Python.

```
>>> x = 10
>>> def func_1():
           x = 15
           print(a) # gives an error.
           x = 20
>>> func_1()
```

Here, the print statement gives an error since it tries to a variable that is not accepted as a global variable anymore.

Python Keywords

There are **`global`** and **`nonlocal`** keywords in the language. Here are two examples of them.

```
>>> y = 33
>>> def func_2():
           global y
           y = 66
           print(a)              # prints '66'
>>> func_2()

----------------------------------------------

>>> def func_3():
           z = 200
           def func_4():
```

```
                        nonlocal z
                        z = 300
                        w = 300
                        print(z)   # prints '300'
                        print(w)   # prints '300'
            func_4()
            print(a)               # prints '300'
>>> func_3()
```

In the 2nd example, if the **`nonlocal`** keyword was not there, then the third print statement would print '200' as output.

Studying Scoping

For one week, I have been studying from many different sources from different platforms, and I have had different methods of studying for almost every language.

The Sources

I have studied from the codes in dijkstra server, and on many websites. I compiled all the code blocks that I saw on the web, and tried to compile them in online compilers, then, by using what I have learned, I wrote original code. Stack OverFlow was one of the sources that I have used most.

Other than the online sources, I have focused on what the courses give to me. From CS315 courses, Professor Guvenir teached what the keywords in JavaScript, Python 3, and Perl are used for, and asked them in midterm also. Thus, I was already experienced with them, but PHP was a bit newer for me amongst them.

I also checked our textbook, however, it did not bring many things to this homework. It helped me a lot when I was studying for the midterm, nonetheless.

The Languages

I, firstly, want to discuss my strategy on the Dart language. Since I have been working as a Flutter developer for almost six months, I already knew what the syntax and basics are, but I still learned many things about the language, thanks to this homework. I think Dart is designed to develop the front end of the applications, so both its syntax and scoping rules are evolved in this direction. There are some scoping rules that solves many problems during Android development.

For PHP and Python, I was completely lost at the first moment. I still do not feel like a professional at them, definitely. However, I like the Perl language, even though I have had literally zero experience with it.

For my CS319 project, I have been learning JavaScript for a couple of weeks, since we are to use React.JS. However, to know only a framework of a language is completely different from to know its scoping rules. In React.JS, most of the hard work is not about variables, thus, I did not know much about them at the start.

Communication

I communicated with some of my friends about how they study, like, asking them which websites they are using, however, there was not a situation of collaboration, of course. All the code that I wrote are original, and their discussions, descriptions and comments are so.

Experimenting

I used online compilers in the first place, when I was first learning the concepts. The reason for using online compilers instead of the dijkstra machine is that they are simply easier to use. dartpad.dev was the only online compiler that I used for Dart language. For Perl and Python 3, I used onlinegdb.com. For PHP and JavaScript, I used w3schools.com. W3Schools is a website where one can learn about web development, thus, it was useful for JavaScript and PHP.

Conclusion

Except the Dart language, 90% of my knowledge for the topic of this homework came from web sites.