Εισαγωγή στην Επιστήμη Υπολογιστών

Εργαστηριακές Διαλέξεις - Εισαγωγή σε JavaScript

Javascript: Σύντομες σημειώσεις στην γλώσσα.

Επιμέλεια εκπαιδευτικού υλικού: Αναστάσιος Δρακόπουλος

Ακαδημαϊκό έτος 2024-2025

Πίνακας περιεχομένων

Σύντομες σημειώσεις στην Javascript. 4
Μεταβλητές στην JavaScript: Τοπικές και Καθολικές 4
Τοπικές και Καθολικές Μεταβλητές 4
Τρόποι Δήλωσης Μεταβλητών 4
Τύποι Δεδομένων στην JavaScript 5
Ολοκληρωμένο Παράδειγμα 6
Δομές Επιλογής στην JavaScript 8
1. Η if και else 8
2. Η else if 8
3. Η switch 9
Σύνθετα Παραδείγματα 9
Παράδειγμα 1: Αξιολόγηση Τιμής Προϊόντος 9
Παράδειγμα 2: Διαχείριση Εντολών Χρήστη με switch 10
Σημεία Προσοχής 10
Συνοψίζοντας 10
Δομές Επανάληψης στην JavaScript 12
Τύποι Δομών Επανάληψης 12
1. Η for 12
2. Η while 12
3. Η do...while 13
4. Η for...in 13
5. Η for...of 14
Σύνθετα Παραδείγματα 14
Παράδειγμα 1: Εύρεση Ζυγών Αριθμών 14
Παράδειγμα 2: Υπολογισμός Αθροίσματος με while 14
Παράδειγμα 3: Επαλήθευση Ελάχιστης Εισαγωγής με do...while 15
Παράδειγμα 4: Επανάληψη σε Πίνακα με for...of 15
Παράδειγμα 5: Επανάληψη σε Ιδιότητες Αντικειμένου με for...in 15
Λογικοί Τελεστές και Διακοπή Δομών 15
Συνοψίζοντας 16
1. Χρήση του Object Literal 16
2. Χρήση του Constructor Function 17
3. Χρήση της Object.create() 17
4. Χρήση της class (ES6 Syntax) 18
5. Χρήση του Object Constructor 18
6. Χρήση JSON (JavaScript Object Notation) 19
Συνοπτικός Πίνακας Τρόπων Ορισμού Αντικειμένων 19
Συναρτήσεις στην JavaScript 20
Τρόποι Ορισμού Συναρτήσεων 20
Πέρασμα Παραμέτρων: Με Τιμή και Αναφορά 21
Παραδείγματα Συναρτήσεων 21
Παράδειγμα 1: Συνάρτηση χωρίς παραμέτρους 21
Παράδειγμα 2: Συνάρτηση με παραμέτρους 22
Παράδειγμα 3: Επιστροφή τιμής 22
Παράδειγμα 4: Βέλη Συνάρτηση 22
Παράδειγμα 5: Ανώνυμη Συνάρτηση 22
Παράδειγμα 6: Συνάρτηση με προεπιλεγμένες παραμέτρους 22
Παράδειγμα 7: Αναδρομή 22
Παράδειγμα 8: Συνάρτηση ως τιμή 23
Παράδειγμα 9: Πέρασμα πίνακα ως παράμετρος 23
Παράδειγμα 10: Κατασκευή αντικειμένων με Constructor Function 23

Συνοψίζοντας 24
Αντικειμενοστραφής Προγραμματισμός (Object-Oriented Programming) στην JavaScript 25
Τρόποι Υλοποίησης OOP στην JavaScript 25
10 Αναλυτικά Παραδείγματα Προγραμμάτων 25
1. Δημιουργία Αντικειμένου με Object Literal 25
2. Δημιουργία Αντικειμένων με Constructor Function 25
3. Δημιουργία Κλάσης με ES6 Syntax 26
4. Κληρονομικότητα με extends 26
5. Χρήση Object.create για Κληρονομικότητα 27
6. Προσθήκη Μεθόδων με prototype 27
7. Πολυμορφισμός με Υποκλάσεις 27
8. Static Methods 28
9. Encapsulation (Ιδιότητες με Private Scope) 28
10. Χρήση Getters και Setters 28
Συμπεράσματα 29

Σύντομες σημειώσεις στην Javascript.

Μεταβλητές στην JavaScript: Τοπικές και Καθολικές

Η **JavaScript** είναι μια γλώσσα δυναμικά τυποποιημένη, όπου οι μεταβλητές μπορούν να περιέχουν διαφορετικούς τύπους δεδομένων σε διαφορετικές χρονικές στιγμές. Οι μεταβλητές μπορούν να είναι **τοπικές** ή **καθολικές** ανάλογα με το πού και πώς δηλώνονται.

Τοπικές και Καθολικές Μεταβλητές

1. **Καθολικές Μεταβλητές:**

- Δηλώνονται εκτός οποιασδήποτε συνάρτησης, μπλοκ ή μεθόδου.
- Είναι διαθέσιμες σε όλο το πρόγραμμα.
- Προσβάσιμες από οποιοδήποτε μέρος του κώδικα.

