Προγραμματισμός και Συστήματα στον Παγκόσμιο Ιστό

Εισαγωγή στη JavaScript

Δρ. Δημήτριος Κουτσομητρόπουλος
Ιωάννης Γαροφαλάκης, καθηγητής

Περιεχόμενα

Σήμερα
- Εισαγωγή στην JavaScript (‘vanilla’ JS)
- Βασικά χαρακτηριστικά της γλώσσας
- Χειρισμός συμβάντων (event handling)
- DOM

Την επόμενη φορά
- Εκφράσεις συναρτήσεων
- Promises
- Συναρτησιακός προγραμματισμός στη JavaScript
  - Currying, Closures, Lambda functions
- Frameworks

+

παράγει

Περιγράφει το περιεχόμενο και τη δομή της σελίδας

Περιγράφει την εμφάνιση και τα στυλ της σελίδας

Μια ιστοσελίδα… που δεν κάνει τίποτα

Τι μπορεί να «κάνει» μια σελίδα;

- Να δέχεται είσοδο από το χρήστη
- Να αποθηκεύει την είσοδο από το χρήστη
- Να εμφανίζει και να εξαφανίζει στοιχεία όταν ο χρήστης αλληλεπιδρά με τη σελίδα
- ….

*Η **JavaScript** είναι η μόνη γλώσσα προγραμματισμού που μπορεί να εκτελέσει εγγενώς ο φυλλομετρητής*

JavaScript

Δημιουργήθηκε το 1995 από τον Brendan Eich
- Συνιδρυτή του Mozilla
- Αρχικά μόνο για τον Netscape (μετέπειτα Firefox)

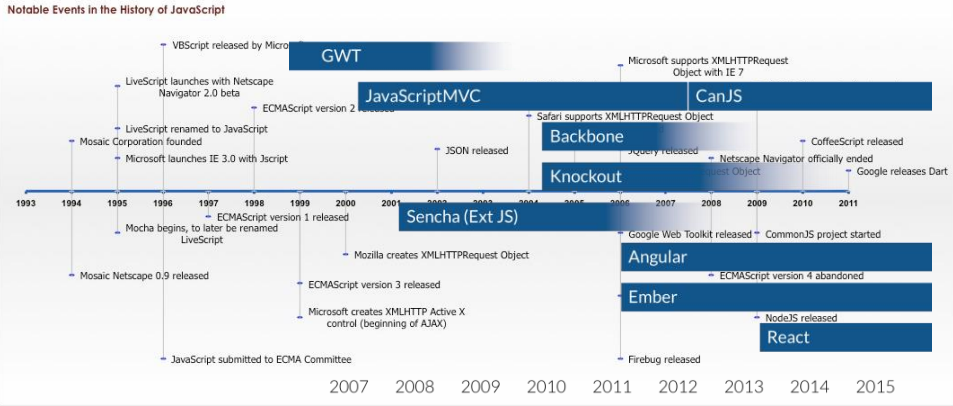
Τυποποιήθηκε το 1996 ως ECMAScript (ES)
- Από το European Computer Manufacturers Association (ECMA)

Η Javascript δεν έχει σχέση με τη Java
- Ονομάστηκε έτσι για εμπορικούς λόγους

Η πρώτη έκδοση γράφτηκε μόλις σε 10 μέρες

"I was under marketing orders to make it look like Java but not make it too big for its britches … [it] needed to be a silly little brother language." (πηγή)

Εξέλιξη



Σήμερα: Έκδοση ECMAScript 2018
Σημαντική Έκδοση: ECMAScript 2015 (ES6)

Πηγή: B. Williamson, 'Intro JS'. http://goodbedford.github.io/introJs/

ECMAScript 2015 (ES6)

