ΠΕΡΙΕΧΟΜΕΝΑ:

1. Δομές Επανάληψης
    1. do…while
    2. while
    3. for
        1. Ασκήσεις
2. Εμφωλιασμένοι Βρόχοι
3. Break και Continue
    1. Ετικέτες Διακοπής Βρόχου

Γεώργιος Κ.

Σμαραγδένιος Χορηγός Μαθήματος

Νικόλαος Β.

Χρυσός Χορηγός Μαθήματος

ΜΑΘΗΜΑ 5: Δομές Επανάληψης — 1.1. Η δομή επανάληψης while

JavaScript psounis

- Με τον όρο **"Επανάληψη"** εννοούμε τις τρεις εντολές της JavaScript **(for, while, do..while)**
  - οι οποίες **μας επιτρέπουν να επαναλάβουμε πολλές φορές τον ίδιο κώδικα.**
  - (το οποίο είναι εξαιρετικά χρήσιμο και κύριο χαρακτηριστικό του προγραμματισμού)

while

- Συντακτικό:

```
[προηγούμενες εντολές]

while(condition) {
   statement1
   ...
   statementN
}

[επόμενες εντολές]
```

- Εξήγηση:
  - Ελέγχεται η συνθήκη
  - Αν η συνθήκη είναι true(-thy)
    - εκτελούνται οι εντολές
    - μεταφέρεται ο έλεγχος πάλι στη συνθήκη (αρχή)
  - Αν η συνθήκη είναι false(-y)
    - Βγαίνουμε από τη while και προχωράμε στις επόμενες εντολές

Παρατηρήσεις:

- Μία εντολή επανάληψης αναφέρεται συχνά και σαν **βρόχος (ή loop)**
- Ενώ οι εντολές μέσα στο βρόχο αναφέρονται και σαν **σώμα του βρόχου** και αποτελούν ένα **μπλοκ κώδικα**.
- Αν βάλουμε μόνο μία εντολή μέσα στο σώμα του βρόχου, τότε μπορούμε να παραλείψουμε τα άγκιστρα.
- Συνήθως μεταφράζεται "Επανάλαβε"

Παράδειγμα 1: while.java

```
let i;

i=0;
while(i<=5) {
  console.log(i);
  i++;
}
```

Άσκηση 1:

Τυπώστε επαναληπτικά όλους τους περιττούς αριθμούς από το 1 έως το 55

Άσκηση 2:

Τυπώστε επαναληπτικά όλους τους άρτιους αριθμούς από το 120 έως το 0 (φθίνουσα σειρά)

Άσκηση 3:

Βρείτε το λάθος στον παρακάτω τμήμα κώδικα:

```
i=100;
while (i>=0) {
   System.out.println(i);
   i++;
}
```

do...while

- Συντακτικό:

```
[προηγούμενες εντολές]

do {
   statement1
   ...
   statementN
} while (condition);


[επόμενες εντολές]
```

- Εξήγηση:
  - Εκτελούνται οι εντολές
  - Ελέγχεται η συνθήκη
  - Αν η συνθήκη είναι true(-thy)
    - μεταφέρεται ο έλεγχος πάλι στην αρχή (εκτέλεση των εντολών
  - Αν η συνθήκη είναι false(-y)
    - Βγαίνουμε από το loop και προχωράμε στις επόμενες εντολές

Παρατηρήσεις:

- Το σώμα του βρόχου θα εκτελεστεί τουλάχιστον μία φορά
- Μπορούμε να παραλείψουμε τα άγκιστρα, αν το σώμα έχει ακριβώς μία εντολή.

Παράδειγμα 2: do_while.java

```
let i;

i=0;
do {
  console.log(i);
  i++;
} while (i<=5);
```

Άσκηση 4:

Χρησιμοποιώντας τη δομή επανάληψης do..while τυπώστε τα τετράγωνα των αριθμών από το 0 έως το 100. Σε κάθε γραμμή να τυπώνεται ο αριθμός και το τετράγωνό του στη μορφή:

```
5^2=25
```

Παρατήρηση: Για το σωστό "στήσιμο" μιας επαναληπτικής δομής, πρέπει να σκεφτόμαστε:

- Την αρχικοποίηση των απαραίτητων μεταβλητών πριν την έναρξη της επανάληψης
- Την τροποποιήση των μεταβλητών σε κάθε επαναληπτικό βήμα.
- Τη συνθήκη τερματισμού.