Παράδειγμα:

```
let globalVar = "Είμαι καθολική!"; // Καθολική μεταβλητή

function displayGlobalVar() {
  console.log(globalVar); // Πρόσβαση στην καθολική μεταβλητή
}

displayGlobalVar(); // Εμφανίζει: Είμαι καθολική!
console.log(globalVar); // Εμφανίζει: Είμαι καθολική!
```

[https://ideone.com/VTYpbb]

2. **Τοπικές Μεταβλητές:**

- Δηλώνονται μέσα σε συνάρτηση, μπλοκ ή μέθοδο.
- Είναι διαθέσιμες μόνο μέσα στο πεδίο όπου δηλώνονται.
- Δεν είναι προσβάσιμες εκτός της περιοχής τους.

Παράδειγμα:

```
function displayLocalVar() {
  let localVar = "Είμαι τοπική!";
  console.log(localVar); // Εμφανίζει: Είμαι τοπική!
}

displayLocalVar();
// console.log(localVar); // Σφάλμα: localVar is not defined
```

{ https://ideone.com/YLDXef }

Τρόποι Δήλωσης Μεταβλητών

1. var**:**

- Παλιότερος τρόπος δήλωσης.
- Δεν περιορίζεται από μπλοκ (block scope), αλλά μόνο από συναρτήσεις (function scope).

- Αν δηλωθεί χωρίς τη λέξη var, γίνεται αυτόματα καθολική.

Παράδειγμα:

```
if (true) {
   var x = 10; // Καθολική σε επίπεδο συναρτήσεων
}
console.log(x); // Εμφανίζει: 10
```

2. let:

- Εισήχθη με ES6.
- Υποστηρίζει περιορισμό σε μπλοκ (block scope).
- Προτιμάται για τοπικές μεταβλητές.

Παράδειγμα:

```
if (true) {
   let y = 20; // Τοπική στο μπλοκ
   console.log(y); // Εμφανίζει: 20
}
// console.log(y); // Σφάλμα: y is not defined
```

{ https://ideone.com/C7P9ZZ }

3. const:

- Δηλώνει σταθερές (constant variables).
- Υποστηρίζει περιορισμό σε μπλοκ (block scope).
- Δεν μπορεί να επαναπροσδιοριστεί ή να αλλάξει αναφορά (για αντικείμενα).

Παράδειγμα:

```
const z = 30;
console.log(z); // Εμφανίζει: 30

// z = 40; // Σφάλμα: Assignment to constant variable
```

{ https://ideone.com/QmBGP9 }

Τύποι Δεδομένων στην JavaScript

1. **Αριθμητικοί (Number):**

- Περιλαμβάνει ακέραιους και δεκαδικούς αριθμούς.

```
let num = 42;
let decimal = 3.14;
```

2. **Κείμενο (String):**

- Περιλαμβάνει αλφαριθμητικούς χαρακτήρες.

```
let text = "Hello, world!";
```

3. **Λογικοί (Boolean):**

- Έχει δύο τιμές: true ή false.

```
let isTrue = true;
```

4. **Μηδενικό (Null):**

- Δείχνει σκόπιμη απουσία τιμής.

```
let empty = null;
```

5. **Μη ορισμένο (Undefined):**

- Υποδεικνύει ότι μια μεταβλητή δηλώθηκε αλλά δεν έχει εκχωρηθεί τιμή.

```
let unknown;
console.log(unknown); // Εμφανίζει: undefined
```

{ https://ideone.com/czcsqb }

6. **Αντικείμενα (Object):**

- Χρησιμοποιούνται για την αποθήκευση συλλογών δεδομένων και πιο πολύπλοκων οντοτήτων.

```
let person = {
  name: "John",
  age: 30
};
```

7. **Συμβολικά (Symbol):**

- Δημιουργεί μοναδικά αναγνωριστικά για αντικείμενα.

```
let sym = Symbol("unique");
```

Ολοκληρωμένο Παράδειγμα

```
// Καθολική μεταβλητή
let globalMessage = "Καλώς ήρθατε στο πρόγραμμα!";

// Συνάρτηση που διαχειρίζεται τοπικές μεταβλητές
function processVariables() {
  let localMessage = "Είμαι μόνο για αυτή τη συνάρτηση!";
  const pi = 3.14159;

  console.log(globalMessage); // Πρόσβαση στην καθολική μεταβλητή
  console.log(localMessage); // Πρόσβαση στην τοπική μεταβλητή
```

```
    console.log(`Η τιμή του pi είναι: ${pi}`);
}

// Εκτέλεση της συνάρτησης
processVariables();

// Προσπάθεια πρόσβασης σε τοπική μεταβλητή
// console.log(localMessage); // Σφάλμα: localMessage is not defined

// Παράδειγμα αντικειμένου
let car = {
    brand: "Toyota",
    model: "Corolla",
    year: 2020,
    displayInfo: function () {
        return `${this.brand} ${this.model}, Έτος: ${this.year}`;
    }
};

console.log(car.displayInfo()); // Εμφανίζει: Toyota Corolla, Έτος: 2020

// Παράδειγμα συμβολοσειράς και αριθμητικών πράξεων
let name = "Maria";
let age = 25;
console.log(`${name} είναι ${age} ετών.`); // Εμφανίζει: Maria είναι 25 ετών.
```

{ https://ideone.com/Fcc7Ip }

Συμπεράσματα:

- Οι μεταβλητές μπορούν να δηλωθούν με var, let ή const ανάλογα με τις απαιτήσεις περιοχής και αμεταβλητότητας.
- Η κατανόηση του scope (τοπικό ή καθολικό) είναι κρίσιμη για την αποφυγή σφαλμάτων.
- Οι τύποι δεδομένων προσφέρουν μεγάλη ευελιξία στη διαχείριση δεδομένων στην εφαρμογή σας.

