ΠΕΡΙΕΧΟΜΕΝΑ:

1. Κατασκευή Συμβολοσειράς
2. Συναφής Λειτουργικότητα με Πίνακες
3. Περισσότερα για τα template literals
4. Μέθοδοι Συμβολοσειρών
    1. «Διόρθωση» Συμβολοσειράς
    2. Μέρη και Επανάληψη Συμβολοσειράς
    3. Αναζήτηση και Αντικατάσταση

Ανδρέας Γ.
Σμαραγδένιος Χορηγός Μαθήματος

Δέσποινα Μωραΐτη
Χρυσός Χορηγός Μαθήματος

ΜΑΘΗΜΑ 9.2: Συμβολοσειρές

1. Κατασκευή Συμβολοσειράς

JavaScript psounis

Υπενθυμίσεις για τις Συμβολοσειρές (από το μαθ. 2)

- Είναι immutable (που σημαίνει ότι δεν υπάρχει τρόπος να τροποποιήσουμε μία συμβολοσειρά από τη στιγμή που την κατασκευάσαμε).
- Κωδικοποίηση Χαρακτήρων με UTF-16 (1 ή 2 bytes)
- Literals σε 'μονά' ή "διπλά" εισαγωγικά.
- Διευκόλυνση με τα `string templates` (backticks)
- Λειτουργικότητα:
  - str[i]: Χαρακτήρας στη θέση i (επιστρέφει συμβ/ρα)
  - str.length: Το μήκος της συμβολοσειράς
- Τελεστές επί strings: +, +=, καθώς και οι σχεσιακοί (==, <, κ.λπ.)

Σημειώσεις:

- Ένα string literal είναι primitive (δεν είναι αντικείμενο, δεν έχει π.χ. μεθόδους)
- Ωστόσο μπορούμε να έχουμε πρόσβαση σε μεθόδους σε string literal, π.χ. μπορούμε να γράψουμε:
  - "string".length
  - Αντίστοιχα με τους αριθμούς, λαμβάνει χώρα η ενθυλάκωση (boxing) της συμβ/ρας σε αντικείμενο και έπειτα καλείται η μέθοδος (ή το property).
- Εσωτερικά, η συμβολοσειρά είναι ένας μη τροποποιήσιμος πίνακας χαρακτήρων.

Κατασκευή Συμβολοσειράς:

- Εκτός από το string literal, μπορούμε να χρησιμοποιήσουμε και:
  - **new String(s)**: Παράγει νέο αντικείμενο - συμβολοσειρά (συνάρτηση - κατασκευαστής)
  - **String(s)**: Μετατρέπει το s σε συμβολοσειρά (συνάρτηση)

Παράδειγμα 1: string

```
let str = new String("a string");
let str2 = String("a string");
let str3 = "a string";
console.log(str, typeof str);
console.log(str2, typeof str2, str3, typeof str3);
```

Παράδειγμα 2: boxing

```
let str = "a string";
console.log(str.length);
console.log(Object(str));
```

```
8
▼ String
    0: "a"
    1: " "
    2: "s"
    3: "t"
    4: "r"
    5: "i"
    6: "n"
    7: "g"
    length: 8
  ▶ [[Prototype]]: String
    [[PrimitiveValue]]: "a string"
```

Επανάληψη επί Συμβολοσειράς:

- Επανάληψη με την for..in:
  - Όπως και στους πίνακες, επιστρέφει τα ονόματα των properties (θέσεις του πίνακα).
- Επανάληψη με την for..of:
  - Έχει οριστεί επαναληπτής (βλ. μαθ.9.1) που επιστρέφει τους διαδοχικούς χαρακτήρες της συμβολοσειράς.

Παράδειγμα 3: loops

```
for (let pos in "string")
   console.log(`position: ${pos}`);

for (let char of "string")
   console.log(`character: ${char}`);
```

```
position: 0
position: 1
position: 2
position: 3
position: 4
position: 5
character: s
character: t
character: r
character: i
character: n
character: g
```

Μέθοδοι (με ίδια συμπεριφορά όπως στους πίνακες):

- Λόγω της συνάφειας με τους πίνακες, περιέχει αρκετές από τις μεθόδους που είδαμε στα μαθήματα 7.1 - 7.2
  - Πρακτικά είναι αυτές που δεν προκαλούν τροποποίηση της συμβολοσειράς.

