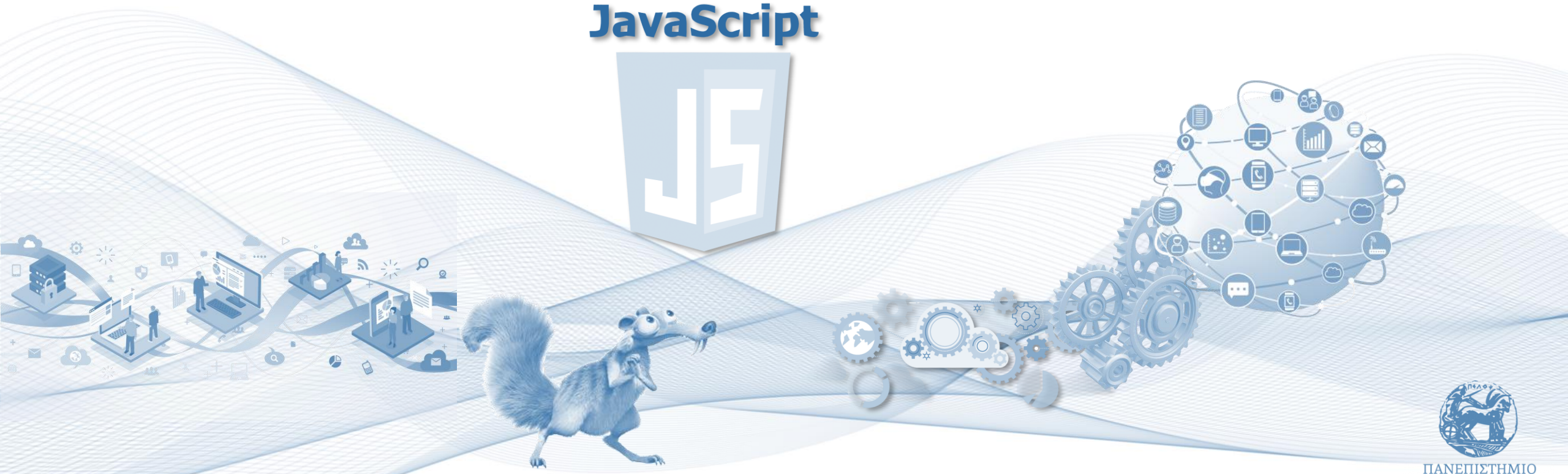
Τμήμα Ηλεκτρολόγων Μηχανικών
& Μηχανικών Υπολογιστών
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΛΟΠΟΝΝΗΣΟΥ
ΗΜΜΥ
Πάτρα

ΤΕΧΝΙΚΕΣ ΠΡΟΓΡΑΜΜΑΤΙΜΟΥ ΥΠΟΛΟΓΙΣΤΩΝ

Διάλεξη IV

JavaScript από πιο κοντά: Functions – Branch & Loop

Γιάννης Τζήμας, Καθηγητής



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΛΟΠΟΝΝΗΣΟΥ
UNIVERSITY of the PELOPONNESE

Τι θα δούμε σήμερα…

- Σε ένα δεύτερο επίπεδο θα δούμε τα ακόλουθα:
  - Συναρτήσεις – Functions,
  - Δυνατότητες Επιλογής ανάμεσα σε διάφορες εναλλακτικές και
  - Αξιοποίηση της Επανάληψης

Ας ρίξουμε τα ζάρια…

- Ας προσπαθήσουμε να φτιάξουμε ένα πρόγραμμα που εξομοιώνει το ρίξιμο ζαριών.
- Τι θα χρειαστούμε;

```
1  var firstDie = 5;
2  var secondDie = 3;
3
4  console.log(firstDie);
5  console.log(secondDie);
6  console.log(firstDie + secondDie);
```

```
C:\Users\Administrator\Documents\examples>node functions.js
5
3
8
```

Το μόνο πρόβλημα είναι ότι παίρνουμε **πάντα ίδιο αποτέλεσμα…**

Θα πρέπει λοιπόν να εισάγουμε κάποια **τυχαιότητα**!