Δομές Επιλογής στην JavaScript

Η **JavaScript** παρέχει δομές επιλογής για την εκτέλεση διαφορετικών τμημάτων κώδικα ανάλογα με τις συνθήκες. Αυτές περιλαμβάνουν:

1. if **και** else
2. else if
3. switch

Οι δομές επιλογής χρησιμοποιούνται για τη λήψη αποφάσεων με βάση λογικές συνθήκες.

1. Η if και else

Η if επιτρέπει την εκτέλεση ενός μπλοκ κώδικα αν μια συνθήκη είναι αληθής. Η else εκτελείται αν η συνθήκη είναι ψευδής.

Σύνταξη:

```
if (condition) {
   // Εκτέλεση αν η συνθήκη είναι true
} else {
   // Εκτέλεση αν η συνθήκη είναι false
}
```

Παράδειγμα:

```
let age = 18;

if (age >= 18) {
   console.log("Μπορείτε να ψηφίσετε."); // Εκτελείται αν το age είναι 18 ή περισσότερο
} else {
   console.log("Δεν μπορείτε να ψηφίσετε."); // Εκτελείται αν το age είναι κάτω από 18
}
```

{ https://ideone.com/T2dwov }

2. Η else if

Η else if επεκτείνει την if επιτρέποντας πολλαπλές συνθήκες.

Σύνταξη:

```
if (condition1) {
   // Εκτέλεση αν condition1 είναι true
} else if (condition2) {
   // Εκτέλεση αν condition2 είναι true
} else {
   // Εκτέλεση αν καμία από τις συνθήκες δεν είναι true
}
```

Παράδειγμα:

```
let grade = 85;
```

```
if (grade >= 90) {
   console.log("Άριστα");
} else if (grade >= 75) {
   console.log("Πολύ καλά");
} else if (grade >= 50) {
   console.log("Καλώς");
} else {
   console.log("Ανεπιτυχής");
}
```

{ https://ideone.com/RfDdpP }

3. Η switch

Η switch χρησιμοποιείται για την επιλογή ενός μπλοκ κώδικα από πολλές περιπτώσεις (cases). Είναι πιο αποδοτική όταν υπάρχουν πολλές τιμές προς έλεγχο.

Σύνταξη:

```
switch (expression) {
   case value1:
      // Εκτέλεση αν το expression ταιριάζει με value1
      break;
   case value2:
      // Εκτέλεση αν το expression ταιριάζει με value2
      break;
   default:
      // Εκτέλεση αν το expression δεν ταιριάζει με καμία τιμή
}
```

Παράδειγμα:

```
let day = "Monday";

switch (day) {
   case "Monday":
      console.log("Καλή αρχή εβδομάδας!");
      break;
   case "Friday":
      console.log("Είναι Παρασκευή!");
      break;
   default:
      console.log("Μια συνηθισμένη ημέρα.");
}
```

{ https://ideone.com/aff14O }

Σύνθετα Παραδείγματα

Παράδειγμα 1: Αξιολόγηση Τιμής Προϊόντος

```
let price = 100;
```

```
if (price < 50) {
   console.log("Φτηνό προϊόν");
} else if (price >= 50 && price <= 150) {
   console.log("Μεσαίας τιμής προϊόν");
} else {
   console.log("Ακριβό προϊόν");
}
```

{ https://ideone.com/mqBRl6 }

Παράδειγμα 2: Διαχείριση Εντολών Χρήστη με switch

```
let command = "exit";

switch (command) {
   case "start":
      console.log("Η εφαρμογή ξεκίνησε.");
      break;
   case "stop":
      console.log("Η εφαρμογή σταμάτησε.");
      break;
   case "exit":
      console.log("Έξοδος από την εφαρμογή.");
      break;
   default:
      console.log("Άγνωστη εντολή.");
}
```

{ https://ideone.com/H7838G }

Σημεία Προσοχής

1. **Οι συνθήκες είναι λογικές εκφράσεις:**

- Πρέπει να επιστρέφουν true ή false.

2. break **στη** switch**:**

- Χρησιμοποιείται για να αποφευχθεί η εκτέλεση των επόμενων περιπτώσεων.

3. **Λογικοί Τελεστές:**

- && (AND): Ελέγχει αν όλες οι συνθήκες είναι αληθείς.
- || (OR): Ελέγχει αν τουλάχιστον μία συνθήκη είναι αληθής.

Παράδειγμα:

```
let userRole = "admin";

if (userRole === "admin" || userRole === "superuser") {
   console.log("Έχετε πλήρη πρόσβαση.");
} else {
   console.log("Περιορισμένη πρόσβαση.");
}
```

{ https://ideone.com/tpQqUM }

Συνοψίζοντας

- Η if-else είναι κατάλληλη για απλές ή ιεραρχικές συνθήκες.
- Η else if χρησιμοποιείται για πολλαπλές εναλλακτικές.
- Η switch είναι αποδοτική για πολλαπλές τιμές της ίδιας μεταβλητής.
- Τα σχόλια στον κώδικα βοηθούν στη βελτίωση της κατανόησης και της συντήρησης.

