ΠΕΡΙΕΧΟΜΕΝΑ:

Ο τέλεστής this

- Μέχρι τώρα έχουμε δει:
  - Σε μία **συνάρτηση-κατασκευαστή**, το this αναφέρεται στο αντικείμενο που κατασκευάζεται
  - Σε μία **μέθοδο αντικειμένου**, το this αναφέρεται στο ίδιο το αντικείμενο που περιέχει τη μέθοδο.

- Βλέπουμε περαιτέρω:
  - Το this **στο global context** αναφέρεται στο Window
  - Σε μία **συνάρτηση** (που δεν περιέχεται σε αντικείμενο) και δεν είμαστε σε strict mode: αναφέρεται στο Window
  - Σε μία **συνάρτηση** (που δεν περιέχεται σε αντικείμενο) και είμαστε σε strict mode: είναι undefined

Παράδειγμα 1: this_methods

```
// An object
let ob = {
   x: 1,
   f: function() { return this; }
};
console.log(ob.f());
// A constructor function
function Constructor() {
   this.x = 1;
}
Constructor.prototype.f = function() { return this; }
let ob2 = new Constructor();
console.log(ob2.f());
// A class
class Class {
   constructor() {
      this.x = 1;
   }
   method() { return this; }
}
let ob3 = new Class();
console.log(ob3);
```

```
▼ Object
  ▶ f: ƒ ()
    x: 1
  ▶ [[Prototype]]: Object
```

```
▼ Constructor
    x: 1
  ▼ [[Prototype]]: Object
    ▶ f: ƒ ()
    ▶ constructor: ƒ Constructor()
```

```
▼ Class
    x: 1
  ▼ [[Prototype]]: Object
    ▶ constructor: class Class
    ▶ method: ƒ method()
```

Παράδειγμα 2: this_functions

```
console.log(this);

// A function
function myfunc1() {
   return this;
}
console.log(myfunc1());


// on inner functions
function outer() {
   function inner() {
      return this;
   }

   return inner();
}
console.log(outer());
```

strict mode

```
▶ Window
undefined
undefined
```

non-strict mode

```
▶ Window {window: Window, se
▶ Window {window: Window, se
▶ Window {window: Window, se
```

ΜΑΘΗΜΑ 10.3: this και Μέθοδοι Συναρτήσεων 1.1. this σε arrow functions

- Όσα είπαμε ισχύουν για τυπικές συναρτήσεις.
- Σε arrow functions, ο τελεστης this αναφέρεται στο περιβάλλον εκτέλεσης που τις κάλεσε.

Παράδειγμα 3: this_arrow_functions

```
// An arrow function enclosed in global EC
console.log((()=>this)());

// An arrow function enclosed in function EC
function f() {
   console.log((()=>this)());
}
f();

// An arrow function enclosed in a method
let ob = {
   x: 1,
   f: function() {
      return (()=>this)();
   }
}
console.log(ob.f());
```

```
▸ Window {window: Window, self: W
undefined
▸ {x: 1, f: f}
```

Άσκηση 1:

- Υπολογίστε, χωρίς να τρέξετε τον κώδικα, τι επιστρέφει το ακόλουθο πρόγραμμα:

```
// An arrow function as a method
let ob = {
   x: 1,
   f: ()=>this
}
console.log(ob.f());
```

ΜΑΘΗΜΑ 10.3: this και Μέθοδοι Συναρτήσεων

2.1. Μέθοδοι: call() και apply()

JavaScript psounis

Μέθοδοι αντικειμένου - συνάρτησης:

- Βρίσκουμε τις συναρτήσεις που μελετάμε, στο πρωτότυπο του αντικειμένου - συνάρτησης (Function.prototype):

Μέθοδος	Επεξήγηση
call(this, ...args)	Έστω συνάρτηση f(x,y). Καλούμε την call ως: f.call(this, x, y): Η συνάρτηση f καλείται έχοντας θέσει το this της συνάρτησης ίσο με το όρισμα.

- Ίδια λειτουργικότητα, με διαφορετικές παραμέτρους έχει η μέθοδος:

Μέθοδος	Επεξήγηση
apply(this, args)	Έστω συνάρτηση f(x,y). Καλούμε την apply ως: f.apply(this, [x, y]): Η συνάρτηση f καλείται έχοντας θέσει το this της συνάρτησης ίσο με το όρισμα.

Παράδειγμα 3: call

```
function greet(greeting, question) {
  console.log(`${greeting}, my name is ${this.name}. ${question}`);
}

const person1 = { name: "Tom" };
const person2 = { name: "Bob" };

greet.call(person1, "Hello", "How is it going?");
greet.call(person2, "Hi", "Are you ok?");
```

```
Hello, my name is Tom. How is it going?
Hi, my name is Bob. Are you ok?
```

Παράδειγμα 4: apply

```
function greet(greeting, question) {
  console.log(`${greeting}, my name is ${this.name}. ${question}`);
}

const person1 = { name: "Tom" };
const person2 = { name: "Bob" };

greet.apply(person1, ["Hello", "How is it going?"]);
greet.apply(person2, ["Hi", "Are you ok?"]);
```

```
Hello, my name is Tom. How is it going?
Hi, my name is Bob. Are you ok?
```

Προχωρημένες Χρήσεις:

- Ένα αντικείμενο μπορεί να «δανειστεί» μία μέθοδο από ένα άλλο αντικείμενο.

