ΠΕΡΙΕΧΟΜΕΝΑ:

1. Εσωτερικές Συναρτήσεις
2. Εργοστάσια Συναρτήσεων
3. Εμβέλεια Ονομάτων
4. Περιβάλλοντα Εκτέλεσης και Στοίβα Κλήσεων
5. Το Αντικείμενο - Συνάρτηση
6. Ασκήσεις

Γιώργος Φραγκιαδάκης
Σμαραγδένιος Χορηγός Μαθήματος

Δημήτρης Α.
Χρυσός Χορηγός Μαθήματος

ΜΑΘΗΜΑ 10.1: Συναρτήσεις: Εμβέλεια κ.α.

1. Εσωτερικές Συναρτήσεις

JavaScript psounis

- Οι **εσωτερικές συναρτήσεις (inner functions):**
  - Είναι **συναρτήσεις που ορίζονται μέσα σε συναρτήσεις**
  - Και μπορουν να κληθούν μόνο από τη συνάρτηση μέσα στην οποία έχουν οριστεί.

Παράδειγμα 1: inner_function

```
function outer() {
   function inner() {
      console.log("inner");
   }
   console.log("outer");
   inner();
   inner();
}
outer();
```

Η inner() είναι εσωτερική συνάρτηση στην outer()
Σημειώστε ότι δεν μπορούμε να καλέσουμε την inner() π.χ. από το κυρίως πρόγραμμα.

Οι εσωτερικές συναρτήσεις είναι χρήσιμες όταν:

- Για να προφυλάξουμε τις εσωτερικές συναρτήσεις από ότι συμβαίνει εκτός της συνάρτησης.
- Για να κρατήσουμε τον κώδικα καθαρό, χωρίς περιττούς ορισμούς συναρτήσεων
- Σαν περίβλημα σε πιο περίπλοκες συναρτήσεις (βλ. παράδειγμα 2) και στα εργοστάσια συναρτήσεων (επομ.διαφ.))

Παράδειγμα 2: fibonacci

```
function fibonacci(n) {
   function rec(n) {
      if (n===0)
         return 0;
      else if (n===1)
         return 1;
      else
         return rec(n-1)+rec(n-2);
   }

   if (Number.isNaN(n) || !Number.isInteger(n) || n<0)
      return undefined;
   else
      return rec(n);
}

console.log(fibonacci("two"));
console.log(fibonacci(5.5));
console.log(fibonacci(-1));
console.log(fibonacci(1));
console.log(fibonacci(10));
```

- Τα **εργοστάσια συναρτήσεων (function factories):**
  - Είναι **συναρτήσεις που κατασκευάζουν συναρτήσεις.**
  - Συγκεκριμένα, **επιστρέφουν μια εσωτερική συνάρτηση, στην οποία έχουν πάρει συγκεκριμένες τιμές κάποιες μεταβλητές.**

Παράδειγμα 2: function_factory

```
function factoryPower(base) {
   function toPower(n) {
      return base**n;
   }

   return toPower;
}

let powerOfTwo = factoryPower(2);
console.log(powerOfTwo(10));

let powerOfThree = factoryPower(3);
console.log(powerOfThree(3));
```

Η factory_power:

- Πρώτα ορίζει μια συνάρτηση(toPower) ορίζει μια συνάρτηση, στην οποία η μεταβλητή base έχει αρχικοποιηθεί με την τιμή του ορίσματος.
- Έπειτα την επιστρέφει.

Άσκηση 1:

Ορίστε το εργοστάσιο δευτεροβάθμιων πολυωνύμων, δηλαδή συναρτήσεων της μορφής $f(x) = ax^2 + bx + c$

Παράδειγμα κλήσης:

```
let myPolynomial = factory(1, 1, 1);
console.log(myPolynomial(2));
```

που θα τυπώνει 7 (αφού $\text{pol}(x) = x^2 + x + 1$)

Άσκηση 2:

Εκτελέστε το ακόλουθο πρόγραμμα:

```
function counter() {
   let cnt = 0;
   return function() {
      cnt++;
      return cnt;
   }
}
let cnt1 = counter();
let cnt2 = counter();
console.log(cnt1());
console.log(cnt1());
console.log(cnt2());
console.log(cnt2());
```

Παρατηρήστε τη συμπεριφορά της μεταβλητής cnt στα αντικείμενα - συναρτήσεις που κατασκευάζονται.

