JavaScript

psounis

ΠΕΡΙΕΧΟΜΕΝΑ:

1. Τελεστές
2. Προτεραιότητα Τελεστών
3. Οι τελεστές == και ===
4. Περισσότερα για τους Λογικούς Τελεστές
5. Bitwise τελεστές

Νικόλαος Βασιλειάδης

Σμαραγδένιος Χορηγός Μαθήματος

Νικόλαος Β.

Σμαραγδένιος Χορηγός Μαθήματος

ΜΑΘΗΜΑ 4.1: Τελεστές

1. Τελεστές

JavaScript psounis

- Με χρήση απλών παραστάσεων (εκφράσεις, expressions), μπορούμε να κατασκευάσουμε άλλες, πιο συνθετες χρησιμοποιώντας **τελεστές (operators)**
  - Υπάρχουν οι αριθμητικοί, σχεσιακοί, λογικοί τελεστές καθώς και οι τελεστές καταχώρησης και bitwise τελεστές.

Αριθμητικοί Τελεστές (arithmetic)

- +, -, *, /, %, ** **(Συνήθεις μαθηματικοί τελεστές)**
  - Εφαρμόζονται σε αριθμούς (το + εφαρμόζεται και σε συμβολοσειρές - συνένωση).
- +=, -=, *=, /=, %= **(Τελεστές Επαυξημένης Καταχώρησης)**
  - Αριστερά μπαίνει μεταβλητή και δεξιά παράσταση
  - Είναι συντομογραφία πιο σύνθετης έκφρασης:
    - π.χ. το **a+=3** είναι συντομογραφία του **a = a + 3**
- ++, -- **(Μοναδιαία αύξηση/μείωση)**
  - Προθεματική μορφή (π.χ. ++a) πρώτα εκτελεί την αύξηση κατά 1 της μεταβλητής και μετά υπολογίζει την παράσταση στην οποία ανήκει.
  - Μεταθεματική μορφή (π.χ. a++) πρώτα υπολογισμός της παράστασης που ανήκει και έπειτα αύξηση κατά 1.

Παράδειγμα 1: increment

```
let x=0;
console.log(x++);
console.log(++x);
```

Σχεσιακοί Τελεστές (relational)

- === **(αυστηρή ισότητα)**, == **(ισότητα)**
- !== **(αυστηρά όχι ίσα)** , != **(όχι ίσα)**
- **>, <, >=, <= (μεγαλύτερο από κ.ο.κ.)**

Αυστηρή ισότητα είναι ισότητα και κατά τον τύπο δεδομένων, χωρίς να λάβουν χώρα μετατροπές (βλέπε επόμενη διαφάνεια)

Λογικοί Τελεστές (logical)

- **&& (και)**
- **|| (ή)**
- **! (όχι)**

Ιδιαιτερότητες με truthy/falsy τιμές, κάνουν lazy evaluation της παράστασης (βλ. επόμενη διαφάνεια)

Παράδειγμα 2: logical

```
console.log(8+1===9);
console.log(6*2!==7-3 && 5/4>4%2);
console.log(5/2===2 || 5/0===0);
console.log(!(0===0));
console.log("1"==1);
let x = 0 || 5/0;
console.log(x);
```

Η προτεραιότητα των τελεστών (από μεγαλύτερη προς μικρότερη):
(google: mdn js operator precedence)

Level	Operator	Description	Associativity
21	()	grouping	left-to-right
	.	member access	left-to-right
	[]	computed memb.access	left-to-right
20	new	new (with args)	n/a
	()	function call	left-to-right
	?.	optional chaining	left-to-right
19	new	new (without args)	right-to-left
18	++,--	postfix inc/dec	n/a
	!, ~	not (logical, bitwise)	
	+, -	unary plus, negation	
17	++,--	prefix inc/dec	right-to-left
	typeof, void, delete, await	typeof, void delete, await	
16	**	exponentation	right-to-left
15	* /, %	multiplication division, remainder	left-to-right
14	+, -	addition, subtraction	left-to-right
13	<<, >>, >>>	bitwise shift	left-to-right
12	<, <=, >, >= in, instanceof	comparison operators in, instanceof	left-to-right
11	==,!=,===,!==	equality operators	left-to-right
10	&	bitwise AND	left-to-right
9	^	bitwise XOR	left to right
8	|	bitwise OR	left to right
7	&&	logical AND	left to right
6	||	logical OR	left to right
5	??	Nullish Coalescing Op	left-to-right
4	?:	ternary	right to left
3	= += -= **= *= /= %= &= ^= |= <<= >>= >>>= &&= ||= ??=	augmented assignment	right-to-left
2	yield, yield*	yield	right-to-left
1	,	comma /sequence	left-to-right