Μέθοδος	Επεξήγηση
indexOf(str[,n])	Θέση του str [ξεκινώνας από το n]
lastIndexOf(str[,n])	Θέση του str [ξεκινώντας από το -1]
includes(str)	true/false, αν υπάρχει η str
slice(start[,end])	Κομμάτι από το start [έως το end-1]
concat(str2)	invoking + str2

Παράδειγμα 4: as_in_arrays

```
let str = "Sample String";
console.log(`length: ${str.length}`);
console.log(`char at pos 1: ${str[1]}`);
console.log(`index of 'S': ${str.indexOf("S")}`);
console.log(`lastIndex of 'S': ${str.lastIndexOf("S")}`);
console.log(`includes 'tr': ${str.includes("tr")}`);
console.log(`slice positions 2..4: ${str.slice(2,5)}`);
console.log(`concat with itself: ${str.concat(str)}`);
```

ΜΑΘΗΜΑ 9.2: Συμβολοσειρές — 3. Περισσότερα για τα Template Literals

JavaScript psounis

Υπενθυμίσεις:
- Εντίθενται σε backticks (`)
- Μέσα σε ${…} γράφουμε εκφράσεις που υπολογίζονται σε μία τιμή (η οποία και ενσωματώνεται στη συμβολοσειρά)

Template Literals πολλών γραμμών:
- Τα template literals μπορούν να εκτείνονται σε πολλές γραμμές (η αλλαγή γραμμής διατηρείται ως χαρακτήρας και τυχόν whitespace διατηρείται)

Σημείωση:
- Πριν την ES6 χρησιμοποιούταν ο χαρακτήρας backslash για την αλλαγή γραμμής σε ένα string.

Παράδειγμα 5: multiline

```
let str = `a multiline
        string`;
let str2 = "a multiline \n\
        string";
console.log(str);
console.log(str2);
```

```
a multiline
        string
a multiline
        string
```

Tagged Template Literals:
- Προχωρημένος τρόπος διαχείρισης template strings μέσω συναρτήσεων (που θα δούμε μέσω ενός παραδείγματος)

Παράδειγμα 6: tagged_templates

```
function tagged(strings, arg1, arg2) {
  for (let s of strings)
    console.log(s);
  console.log(arg1);
  console.log(arg2);

  return arg1;
}
let result = tagged`XX${"first"}YY${"second"}ZZ`;
console.log(result);
```

```
top ▾ | Filter
XX
YY
ZZ
first
second
first
```

Τρόπος Λειτουργίας:
- Το tagged literal χωρίζει τα μέρη σε δύο ομάδες: Αυτά που είναι μεταξύ των ${} και τα ${}.
- Αυτά διοχετεύονται στη συνάρτηση - διαχειριστή του tag literal:
  - Η 1η ομάδα διοχετεύεται στη συνάρτηση ως πίνακας στο 1ο όρισμα
  - Τα επόμενα ορίσματα είναι οι τιμές των διαδοχικών ${} στο tagged literal.
- Τελικά η συνάρτηση επιστρέφει οποιαδήποτε σύνθεση αυτών των τιμών ως συμβολοσειρά:
  - Με τον τρόπο αυτό έχουμε απόλυτο έλεγχο στο τι θα επιστρέφει το template string, ενσωματώνοντας οποιαδήποτε προγραμματιστική λειτουργικότητα επιθυμούμε.

Μέθοδοι για το case:

Μέθοδος	Επεξήγηση
toLowerCase()	Όλοι οι χαρακτήρες με μικρά
toUpperCase()	Όλοι οι χαρακτήρες σε κεφαλαία

Αφαίρεση whitespace (κενά, tabs, αλλαγές γραμμής κ.α.) στα άκρα:

Μέθοδος	Επεξήγηση
trim()	Αφαιρεί το whitespace και από τα δύο άκρα της συμβ/ράς
trimStart()	Αφαιρεί το whitespace από την αρχή της συμβ/ράς
trimEnd()	Αφαιρεί το whitespace από το τέλος της συμβ/ράς

Γέμισμα στα άκρα:

Μέθοδος	Επεξήγηση
padStart(n[, str])	Γεμίζει τη συμβολοσειρά με επαναλήψεις της str από αριστερά, μέχρι να γίνει μήκους n
padEnd(n[, str])	Γεμίζει τη συμβολοσειρά με επαναλήψεις της str από δεξία, μέχρι να γίνει μήκους n

- Το 2ο όρισμα είναι προαιρετικό. Αν το παραλείψουμε συμπληρώνεται με κενά.

Παρατήρηση:

- Αφού η συμβολοσειρά είναι immutable, όλες αυτές οι μέθοδοι επιστρέφουν νέα συμβολοσειρά με την ενέργεια που περιγράφεται στην επεξήγηση.
- Είναι συχνό να γίνονται ενέργειες επί της επιστρεφόμενης συμβολοσείρας κ.ο.κ. και να έχουμε μια αλυσίδωση ενεργειών, π.χ. str.toLowerCase().trim().padStart(10)

Παράδειγμα 7: string_adjustments

```
let str = "   Factotum "
console.log(`trimStart : |${str.trimStart()}|`);
console.log(`trimEnd   : |${str.trimEnd()}|`);
console.log(`trim      : |${str.trim()}|`);

str = "AaBb";
console.log(`toLowerCase: |${str.toLowerCase()}|`);
console.log(`toUpperCase: |${str.toUpperCase()}|`);

str = "abcd"
console.log(`padStart     : |${str.padStart(10, "-")}|`);
console.log(`padEnd       : |${str.padEnd(10, "-")}|`);

str = " Jim";
console.log(`|${str.toLowerCase().trim().padStart(10)}|`);
```