Παράδειγμα 5: call_usage

```
let object1 = {
   name: "Bob",
   toString() {
      console.log(this.name);
   }
}

let object2 = {
   name: "Tom"
}

object1.toString.call(object2);
```

Άσκηση 2:

- Ορίστε ένα αντικείμενο με όνομα utils. Να περιέχει μία μέθοδο που μετράει το πλήθος των εμφανίσεων μίας τιμής σε ένα array-like αντικείμενο.
- Ορίστε ένα array-like αντικείμενο και δανειστείτε την παραπάνω μέθοδο για να μετρήσετε το πλήθος εμφανίσεων μιας συγκεκριμένης τιμής.

- Παλιότερα, η apply χρησιμοποιούνταν για να απλώσουμε τα περιεχόμενα ενός πίνακα ως ορίσματα (με την ES6 αυτό γίνεται πολύ πιο κομψά με τον spread operator)

Παράδειγμα 6: call_usage2

```
let array = [1,2,3,4,5];

function max() {
   let v = arguments[0];
   for (let item of arguments)
      if (item > v)
         v = item;
   return v;
}

// ES6 +
console.log(max(...array));

// pre-ES6
console.log(max.apply(null, array));
```

Η μέθοδος bind():

Μέθοδος	Επεξήγηση
bind(this, …args)	Έστω συνάρτηση f(x,y). Καλούμε την bind ως: f.bind(this, x, y): Δεν εκτελεί την f (όπως η call()), αλλά επιστρέφει μία νέα συνάρτηση, που τρέχει την f επί του ορίσματος this που διοχετεύσαμε (ώστε να μπορούμε να την καλέσουμε πολλές φορές)

Παράδειγμα 7: bind

```
function greet(greeting, question) {
   console.log(`${greeting}, my name is ${this.name}. ${question}`);
}

const person1 = { name: "Tom" };
const person2 = { name: "Bob" };

let greetTom = greet.bind(person1, "Hello", "How is it going?");
let greetBob = greet.bind(person2, "Hi", "Are you ok?");

greetTom();
greetTom();
greetBob();
greetBob();
```

Παρατήρηση:

- Με την bind() αποφεύγουμε ένα συνηθισμένο λάθος: Να εξάγουμε μία συνάρτηση από ένα αντικείμενο και να περιμένουμε να έχει την ίδια συμπεριφορά εφόσον την καλέσουμε.

Παράδειγμα 8: apply

```
let object = {
   x: 1,
   getX: function() {
      return this.x;
   }
}

// wrong
let extracted = object.getX;
// console.log(extracted()); // error

// correct
let extracted2 = object.getX.bind(object);
console.log(extracted2());

// correct
console.log(extracted.bind(object)());
```

ΜΑΘΗΜΑ 10.3: this και Μέθοδοι Συναρτήσεων

2.3. Η μέθοδος: toString()

JavaScript psounis

Η μέθοδος toString():

Μέθοδος	Επεξήγηση
toString()	Εφόσον δεν την επαναορίσουμε επιστρέφει τον κώδικα της συνάρτησης

Παράδειγμα 9: toString

```
function f() {
   console.log("hey");
}

console.log(f);
console.log(f.toString, f.toString());
f.toString = () => "f";
console.log(f.toString, f.toString());
```

Μία ενδιαφέρουσα χρήση (και εξαιρετικά επικίνδυνη):

- Η μέθοδος eval τρέχει τον κώδικα που δέχεται ως όρισμα ως συμβολοσειρά.
- Ωστόσο θεωρείται επικίνδυνη μέθοδος, ειδικά αν η συμβολοσειρά μπορεί να τροποποιηθεί από κάποιον «κακό». Π.χ. αν η συμβολοσειρά διαβάζεται από κάποιο πλαίσιο κειμένου.

Παράδειγμα 10: toString2

```
function f() {
   console.log("hey");
}

console.log(f);
eval("(" + f.toString().replace("\n", "") + ")()");
```

Άσκηση 3:

- Ορίστε ένα αντικείμενο piko:
  - Να περιέχει το όνομά του (name) και το είδος του (type).
  - Αρχικοποιήστε το με τιμές name="piko", type="dog"
  - Ορίστε μία μέθοδο toString() που να τυπώνει το είδος και το όνομά του αντικειμένου
- Ορίστε ένα αντικείμενο miou:
  - Να περιέχει το όνομά του (name) και το είδος του (type).
  - Αρχικοποιήστε το με τιμές name="miou", type="cat"
- Στο κυρίως πρόγραμμα, τυπώστε τον piko, και έπειτα δανειστείτε τη μέθοδο toString του piko, για να τυπώστε τις πληροφορίες για την γάτα miou.