Ας απομονώσουμε κάποιο τμήμα του κώδικα... - Σχόλια

```
1  var firstDie = 5;
2  var secondDie = 3;
3
4  //console.log(firstDie);
5  //console.log(secondDie);
6  //console.log(firstDie + secondDie);
```

Παρατηρείστε ότι δεν υπάρχει έξοδος

```
C:\Users\Administrator\Documents\examples>node functions.js

C:\Users\Administrator\Documents\examples>
```

Σχόλια: μου επιτρέπουν να τεκμηριώσω τον κώδικά μου. Ο σχολιασμός είναι για τους προγραμματιστές. Ο Η/Υ τα αγνοεί.

```
1  var x = 5; //Αυτό είναι ένα σχόλιο
```

Και ας βάλουμε λίγη τυχαιότητα στο πρόγραμμά μας...

```
1  var firstDie = 5;
2  var secondDie = 3;
3
4  console.log(Math.random())
5
6  //console.log(firstDie);
```

Σε αυτή τη γραμμή καλούμε δύο συναρτήσεις, τη **random** και τη **log**.
Η τιμή που επιστρέφει η random περνιέται σαν παράμετρος στη log.

```
C:\Users\Administrator\Do
0.16515585618381712

C:\Users\Administrator\Do
0.49948770077409965

C:\Users\Administrator\Do
0.40579323197043626

C:\Users\Administrator\Do
0.9190875754130274
```

Ορολογία

`Math.random()`

Λέμε ότι:

- Καλούμε τη συνάρτηση

ή στα αγγλικά

- “Invoking” the function
- “Calling” the function
- “Running” the function

Όλα τα παραπάνω αναφέρονται στο ίδιο ακριβώς.

Εννοούμε ότι θα εκτελεστεί ο κώδικας που υπάρχει εντός της συνάρτησης.

Κάναμε ένα βήμα,
αλλά ας έρθουμε πιο κοντά στους αριθμούς των ζαριών…

```
1  var firstDie = 5;
2  var secondDie = 3;
3
4  console.log(6*Math.random())
5
6  //console.log(firstDie);
```

```
C:\Users\Administrator\D
4.231911607648955

C:\Users\Administrator\D
0.38959873864978345

C:\Users\Administrator\D
1.1866536034271316
```

```
1  var firstDie = 5;
2  var secondDie = 3;
3
4  var value = 6*Math.random();
5  var roll = Math.ceil(value);
6
7  console.log(roll);
```

```
C:\Users\Admini
2

C:\Users\Admini
5

C:\Users\Admini
3

C:\Users\Admini
5
```

- Μερικές δοκιμές για το σπίτι:
  - Τι θα συνέβαινε αν χρησιμοποιούσαμε τη **`Math.round()`** ή τη **`Math.floor()`**;

Και ολοκληρώνοντας…

```
1  var firstDie = Math.ceil(6*Math.random());
2  var secondDie = Math.ceil(6*Math.random());
3
4  console.log(firstDie);
5  console.log(secondDie);
6  console.log(firstDie + secondDie);
```

```
C:\Users\Admini
2
6
8

C:\Users\Admini
6
6
12

C:\Users\Admini
1
2
3
```

Προσοχή!
Μπορούμε να **περάσουμε** 0 ή περισσότερες παραμέτρους σε μία συνάρτηση.
Μία συνάρτηση μπορεί να **επιστρέψει** μία τιμή, ή να μην το κάνει.

- Μερικές δοκιμές για το σπίτι:
  - Τι θα συνέβαινε αν βάζατε τις παρενθέσεις σε άλλα σημεία; π.χ.
    `Math.ceil(6) * (Math.random());`
    `Math.ceil(6 * Math.random();`

Επισκεφτείτε τη διεύθυνση:
https://developer.mozilla.org/en-US/docs/Web/JavaScript

Συναρτήσεις – Ας κάνουμε δικές μας συναρτήσεις…

- Εκτός από το να χρησιμοποιήσουμε συναρτήσεις που μας δίνει οι JavaScript, μπορούμε να γράψουμε και **δικές μας** συναρτήσεις.

- Γιατί να το κάνουμε αυτό;
  - Για να γίνει ο κώδικας πιο **ευανάγνωστος**.
  - Για να **μην υπάρχει επανάληψη** του ίδιου κώδικα.

```
1  var firstDie = Math.ceil(6*Math.random());
2  var secondDie = Math.ceil(6*Math.random());
3
4  console.log(firstDie);
5  console.log(secondDie);
6  console.log(firstDie + secondDie);
```

Επιστροφή τιμής

```
var getDieRoll = function() {
    console.log("Ρίχνω ένα ζάρι");
    return Math.ceil(6 * Math.random());
};

var firstDie = getDieRoll();
var secondDie = getDieRoll();

console.log(firstDie);
console.log(secondDie);
console.log(firstDie + secondDie);
```

```
C:\Users\Administr
Ρίχνω ένα ζάρι
Ρίχνω ένα ζάρι
2
6
8

C:\Users\Administr
Ρίχνω ένα ζάρι
Ρίχνω ένα ζάρι
2
4
6
```