Εξαγωγή χαρακτήρα:

- Εκτός από την εξαγωγή με τον **τελεστή []**, έχουμε και την:

Μέθοδος	Επεξήγηση
at(n)	Χαρακτήρας στη θέση n

  - Σε αντίθεση με τον τελεστή [], δέχεται και αρνητικές τιμές (μέτρα από το τέλος της συμβολοσειράς: -1, -2, …)

Μέρος Συμβολοσειράς

- Εκτός από την **slice(start[,end])**, έχουμε και την:

Μέθοδος	Επεξήγηση
substring(start[, end])	Μέρος της συμβολοσειράς από το start [έως το end-1]

  - Παρατηρείται διαφορά στις δύο μεθόδους μόνο σε εκκεντρικές περιπτώσεις (π.χ. αν start>end: Η slice επιστρέφει την κενή συμβολοσειρά και η substring εναλλάσσει τις τιμές των ορισμάτων)

Χωρισμός σε Μέρη

Μέθοδος	Επεξήγηση
split(separator)	Επιστρέφει πίνακα συμβολοσειρών, χωρίζοντας την αρχική συμβολοσειρά σε μέρη, με διαχωριστή τον separator

Επανάληψη Συμβολοσειράς

- Αρκετά Χρήσιμη είναι και η Μέθοδος:

Μέθοδος	Επεξήγηση
repeat(n)	Επαναλαμβανει τη συμβολοσειρά n φορές

- Επαναλαμβάνει την ίδια συμβολοσειρά n φορες, σε αντίθεση με τη **array.join(str)** που παραθέτει τις συμβολοσειρές του πίνακα, ενωμένες με τη συμβολοσειρά που δέχεται ως όρισμα

Παράδειγμα 8: string_parts

```
let str = "A Sample String";
console.log(`char at pos 1: ${str[1]}, ${str.at(1)}`);
console.log(`char at pos -1: ${str[str.length - 1]}, ${str.at(-1)}`);
console.log(`substring positions 2..4: ${str.substring(2,5)},
${str.slice(2,5)}`);

let splitted = str.split(" ");
console.log(`split with ' ': ${splitted}`);
let joined = splitted.join("-");
console.log(`joined with '-': ${joined}`);

let repeated = str.split(" ")[0].repeat(10);
console.log(`repeated: ${repeated}`);
```

ΜΑΘΗΜΑ 9.2: Συμβολοσειρές — 4.3. Αναζήτηση και Αντικατάσταση

JavaScript psounis

Αναζήτηση Συμβολοσειράς:

- Εκτός από την αναζήτηση με τις μεθόδους **indexOf(elem[,n]), lastIndexOf(elem[,n]), includes(str)**, έχουμε και τις:

Μέθοδος	Επεξήγηση
search(regexp)	Θέση που εντοπίζεται η regexp
startsWith(str[,pos])	true/false [ξεκινώντας από το pos]
endsWith(str[,pos])	true/false [μέχρι το pos]

Αντικατάσταση σε Συμβολοσειρά

Μέθοδος	Επεξήγηση
replace(regexp, replacement)	Αντικαθιστά τις εμφανίσεις της regexp (την πρώτη φορά αν είναι συμβολοσειρά ή παραπάνω αν είναι κανονική έκφραση) με τη συμβολοσειρά (ή συνάρτηση) replacement
replaceAll(regexp, replacement)	Αντικαθιστά όλες τις εμφανίσεις της regexp με τη replacement, εφόσον αυτή είναι συμβολοσειρά.

Κανονικές Εκφράσεις:

- Η search(), διαφοροποιείται από π.χ. την indexOf(), στο γεγονός ότι δέχεται ως όρισμα κανονική έκφραση (regular expression).
- Οι κανονικές εκφράσεις είναι σύνθετες αναπαραστάσεις προτύπων συμβολοσειράς
  - π.χ. η κανονική έκφραση /aa.*bb.*cc/ εκφράζει τις συμβολοσειρές που ξεκινούν με aa, περιέχουν bb και τελειώνουν με cc
- Άλλες μέθοδοι συμβολοσειρών που υποστηρίζουν κανονικές εκφράσεις είναι οι:

Μέθοδος	Επεξήγηση
match(reg)	Πίνακας με τα ταιριάσματα της κανονικής έκφρασης reg
matchAll(reg)	Iterator με τα ταιριάσματα της κανονικής έκφρασης reg

- (Θα αφιερώσουμε το επόμενο μάθημα στις regexp)

Παράδειγμα 9: string_parts

```
let str = "A Sample String";
console.log(`search for /S.*e/: ${str.search(/S.*e/)}`);
console.log(`starts with 'Sample' at 2: ${str.startsWith("Sample", 2)}`);
console.log(`ends with 'String': ${str.endsWith("String")}`);

console.log(`matches /S../: ${str.match(/S../g)}`);

console.log(`replace 'S' with 'Xa': ${str.replace("S","Xa")}`);
console.log(`replaceAll 'S' with 'Xa': ${str.replaceAll("S","Xa")}`);
```