2021–2022 SPRING SEMESTER

CS315 – HW1

Scoping in Dart, Javascript, Perl, PHP and Python

NAME: ABDULLAH RIAZ

ID : 22001296

COURSE:CS315 - PROGRAMMING LANGUAGES

SECTION: 01

DATE: 10/04/2022

Dart

Scoping Rule: Static Scoping

```
main() {

var x = "Global X";

foo(){
      x = "foo local";
      print("\n In Top Foo: " + x);


      void nestedFoo(){
      x = "nestedFoo X var";
      print("\n In nested Foo: " + x);

      }


      void nestedfooCaller(){
      x = "\n foo caller x var";
      nestedFoo();

      }

      nestedFoo();
      nestedfooCaller();
}

foo();

}
```

Dart, like other modern programming languages, uses static scoping. This means a nested function, *nestedFoo* in this case, can access its closest *x* variable from both inside its own scope and its parent functions. In this case, despite x being declared in *nestedfooCaller, nestedFoo* prints out the x declared locally inside itself. In case, x is not defined inside the function(or commented out) it would print the x defined in *foo.* If that is also not defined it keeps on going above till it finds one such as the global x variable. If it doesn't find any, it gives out an error.

JavaScript

Scoping Rule: Static Scoping

```
var x = "Global X"

function foo(){
      x = "foo local"  ;
      console.log("\nIn Top Foo: ", x);


      function nestedFoo(){
      x = "nestedFoo X var"
      console.log("\nIn nested Foo: ", x)
      };

      function nestedfooCaller(){
      x = "\nfoo caller x var"
      nestedFoo()
      };

      nestedFoo();
      nestedfooCaller();
}

foo();
```

Similar to Dart and other modern programing language, Javascript also has static scoping which means it has a inside to top approach where it checks the *nestedfoo* for a declared x variable and keep on going above till it finds an x variable

Perl

Scoping Rule: Both Dynamic and Static Scoping

```
$x = "Global X";

sub foo{
      $x = "foo local"  ;
      print "\nIn Top Foo:  $x" ;


      sub nestedFoo{
      # $x = "nestedFoo X var";   commented out to demonstrate dynamic
scope
      print "\nIn nested Foo:  $x";
      }

      sub nestedfooCaller{
      local $x = "foo caller x var";
      nestedFoo();
      }

      nestedFoo();
      nestedfooCaller();
}

foo();
```

Perl is a unique language compared to the rest of the languages discussed here. It supports both dynamic scoping(using local keyword) and static scoping. Compared to the rest of the languages here, when x is declared using local inside nestedfooCaller, the nestedFoo function first looks inside itself for x and then moves on to nestedfooCaller(its caller) to check for the availability of the x variable. This is the effect of dynamic scoping, where apart from function checking availability of variables inside itself, it also checks the caller function for the variable.

In the snippet below where the local identifier is removed from the x variable in *nestedFooCaller,* it behaves like a static scoped language.

```
$x = "Global X";

sub foo{
      $x = "foo local"  ;
      print "\nIn Top Foo:  $x" ;


      sub nestedFoo{
      $x = "nestedFoo X var";
      print "\nIn nested Foo:  $x";
      }
      sub nestedfooCaller{
      my $x = "foo caller x var";
      nestedFoo();
      }

      nestedFoo();
      nestedfooCaller();
}

foo();
```

PHP

Scoping Rule: Static Scoping

```
<?php

function foo() {

      $x = "foo local";
      echo "In Top Foo: " . $x;

      function nestedFoo() {
      $x = "nestedFood X var";
      echo "In nested Foo: " . $x;
      }

      function nestedfooCaller() { // calls the nestedFoo Function
      $x = "foo caller x var";
      // nestedFoo();     //this cannot be called because php has function
scope
      }

      nestedFoo();
      nestedfooCaller();

}

foo();
```

```
?>
```

PHP language is different from other languages in that although it behaves similarly to static scoped languages, it also has function scoping that affects the usage of variables. If the *nestedFoo()* is uncommented in the code snippet above, it gives an error while running. Function scope means that unless nestedFoo() is not declared inside the caller, it will not run. This is why it has no issue being called from inside the foo() parent function

Python

Scoping Rule: Static Scoping

```
x = "Global X"

def foo():
      x = "foo local"
      print("\nIn Top Foo: ", x)


      def nestedFoo():
      x = "nestedFoo X var"
      print("\nIn nested Foo: ", x)

      def nestedfooCaller():        #calls the nestedFoo Function
      x = "foo caller x var"
      nestedFoo()

      nestedFoo()
      nestedfooCaller()

foo()
```

Python is a statically scoped language. It behaves the same as the rest of the statically scoped languages discussed here.

Learning Strategy:

I first tried to comprehend the logic behind static scoping and dynamic scoping using the notes from class as well as from the homework prompt. I also viewed a few videos that explain the

differences. After getting an idea of how it is supposed to work, I tested out the logic using Perl with my and local keywords as it was the only language supporting both types of scoping. Once I understood the difference in results when static and dynamic scoping were used, I wrote a program in Python since I feel most comfortable with that language. After that, it was a matter of translating the python code into the other languages, and it was quite easy to do so (although I had to double-check the function and variable declaration multiple times as I got confused between them). My main tool for writing and testing the code was the online compilers mentioned in the references. Once it was working there, I made files for each language and ran them on Dijkstra to confirm it worked on that system as well.
After that, the final step was inserting the snippets and writing the relevant descriptions for each of the languages.

References:

Python online compiler. Online Python Compiler (Interpreter). (n.d.). Retrieved April 12, 2022, from https://www.programiz.com/python-programming/online-compiler/

Execute dart online. Online Dart Compiler. (n.d.). Retrieved April 12, 2022, from https://www.tutorialspoint.com/execute_dart_online.php

Execute perl online. Online Perl Compiler. (n.d.). Retrieved April 12, 2022, from https://www.tutorialspoint.com/execute_perl_online.php