Γιατί όμως αυτό το αποτέλεσμα;

```
var getDieRoll = function() {
    console.log("Ρίχνω ένα ζάρι");
    return Math.ceil(6 * Math.random());
};

var firstDie = getDieRoll();
var secondDie = getDieRoll();

console.log(firstDie);
console.log(secondDie);
console.log(firstDie + secondDie);
```

```
C:\Users\Administr
Ρίχνω ένα ζάρι
Ρίχνω ένα ζάρι
2
6
8

C:\Users\Administr
Ρίχνω ένα ζάρι
Ρίχνω ένα ζάρι
2
4
6
```

```
var getDieRoll = function() {
    console.log("Ρίχνω ένα ζάρι");
    return Math.ceil(6 * Math.random());
};

var firstDie = getDieRoll();
console.log(firstDie);

var secondDie = getDieRoll();
console.log(secondDie);
console.log(firstDie + secondDie);
```

```
C:\Users\Administr
Ρίχνω ένα ζάρι
2
Ρίχνω ένα ζάρι
5
7

C:\Users\Administr
Ρίχνω ένα ζάρι
6
Ρίχνω ένα ζάρι
5
11
```

Παράμετροι – Parameters

```
var getDieRoll = function(var1, var2, var3) {
    console.log("Ρίχνω ένα ζάρι");
    return Math.ceil(6 * Math.random());
};

var firstDie = getDieRoll(1, 2, 3);
console.log(firstDie);
```

Παράμετροι:
Επιτυγχάνουν την παραμετροποίηση της συνάρτησης.

Πέρασμα Παραμέτρων

Και αν το ζάρι έχει 10 πλευρές;

```
C:\Users\A
10
9
10
5
14
```

```
var getDieRoll = function(dieSize) {
    console.log(dieSize);
    return Math.ceil(dieSize * Math.random());
};

var firstDie = getDieRoll(10);
console.log(firstDie);

var secondDie = getDieRoll(10);
console.log(secondDie);
console.log(firstDie + secondDie);
```

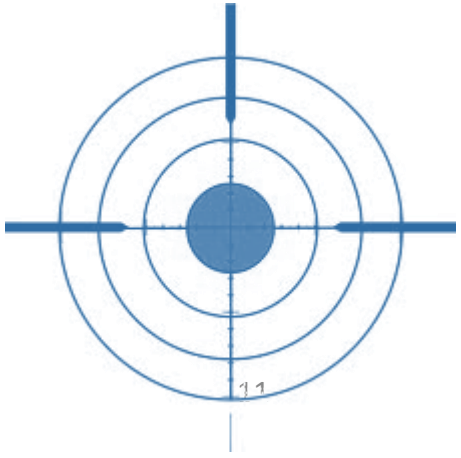
Εμβέλεια Μεταβλητών - Variable Scope

- Όταν λέμε εμβέλεια μεταβλητών εννοούμε το σε ποια σημεία του προγράμματος είναι διαθέσιμη η μεταβλητή.

```
var getDieRoll = function(dieSize) {
    console.log(dieSize);
    return Math.ceil(dieSize * Math.random());
};

var firstDie = getDieRoll(10);
console.log(firstDie);

var secondDie = getDieRoll(10);
console.log(secondDie);
console.log(firstDie + secondDie);
```

Εμβέλεια - Scope

`var x;` // σε ποια σημεία μπορώ να χρησιμοποιήσω τη x;



Εμβέλεια Μεταβλητών – Ένα παράδειγμα...

```
var getDieRoll = function(dieSize) {
    var result = Math.ceil(dieSize * Math.random());
    return result;
};

var firstDie = getDieRoll(10);
var secondDie = getDieRoll(10);

console.log(result);

console.log(firstDie);
console.log(secondDie);
console.log(firstDie + secondDie);
```

```
C:\Users\Administrator\Documents\examples>node functions.js
C:\Users\Administrator\Documents\examples\functions.js:9
console.log(result);
                    ^

ReferenceError: result is not defined
    at Object.<anonymous> (C:\Users\Administrator\Documents\ex
    at Module._compile (module.js:570:32)
```

Το ίδιο ακριβώς λάθος θα επιστραφεί αν χρησιμοποιήσουμε τη dieSize

- Το λάθος που μας επιστρέφει είναι ότι η result δεν έχει οριστεί.
- Αυτό συμβαίνει γιατί η result είναι αυτό που ονομάζουμε **τοπική** μεταβλητή (local).
- Τοπική γιατί έχει οριστεί εντός μιας συνάρτησης, οπότε η **εμβέλειά** της είναι εντός του σώματος της συνάρτησης.