Δομές Επανάληψης στην JavaScript

Οι **δομές επανάληψης** (loops) στην JavaScript χρησιμοποιούνται για την εκτέλεση ενός μπλοκ κώδικα πολλές φορές, είτε για συγκεκριμένο αριθμό επαναλήψεων είτε μέχρι να ικανοποιηθεί μια συνθήκη.

Τύποι Δομών Επανάληψης

1. for
2. while
3. do...while
4. for...in
5. for...of

1. Η for

Η πιο κοινή δομή επανάληψης, χρησιμοποιείται όταν ο αριθμός των επαναλήψεων είναι γνωστός εκ των προτέρων.

Σύνταξη:

```
for (initialization; condition; increment/decrement) {
   // Κώδικας προς εκτέλεση
}
```

Παράδειγμα:

```
// Εμφάνιση αριθμών από 1 έως 5
for (let i = 1; i <= 5; i++) {
   console.log(`Αριθμός: ${i}`);
}
```

{ https://ideone.com/Te7k1d }

Εξήγηση:

- initialization: Δηλώνει και αρχικοποιεί μια μεταβλητή (π.χ., let i = 1).
- condition: Ελέγχεται πριν από κάθε επανάληψη (π.χ., i <= 5).
- increment/decrement: Αυξάνει ή μειώνει τη μεταβλητή μετά από κάθε επανάληψη (π.χ., i++).

2. Η while

Η while επαναλαμβάνει ένα μπλοκ κώδικα όσο μια συνθήκη είναι αληθής.

Σύνταξη:

```
while (condition) {
   // Κώδικας προς εκτέλεση
}
```

Παράδειγμα:

```
// Εμφάνιση αριθμών από 1 έως 5
let i = 1;
while (i <= 5) {
   console.log(`Αριθμός: ${i}`);
   i++; // Αύξηση του i
}
```

{ https://ideone.com/NHypEk }

Εξήγηση:

- Η συνθήκη ελέγχεται πριν από κάθε επανάληψη. Αν είναι false, ο βρόχος τερματίζει.

3. Η do...while

Η do...while εκτελεί τον κώδικα τουλάχιστον μία φορά, επειδή η συνθήκη ελέγχεται στο τέλος κάθε επανάληψης.

Σύνταξη:

```
do {
   // Κώδικας προς εκτέλεση
} while (condition);
```

Παράδειγμα:

```
// Εμφάνιση αριθμών από 1 έως 5
let i = 1;
do {
   console.log(`Αριθμός: ${i}`);
   i++;
} while (i <= 5);
```

{ https://ideone.com/CF692o }

Εξήγηση:

- Η do...while εκτελείται τουλάχιστον μία φορά, ακόμη και αν η συνθήκη είναι false.

4. Η for...in

Η for...in χρησιμοποιείται για την επανάληψη πάνω στα κλειδιά ενός αντικειμένου.

Σύνταξη:

```
for (key in object) {
   // Κώδικας προς εκτέλεση
}
```

Παράδειγμα:

```
let person = { name: "John", age: 30, city: "Athens" };
```

```
for (let key in person) {
   console.log(`${key}: ${person[key]}`);
}
```

{ https://ideone.com/VP0VH2 }

Εξήγηση:

- Η key αναπαριστά κάθε κλειδί του αντικειμένου.
- Η πρόσβαση στην τιμή γίνεται με object[key].

5. Η for...of

Η for...of χρησιμοποιείται για την επανάληψη πάνω στα στοιχεία ενός αντικειμένου που είναι επαναλήψιμο (π.χ., πίνακες, συμβολοσειρές).

Σύνταξη:

```
for (value of iterable) {
   // Κώδικας προς εκτέλεση
}
```

Παράδειγμα:

```
let numbers = [10, 20, 30];

for (let num of numbers) {
   console.log(`Αριθμός: ${num}`);
}
```

{ https://ideone.com/TztZfK }

Εξήγηση:

- Η value αναπαριστά κάθε στοιχείο της επαναλήψιμης δομής.

Σύνθετα Παραδείγματα

Παράδειγμα 1: Εύρεση Ζυγών Αριθμών

```
for (let i = 1; i <= 10; i++) {
   if (i % 2 === 0) {
      console.log(`Ζυγός αριθμός: ${i}`);
   }
}
```

{ https://ideone.com/B4lQHu }

Παράδειγμα 2: Υπολογισμός Αθροίσματος με while

```
let sum = 0;
let i = 1;

while (i <= 5) {
```

```
    sum += i; // Πρόσθεση του i στο sum
    i++;
}

console.log(`Το άθροισμα είναι: ${sum}`);
```

{ https://ideone.com/FDrQIb }

Παράδειγμα 3: Επαλήθευση Ελάχιστης Εισαγωγής με do...while

```
let input;

do {
    input = prompt("Εισάγετε έναν αριθμό μεγαλύτερο από 10:");
} while (input <= 10);

console.log(`Σωστή εισαγωγή: ${input}`);
```

{ https://ideone.com/H4mM6K }

Παράδειγμα 4: Επανάληψη σε Πίνακα με for...of

```
let fruits = ["Μήλο", "Μπανάνα", "Πορτοκάλι"];

for (let fruit of fruits) {
    console.log(`Φρούτο: ${fruit}`);
}
```

