SAE INSTITUTE

WMNFE2010 | FRONT-END DEVELOPMENT

# JavaScript #11

## JavaScript Objects

# ΠΕΡΙΕΧΟΜΕΝΑ

# Περιεχόμενα

- **JavaScript** Objects
  - Instantiation
  - Properties
  - Methods
  - **this** keyword
  - Constructor function

# JAVASCRIPT OBJECTS

# JS Objects

## Data Types

Αν εξαιρέσουμε τα **7** *primitive data types* της **JavaScript** (**number**, **string**, **boolean**, **bigint**, **symbol**, **undefined**, **null**).

Όλα τα υπόλοιπα είναι αντικείμενα:

- Σχεδόν οτιδήποτε μπορεί να δημιουργηθεί με το keyword **new** (**Array**, **Date** κ.λπ.)
- Οι συναρτήσεις (παρόλο που στη χρήση του τελεστή **typeof** απαντούν με **function**)

Ο τύπος **null** ανήκει στα *primitive data types*, παρόλο που **typeof null === 'object'**!.

# JS Objects

## OOP

Η JavaScript δεν είναι μια "τυπική" αντικειμενοστρεφής γλώσσα, όπως οι **Java**, **C#** κ.α.

- Ή τουλάχιστον δεν ήταν μέχρι την έκδοση *ECMAScript 2015*, όπου πλέον υπάρχει και το keyword **class**.
- Πριν την έκδοση αυτή, οι αντικειμενοστρεφής της ιδιότητες προέκυπταν από τα **Prototypes**.

Έχουμε συναντήσει ήδη αρκετά **Objects**

- **Array**, **Document**, **Element** κ.λπ.

# INSTANTIATION

# Instantiation

Δύο βασικοί τρόποι δημιουργίας ενός αντικειμένου:

```
const obj = new Object();

obj.property1 = value1;
obj.property2 = value2;
```

```
// JSON

const obj = {
  property1: value1,
  property2: value2
};
```

# Instantiation

## Παράδειγμα #1

```
let student = new Object();

student.firstname = 'Joan';
student.lastname = 'Doe';
student.age = 21;
```

# Instantiation

## Παράδειγμα #2

```
let student = {
  firstname: 'Joan',
  lastname: 'Doe',
  age: 21
};
```

# PROPERTIES

# Properties

## Γενικά

- Αυτό που χαρακτηρίζει ένα αντικείμενο, είναι οι ιδιότητές του.
- Πρόκειται για τιμές που σχετίζονται με/ανήκουν σε συγκεκριμένο αντικείμενο.
- Η προσπέλαση σε αυτές γίνεται:
  - με το *dot notation*, π.χ. **array.length**
  - με το *square bracket notation*, π.χ. **element['value']**

# Properties

## Παραδείγματα

```
const arr = [1, 2, 3, 4];
console.log(arr);
console.log(arr.length);

const form = document.getElementById('my-form');
console.log(form.elements['firstname'].value);
```

# Properties

## Συλλογή από key-value pairs

- Τα αντικείμενα στη **JavaScript** μπορούν να θεωρηθούν και ως συλλογές από ιδιότητες, τις οποίες μπορείς να προσπελάσεις με κάποιο *κλειδί*
  - Σε αντίθεση με τους πίνακες, όπου χρησιμοποιείς τη *θέση*
- Είναι κοντά σε έννοιες όπως **dictionary** / **property bag** κ.α.
  - Από την έκδοση *ECMAScript 2015* υπάρχουν και τα αντικείμενα **Map** και **Set**

# Properties

## Παράδειγμα #1

Καμιά φορά θυμίζουν πίνακες ή άλλες δομές...

```
const studentNames = [
  "Luke Skywalker",
  "Darth Vader",
  "Leia Organa"
];
console.log(studentNames[0]);

const studentNames = {
  s001: "Luke Skywalker",
  s002: "Darth Vader",
  s003: "Leia Organa"
};
console.log(studentNames["s001"]);
```