Παρατηρήσεις:

- Αριθμητικοί Τελεστές: Συνήθης μαθηματική προτεραιότητα
- Σχεσιακοί Τελεστές: Μικρότερη προτεραιότητα από τους αριθμητικούς τελεστές.
- Λογικοί Τελεστές: Μικρότερη Προτεραιότητα από τους σχεσιακούς τελεστές.

Και ασφαλώς δεν ξεχνάμε ότι μπορούμε να καθορίσουμε την προτεραιότητα με παρενθέσεις οποτεδήποτε δεν είμαστε σίγουροι για την σειρά των πράξεων

3. Οι τελεστές == και ===

Ο τελεστής === (αυστηρή ισότητα - strict equality)

- αφού υπολογίζει τους τελεστέους σε τιμές,
- Αν αυτές έχουν τον ίδιο τύπο δεδομένων ΚΑΙ την ίδια τιμή, τότε επιστρέφει true (αλλιώς επιστρέφει false)

Ειδικές Περιπτώσεις:

- Ισχύουν (είναι true):
  - undefined === undefined
  - null === null
- Δεν ισχύει (είναι false)
  - NaN === NaN
- Δύο συμβολοσειρές είναι ίσες αν και μόνο αν περιέχουν ακριβώς τους ίδιους χαρακτήρες

Ο τελεστής !== (αυστηρή ανισότητα - strict inequality)

- είναι αληθής όταν ο τελεστής === είναι ψευδής κ' αντίστροφα

Παράδειγμα 3: strict_inequality

```
console.log(8+1 === 9);
console.log(8+1 === "9");
console.log(1/0 === 5/0);
console.log("1" + 1 !== "11");
```

Ο τελεστής == (ισότητα - equality)

- αφού υπολογίζει τους τελεστέους σε τιμές,
- ελέγχει αν είναι αυστηρά ίσοι. Αν ναι, υπολογίζεται σε true, αλλιώς επαναλαμβάνει κάνοντας **μετατροπές**:
  - Αριθμός με Συμβολοσειρά: Η συμβολοσειρά μετατρέπεται σε αριθμό (αν είναι εφικτό, αλλιώς false).
  - true με άλλο τ.δ.: Το true γίνεται 1
  - false με άλλο τ.δ.: Το false γίνεται 0

- undefined, null είναι ίσα μόνο με τον εαυτό τους.
  - Με την εξαίρεση ότι: undefined == null
- (σε επόμενο μάθημα) τα αντικείμενα μετατρέπονται αυτόματα σε primitive με τη μέθοδο valueOf()

Παράδειγμα 4: equality

```
console.log(8+1 == "9");
console.log(true == 1);
console.log(false == "0");
console.log(undefined != null);
```

Ο τελεστής != (ανισότητα - inequality)

- είναι αληθής όταν ο τελεστής == είναι ψευδής κ' αντίστροφα

JS Beef:

- Θα χρησιμοποιούμε πάντα τους αυστηρές εκδοχές των τελεστών

4. Περισσότερα για τους Λογικούς Τελεστές

Λογικός Τελεστής OR (||)

- Ο συνηθισμένος τρόπος είναι να τη χρησιμοποιούμε επί αποτελεσμάτων πράξεων που έχουν boolean αποτέλεσμα, όπως π.χ. x<0 || x>10
  - οπότε η τιμή τους υπολογίζεται με βάση τον πίνακα αλήθειας του OR (true, αν τουλάχιστον ένα από τα δύο είναι true)
- **Προσοχή** ότι γίνεται lazy evaluation, δηλ:
  - Αν το 1ο μέρος του A || B είναι true, τότε το αποτέλεσμα είναι true (και το B δεν υπολογίζεται)
  - Αν το 1ο μέρος του A && B είναι false, τότε το αποτέλεσμα είναι false (και το B δεν υπολογίζεται)

Υπενθύμιση (Μάθημα 2):

- falsy τιμές: 0, undefined, null, NaN και η κενή συμβ/ρά “”
- truthy τιμές: Κάθε άλλη τιμή

- Αν ωστόσο το A είναι κάποια truthy τιμή: επιστρέφεται το ίδιο το A (μόνο αν ήταν σε κάποιο λογικό έλεγχο, μετατρέπεται σε true)
- Ενώ αν το A έχει falsy τιμή, τότε το αποτέλεσμα είναι το B (!)