Εμβέλεια Μεταβλητών – Ένα παράδειγμα…

```
var getDieRoll = function(dieSize) {
    var result = Math.ceil(dieSize * Math.random());
    return result;
};

var showResult = function() {
    console.log(firstDie);
    console.log(secondDie);
    console.log(firstDie + secondDie);
};

var firstDie = getDieRoll(10);
var secondDie = getDieRoll(10);
showResult();
```

Quiz:

Υπάρχουν 4 καθολικές μεταβλητές στο πρόγραμμα. Μπορείτε να τις βρείτε;

```
C:\Users\Administrator\
10
1
11

C:\Users\Administrator\
4
10
14
```

- Σε αντίθεση οι μεταβλητές **`firstDie`** και **`secondDie`** μπορούν να χρησιμοποιηθούν εντός της **`showResult`**;
- Η απάντηση είναι **ναι** γιατί είναι καθολικές μεταβλητές (**global**).

- Έχουν **εμβέλεια** δηλαδή σε οποιοδήποτε σημείο του προγράμματος.

Τι είδαμε μέχρι τώρα;

- Μάθαμε πώς να ορίσουμε μία συνάρτηση και πως επιστρέφουμε μία τιμή από συνάρτηση.

- Μάθαμε επίσης πως μπορούμε να περάσουμε παραμέτρους σε μία συνάρτηση και κυρίως μάθαμε πώς να οργανώνουμε τον κώδικά μας ώστε να είναι πιο ευανάγνωστος.

- Τέλος, μάθαμε τι είναι η εμβέλεια των μεταβλητών. Μάθαμε ότι υπάρχουν τοπικές και καθολικές μεταβλητές.

Ελέγχοντας τη Ροή του Προγράμματος: Διακλάδωση (Branching) – if

- Μένοντας στο παράδειγμα με τα ζάρια…

```
var getDieRoll = function(dieSize) {
    var result = Math.ceil(dieSize * Math.random());
    return result;
};
var roll = getDieRoll(6);
console.log("You rolled a " + roll);
```

```
C:\Users\Administra
You rolled a 5

C:\Users\Administra
You rolled a 3
```

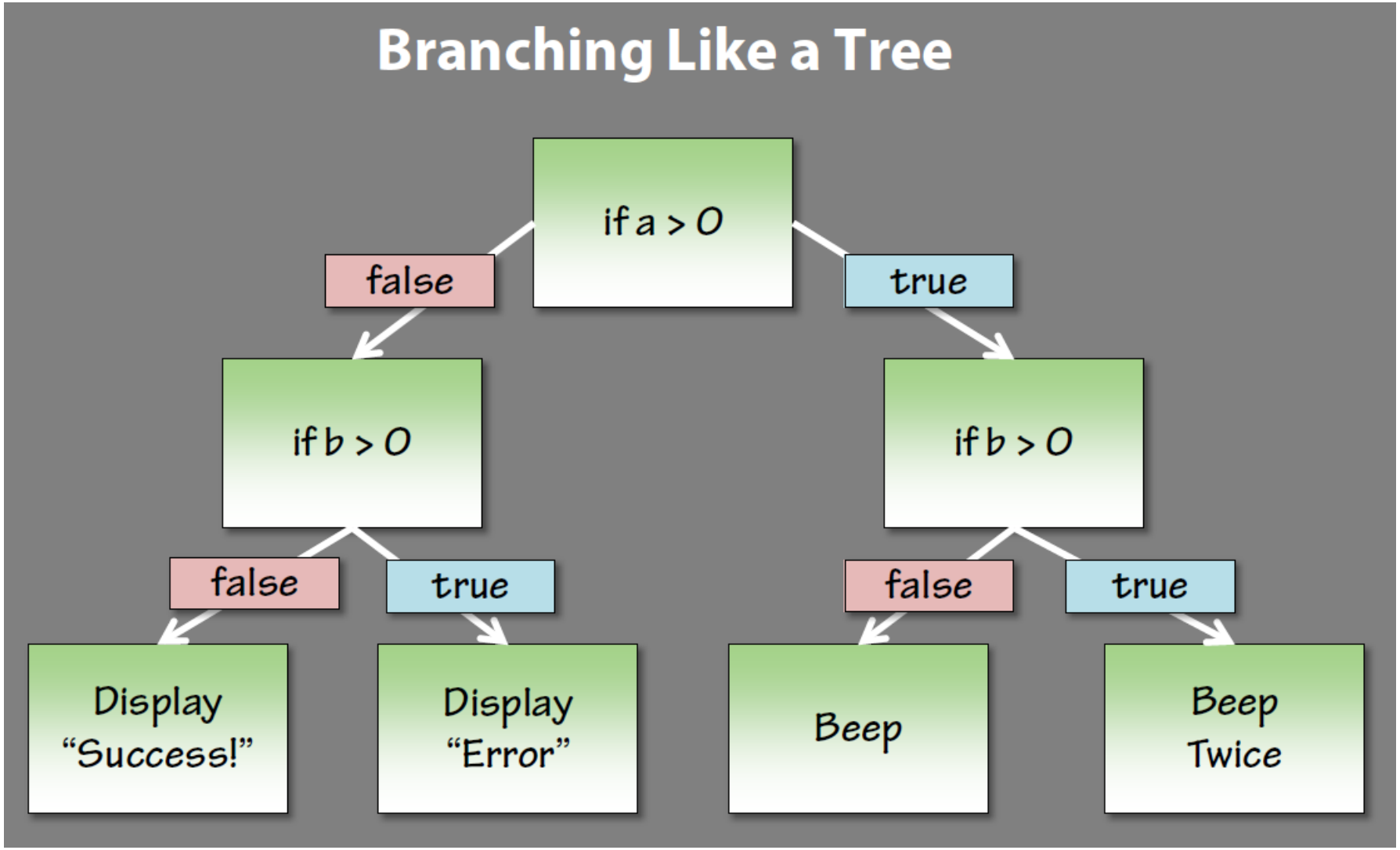
- Και αν θέλαμε να αξιολογήσουμε το αποτέλεσμα; Δηλαδή καλή ζαριά να είναι το 5 ή το 6.

```
var getDieRoll = function(dieSize) {
    var result = Math.ceil(dieSize * Math.random());
    return result;
};
var roll = getDieRoll(6);

if(roll >= 5){
    console.log("Great roll!");
}
console.log("You rolled a " + roll);
```

```
C:\Users\Adminis
You rolled a 2

C:\Users\Adminis
Great roll!
You rolled a 5
```