# Properties

## Παράδειγμα #2

... αλλά δεν είναι αυτή η κύρια "αποστολή" τους

```
const brands = [
  "Audi",
  "Mercedes",
  "Volkswagen"
];
console.log(brands[1]);

const car = {
  brand: "Audi",
  model: "R8",
  color: "Red"
};
console.log(car["model"]);
```

# METHODS

# Methods

## Γενικά

- Οι μέθοδοι είναι ενέργειες που αφορούν σε κάποιο αντικείμενο
- Είναι στην ουσία ιδιότητες οι οποίες περιέχουν τον ορισμό μιας συνάρτησης
- Έχουν άμεση πρόσβαση στις άλλες ιδιότητες του αντικειμένου

# Methods

## Παράδειγμα #1

```
let user = {
  firstname: "Slim",
  lastname: "Shady",
  greet() {
    alert(`My name is ${this.firstname} ${this.lastname}`);
  }
};

user.greet();
```

# Methods

## Παράδειγμα #2

```
let car = {
  brand: "Mercedes",
  model: "SLK 55 AMG",
  color: "Black",
  paint: function(newColor) { this.color = newColor; }
};

console.log(car.color);
car.paint('Red');
console.log(car.color);
```

# THIS KEYWORD

# this keyword

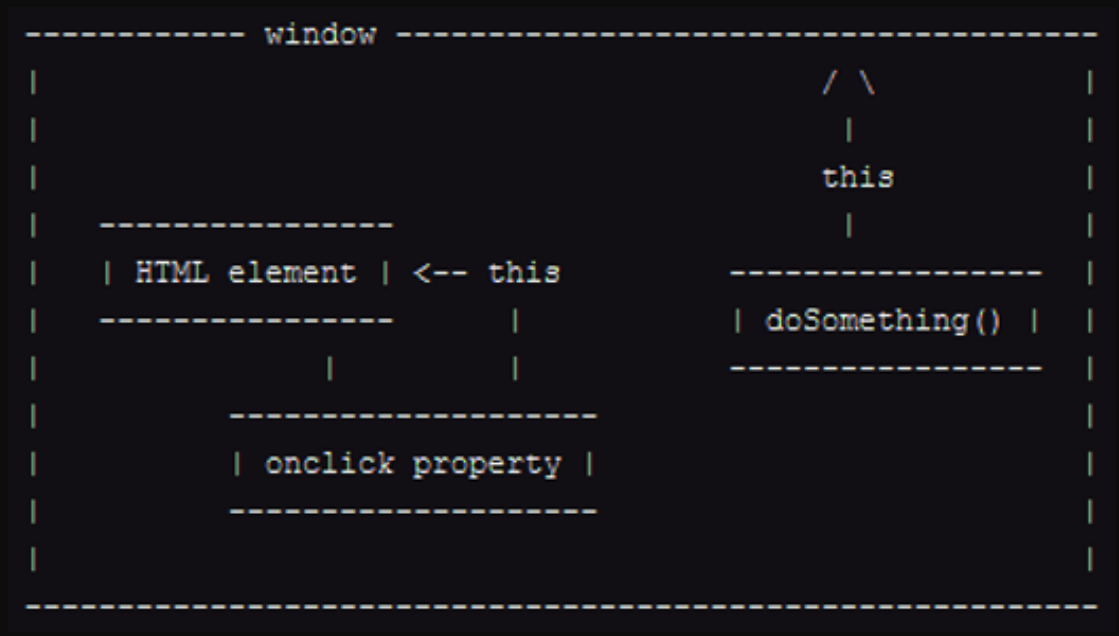
## Γενικά

- Με τη λέξη-κλειδί **this** έχουμε πρόσβαση στο πλαίσιο (*context*) εκτέλεσης κάποιου τμήματος κώδικα
- Στα πλαίσια μιας συνάρτησης, αφορά στον τρόπο που εκτελέστηκε η συνάρτηση αυτή
  - Π.χ. στα πλαίσια ενός *event handler*, μέσω του **this** έχουμε πρόσβαση στο στοιχείο που αφορά το γεγονός
  - Προσοχή με τις *arrow functions*
- Στα πλαίσια ενός αντικειμένου, μπορούμε να έχουμε πρόσβαση στις ιδιότητές του

# this keyword

## Context

Αναλόγως το πλαίσιο (*context*) το οποίο χρησιμοποιείται, αναφέρεται και σε διαφορετικό αντικείμενο.