ΜΑΘΗΜΑ 10.1: Συναρτήσεις: Εμβέλεια κ.α.

3. Εμβέλεια Ονομάτων

JavaScript psounis

- Με τον όρο **εμβέλεια (scope)** ενός ονόματος (π.χ. μεταβλητής, συνάρτησης, αντικειμένου) εννοούμε:
  - Την περιοχή του προγράμματος στην οποία έχουμε πρόσβαση σε αυτό το όνομα (Συνεπώς τα ονόματα ανήκουν σε εμβέλειες)
- Υπάρχουν οι ακόλουθες εμβέλειες:
  - **Τοπική Εμβέλεια ονόματος σε μπλοκ(block scope)**
    - Ονόματα που ορίζονται σε μπλοκ κώδικα (π.χ. σε block δομής ελέγχου ή δομής επανάληψης).
    - Ονομάζουμε τα ονόματα: Τοπικά σε μπλοκ.
    - Τα ονόματα είναι ορατά μόνο στο block.
  - **Τοπική Εμβέλεια ονόματος σε Συνάρτηση (local scope):**
    - Ονόματα που έχουν οριστεί στη συνάρτηση (περιλαμβάνονται οι παράμετροι στης συνάρτησης)
    - Ονομάζουμε τις μεταβλητές: Τοπικές σε συνάρτηση.
    - Είναι ορατά μόνο στον κώδικα της συνάρτησης
    - Δημιουργούνται όταν γίνεται η δήλωσή τους
  - **Τοπική Εξωτερική Εμβέλεια ονόματος (enclosing scope):** Η εμβέλεια σε μία συνάρτηση που περιέχει εσωτερικές συναρτήσεις.
    - Ονόματα που έχουν οριστεί στην συνάρτηση που περιέχει εσωτερικές συναρτήσεις
    - Ορατά από τη συνάρτηση και τις εσωτερικές της συναρτήσεις
  - **Καθολική Εμβέλεια (global scope):**
    - Ονόματα που ορίζονται εκτός συναρτήσεων.
    - Ονομάζουμε τα ονόματα: Καθολικά (global).
    - Ορατά παντού στον κώδικα.
- **Προσοχή!** Σε περίπτωση συγκρούσεων ονομάτων, επικρατεί το όνομα που βρίσκεται πιο πάνω στην παραπάνω ιεραρχία.

Παράδειγμα 3: scope

```
let x=1;

function outer() {
  let x=2;

  function inner() {
    let x=3;
    console.log(`inner: ${x}`);
    if (true) {
      let x=4;
      console.log(`block: ${x}`);
    }
  }

  console.log(`outer: ${x}`);
  inner();
}

console.log(`global: ${x}`);
outer();
```

Σημείωση:

- Τα προηγούμενα ισχύουν για δήλωση με let/const.
- Για **δήλωση με var σε συνάρτηση:**
  - **Γίνεται hoisting στην αρχή της συνάρτησης**.
  - Και θυμόμαστε:
    - Καθολικές μεταβλητές με var: hoisted -> αρχή του script.
    - Τοπικές μεταβλητές με var: hoisted -> αρχή της συνάρτησης.