Η Λογική Διακλάδωσης (Branching Logic) σε ένα Πρόγραμμα

- Η λογική αυτή υπάρχει σε απεριόριστο αριθμό προγραμμάτων.
- Μπορείτε να σκεφτείτε μερικές περιπτώσεις;



Τελεστές – Operators

- Τελεστές Σύγκρισης (Comparison Operators)
  - `>` Μεγαλύτερο
  - `>=` Μεγαλύτερο ή ίσο
  - `<` Μικρότερο
  - `<=` Μικρότερο ή ίσο
  - `==` Ίσο
  - `!=` Διαφορετικό

```
if(roll != 5){
    console.log("Great roll!");
}
```

- Λογικοί Τελεστές (Logical Operators)
  - `&&` AND
  - `||` OR
  - `!` NOT

```
if(roll >= 2 && roll <= 2){
    console.log("Great roll!");
}
if(roll == 1 || roll == 6){
    console.log("Great roll!");
}
if(!(roll == 1 || roll == 6)){
    console.log("Great roll!");
}
```

- Μία δοκιμή με ότι μάθαμε:
  Μία πολύ καλή ζαριά είναι:
  - Όχι το 1
  - Το 1, 2 ή το 3
  - Το 3, 4 ή το 5
  - Το 1, 2, 5 ή το 6

Παραλλαγές του if (Nesting & Else)

- Nesting

```
if(roll >= 3 && roll <= 5){
    console.log("Great roll!");
}
```

Είναι το ίδιο

```
if(roll >= 3) {
    if(roll <= 5){
        console.log("Great roll!");
    }
}
```

Στοίχιση: Ο Κώδικας είναι πιο ευανάγνωστος.

- **Else**

```
if(roll >= 3 && roll <= 5){
    console.log("Great roll!");
}
else {
    console.log("That was an ok roll");
}
```

```
C:\Users\Administrat
That was an ok roll
You rolled a 6

C:\Users\Administrat
Great roll!
You rolled a 4
```

```
if(roll >= 3 && roll <= 5){
    console.log("Great roll!");
}
else if (roll ==1) {
    console.log("Terrible roll!");
}
else {
    console.log("That was an ok roll");
}
```

```
C:\Users\Administrato
Terrible roll!
You rolled a 1

C:\Users\Administrato
That was an ok roll
You rolled a 2
```

Ελέγχοντας τη Ροή του Προγράμματος: Κάνοντας Βρόχο (Looping) – while

```
var getDieRoll = function(dieSize) {
    var result = Math.ceil(dieSize * Math.random());
    return result;
};
var roll = getDieRoll(6);

while (roll<4){
    console.log(roll);
    roll = getDieRoll(6);
}

if(roll >= 3 && roll <= 5){
    console.log("Great roll!");
}
else if (roll ==1) {
    console.log("Terrible roll!");
}
else {
    console.log("That was an ok roll");
}
console.log("You rolled a " + roll);
```