# CONSTRUCTOR FUNCTION

# Constructor function

## Γενικά

Με την σύνταξη **{...}** μπορούμε να φτιάχνουμε ένα αντικείμενο "τη φορά".

Αν έχουμε σκοπό να δημιουργήσουμε πολλά παρόμοια αντικείμενα, είναι χρήσιμο να δημιουργήσουμε μια **constructor function** για αυτή τη δουλειά, και να την καλούμε με τον τελεστή **new**.

Εξασφαλίζουμε ίδια δομή σε όλα τα αντικείμενα, και δεν επαναλαμβάνουμε κώδικα άσκοπα.

# Constructor function

## Παράδειγμα #1

```
function Car(b, m, c) {
  this.brand = b;
  this.model = m;
  this.color = c;
}

let car1 = new Car('Audi', 'R8', 'Silver');
let car2 = new Car('Ferrari','488 GT','Red');

console.log(car1);
console.log(car2);
```

# Constructor function

## Παράδειγμα #2

```
function Warning(message) {
  this.prefix = "ΠΡΟΣΟΧΗ: ";
  this.message = message;
  this.show = function () {
    alert(`${this.prefix}: ${this.message}!`);
  };
}

let warn1 = new Warning('Έρχεται κακοκαιρία');
warn1.show();

let warn2 = new Warning('Άκουσα κάποιον να βήχει');
warn2.show();
```

# Constructor function

## Σημείωση

Στο προηγούμενο παράδειγμα, θα ήταν ίσως προτιμότερο να χρησιμοποιηθεί το εξής:

```
Warning.prototype.show = function () {
  alert(`${this.prefix}: ${this.message}!`);
};
```

# HOMEWORK

# Homework #1

## Triangle

Δημιουργήστε ένα script στο οποίο θα ορίζεται μία **constructor function** με όνομα *Triangle*.

Τα αντικείμενα που θα δημιουργεί η **function** αυτή θα πρέπει να έχουν δύο ιδιότητες, τις **base** και **height**, καθώς και μία μέθοδο **area()**. Η μέθοδος αυτή θα πρέπει να υπολογίζει και θα επιστρέφει το εμβαδόν του τριγώνου.

**Σημείωση**: **εμβαδό τριγώνου = βάση * ύψος / 2**

# Homework #2

## ImageGallery

Δημιουργήστε ένα script στο οποίο θα ορίζεται μία **constructor function** με όνομα *ImageGallery*.

Ορίστε όσες ιδιότητες (**properties**) και όσες μεθόδους (**methods**) θέλετε μέσα σε αυτή, οι οποίες θεωρείτε πως θα ήταν χρήσιμες σε ένα τέτοιο αντικείμενο.

# Χρήσιμα links

JavaScript data types and data structures - JavaScript | MDN
https://developer.mozilla.org/en-US/docs/Web/JavaS…

Data Structures: Objects and Arrays :: Eloquent JavaScript
https://eloquentjavascript.net/04_data.html

Constructor, operator "new"
https://javascript.info/constructor-new

Working with objects - JavaScript | MDN
https://developer.mozilla.org/en-US/docs/Web/JavaS…

# Extra info

- Object prototypes - Learn web development | MDN
  https://developer.mozilla.org/en-US/docs/Learn/Java…
- Map - JavaScript | MDN
  https://developer.mozilla.org/en-US/docs/Web/JavaS…
- Classes - JavaScript | MDN
  https://developer.mozilla.org/en-US/docs/Web/JavaS…
- What is JSON? A better format for data exchange | InfoWorld
  https://www.infoworld.com/article/3222851/what-is-js…

# THANK YOU!