Παράδειγμα 5: or

```
console.log(5<3 || 1 < 2);
console.log("" || 2);
console.log("a" || 1);
```

Σημείωση:

- Το προηγούμενο παράδειγμα ίσως φαίνεται παράξενο σε κάποιον που έρχεται από C/C++/Java
- Ωστόσο, πρέπει να το καταλάβουμε: Η JS έχει σχεδιαστεί έτσι ώστε να μην προκαλεί σφάλματα.
- Συνεπώς τέτοιες συμπεριφορές είναι συχνές: Θα γίνεται αλλαγή τύπου (type coercion) ώστε πάντα να επιστρέφεται αποτέλεσμα.
- Οι προγραμματιστές εκμεταλλεύονται αυτή τη συμπεριφορά για να γράψουν προγράμματα που κάνουν έμμεσους ελέγχους (βλ. επόμενο παράδειγμα)

Παράδειγμα 6: or2

```
let surname = null;
let result = surname || "No Value";
console.log(result);
surname = "Doe";
result = surname || "No Value"
console.log(result);
```

Λογικός Τελεστής AND:

- Εντελώς αντίστοιχα για την έκφραση A&&B:
  - Αν A είναι false/y επιστρέφεται το A
  - Αλλιώς επιστρέφεται το B

Παρατήρηση:

- Γενικά θα αποφεύγουμε να χρησιμοποιούμε τέτοιου τύπου κόλπα, ώστε ο κώδικάς μας να είναι κατά το δυνατόν ευανάγνωστος.

5. Bitwise Τελεστές

Τελεστές bit: Οι τελεστές bit κάνουν πράξεις bit σε αριθμούς (number και biginteger). Αυτοί είναι:

- **Λογικό ΚΑΙ (&).** Π.χ.:

	0	0	1	0	1	0	1	1	(43)
&	0	1	1	0	0	1	1	1	(103)
	0	0	1	0	0	0	1	1	(35)

- **Λογικό Ή (|).** Π.χ.:

	0	0	1	0	1	0	1	1	(43)
|	0	1	1	0	0	1	1	1	(103)
	0	1	1	0	1	1	1	1	(111)

- **Λογικό ΟΧΙ(~).** Π.χ.:

~	0	0	1	0	1	0	1	1	(43)
	1	1	0	1	0	1	0	0	(212)

- **Λογικό XOR(^).** Π.χ.:

	0	0	1	0	1	0	1	1	(43)
^	0	1	1	0	0	1	1	1	(103)
	0	1	0	0	1	1	0	0	(76)

Τελεστές ολίσθησης (μόνο για ακέραιες μεταβλητές):

- Δεξιά ολίσθηση n θέσεων (>>). Πχ. 5>>1 == 2
- Αριστερή ολίσθηση n θέσεων (<<). Πχ. 5<<1 == 10
- Δεξιά ολίσθηση n θέσεων, με γέμισμα μηδενικών (>>>)

Υπάρχουν και οι τελεστές επαυξημένης καταχώρησης:

- &=, |=, ^=,
- >>=, <<=, >>>=

π.χ. η εντολή a>>=1 είναι ισοδύναμη με την a=a>>1.

Παράδειγμα 7: bitwise.html

```
console.log(43 & 103);
console.log(43 | 103);
console.log(~1);
console.log(43 ^ 103);
console.log(5 >> 1);
console.log(5 << 1);
console.log(-1 << 1);
console.log(-1 >> 1);
console.log(-1 >>> 1);
let x = 1;
x |= 3;
console.log(x);
x &= 2;
console.log(x);
```

Ιδιαιτερότητες της JS:

- Προτού εκτελέσει μία bitwise πράξη, οι τελεστέοι μετατρέπονται σε ακεραίους (με αποκοπή δεκαδικών ψηφίων) και το αποτέλεσμα μετατρέπεται πάλι σε floating point.
- Γενικά οι τελεστές αυτοί, σπάνια χρησιμοποιούνται στην πράξη.