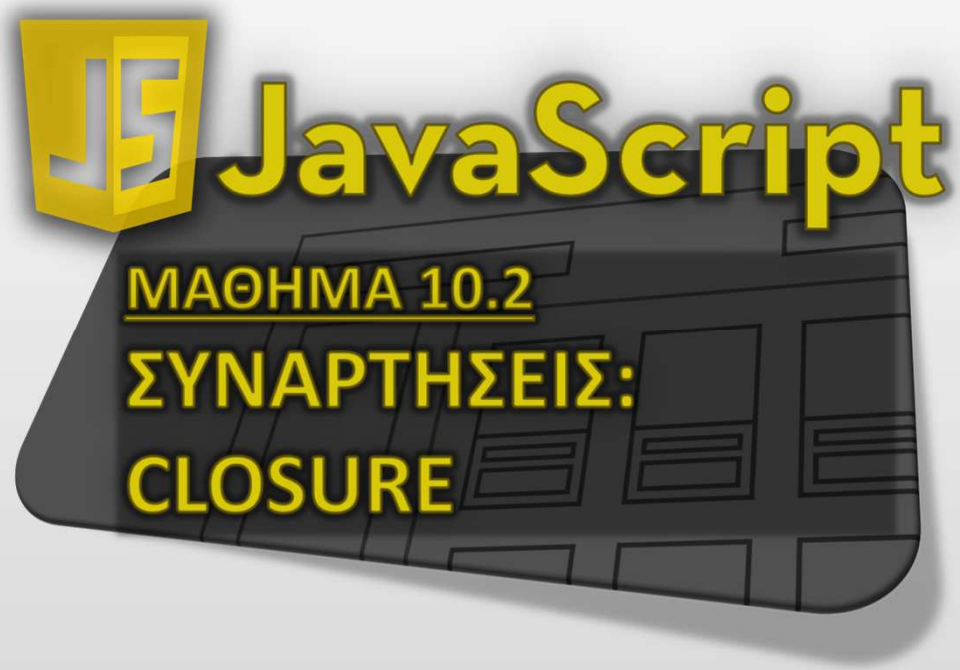


ΠΕΡΙΕΧΟΜΕΝΑ:

ΜΑΘΗΜΑ 10.2: Συναρτήσεις: Closure

1. Λεκτική Εμβέλεια

JavaScript psounis

Λεκτική Εμβέλεια (Lexical Scoping):

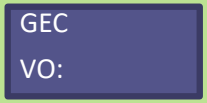
- Η εμβέλεια που βλέπει μία συνάρτηση καθορίζεται από το σημείο που η συνάρτηση ορίστηκε (και όχι από το σημείο που η συνάρτηση κλήθηκε)

Παράδειγμα 1: lexical

```
let a = 1;

function f() {
  return a;
}

function g() {
  let a = 2;
  return f();
}

console.log(g());
```

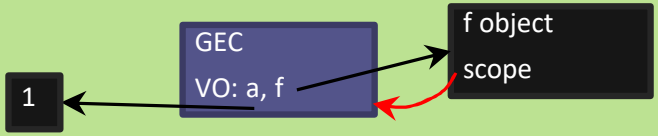
Πως τρέχει το παραπάνω πρόγραμμα:

- Αρχικά δημιουργείται το καθολικό περιβάλλον εκτέλεσης:

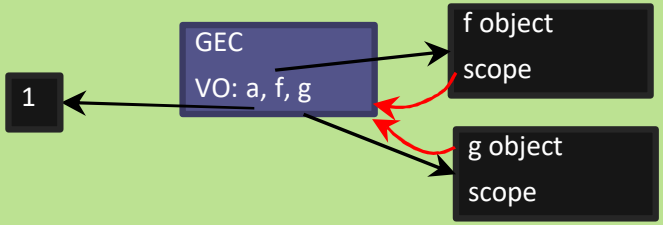


- Δήλωση της a: Προστίθεται στο VO της καθολικής εμβέλειας:

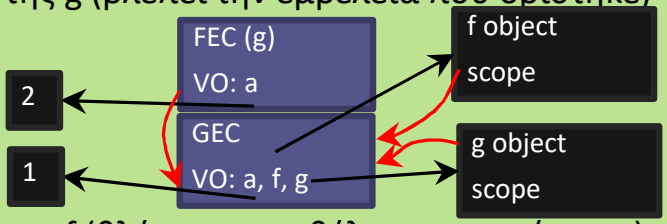
- Δήλωση της f: Προστίθεται στο VO της καθολικής εμβέλειας. Το αντικείμενο «βλέπει» την καθολική εμβέλεια



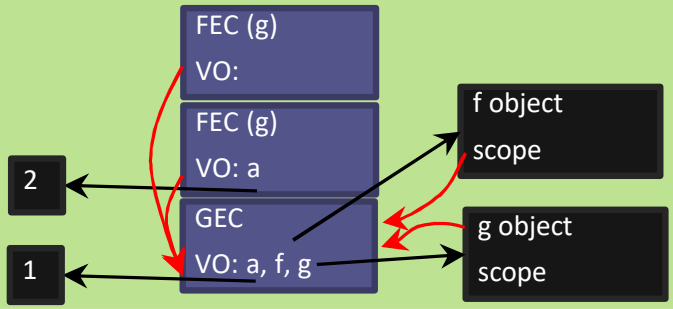
- Δήλωση της g: Προστίθεται στο VO της καθολικής εμβέλειας. Το αντικείμενο «βλέπει» την καθολική εμβέλεια



- Κλήση της g (βλέπει την εμβέλεια που ορίστηκε)



- Κλήση της f (βλέπει την εμβέλεια που ορίστηκε)



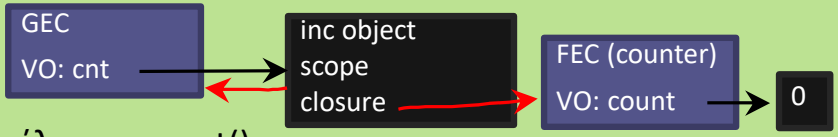
Κλειστότητα (Closure):

- Ο συνδυασμός του περιβάλλοντος εκτέλεσης συνάρτησης με την εμβέλεια με την οποία έχει συσχετιστεί (όπως είδαμε, αυτήν στην οποία ορίστηκε), λέγεται **κλειστότητα (closure)**.
- Η πιο ενδιαφέρουσα ιδιότητα της κλειστότητας, είναι ότι η εμβέλεια παραμένει, ακόμη και αν η κλήση συνάρτησης που την όρισε, έχει ολοκληρωθεί.
- Το μελετάμε με ένα παράδειγμα.

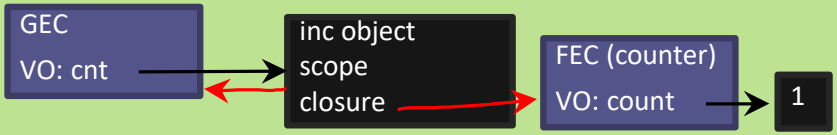
Παράδειγμα 2: closure

```
function counter() {
  let count = 0;

  function inc() {
    count++;
    return count;
  }
  return inc;
}

let cnt = counter();
console.log(cnt()); // 1
console.log(cnt()); // 2

let cnt2 = counter();
console.log(cnt2()); // 1
console.log(cnt2()); // 2
```

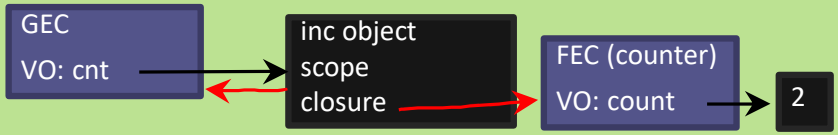
- Κατάσταση μετά την εντολή let cnt = counter():



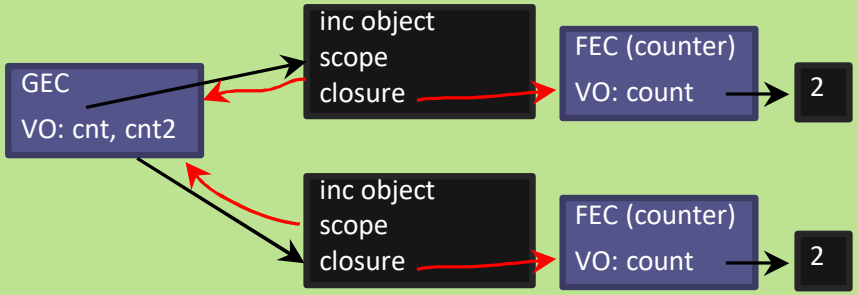
- 1η Εκτέλεση της cnt():