for

- Συντακτικό:

```
[προηγούμενες εντολές]

for (initialization; condition; step) {
   statement1;
   ...
   statementN;
}

[επόμενες εντολές]
```

- Εξήγηση:
  - Τρέχει η εντολή αρχικοποίησης
  - Ελέγχεται η συνθήκη
  - Αν η συνθήκη είναι true(-thy)
    - Τρέχουν οι εντολές, έπειτα το step και ο έλεγχος μεταφέρεται ξανά στον έλεγχο της συνθήκης
  - Αν η συνθήκη είναι false(-y)
    - Βγαίνουμε από το loop και προχωράμε στις επόμενες εντολές

Παρατηρήσεις:

- Αν έχουμε μόνο μία εντολή στο σώμα του βρόχου, τότε τα άγκιστρα μπορούν να παραλειφθούν.

Παράδειγμα 3: for1.java

```
let k, square;

for (k=10; k>=0; k--) {
  square = k*k;
  console.log(square);
}
```

Άσκηση 5:

Υλοποιήστε το παράδειγμα 3, χρησιμοποιώντας τη δομή while και έπειτα τη δομή επανάληψης do..while.

Σημείωση:

- Μπορούμε να έχουμε δύο ή παραπάνω εντολές στην αρχικοποίηση, ενώνοντας τις με κόμμα (βλ. for2.html). Π.χ.:

```
for (let i=0, j=0; i<=5 && j<=5; i++, j+=2)
  console.log(`i: ${i}, j: ${j}`);
```

- Οποιοδήποτε από τα στοιχεία μπορεί να είναι κενό, π.χ.:

```
for(;;);
```

Σημείωση: Υπάρχουν ακόμη δύο εκδοχές της for:

- for/in (δουλεύει επί αντικειμένων (objects))
- for/of (δουλεύει επί iterable αντικειμένων, όπως οι πίνακες)
- Θα τις μελετήσουμε στα αντίστοιχα μαθήματα.

Άσκηση 6:

Κατασκευάστε πρόγραμμα το οποίο:

- Θα τυπώνει όλους τους άρτιους αριθμούς από το 10 έως το 20
- Θα τυπώνει όλους τους περιττούς αριθμούς από το 19 έως το 11 (φθίνουσα σειρά)
- Θα τυπώνει εκείνους τους περιττούς αριθμούς από το 1 έως το 29 που είναι επίσης πολλαπλάσια του 3.

Κάθε ομάδα αριθμών να τυπώνεται σε μία γραμμή (οι αριθμοί να είναι χωρισμένοι με κενά)

Άσκηση 7:

Κατασκευάστε ένα πρόγραμμα το οποίο υπολογίζει το άθροισμα των περιττών αριθμών από το 1 έως το 99.

Άσκηση 8: Η ακολουθία 3n+1

Ένα γνωστό μαθηματικό πρόβλημα είναι το εξής: Ξεκινώντας από έναν φυσικό αριθμό n>=2:

- Αν το n είναι άρτιος, τότε μειώνουμε το n στο μισό και επαναλαμβάνουμε.
- Αν το n είναι περιττός, τότε θέτουμε το n ίσο με 3n+1 και επαναλαμβάνουμε.
- Τερματίζουμε όταν το n γίνει ίσο με το 1

Κατασκευάστε ένα πρόγραμμα που εκτυπώνει την ακολουθία των φυσικών από το n μέχρι το 1.

[Σημείωση: Το μαθηματικό πρόβλημα είναι αν για κάθε φυσικό αριθμό, η ακολουθία δεν είναι άπειρη, δηλαδή κάποια στιγμή γίνεται ίσο με το 1 και τερματίζει]

Εμφωλιασμένοι Βρόχοι

- Οι τρεις εντολές επανάληψης, είναι απλά τρεις ακόμη (πιο περίπλοκες) εντολές.
- Έτσι είναι συχνό να έχουμε μια εντολή επανάληψης μέσα σε μια εντολή επανάληψης.
  - που συχνά αναφέρεται σαν **nested loops** (εμφωλιασμένοι βρόχοι)

Παράδειγμα 4: nested_loops

Το παρακάτω πρόγραμμα υπολογίζει την προπαίδεια των 1,2,…, 9

```
for (let i=1; i<=10; i++) {
  for (let j = 1; j <= 10; j++)
    console.log(`${i}*${j} = ${i * j}`);
  console.log("=======");
}
```

Ο εσωτερικός βρόχος τρέχει για κάθε τιμή του i του εξωτερικού βρόχου.

