WMNFE 2210

FRONT-END DEVELOPMENT

JavaScript #3 | Loops

CREATIVE MEDIA EDUCATION

ΠΕΡΙΕΧΟΜΕΝΑ

Περιεχόμενα

- Τελεστές
  - **+=**, **-=**, ***=**, **/=**
  - **++**, **--**
- Εντολές επανάληψης / Loops
  - Εντολή **while**
  - Εντολή **do..while**
  - Εντολή **for**
  - Εντολές **break** & **continue**
  - Μετατροπές

ΤΕΛΕΣΤΕΣ

Τελεστές

+=, -=, *=, /=

Πέρα από τον **βασικό** τελεστή ανάθεσης **=** (*assignment operator*), υπάρχουν και άλλοι, που συνδυάζουν τον τελεστή ανάθεσης με κάποιον αριθμητικό τελεστή.

Δεν προσφέρουν κάτι καινούργιο, αλλά έναν διαφορετικό, πιο συνοπτικό τρόπο, για συνηθισμένες ενέργειες / πράξεις.

```
i += 1;    // i = i + 1;
i -= 1;    // i = i - 1;
i *= 1;    // i = i * 1;
i /= 1;    // i = i / 1;
```

Τελεστές

++, --

Οι τελεστές αυτοί μπορεί να μην έχουν αποκλειστική χρήση μέσα στις επαναλήψεις, αλλά τους συναντάμε συχνά εκεί.

Η αύξηση ή η μείωση ενός αριθμού κατά *1* είναι μια αρκετά συνηθισμένη λειτουργία, για αυτό και υπάρχουν ειδικοί τελεστές για το σκοπό αυτό.

- **++** = αύξηση κατά *1*
- **--** = μείωση κατά *1*

Τελεστές

++, --

Χρήση

i++ ή **++i**

i-- ή **--i**

Προσοχή: Έχει σημασία αν ο τελεστής βρίσκεται πριν ή μετά την μεταβλητή.

Καθορίζεται, με αυτόν τον τρόπο, αν η τιμή της μεταβλητής αλλάξει πριν ή μετά την εκτέλεση της εντολής στην οποία περιλαμβάνεται.

Τελεστές

Παράδειγμα

```
let counter = 5;

counter++; // counter = counter + 1; ή counter += 1;
--counter; // counter = counter - 1; ή counter -= 1;

console.log(counter++);
// Ισοδύναμο με:
// console.log(counter);
// counter = counter + 1;

console.log(--counter);
// Ισοδύναμο με:
// counter = counter - 1;
// console.log(counter);

// Τι θα εμφανιστεί στην κονσόλα;
```

LOOPS

Loops

Loops

Με τις επαναλήψεις μπορούμε να επαναλάβουμε ένα τμήμα κώδικα, το οποίο θα εκτελεστεί είτε για συγκεκριμένο, είτε για άγνωστο πλήθος επαναλήψεων.

Οι κλασικοί βρόχοι επανάληψης, στη **JavaScript** είναι οι:

- **for** loops
- **while** loops
- **do..while** loops

Εντολή while

Σύνταξη

```
while (condition) {
  statements
}
```

Οι εντολές μέσα στο σώμα της **while** εκτελούνται **όσο** η συνθήκη είναι *αληθής* (**true**).

Εντολή while

Παρατηρήσεις

- Με μια εντολή επανάληψης υπάρχει ο κίνδυνος να πέσουμε σε *"ατέρμων βρόχο"* / *"infinite loop"*
- Πρέπει να φροντίζουμε ώστε, είτε η συνθήκη ελέγχου της επανάληψης να γίνει κάποια στιγμή **false**, είτε να σταματήσει η εκτέλεση της επανάληψης με άλλα μέσα (π.χ. εντολή **break**)

Εντολή while

Παράδειγμα #1

```
let i

// Display numbers from 0 through 4
i = 0;
while (i < 5) {
  console.log(i);
  i = i + 1;      // i++; i += 1;
}

// Display numbers from 1 through 5
i = 0;
while (i < 5) {
  i = i + 1;      // i++; i += 1;
  console.log(i);
}
```

Εντολή while

Παράδειγμα #2

```
let i;

i = 0;
while (i < 5) {
  document.write("Counter i = " + i + "<br>");
  i = i + 1;
}

// Same result
i = 0;
while (i < 5) {
  document.write(`Counter i = ${i++}<br>`);
}
```

Εντολή do...while

Σύνταξη

```
do {
  statements
} while (condition);
```

Όμοια λειτουργία με την **while**, μόνο που οι εντολές εκτελούνται σίγουρα μια φορά.

Εντολή do...while

Παράδειγμα #1

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

Εντολή do...while

Παράδειγμα #2

```
let answer, age;

do {
  age = parseInt(prompt('What is your age?'));
} while (isNaN(age));
```

CLASSWORK

Classwork

Classwork #1

Χρησιμοποιώντας την εντολή **while**, φτιάξτε ένα παιχνίδι που να μαντεύει τον αριθμό που σκέφτεται ο παίκτης!