ΜΑΘΗΜΑ 10.1: Εμβέλεια κ.α. — 4. Περιβάλλοντα Εκτέλεσης και Στοίβα Κλήσεων

JavaScript psounis

Περιβάλλοντα Εκτέλεσης (Execution Contexts):

- (Υπενθύμιση από μάθημα 3) **Καθολικό Περιβάλλον Εκτέλεσης (Global Execution Context)**:
  - Δημιουργείται αυτόματα όταν αρχίζει να τρέχει η εφαρμογή μας.
  - Περιέχει το αντικείμενο global (πολλές χρήσιμες ιδιότητες, όπως π.χ. το document που περιέχει τον HTML κώδικα)
  - Περιέχει τα ονόματα που έχουν δηλωθεί στον κώδικα (π.χ. μεταβλητές, συναρτήσεις κ.λπ.) **κ.α.**
- **Περιβάλλον Εκτέλεσης Συνάρτησης (Function (ή Local) Execution Context):**
  - Δημιουργείται αυτόματα όταν γίνεται κλήση συνάρτησης
  - Περιέχει τα ονόματα που έχουν δηλωθεί στον κώδικα της συνάρτησης (π.χ. μεταβλητές, συναρτήσεις κ.λπ.) **κ.α.**

Παράδειγμα 4: execution_contexts

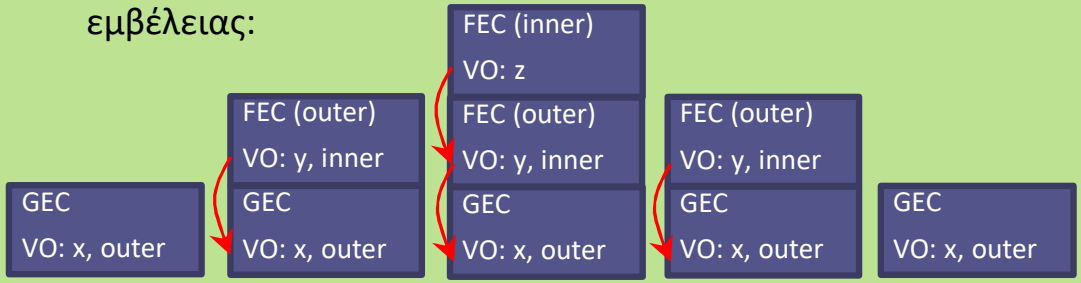
```
function outer() {
  function inner() {
    let z = 3;
    console.log(x, y, z);
  }
  let y = 2;
  console.log(x, y);
  inner();
}
let x = 1;
console.log(x);
outer();
```

Στοίβα Κλήσεων (Call Stack):

- Διατηρείται κατά την εκτέλεση του προγράμματός μας, είναι η δομή που συγκρατεί τις ενεργές κλήσεις συνάρτησης.
- Αρχικά περιέχει μόνο το καθολικό περιβάλλον εκτέλεσης
- Όταν γίνεται κλήση συνάρτησης, το περιβάλλον εκτέλεσής της στοιβάζεται επί του προηγούμενου ενεργού περιβάλλοντος εκτέλεσης.

Στο παράδειγμα 4:

- (βλ. και βίντεο για τον τρόπο που παρακολουθούμε τα περιβάλλοντα εκτέλεσης στον google chrome)
- Οι διαδοχικές φάσεις του call stack με τα περιβ. εκτέλεσης
- Σε κάθε περιβάλλον εκτέλεσης, αναφέρονται τα ονόματα που περιέχει (variable object), δηλ. τα ονόματα της τοπικής εμβέλειας:



Παρατήρηση:

- Κάθε κόκκινο βέλος δείχνει τη σειρά με την οποία θα αναζητηθούν τα ονόματα στην αλυσίδα εμβελειών (**scope chain**)
- Π.χ. στο μεσαίο παράδειγμα, πρώτα γίνεται αναζήτηση στην εμβέλεια του FEC(inner), αν δεν βρεθεί στο FEC(outer) κ.ο.κ.

Το αντικείμενο - συνάρτηση (function object):

- (Υπενθύμιση από μάθημα 7) Η δήλωση μιας συνάρτησης κατασκευάζει ένα αντικείμενο-συνάρτηση.