- 2η εκτέλεση της cnt():



- 2η εκτέλεση της cnt2():



ΜΑΘΗΜΑ 10.2: Closure — 2.1. Εφαρμογή: Ιδιωτικές Μεταβλητές Αντικειμένων

JavaScript psounis

1η εφαρμογή των closures:

- Μπορούμε να δημιουργήσουμε αντικείμενα που έχουν «πραγματικά» ιδιωτικά μέλη.

Πως το επιτυγχάνουμε:

- Η εξωτερική συνάρτηση ορίζει ιδιωτικό μέλος
- Η εσωτερική συνάρτηση ορίζει το αντικείμενο, το οποίο έχει πρόσβαση στο ιδιωτικό μέλος.

Παράδειγμα 3: private

```
function createObject() {
   let somethingPrivate = 0;

   function theObject() {
      let object = {
         "key1": "value1",
         "key2": "value2",
         incPrivateValue() {
            somethingPrivate++;
         },
         getPrivateValue() {
            return somethingPrivate;
         }
      }
      return object;
   }
   return theObject();
}

let object = createObject();
object.incPrivateValue();

for (let key in object)
   console.log(key);
console.log(object.getPrivateValue());
```

Άσκηση 1:

- Κατασκευάστε μια συνάρτηση που επιστρέφει αντικείμενα που μοντελοποιούν άτομα:
  - Ιδιωτικές Μεταβλητές: Age και Name
  - Μέθοδοι:
    - getName() και setName(name)
    - getAge()
    - birthday(): Αυξάνει την ηλικία κατά 1.
    - toString(): Μορφοποιημένη εκτύπωση
- Η συνάρτηση θα παίρνει σαν όρισμα την ηλικία και το όνομα, θα αρχικοποιεί τις ιδιωτικές μεταβλητές και έπειτα θα επιστρέφει το αντικείμενο - άτομο.

Ελέγξτε το πρόγραμμά σας με τον παρακάτω κώδικα (και την επιθυμητή εκτύπωση)

```
let person = Person('John', 22);
console.log("" + person);
person.birthday();
person.birthday();
console.log("" + person);
```

2η εφαρμογή των closures:

- Currying: Διάσπαση μιας συνάρτησης που παίρνει πολλά ορίσματα σε επιμέρους συναρτήσεις που παίρνουν ένα όρισμα.

Παράδειγμα 4: currying

```
function multiply(a) {
  return function(b) {
    return function(c) {
      return a * b * c;
    }
  }
}

let multiplyByTwo = multiply(2);
console.log(multiplyByTwo(3)(4));

let multiplyBySix = multiplyByTwo(3);
console.log(multiplyBySix(4));
```

Παρατήρηση:

- Το currying δημιουργεί εκτελέσεις συναρτήσεων με αρκετά ενδιαφέρον συντακτικό!

Παράδειγμα 5: logic_gates

```
function not(a) {
  return !a;
}

function and(a) {
  return function(b) {
    return a && b;
  }
}

function or(a) {
  return function(b) {
    return a || b;
  }
}

console.log(not(true));
console.log(and(true)(false));
console.log(or(true)(not(false)));
```

```
false
false
true
```

Άσκηση 2:

- Υπολογίστε, χωρίς να τρέξετε τον κώδικα, τι επιστρέφει το ακόλουθο πρόγραμμα:

```
function makeFunctions() {
  let funcs = [];

  for (let i=0; i<10; i++) {

    function func() {
      return i;
    }

    funcs.push(func);
  }

  return funcs;
}

let functions = makeFunctions();
console.log(functions[3]());
```

Άσκηση 3:

- Υπολογίστε, χωρίς να τρέξετε τον κώδικα, τι επιστρέφει το ακόλουθο πρόγραμμα:

```
function makeFunctions() {
  let funcs = [];
  let i;

  for (i=0; i<10; i++) {

    function func() {
      return i;
    }

    funcs.push(func);
  }

  return funcs;
}

let functions = makeFunctions();
console.log(functions[3]());
```