{ https://ideone.com/sGnRBz }

Παράδειγμα 5: Επανάληψη σε Ιδιότητες Αντικειμένου με for...in

```
let car = { brand: "Toyota", model: "Corolla", year: 2022 };

for (let key in car) {
    console.log(`${key}: ${car[key]}`);
}
```

{ https://ideone.com/JXL9uo }

Λογικοί Τελεστές και Διακοπή Δομών

1. break:

- Τερματίζει τη δομή επανάληψης.

```
for (let i = 1; i <= 10; i++) {
    if (i === 5) break; // Σταματά όταν το i είναι 5
    console.log(i);
}
```

{ https://ideone.com/2Bcq4N }

2. continue:

- Παραλείπει την τρέχουσα επανάληψη και συνεχίζει με την επόμενη.

```
for (let i = 1; i <= 10; i++) {
   if (i % 2 === 0) continue; // Παραλείπει ζυγούς αριθμούς
   console.log(i);
}
```

{ https://ideone.com/G6weeF }

Συνοψίζοντας

- for: Χρησιμοποιείται για επαναλήψεις συγκεκριμένου αριθμού.
- while: Χρησιμοποιείται όταν δεν γνωρίζουμε τον αριθμό των επαναλήψεων εκ των προτέρων.
- do...while: Εγγυάται τουλάχιστον μία εκτέλεση του μπλοκ.
- for...in: Χρησιμοποιείται για ιδιότητες αντικειμένων.
- for...of: Χρησιμοποιείται για τιμές επαναλήψιμων αντικειμένων.

Η κατανόηση αυτών των δομών επανάληψης είναι κρίσιμη για τη διαχείριση δεδομένων και την αυτοματοποίηση διαδικασιών στην JavaScript.

1. Χρήση του Object Literal

Ο πιο απλός και συχνός τρόπος για τον ορισμό ενός αντικειμένου.

Σύνταξη:

```
let objectName = {
   property1: value1,
   property2: value2,
   methodName: function() {
      // Λειτουργία
   }
};
```

Παράδειγμα:

```
let person = {
   firstName: "John", // Ιδιότητα
   lastName: "Doe",
   age: 30,
   greet: function() { // Μέθοδος
      console.log(`Hello, my name is ${this.firstName} ${this.lastName}.`);
   }
};

// Πρόσβαση στις ιδιότητες
console.log(person.firstName); // Εμφανίζει: John
console.log(person.age); // Εμφανίζει: 30
```

```
// Κλήση μεθόδου
person.greet(); // Εμφανίζει: Hello, my name is John Doe.
```

{ https://ideone.com/wt3tG8 }

2. Χρήση του Constructor Function

Οι συναρτήσεις κατασκευής χρησιμοποιούνται για τη δημιουργία πολλών αντικειμένων με την ίδια δομή.

Σύνταξη:

```
function ConstructorName(param1, param2) {
   this.property1 = param1;
   this.property2 = param2;
}
```

Παράδειγμα:

```
function Car(brand, model, year) {
   this.brand = brand;
   this.model = model;
   this.year = year;
   this.displayInfo = function() {
      console.log(`${this.brand} ${this.model} (${this.year})`);
   };
}

// Δημιουργία αντικειμένων
let car1 = new Car("Toyota", "Corolla", 2020);
let car2 = new Car("Honda", "Civic", 2019);

// Πρόσβαση στις ιδιότητες
console.log(car1.brand); // Εμφανίζει: Toyota
console.log(car2.year); // Εμφανίζει: 2019

// Κλήση μεθόδου
car1.displayInfo(); // Εμφανίζει: Toyota Corolla (2020)
```

{ https://ideone.com/4LQlNC }

3. Χρήση της Object.create()

Η μέθοδος Object.create() χρησιμοποιείται για τη δημιουργία αντικειμένων με βάση ένα υπάρχον πρωτότυπο.

Σύνταξη:

```
let newObject = Object.create(existingObject);
```

Παράδειγμα:

```
let animal = {
```

```
    type: "Unknown",
    sound: function() {
        console.log("Some generic sound...");
    }
};

let dog = Object.create(animal);
dog.type = "Dog";
dog.bark = function() {
    console.log("Woof!");
};

// Πρόσβαση σε ιδιότητες και μεθόδους
console.log(dog.type); // Εμφανίζει: Dog
dog.sound(); // Εμφανίζει: Some generic sound...
dog.bark(); // Εμφανίζει: Woof!
```

{ https://ideone.com/UNhdCe }

4. Χρήση της class (ES6 Syntax)

Οι κλάσεις εισήχθησαν στην ES6 και παρέχουν έναν καθαρότερο τρόπο για τη δημιουργία αντικειμένων μέσω ενός προτύπου.

Σύνταξη:

```
class ClassName {
    constructor(param1, param2) {
        this.property1 = param1;
        this.property2 = param2;
    }

    methodName() {
        // Λειτουργία
    }
}
```

Παράδειγμα:

```
class Person {
    constructor(firstName, lastName, age) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
    }

    greet() {
        console.log(`Hello, my name is ${this.firstName} ${this.lastName}.`);
    }
}

// Δημιουργία αντικειμένων
let person1 = new Person("Alice", "Smith", 25);
let person2 = new Person("Bob", "Brown", 30);
```

```
// Πρόσβαση σε ιδιότητες
console.log(person1.firstName); // Εμφανίζει: Alice
console.log(person2.age); // Εμφανίζει: 30

// Κλήση μεθόδου
person1.greet(); // Εμφανίζει: Hello, my name is Alice Smith.
```

{ https://ideone.com/PVyJDI }

5. Χρήση του Object Constructor

Η μέθοδος αυτή χρησιμοποιεί τον ενσωματωμένο κατασκευαστή Object.

