PRESENTATION A

- What does it mean to **define a function**? Give an example in the following languages

a “snippet” of code that you can use over and over again, rather than writing it out multiple times.

  - Javascript:

```
function hello(function parameters){

function body

};

hello();  //here you call the function
```

- What does it mean to **define a function**? Give an example in the following languages

a “chunk” of code that you can use over and over again, rather than writing it out multiple times.

  - C#:

```
static void MyMethod() {

 Console.WriteLine("I just got executed!");

}

static void Main(string[] args){

MyMethod();

}
```

- What does it mean to **define a function**? Give an example in the following languages

a “chunk” of code that you can use over and over again, rather than writing it out multiple times.

- Java:

```
public class Main {
  static void myMethod() {
    //code to be executed in here;
  }
 public static void main(String[] args) {
    myMethod();
  }
}
```

- What is the relationship between **defining a variable** and the **scope**?

scoop refers to the current context of code, which determines the accessibility of variables.

There are 2 types of scoop:

- Global
- Local

- Show an example of accessing a **global variable** inside a function in the following languages:

    - Javascript

```
const myVariable = “Hello”;

function sayHello(){

        console.log(myVariable);

}

sayHello();
```

- Show an example of accessing a **global variable** inside a function in the following languages:
  - C#

```
static class MyMethod() {

 public static String executed = “I just got executed!";

}

static void Main(string[] args){

Console.WriteLine(myMethod.executed);

}
```

- Show an example of accessing a **global variable** inside a function in the following languages:
    - Java - Java actually doesn't have the concept of Global variable, it is known as class variable (static field)

```
public class Globals {

                 public static int globalInt = 0;

}

public class Main {

  static void myMethod() {

    System.out.println(globalInt);

  }

 public static void main(String[] args) {

    myMethod();

  }

}
```

- What is **nested scope**? Show an example of a nested scope in the following languages:

Blocks and functions created inside other blocks and functions

    - Javascript

```
var scope1 = "global scope";
function MyFunc() {
   var scope2 = "local scope";
   function AnotherFunc() {
      return scope1 + scope2;
   }
   return AnotherFunc();
}
MyFunc()
```

- What is **nested scope**? Show an example of a nested scope in the following languages:
    - C#

```
public int InnerScopeMethod() {
            int i = 10;
            //Inner Scope
            {
                  int j = 100;//j will not work outside of the scope.
                  i *= j;
            }
            //Scope Ends Here

            return i;
      }
```

- What is **nested scope**? Show an example of a nested scope in the following languages:

    - Java

```
public class J {
        public static void main(String[] args) {
              {
              // outer
              int a = 1; ;
                      { // inner
                              System.out.println(a);
                               int b = 3;
                              System.out.println(a+b);
                      }
               }
       }
}
```

- In which of the following programming languages is it possible to define a function as value? Give an example.
    - Javascript: const myFunction = () =>{}
    - C# - no, the functions returns a value
    - Java - same has c#

- What does this **script return**?

```
function bar() {
   return foo;
   foo = 10;
   function foo() { }
   var foo = '11';
}
alert(typeof bar());
```

It returns an alert saying function because the return give us a function

- What are **arrow functions** in Javascript? Show an example of declaring a function with or without arrows

Arrow functions allow us to write shorter function syntax.
Instead of the function keyword, it uses an arrow ((parameters)=>{code}) made up of an equal sign and a greater-than character

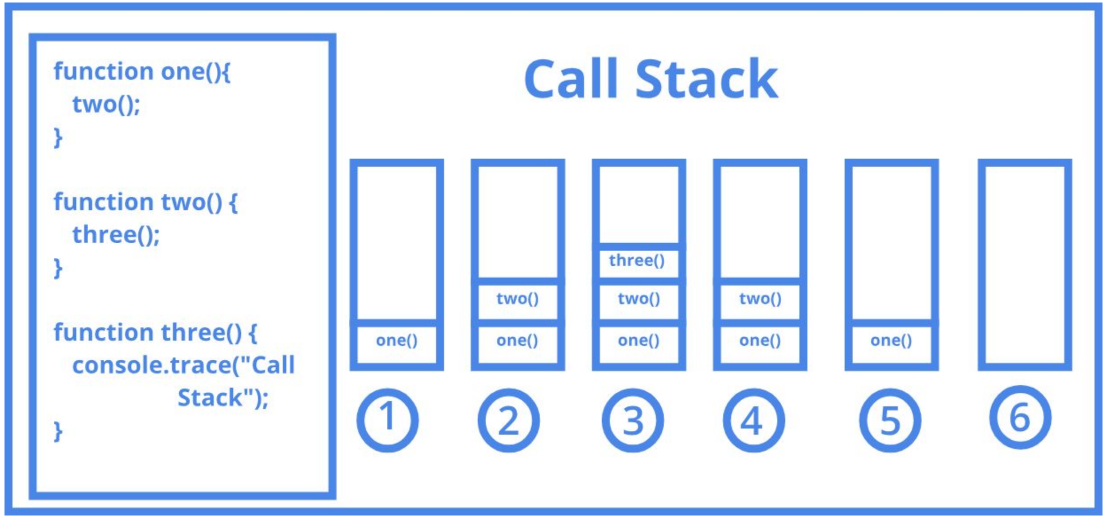
```
const hello = function() {
  return "Hello World!";
}
```

