Τμήμα Πληροφορικής Εισαγωγή στην Επιστήμη Υπολογιστών

Εισαγωγή στη γλώσσα Javascript

Επιμέλεια: Άννα Κεφάλα

Εισαγωγή στη γλώσσα Javascript

2ο μέρος: δομές ελέγχου και συναρτήσεις

Δομές ελέγχου

- Υποθετικές Συνθήκες (conditional)
  - **if** κάποια συνθήκη είναι αληθής → εκτέλεσε το αντίστοιχο block κώδικα
  - **else** (αν η συνθήκη if είναι ψευδής) → εκτέλεσε το αντίστοιχο block κώδικα
  - **else if** όταν υπάρχει και εναλλακτική συνθήκη ελέγχου και η $1^{η}$ συνθήκη ελέγχου if είναι ψευδής
  - **switch** για προσδιορισμό περισσότερων διαφορετικών blocks κώδικα προς εκτέλεση
- Επανάληψη (loops)
  - **for - loops** επανάληψη ενός block κώδικα για έναν αριθμό επαναλήψεων
  - **for/in - loops** επανάληψη ενός block κώδικα βάση των properties ενός object
  - **for/of - loops** επανάληψη ενός block κώδικα βάση των τιμών ενός iterable object
  - **while - loops** επανάληψη ενός block κώδικα όσο μία συνθήκη ελέγχου είναι αληθής
  - **do/while** επανάληψη ενός block κώδικα όσο μία συνθήκη ελέγχου είναι αληθής

Δομές ελέγχου | υποθετικές συνθήκες

- Εκτέλεση διαφορετικού block εντολών ανάλογα με την τιμή κάποιας συνθήκης ελέγχου

```
const time = new Date().getHours(); // επιστρέφει την ώρα (0 – 23) με βάση την τοπική ώρα
let greeting;
if (time < 12) {
   greeting = "Καλημέρα";
} else if (time < 18) {
    greeting = "Χαίρετε";
} else {
   greeting = "Καλησπέρα";
}
```

[*01-ifElse.html*]

Δομές ελέγχου | υποθετικές συνθήκες (2)

- Εκτέλεση διαφορετικού block εντολών βάσει διαφορετικών περιπτώσεων (cases)

```
let day;
switch (new Date().getDay())
{
  case 0:
     day = "Κυριακή";
     break;
  case 1:
     day = "Δευτέρα";
     break;
  case 2:
     day = "Τρίτη";
     break;
  case 3:
     day = "Τετάρτη";
     break;
  case 4:
     day = "Πέμπτη";
     break;
  case 5:
     day = "Παρασκευή";
     break;
  case  6:
     day = "Σάββατο";
}
```

[02-switch.html]

Δομές ελέγχου | επανάληψη [for]

- **for**: εκτέλεση ενός block εντολών κατ' επανάληψη βάσει της τιμής κάποιας συνθήκης

```
for (expression 1; expression 2; expression 3) {
        // code block to be executed
}
```

Expression 1: εκτελείται 1 φορά **πριν** την εκτέλεση του block του κώδικα

Expression 2: ορίζει τη **συνθήκη εκτέλεσης** του block του κώδικα

Expression 3: εκτελείται κάθε φορά **μετά** την εκτέλεση του block του κώδικα

Δομές ελέγχου | επανάληψη [for] (2)

```
let text = "";
for (let i = 0; i < 5; i++) {
        text += "Ο αριθμός είναι: " + i + "\n";
}
```

όλα τα παραδείγματα για loops:
[03-loops.html]

Δομές ελέγχου | επανάληψη [for in]

- **for in**: εντολή επανάληψης για τις ιδιότητες ενός αντικειμένου

```
for (key in object) {
        // code block to be executed
}
```

- αντίστοιχα για τα στοιχεία ενός πίνακα

```
for (variable in array) {
        // code block to be executed
}
```

Δομές ελέγχου | επανάληψη [for in] (2)

```
const person = {fname:“Jane", lname:"Doe", age:18};
let txt = "";
for (let x in person) {
  txt += person[x] + " ";
}
```

```
const cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat", "Audi"];
let txt = "";
for (let x in cars) {
  txt += cars[x] + "\n";
}
```

Δομές ελέγχου | επανάληψη [for of]

- **for of** loop: χρησιμοποιείται για εκτέλεση block εντολών για κάθε στοιχείο μιας συλλογής αντικειμένων όπως Arrays, Strings, Lists, κλπ.

```
for (variable of iterable) {
        // code block to be executed
}
```

```
const cars = ["BMW", "Volvo", "Saab",
"Ford", "Fiat", "Audi"];

let txt1 = "";
for (y of cars) {
        txt1 += y + "\n";
}
```

```
let phrase = "Carpe Diem";
  let txt4 = "";
  for (l of phrase) {
    txt4 += l + "\n";
  }
```

Δομές ελέγχου | επανάληψη [while, do .. while]

- **while** loop χρησιμοποιείται για επαναληπτική εκτέλεση block εντολών όσο είναι αληθής κάποια συνθήκη

```
let txt5 = "";
let i = 0;
while (i < 10) {
        txt5 += "\n The number is " + i;
        console.log(i);
        i++;
}
```