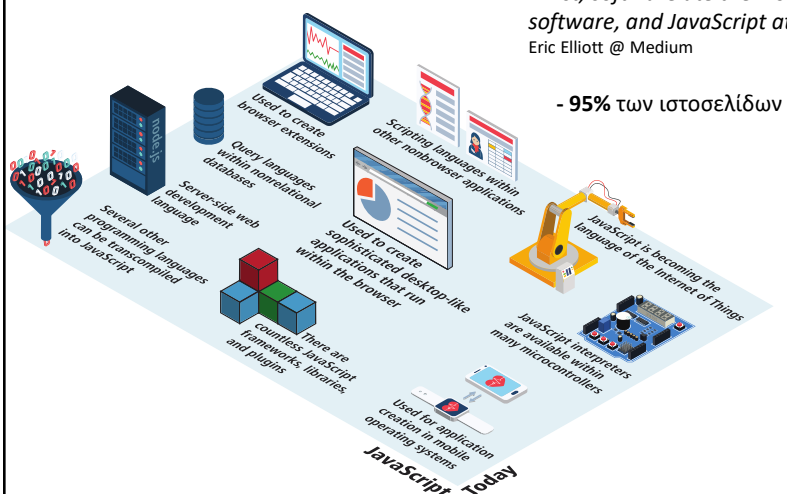
Νέα χαρακτηριστικά

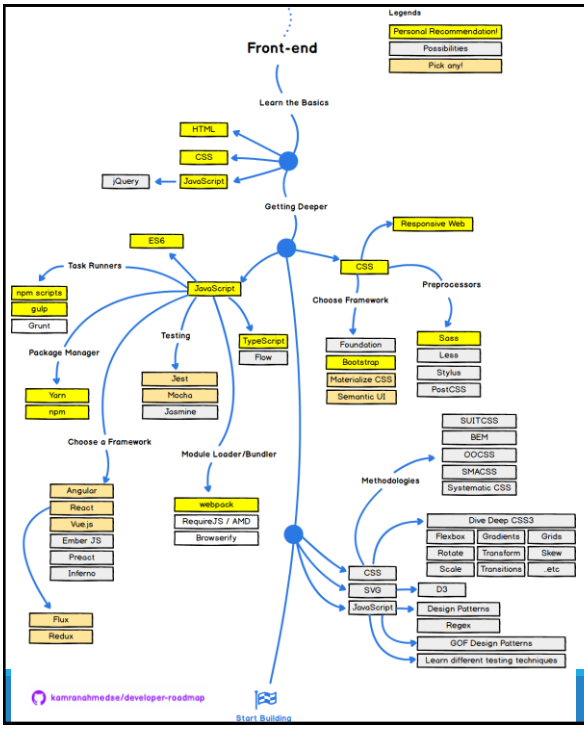
- Υποστήριξη για κλάσεις
- Iterators και for/of loops
- Promises
  - Fetch API
  - Αντικαθιστά το XMLHttpRequest (AJAX style)
- Let, const αντί για var (εμβέλεια βρόχου)
- Arrow functions (lambdas) =>
- currying

Η JavaScript σήμερα

"First, software ate the world, the web ate software, and JavaScript ate the Web".
Eric Elliott @ Medium

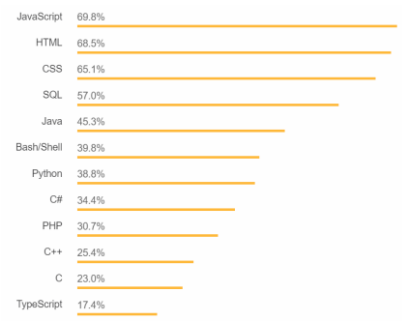
- 95% των ιστοσελίδων χρησιμοποιούν JS



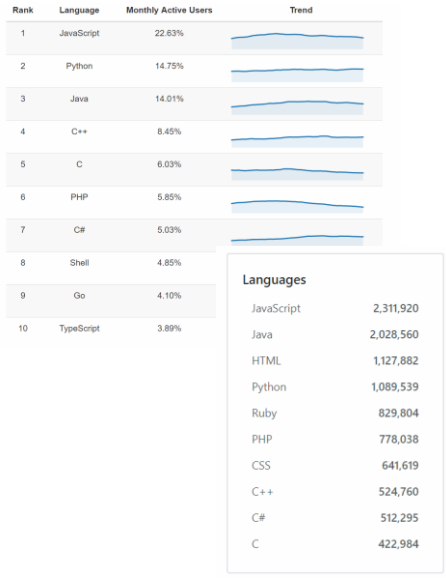


Το οικοσύστημα της JavaScript (και του web development)

StackOverflow Developer Survey (2018)



Github (April, 2018)



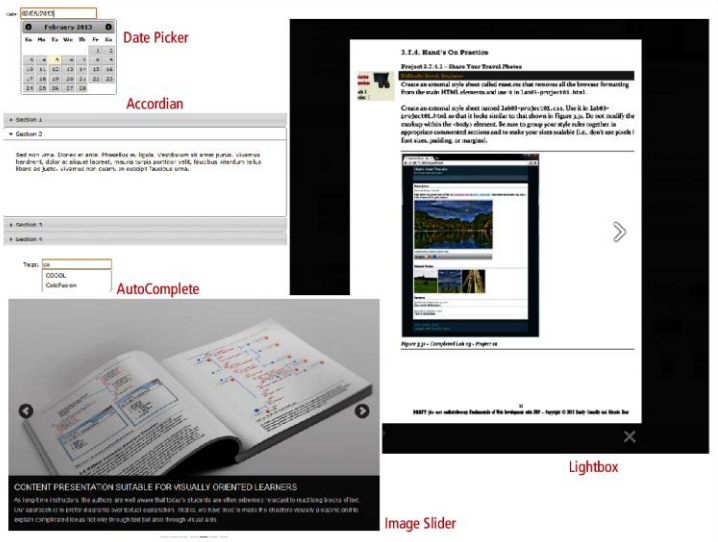
Rank	Language	Monthly Active Users	Trend
1	JavaScript	22.63%	
2	Python	14.75%	
3	Java	14.01%	
4	C++	8.45%	
5	C	6.03%	
6	PHP	5.85%	
7	C#	5.03%	
8	Shell	4.85%	
9	Go	4.10%	
10	TypeScript	3.89%	