- Ζητήστε αρχικά από τον παίκτη να σκεφτεί έναν αριθμό μεταξύ του *1* και του *100*.
- Μέσα σε μία επανάληψη ρωτήστε τον αν ο αριθμός που σκέφτεται ο παίκτης είναι το *50* (η μέση, δηλαδή, των *1* και *100*, στρογγυλοποιημένη προς τα κάτω).
- Αν είναι, βρήκαμε τον αριθμό και το παιχνίδι σταματά!
- Αν δεν είναι, ρωτήστε τον παίκτη αν ο αριθμός που σκέφτεται είναι μεγαλύτερος ή μικρότερος του *50*.
- Αν είναι μεγαλύτερος, επαναλάβατε τη διαδικασία, αλλά τώρα για το διάστημα *51* με *100*, καθώς ο αριθμός βρίσκεται σίγουρα εκεί μέσα.
- Αν όχι, συνεχίστε τη διαδικασία για το διάστημα *1* έως *49*, κ.ο.κ.

Εντολή for

Σύνταξη

```
for (initialization; condition; step) {
  statements
}
```

- **initialization**: Εκτελείται **μία φορά** πριν από τον κώδικα του block.
- **condition**: Καθορίζει τη συνθήκη της επανάληψης, ελέγχεται στο τέλος **κάθε** επανάληψης.
- **step**: Εκτελείται μετά από **κάθε** επανάληψη.

Εντολή for

Παράδειγμα

```
// 0 1 2 3 4
for (let i = 0; i < 5; i++) {
  console.log(i);
}

// 5 4 3 2 1
for (let i = 5; i > 0; i--) {
  console.log(i);
}
```

Εντολή break

Λειτουργία

Η εντολή **break** χρησιμοποιείται για να σταματήσει η επανάληψη. Συνήθως συνοδεύεται από μία εντολή **if**.

Παράδειγμα

```
for (let i = 0; i < 10000; i++) {
  if (i > 10) {
    break;
  }

  console.log(i);
}
```

Εντολή continue

Λειτουργία

Εάν θέλουμε να παραλείψουμε τις εντολές που έχουν απομείνει σε κάποιο βήμα μιας επανάληψης και να συνεχίσουμε με το επόμενο χρησιμοποιούμε την εντολή **continue**.

Παράδειγμα

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

  console.log(i);
}
```

CLASSWORK

Classwork

Classwork #2

Χρησιμοποιώντας την εντολή **for**, φτιάξτε ένα Χριστουγεννιάτικο δέντρο!

- Ζητήστε αρχικά, από το χρήστη, το ύψος του δέντρου.
- Χρησιμοποιώντας το χαρακτήρα *, τυπώστε στην κονσόλα (ή αν θέλετε μέσα στη σελίδα), ένα δεντράκι με το ζητούμενο ύψος.
- Αν π.χ., το ύψος θέλουμε να είναι *5* το αποτέλεσμα πρέπει να είναι το εξής:

```
    *
   ***
  *****
 *******
    *
```

Μετατροπές

while → do...while

```
while (condition) {
  statements
}
```

⇓

```
do {
  if (condition) {
    statements
  }
} while (condition)
```

Μετατροπές

do...while → while

```
do {
  statements
} while (condition)
```

⇓

```
statements
while (condition) {
  statements
}
```

Μετατροπές

for ↔ while

```
for (let i = 0; i < n; i++) {
  statements
}
```

⇕

```
let i = 0;

while (i < n) {
  statements
  i++;
}
```

HOMEWORK

Homeworks

Homework #1

Τροποποιήστε το **Classwork #1** (*"μάντεψε τον αριθμό"*) και υλοποιήστε τις εξής βελτιώσεις:

1. Προσθέστε έναν μετρητή που να μετρά των αριθμό των προσπαθειών.
2. Μόλις ο αριθμός βρεθεί και τελειώσει η επανάληψη, εμφανίστε το μήνυμα *"I found the number you were thinking about (x) in n try(ies)!"*
   - όπου **x** & **n** να εμφανίζονται οι κατάλληλες τιμές.
3. Αν οι προσπάθειες ξεπεράσουν τις *7*, να εμφανίζεται το μήνυμα *"There is something fishy about your answers..."* και να σταματά η επανάληψησ.
   - **ΣΗΜ.1**: με την παραπάνω μέθοδο αποκλείεται να μη βρεθεί ο αριθμός σε *7* επαναλήψεις.
   - **ΣΗΜ.2**: το *7* προκύπτει από τον υπολογισμό **Math.floor(Math.log2(100) + 1)**

Homeworks

Homework #2

Υλοποιήστε το παιχνίδι **FizzBuzz** σε JavaScript!

1. Ξεκινήστε δημιουργώντας μία επανάληψη που να εμφανίζει στην κονσόλα τους αριθμούς από το *1* έως το *100*.
2. Αν ένας αριθμός διαιρείται με το *3* (αλλά όχι με το *5*), εμφανίζεται στη θέση του αριθμού η λέξη *"Fizz"*.
3. Αν ένας αριθμός διαιρείται με το *5* (αλλά όχι με το *3*), εμφανίζεται στη θέση του αριθμού η λέξη *"Buzz"*.
4. Αν ένας αριθμός διαιρείται και με το *3* και με το *5*, εμφανίζεται στη θέση του αριθμού η λέξη *"FizzBuzz"*.

Χρήσιμα links

Basic operators, maths
https://javascript.info/operators

Loops: while and for
https://javascript.info/while-for

Loops and iteration - JavaScript | MDN
https://developer.mozilla.org/en-US/docs/Web/JavaS…

Extra info

javascript - What's the difference between using "le…
https://stackoverflow.com/questions/762011/whats-th…

Operator Lookup - Search JavaScript operators
https://www.joshwcomeau.com/operator-lookup/

let | Can I use... Support tables for HTML5, CSS3, etc
https://caniuse.com/let

THANK YOU!