- Με το βρόχο **while** εκτελούνται οι εντολές 0 ή περισσότερες φορές.

```
C:\Users\Administ
2
Great roll!
You rolled a 4

C:\Users\Administ
2
2
2
Great roll!
You rolled a 5
```

Ελέγχοντας τη Ροή του Προγράμματος: Κάνοντας Βρόχο (Looping) – do-while

```
var getDieRoll = function(dieSize) {
    var result = Math.ceil(dieSize * Math.random());
    return result;
};
var roll = getDieRoll(6);
while (roll<4){
    console.log(roll);
    roll = getDieRoll(6);
}
do {
    roll = getDieRoll(6);
    console.log(roll);
} while (roll>4)
if(roll >= 3 && roll <= 5){
    console.log("Great roll!");
}
else if (roll ==1) {
    console.log("Terrible roll!");
}
else {
    console.log("That was an ok roll");
}
console.log("You rolled a " + roll);
```

0 ή περισσότερες φορές

1 ή περισσότερες φορές

```
do {

} while (true)
```

Infinite loop: Θα πρέπει να το αποφεύγουμε!

- Με το **do-while** εκτελούνται οι εντολές 1 ή περισσότερες φορές.

- Για το σπίτι:
  - Αλλάξτε το πρόγραμμα, ώστε πάντα η ζαριά να είναι καλή.

```
var getDieRoll = function(dieSize) {
    var result = Math.ceil(dieSize * Math.random());
    return result;
};
var roll = getDieRoll(6);



for(var i = 0; i < 10; i+=1) {
    console.log(roll);
    roll = getDieRoll(6);
}



if(roll >= 3 && roll <= 5){
    console.log("Great roll!");
}
else if (roll ==1) {
    console.log("Terrible roll!");
}
else {
    console.log("That was an ok roll");
}
console.log("You rolled a " + roll);
```

Ελέγχοντας τη Ροή του Προγράμματος: Κάνοντας Βρόχο (Looping) – for

- Στα **for** loops είναι σύνηθες να χρησιμοποιείται μια απλή μεταβλητή ως μετρητής
  - Ξεκίνα με i ίσο με 0
  - Κάνε βρόχο ενώ το i είναι μικρότερο από 10
  - Πρόσθεσε 1 στο i κάθε φορά που γίνεται ο βρόχος.

- Το i += 1 είναι το ίδιο ακριβώς με το i = i + 1

Ελέγχοντας τη Ροή του Προγράμματος: Κάνοντας Βρόχο (Looping) – for

```
var getDieRoll = function(dieSize) {
    var result = Math.ceil(dieSize * Math.random());
    return result;
};
var roll = getDieRoll(6);

for(var i = 0; i < 10; i+=1) {
    console.log(roll);
    roll = getDieRoll(6);
}

if(roll >= 3 && roll <= 5){
    console.log("Great roll!");
}
else if (roll ==1) {
    console.log("Terrible roll!");
}
else {
    console.log("That was an ok roll");
}
console.log("You rolled a " + roll);
```

Αρχικοποίηση: εκτελείται μία φορά πριν ξεκινήσει το loop

Συνθήκη: εκτελείται πριν από κάθε πέρασμα

Τελική έκφραση: εκτελείται μετά από κάθε πέρασμα.

```
C:\Users\Administr
6
3
1
5
3
6
5
1
1
2
Great roll!
You rolled a 4
```

For και While

```
for(var i = 0; i < 10; i+=1) {
    // ...

}

var i = 0;
while(i<10){
    // ...
    i +=1;
}
```

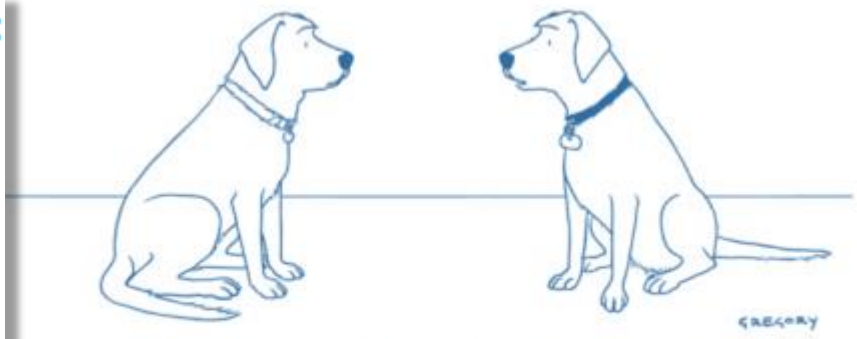
- Είναι διαφορετικές μορφές του ίδιου κώδικα.
- Το **`for`** όμως είναι πιο συμπαγές.