Παράδειγμα 5: nested_loops2

Το πρόγραμμα αυτό εμφανίζει ένα ενδιαφέρον σχήμα:

```
let s;

for (let i=1; i<=10; i++) {
  s = "";
  for (let j = 1; j <= i; j++)
    s += "*";
  console.log(s);
}
```

Παράδειγμα 6: nestedLoops3.java

Το παρακάτω πρόγραμμα εμφανίζει μια παραλλαγή του παρ.5:

```
const N = 10;
let s;

for (let i=0; i<N; i++) {
  s = "";
  for (let j=0; j<N-i-1; j++)
    s+=" ";
  for (let j=0; j<i+1; j++)
    s+="*";
  console.log(s)
}
```

Break

- Αν στη διάρκεια της εκτέλεσης ενός βρόχου αποφασίσουμε ότι θέλουμε να διακόψουμε την εκτέλεση του (άρα να μην εκτελεστούν τα επόμενα βήματά του), χρησιμοποιουμε την εντολή break
- Συνηθισμένο συντακτικό (π.χ. σε for, αλλά ισχύει και για τις άλλες δομές επανάληψης):

```
for (init; cond; step) {
   ...
   if (condition)
      break;
}
```

Continue

- Αν στη διάρκεια της εκτέλεσης ενός βρόχου αποφασίσουμε ότι θέλουμε να παρακάμψουμε την τρέχουσα επανάληψη και να προχωρήσουμε στην επόμενη, χρησιμοποιουμε την εντολή continue
- Συνηθισμένο συντακτικό (π.χ. σε for, αλλά ισχύει και για τις άλλες δομές επανάληψης):

```
for (init; cond; step) {
   ...
   if (condition)
      continue;
}
```

Παράδειγμα 7: break

```
for (let i=0; i<10; i++)
{
  console.log(i);
  if (i===5)
    break;
}
```

Παράδειγμα 8: continue

```
for (let i=0; i<10; i++) {
  if (i%2===0)
    continue;
  console.log(i);
}
```

Ετικέτες Μπλοκ Κώδικα:

- Μπορούμε να "ονομάσουμε" ένα μπλοκ κώδικα με μία ετικέτα.
  - Η ετικέτα είναι ένα όνομα που εμείς επιλέγουμε και μπαίνει μπροστά από το μπλοκ κώδικα ακολουθούμενο από άνω κάτω τελεία.

```
label: {
    ...
}
```

  - και με αντίστοιχο τρόπο μπορούμε να ορίσουμε την ετικέτα ενός μπλοκ κώδικα που ορίζεται από μία δομή επανάληψης, βάζοντας την ετικέτα (με την άνω κάτω τελεία πριν από το βρόχο), π.χ.:

```
label: for (int i=0; i<100; i++) {
    ...
}
```

- Μπορούμε να χρησιμοποιήσουμε την ετικέτα σε μία break ώστε να προσδιορίσουμε ποιο μπλοκ κώδικα επιθυμούμε να διακοπεί.

```
label: for (int i=0; i<100; i++) {
   if (...)
      break label;
}
```

Σημείωση: Οι ετικέτες διακοπής βρόχου είναι πολύ χρήσιμες, όταν θέλουμε διακόψουμε την εκτέλεση εμφωλιασμένων βρόχων.

Παράδειγμα 9: break_with_label

```
outer: for (let i=1; i<=10; i++)
  for (let j=1; j<=10; j++)
    if (i*j===56) {
       console.log(i+","+j);
       break outer;
    }
```

Παρατηρήσεις:

- Στην ουσία είναι ένας πολιτισμένος τρόπος για να προσομοιωθεί η μοναδική χρησιμότητα του goto της C/C++ (το οποίο δεν υπάρχει στη JavaScript)
- Η break ακολουθούμενη από την ετικέτα, μπορεί να αναφέρεται μόνο σε μπλοκ κώδικα που περιέχεται στην ίδια εμβέλεια σε επίπεδο συνάρτησης
  - Δηλαδή δεν μπορούμε να χρησιμοποιήσουμε τις ετικέτες για να μεταφερθούμε οπουδήποτε στον κώδικά μας.
  - **Και η συνιστώμενη χρήση του είναι, μόνον αυτή, δηλαδή να βγαίνουμε «εύκολα» από εμφωλιασμένους βρόχους.**