Σύνταξη:

```
let objectName = new Object();
objectName.property = value;
```

Παράδειγμα:

```
let book = new Object();
book.title = "JavaScript: The Good Parts";
book.author = "Douglas Crockford";
book.year = 2008;

console.log(book.title); // Εμφανίζει: JavaScript: The Good Parts
console.log(book.year); // Εμφανίζει: 2008
```

{ https://ideone.com/htGQGA }

6. Χρήση JSON (JavaScript Object Notation)

Το JSON είναι μια δημοφιλής μορφή για τη δημιουργία αντικειμένων, ειδικά όταν τα δεδομένα έρχονται από API.

Σύνταξη:

```
let objectName = JSON.parse(jsonString);
```

Παράδειγμα:

```
let jsonString = '{"title": "Eloquent JavaScript", "author": "Marijn Haverbeke", "year": 2018}';
let book = JSON.parse(jsonString);

console.log(book.title); // Εμφανίζει: Eloquent JavaScript
console.log(book.author); // Εμφανίζει: Marijn Haverbeke
```

{ https://ideone.com/JDehJm }

Συνοπτικός Πίνακας Τρόπων Ορισμού Αντικειμένων

Τρόπος	Πλεονεκτήματα	Παραδείγματα Χρήσης
Object Literal	Απλότητα, καθαρότητα	Στατικά δεδομένα

Τρόπος	Πλεονεκτήματα	Παραδείγματα Χρήσης
Constructor Function	Ευελιξία για πολλά παρόμοια αντικείμενα	Δημιουργία αντικειμένων με κοινή δομή
Object.create()	Κληρονομικότητα	Βασισμένο σε υπάρχοντα αντικείμενα
class (ES6)	Καθαρό και μοντέρνο στυλ	Κατασκευαστές, σύνθετες εφαρμογές
Object Constructor	Δυναμική δημιουργία αντικειμένων	Χρήση σε δυναμικά περιβάλλοντα
JSON	Εύκολη μεταφορά δεδομένων	Εισαγωγή δεδομένων από API

Με αυτούς τους τρόπους, μπορείτε να επιλέξετε τη μέθοδο που ταιριάζει καλύτερα στις ανάγκες του έργου σας.

Συναρτήσεις στην JavaScript

Στην **JavaScript**, οι συναρτήσεις είναι βασικές δομές που επιτρέπουν την επαναχρησιμοποίηση κώδικα. Είναι μπλοκ κώδικα που εκτελούνται όταν καλούνται. Η γλώσσα υποστηρίζει διαφορετικούς τρόπους ορισμού συναρτήσεων και διαφορετικούς τύπους.

Τρόποι Ορισμού Συναρτήσεων

1. **Δηλωμένες Συναρτήσεις (Function Declaration):**

- Ορίζονται με τη λέξη-κλειδί function.
- Είναι διαθέσιμες σε όλο το πεδίο (hoisting).

Παράδειγμα:

```
function greet() {
  console.log("Hello, world!");
}
```

2. **Εκφράσεις Συναρτήσεων (Function Expression):**

- Αποθηκεύονται σε μια μεταβλητή.
- Δεν είναι διαθέσιμες πριν από τον ορισμό τους.

Παράδειγμα:

```
let greet = function() {
  console.log("Hello, world!");
};
```

3. **Βέλη Συναρτήσεις (Arrow Functions):**

- Εισήχθησαν στην ES6.
- Είναι συντομότερη σύνταξη και δεν έχουν το δικό τους this.

Παράδειγμα:

```
let greet = () => console.log("Hello, world!");
```

4. **Συναρτήσεις Ανωνυμίας (Anonymous Functions):**

- Δεν έχουν όνομα και χρησιμοποιούνται συχνά σε callbacks.

Παράδειγμα:

```
setTimeout(function() {
  console.log("Αυτό είναι μια ανώνυμη συνάρτηση.");
}, 1000);
```

5. **Συναρτήσεις Κατασκευής (Constructor Functions):**

- Δημιουργούνται με το new Function.

Παράδειγμα:

```
let add = new Function("a", "b", "return a + b");
```

Πέρασμα Παραμέτρων: Με Τιμή και Αναφορά

1. **Πέρασμα με τιμή (Pass by Value):**

- Τα πρωτότυπα δεδομένα (π.χ., αριθμοί, συμβολοσειρές, boolean) περνούν ως αντίγραφα.
- Τυχόν αλλαγές μέσα στη συνάρτηση δεν επηρεάζουν την αρχική μεταβλητή.

Παράδειγμα:

```
function increment(value) {
  value++;
  console.log("Μέσα στη συνάρτηση:", value);
}

let num = 5;
increment(num); // Εμφανίζει: Μέσα στη συνάρτηση: 6
console.log("Εκτός συνάρτησης:", num); // Εμφανίζει: 5
```

{ https://ideone.com/kjGUp3 }

2. **Πέρασμα με αναφορά (Pass by Reference):**

- Τα αντικείμενα και οι πίνακες περνούν ως αναφορές.
- Τυχόν αλλαγές μέσα στη συνάρτηση επηρεάζουν την αρχική μεταβλητή.