- Για το σπίτι:
  - Αλλάξτε το πρόγραμμα, ώστε να ρίχνεται το ζάρι 10 φορές και να δείχνει ένα "*" αν το νούμερο είναι μεγαλύτερο ή ίσο από 4 και μία "-" αν όχι.

Και μια μικρή παραλλαγή:

```
for(var i = 0; i < roll; i+=1) {
    console.log("*");
}
```

```
C:\Users\Administ
*
*
*
*
Terrible roll!
You rolled a 1
```

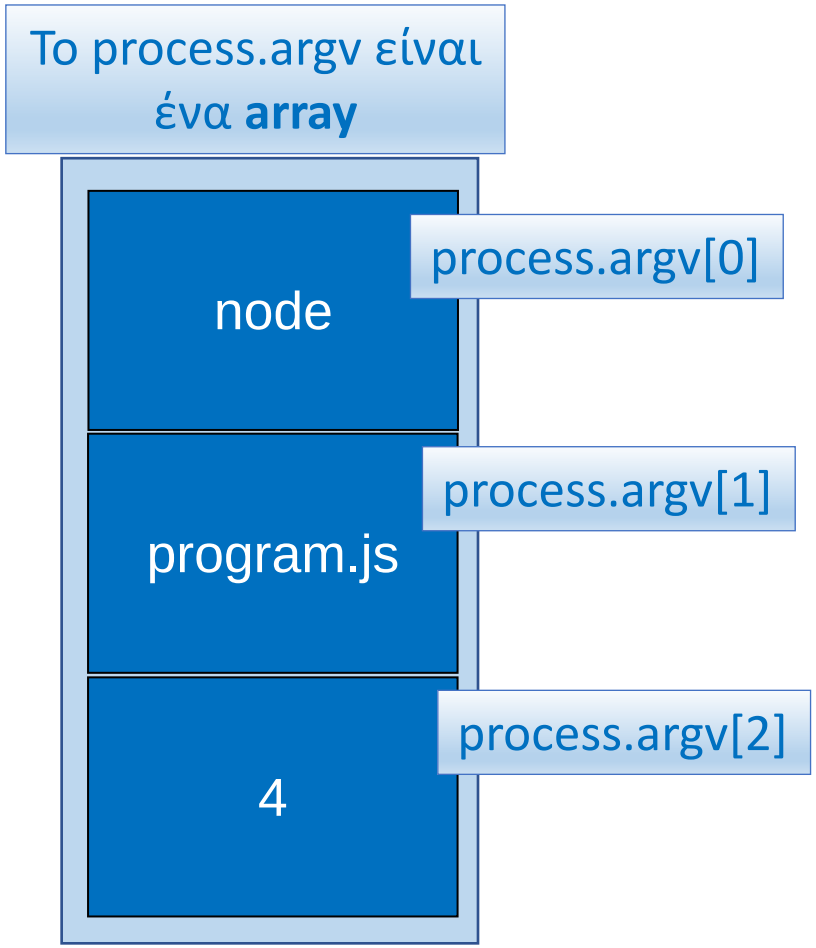


"I had my own blog for a while, but I decided to go back to just pointless, incessant barking."

…και ας κάνουμε μια αναδρομή στα arrays…

- Ένα array είναι μία συλλογή.
- Μπορείτε να βάλετε σχεδόν οτιδήποτε (strings, ή numbers, ή ακόμα και objects) σε ένα array με τη JavaScript.
- Σχεδόν όλες οι γλώσσες προγραμματισμού έχουν arrays, ή κάποια αντίστοιχη έννοια.

Θυμηθείτε το προηγούμενο μάθημα:

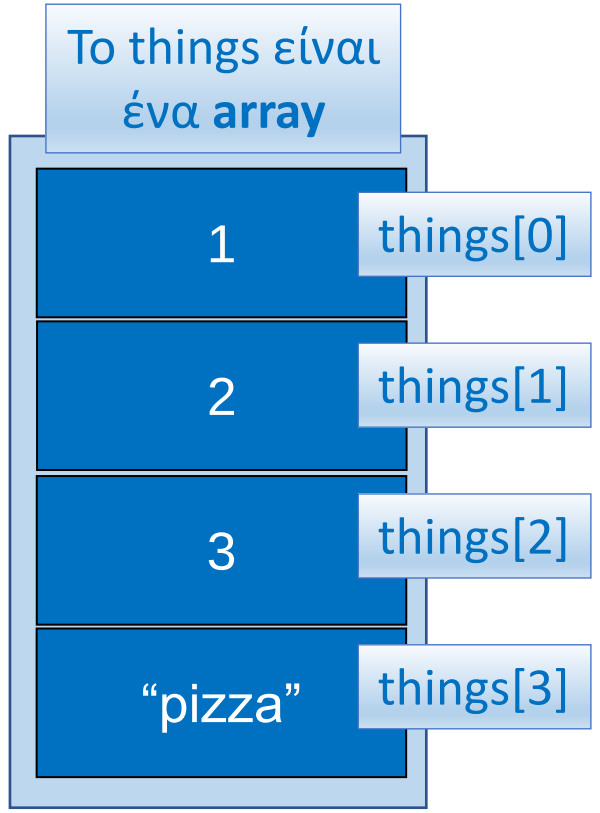


Arrays στη JavaScript

Είναι property (ιδιότητα) του array.

```
var things = [1,2,3,"pizza"];
for(var i = 0; i < things.length; i++) {
    console.log(things[i]);
}
```

```
C:\Users\Adm
1
2
3
pizza
```



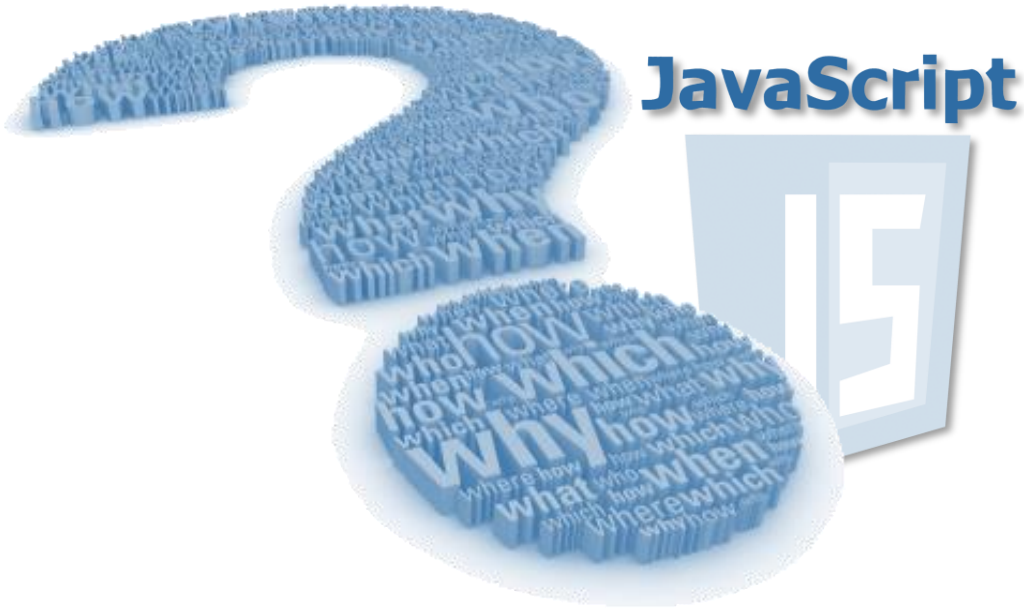
- Το i++ είναι το ίδιο ακριβώς με το i += 1 και το i = i + 1
- Και αν θέλαμε είσοδο από το χρήστη;

```
for(var i = 0; i < process.argv.length; i++) {
    console.log(process.argv[i]);
}
```

```
C:\Users\Administ
john
mary
```

Τι είδαμε μέχρι τώρα;

- Μάθαμε πώς μπορούμε να κάνουμε διακλαδώσεις (branching) για να πάρουμε αποφάσεις και βρόχους (looping).
- Ποιο συγκεκριμένα είδαμε το while και τη διαφορά του με το do-while που έχει σχέση με το ποιος είναι ο ελάχιστος αριθμός φορών που θα εκτελεστεί ο βρόχος.
- Επίσης, μάθαμε το if και το else-if με τα οποία μπορούμε να ελέγχουμε λογικές συνθήκες και με βάση αυτό να παίρνουμε αποφάσεις.
- Είδαμε αναλυτικά τους τελεστές σύγκρισης και τους λογικούς τελεστές στη γλώσσα JavaScript και τέλος
- Χρησιμοποιήσαμε το for για να δημιουργήσουμε βρόχους και να πάρουμε είσοδο από το χρήστη.
- Στο επόμενο μάθημα θα ασχοληθούμε με τα αντικείμενα (objects) και αν θέλετε να ασχοληθείτε στο σπίτι:
  - Κάντε ένα μικρό πρόγραμμα το οποίο να εξομοιώνει μία ρουλέτα.


JavaScript