Παράδειγμα 5: function_object

```
function myfunc() {
  console.log("func");
}
console.log(myfunc);
console.dir(myfunc);
```

```
ƒ myfunc() {
    console.log("func");
}
▼ƒ myfunc()
  length: 0
  name: "myfunc"
  ▶prototype: {constructor: ƒ}
  arguments: (...)
  caller: (...)
  [[FunctionLocation]]: function_object.html…d=RELOAD_ON
  ▶[[Prototype]]: ƒ ()
```

Μέλη του Αντικειμένου - Συνάρτηση:

Μέλος	Επεξήγηση
prototype	Αρχικά κενό. Χρησιμοποιείται μόνο αν η συνάρτηση χρησιμοποιηθεί σαν κατασκευαστής αντικειμένων (βλ. μάθ. 8.1)
name	Το όνομα της συνάρτησης
length	Το πλήθος των παραμέτρων της
arguments	Αρχικοποιείται κατά την κλήση. Είναι ένας πίνακας με όλα τα ορίσματα της κλήσης (βλ. μαθ. 7)
caller	[Deprecated, πρόκειται να αποσυρθεί] Συνάρτηση που την κάλεσε
[[FunctionLocation]]	(Όχι απευθείας πρόσβαση) Σημείο δήλωσης της συνάρτησης (κώδικας της συνάρτησης)

Σημείωση:

- Το πρωτότυπο του αντικειμένου - συνάρτησης, περιέχει χρήσιμες μεθόδους όπως η bind(), call() και apply() που θα δούμε σε επόμενο μάθημα.

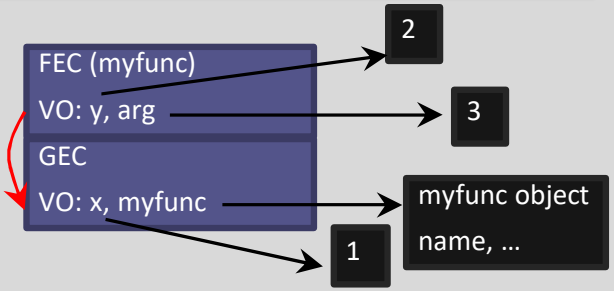
Παρατήρηση:

- Το αντικείμενο-συνάρτηση χρησιμοποιείται για να κατασκευαστεί το Περιβάλλον Εκτέλεσης Συνάρτησης.
- Συνεπώς, όταν γίνεται κλήση συνάρτησης, από το αντικείμενο - συνάρτηση, κατασκευάζεται το περιβάλλον εκτέλεσης και επί αυτού τρέχει ο κώδικας της συνάρτησης

Παράδειγμα 6: function_execution_object

```
function myfunc(arg) {
  let y = 2;
  console.log(x, y, arg);
}
let x = 1;
myfunc(3);
```

call stack όταν γίνεται η κλήση της συνάρτησης (με τις αντίστοιχες αναφορές των ονομάτων)



Άσκηση 3:

Γράψτε μία συνάρτηση με όνομα adder:

- Παίρνει ως όρισμα έναν αριθμό
- Επιστρέφει μία συνάρτηση που προσθέτει στον όρισμα της τον αριθμό.

Παρακάτω ένα παράδειγμα χρήσης της συνάρτησης με τις επιθυμητές εξόδους:

```
let add5 = adder(5);
console.log(add5(2)); // 7
console.log(add5(10)); // 15

let add10 = adder(10);
console.log(add10(2)); // 12
console.log(add10(10)); // 20
```

Άσκηση 4:

Ορίστε μία συνάρτηση με όνομα countVowels η οποία να παίρνει ως όρισμα μία συμβολοσειρά και να επιστρέφει το πλήθος των φωνηέντων που περιέχει [στο λατινικό αλφάβητο τα φωνήεντα είναι: (a, e, i, o, u)]

- Θα πρέπει να είναι case-insensitive (δηλαδή να αγνοεί κεφαλαία - μικρά) Η συνάρτηση θα πρέπει να χρησιμοποιεί μία εσωτερική συνάρτηση με όνομα isVowel η οποία θα παίρνει ως όρισμα έναν χαρακτήρα και θα επιστρέφει true, αν αυτός είναι φωνήεν.