Παράδειγμα:

```
function addProperty(obj) {
  obj.newProperty = "Νέα τιμή";
}

let myObject = { key: "Τιμή" };
addProperty(myObject);
console.log(myObject); // Εμφανίζει: { key: "Τιμή", newProperty: "Νέα τιμή" }
```

{ https://ideone.com/gOnknQ }

Παραδείγματα Συναρτήσεων

Παράδειγμα 1: Συνάρτηση χωρίς παραμέτρους

```
function sayHello() {
   console.log("Hello!");
}

sayHello(); // Εμφανίζει: Hello!
```

{ https://ideone.com/qSxCmq }

Παράδειγμα 2: Συνάρτηση με παραμέτρους

```
function greet(name) {
   console.log(`Hello, ${name}!`);
}

greet("John"); // Εμφανίζει: Hello, John!
```

{ https://ideone.com/OVOyeq }

Παράδειγμα 3: Επιστροφή τιμής

```
function add(a, b) {
   return a + b;
}

let result = add(5, 7);
console.log("Το αποτέλεσμα είναι:", result); // Εμφανίζει: Το αποτέλεσμα είναι: 12
```

{ https://ideone.com/aoSQNa }

Παράδειγμα 4: Βέλη Συνάρτηση

```
let multiply = (a, b) => a * b;

console.log(multiply(3, 4)); // Εμφανίζει: 12
```

{ https://ideone.com/MGcS3A }

Παράδειγμα 5: Ανώνυμη Συνάρτηση

```
let subtract = function(a, b) {
   return a - b;
};

console.log(subtract(10, 3)); // Εμφανίζει: 7
```

{ https://ideone.com/whD3SS }

Παράδειγμα 6: Συνάρτηση με προεπιλεγμένες παραμέτρους

```
function calculateArea(width = 5, height = 10) {
   return width * height;
}

console.log(calculateArea()); // Εμφανίζει: 50
console.log(calculateArea(3, 4)); // Εμφανίζει: 12
```

{ https://ideone.com/d1bVMB }

Παράδειγμα 7: Αναδρομή

```
function factorial(n) {
   if (n === 0) {
      return 1;
   }
   return n * factorial(n - 1);
}

console.log(factorial(5)); // Εμφανίζει: 120
```

{ https://ideone.com/Z7jn4x }

Παράδειγμα 8: Συνάρτηση ως τιμή

```
let square = function(x) {
   return x * x;
};

console.log(square(4)); // Εμφανίζει: 16
```

{ https://ideone.com/xZNQUw }

Παράδειγμα 9: Πέρασμα πίνακα ως παράμετρος

```
function printArray(arr) {
   for (let item of arr) {
      console.log(item);
   }
}

let myArray = [1, 2, 3, 4];
printArray(myArray);
// Εμφανίζει:
// 1
// 2
// 3
// 4
```

{ https://ideone.com/glidfG }

Παράδειγμα 10: Κατασκευή αντικειμένων με Constructor Function

```
function Car(brand, model) {
   this.brand = brand;
```

```
    this.model = model;

    this.displayInfo = function() {
        console.log(`${this.brand} ${this.model}`);
    };
}

let car1 = new Car("Toyota", "Corolla");
car1.displayInfo(); // Εμφανίζει: Toyota Corolla
```

{ https://ideone.com/aIwYcZ }

Συνοψίζοντας

- Η **JavaScript** υποστηρίζει διάφορους τύπους συναρτήσεων, όπως δηλωμένες, εκφράσεις, βέλη και ανώνυμες.
- Το πέρασμα παραμέτρων γίνεται είτε με **τιμή** (για πρωτότυπα δεδομένα) είτε με **αναφορά** (για αντικείμενα/πίνακες).
- Η χρήση συναρτήσεων βελτιώνει την επαναχρησιμοποίηση και την οργάνωση του κώδικα.

Τα παραδείγματα καλύπτουν από βασικές έως πιο σύνθετες περιπτώσεις για εκπαιδευτικούς σκοπούς.

Αντικειμενοστραφής Προγραμματισμός (Object-Oriented Programming) στην JavaScript

Η **JavaScript** υποστηρίζει αντικειμενοστραφή προγραμματισμό (OOP) μέσω αντικειμένων και κλάσεων. Είναι μια πρωτότυπα βασισμένη γλώσσα, όπου τα αντικείμενα μπορούν να δημιουργούνται και να επεκτείνονται δυναμικά.

Τρόποι Υλοποίησης OOP στην JavaScript

1. **Δημιουργία Αντικειμένων με Object Literal**

- Απλός τρόπος δημιουργίας αντικειμένων με συγκεκριμένες ιδιότητες και μεθόδους.

2. **Χρήση Constructor Functions**

- Παλιός τρόπος δημιουργίας αντικειμένων πριν την εισαγωγή των class.

3. **Χρήση** class **(ES6 και νεότερες εκδόσεις)**

- Νεότερος και καθαρότερος τρόπος ορισμού αντικειμένων και κλάσεων.

4. **Κληρονομικότητα μέσω** class **ή** Object.create

- Επέκταση υπαρχουσών κλάσεων ή αντικειμένων.