- **do..while** loop χρησιμοποιείται για επαναληπτική εκτέλεση block εντολών όσο είναι αληθής κάποια συνθήκη

```
let txt6 = ""
let num = 0;

do {
 txt6 += "\n The number is " + num;
   console.log(i);
 num++;
}
while (num < 10);
```

Συναρτήσεις (functions)

- Ορισμός και κλήση (definition & invocation)
- Παράμετροι (parameters) και επιστροφή τιμών (return values)
- Εμβέλεια (scope)

Συναρτήσεις | ορισμός και κλήση

- η συνάρτηση είναι block κώδικα που κάνει μία συγκεκριμένη λειτουργία
- μπορεί να λαμβάνει μία λίστα παραμέτρων και να επιστρέφει μία τιμή εξόδου
- ο κώδικάς της εκτελείται όταν ενεργοποιηθεί (κληθεί) από κάποιο γεγονός ή συνθήκη ενεργοποίησης
  - όταν συμβεί κάποιο γεγονός (event) (π.χ. ο χρήστης κάνει κλικ σε ένα button)
  - όταν καλείται από JavaScript code
  - αυτόματα (self invoked)

```
function name(parameter1, parameter2, parameter3) {
        // code to be executed
}
```

Συναρτήσεις | return

- όταν ο κώδικας βρίσκει εντολή return, σταματάει η εκτέλεση της συνάρτησης και επιστρέφει κάποια τιμή
- αν η συνάρτηση καλείται σε κάποια εντολή, η εκτέλεση του κώδικα θα συνεχιστεί μετά την κλήση της συνάρτησης
- μπορεί η συνάρτηση να παράγει κάποια τιμή που επιστρέφεται στον κώδικα που την καλεί με την εντολή return

Συναρτήσεις | εμβέλεια

- οι μεταβλητές που δηλώνονται μέσα στη συνάρτηση έχουν τοπική (local) εμβέλεια, δηλαδή, μπορούν να χρησιμοποιηθούν μόνο μέσα στο block κώδικα της συνάρτησης.

```
// code here can NOT use carName
function myFunction() {
        let carName = "Volvo";
        // code here CAN use carName
}
// code here can NOT use carName
```

Συναρτήσεις | λόγος ύπαρξης

- με τη χρήση συναρτήσεων ο κώδικας μπορεί να επαναχρησιμοποιηθεί
- γράφεται μία φορά ο κώδικας, χρησιμοποιείται σε περισσότερα σημεία
- μπορεί ο ίδιος κώδικας με διαφορετικές παραμέτρους να παράγει διαφορετικά αποτελέσματα
- οι συναρτήσεις μπορούν να χρησιμοποιηθούν ως μεταβλητές σε εκφράσεις, εκχωρήσεις τιμών και υπολογισμούς

Παράδειγμα: αλλαγή παραγράφου με id

```
<script>
        function myFunction1() {
                document.getElementById("demo1").innerHTML = "Carpe Diem!";
        }
</script>

<p>Όταν πατήσεις το "Click me", θα κληθεί μία function.</p>
<p>Η function θα εμφανίσει ένα μήνυμα.</p>
<button onclick="myFunction1()">Click me</button>
<p id="demo1"></p>
```

[04-functions1.html]

Παράδειγμα: user input (popup window)

```
<p>Πατήστε το κουμπί ανάλογα με την ώρα της ημέρας</p>
<!-- κλήση συνάρτησης με διαφορετικές παραμέτρους με το πάτημα κουμπιών-->
<button onclick="myFunction('καλημέρα')">Είναι πρωί</button>
<button onclick="myFunction('καλησπέρα')">Είναι απόγευμα</button>
<p id="demo"></p>

<script>
function myFunction(message) {
  // user input, αν δεν δώσει πληροφορία, παίρνει ως default τη 2η παράμετρο (Άγνωστος Χ)
  const name = prompt("Ποιο είναι το όνομά σου; ", "Άγνωστος Χ"); // popup window
  document.getElementById("demo").innerHTML = "Καλώς ήλθες <b>" + name + "</b>, " + message + ".";
}
</script>
```

[*04-functions2.html*]

Παράδειγμα: user input (μέσα στη web σελίδα)

```
<label for="userInput">Ποιο είναι το όνομά σου: </label>
<input type="text" id="userInput" name="userName">
<button onclick="getUserInput()">υποβολή</button>
<script>
    function myFunction(message) {
        // πάρε το όνομα του χρήστη από το userInput field
        var name = document.getElementById("userInput").value;
        document.getElementById("demo").innerHTML = "Καλώς ήλθες <b>" + name + "</b>, " +
        message + ".";
    }
</script>
<h2>Πατήστε το κουμπί ανάλογα με την ώρα της ημέρας</h2>
<button onclick="myFunction('καλημέρα')">Είναι πρωί</button>
<button onclick="myFunction('καλησπέρα')">Είναι απόγευμα</button>
<p id="demo"></p>
```

[*04-functions3.html*]