With Arrow Function:

```
const hello = () => {
  return "Hello World!";
}
```

- What is **the call stack**? Search for a graphical example to better understand this concept

is a dynamic LIFO(Last in, First out) data structure, (a stack), that stores information about the active subroutines of a computer program. This kind of stack is also known as an execution stack, control stack, function stack, or runtime stack, and is often described in short form as "the stack".



```
function one(){
  two();
}

function two() {
  three();
}

function three() {
  console.trace("Call
            Stack");
}
```

- What are **optional arguments**? Which of these languages support the optional arguments? Show an example of each one:
    - Javascript - supports

In the following example, if no value is provided for b when multiply is called, b's value would be undefined when evaluating a * b and multiply would return NaN (not a number).

```
function multiply(a, b) {
  return a * b
}

multiply(5, 2)  // 10
multiply(5)     // NaN !
```

- What are **optional arguments**? Which of these languages support the optional arguments? Show an example of each one:
    - C# - supports, the optional arguments needs to have a default value as part of its definition.

```
public void ExampleMethod(int required, string optionalstr = "default string",int optionalint = 10)
```

```
ExampleMethod(1);
```

- What are **optional arguments**? Which of these languages support the optional arguments? Show an example of each one:
    - Java

There isn’t optional arguments in Java

- Who was **first**; the **chicken** or the **egg**? Explain the following code:

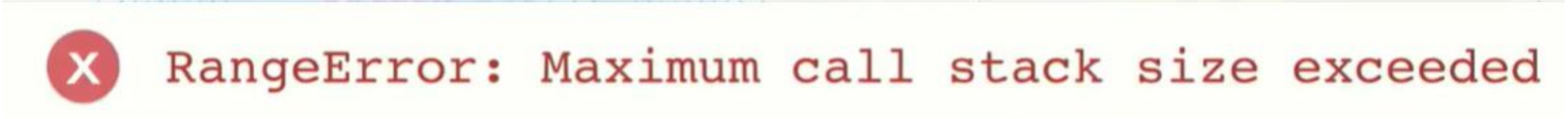
```
function chicken() {
   return egg();
}

function egg() {
   return chicken();
}

console.log(chicken() + " came first."); // → ??
```

It’s an endless loop

The chicken function return egg function and the egg function returns chicken function.

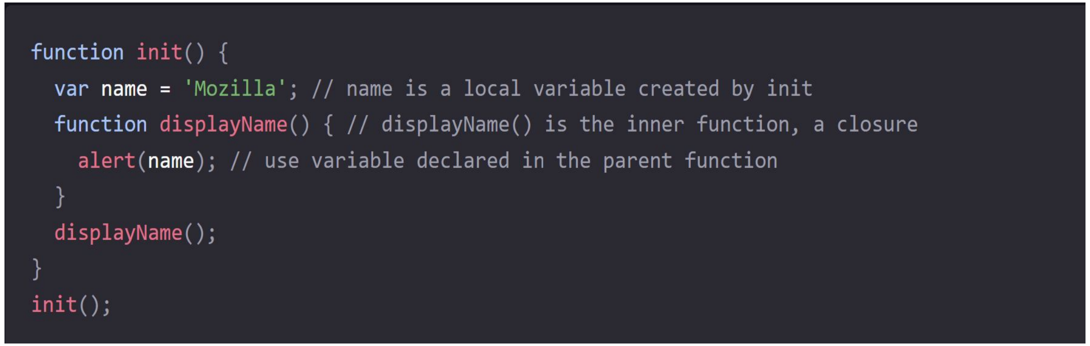


```
RangeError: Maximum call stack size exceeded
```

- What is **closure**? Show a simple practical example of using closure in JavaScript to better understand the concept.

JavaScript variables can belong to the local or global scope. Global variables can be made local (private) with closures.

In other words, a closure gives you access to an outer function's scope from an inner function.



```javascript
function init() {
  var name = 'Mozilla'; // name is a local variable created by init
  function displayName() { // displayName() is the inner function, a closure
    alert(name); // use variable declared in the parent function
  }
  displayName();
}
init();
```

init() creates a local variable called name and a function called displayName(). The displayName() function is an inner function that is defined inside init() and is available only within the body of the init() function. Note that the displayName() function has no local variables of its own. However, since inner functions have access to the variables of outer functions, displayName() can access the variable name declared in the parent function, init().