5. **Χρήση Prototypes**

- Δυνατότητα προσθήκης μεθόδων και ιδιοτήτων σε υπάρχοντα αντικείμενα μέσω prototype.

6. **Polymorphism (Πολυμορφισμός)**

- Διαφορετική υλοποίηση μιας μεθόδου στις υποκλάσεις.

10 Αναλυτικά Παραδείγματα Προγραμμάτων

1. Δημιουργία Αντικειμένου με Object Literal

Κώδικας:

```
let car = {
  brand: "Toyota",
  model: "Corolla",
  year: 2020,
  displayInfo: function() {
    console.log(`${this.brand} ${this.model} (${this.year})`);
  }
};

// Κλήση της μεθόδου
car.displayInfo(); // Εμφανίζει: Toyota Corolla (2020)
```

{ https://ideone.com/KHnlpD }

2. Δημιουργία Αντικειμένων με Constructor Function

Κώδικας:

```
function Person(firstName, lastName) {
  this.firstName = firstName;
  this.lastName = lastName;
```

```
    this.getFullName = function() {
        return `${this.firstName} ${this.lastName}`;
    };
}

let person1 = new Person("John", "Doe");
console.log(person1.getFullName()); // Εμφανίζει: John Doe
```

{ https://ideone.com/iPeD7r }

3. Δημιουργία Κλάσης με ES6 Syntax

Κώδικας:

```
class Animal {
    constructor(name, sound) {
        this.name = name;
        this.sound = sound;
    }

    makeSound() {
        console.log(`${this.name} λέει: ${this.sound}`);
    }
}

let dog = new Animal("Σκύλος", "Γαβ");
dog.makeSound(); // Εμφανίζει: Σκύλος λέει: Γαβ
```

{ https://ideone.com/MDuGK8 }

4. Κληρονομικότητα με extends

Κώδικας:

```
class Animal {
    constructor(name) {
        this.name = name;
    }

    makeSound() {
        console.log(`${this.name} κάνει ήχο.`);
    }
}

class Dog extends Animal {
    makeSound() {
        console.log(`${this.name} λέει: Γαβ`);
    }
}

let dog = new Dog("Σκύλος");
dog.makeSound(); // Εμφανίζει: Σκύλος λέει: Γαβ
```

{ https://ideone.com/HqoX7U }

5. Χρήση Object.create για Κληρονομικότητα

Κώδικας:

```
let animal = {
   name: "Animal",
   makeSound: function() {
      console.log(`${this.name} κάνει ήχο.`);
   }
};

let cat = Object.create(animal);
cat.name = "Γάτα";
cat.makeSound(); // Εμφανίζει: Γάτα κάνει ήχο.
```

{ https://ideone.com/kgSyhj }

6. Προσθήκη Μεθόδων με prototype

Κώδικας:

```
function Car(brand, model) {
   this.brand = brand;
   this.model = model;
}

Car.prototype.displayInfo = function() {
   console.log(`${this.brand} ${this.model}`);
};

let car1 = new Car("Toyota", "Corolla");
car1.displayInfo(); // Εμφανίζει: Toyota Corolla
```

{ https://ideone.com/SuJp9k }

7. Πολυμορφισμός με Υποκλάσεις

Κώδικας:

```
class Shape {
   getArea() {
      return 0;
   }
}

class Circle extends Shape {
   constructor(radius) {
      super();
      this.radius = radius;
   }

   getArea() {
      return Math.PI * this.radius ** 2;
   }
}
```

```
let circle = new Circle(5);
console.log(circle.getArea()); // Εμφανίζει: 78.53981633974483
```

{ https://ideone.com/7zp0QH }

8. Static Methods

Κώδικας:

```
class MathUtil {
  static add(a, b) {
    return a + b;
  }
}

console.log(MathUtil.add(5, 7)); // Εμφανίζει: 12
```

{ https://ideone.com/Tf5elg }

9. Encapsulation (Ιδιότητες με Private Scope)

Κώδικας:

```
class Counter {
  #count = 0; // Ιδιωτική ιδιότητα

  increment() {
    this.#count++;
  }

  getCount() {
    return this.#count;
  }
}

let counter = new Counter();
counter.increment();
console.log(counter.getCount()); // Εμφανίζει: 1
// console.log(counter.#count); // Σφάλμα: Private field '#count' must be declared
```

{ https://ideone.com/kzrDEf }

10. Χρήση Getters και Setters

Κώδικας:

```
class Person {
  constructor(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
  }

  get fullName() {
    return `${this.firstName} ${this.lastName}`;
```

```
    }

    set fullName(name) {
        [this.firstName, this.lastName] = name.split(" ");
    }
}

let person = new Person("John", "Doe");
console.log(person.fullName); // Εμφανίζει: John Doe
person.fullName = "Jane Smith";
console.log(person.fullName); // Εμφανίζει: Jane Smith
```

{ https://ideone.com/P6MGp0 }

Συμπεράσματα

Η **JavaScript** παρέχει πλήρη υποστήριξη για αντικειμενοστραφή προγραμματισμό. Οι κύριες τεχνικές περιλαμβάνουν τη χρήση αντικειμένων, κλάσεων, πρωτοτύπων και κληρονομικότητας. Οι κλάσεις και οι μέθοδοι όπως get, set, και static βελτιώνουν τη σαφήνεια και τη λειτουργικότητα.