Languages

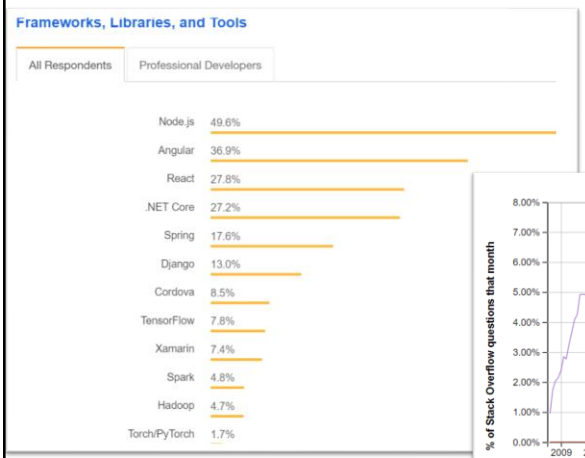
JavaScript	2,311,920
Java	2,028,560
HTML	1,127,882
Python	1,089,539
Ruby	829,804
PHP	778,038
CSS	641,619
C++	524,760
C#	512,295
C	422,984

Δημοφιλία γλωσσών

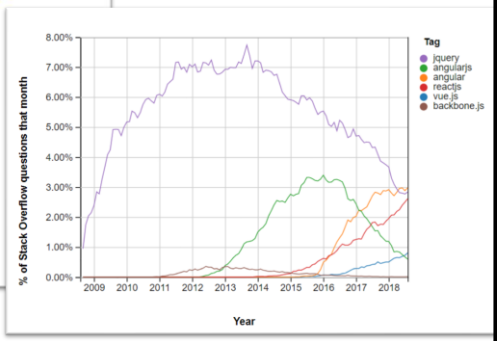
Frameworks/Libraries



JavaScript frameworks και βιβλιοθήκες



StackOverflow, 2018



Βασικά χαρακτηριστικά της JavaScript

JavaScript

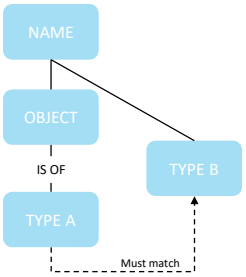
Χαρακτηριστικά γλώσσας προγραμματισμού:

- **Υψηλού Επιπέδου**
- **Διερμηνευόμενη** (interpreted)
- **Δυναμική**
  - Νέες μεταβλητές και συναρτήσεις μπορούν να δημιουργηθούν οποιαδήποτε στιγμή
- **Ασθενών τύπων** (weakly typed)

Σύστημα τύπων (1/3)

Στατικές γλώσσες προγραμματισμού
- Οι μεταβλητές συνδέονται με **ένα** συγκεκριμένο τύπο (typed based)
- Οι έλεγχοι για την ύπαρξη, την ιεραρχία και τον σωστό ορισμό μεθόδων γίνεται κατά την διάρκεια του compile
- «Ρίχνουν» TypeException στην περίπτωση που συνδεθούν με διαφορετικό τύπο από αυτόν που αρχικοποιήθηκαν

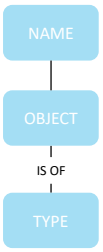
Τέτοιες γλώσσες: Java, C++,…



Σύστημα τύπων (2/3)

Δυναμικές γλώσσες προγραμματισμού
- Οι μεταβλητές δεν συνδέονται με ένα συγκεκριμένο τύπο (untyped based)
- Οι έλεγχοι για την ύπαρξη, την ιεραρχία και τον σωστό ορισμό μεθόδων γίνεται κατά την διάρκεια του runtime (εκτέλεσης)
- Μπορούν να συνδεθούν με διαφορετικό τύπο από αυτόν που αρχικοποιήθηκαν

Τέτοιες γλώσσες: JavaScript, python, php,…



Σύστημα τύπων (3/3)

Η JavaScript **μετατρέπει** τον τύπο σε αυτόν που πρέπει για να πραγματοποιηθεί η οποιαδήποτε εντολή:

```
//JAVASCRIPT
var mStr = "John
Doe";
mStr = 24;
//JAVASCRIPT
var a = "9";
var b = 5;
console.log(a+b);
console.log(a-b);
```

```
//JAVA
String mStr = "John
Doe";
mStr = 24;
//JAVA
String a = "9";
int b = 5;
System.out.println(a+(String)b);
System.out.println(Integer.parseInt(a)-b);
```

⟸ TypeException

JavaScript χρήση (1/3)

Πως χρησιμοποιείται η JavaScript:

1. Ενσωματώνοντας την σε html στοιχείο
   Click me!
2. Χρησιμοποιώντας το <script> στοιχείο στην html
   ```
   <script>
     var a = 5;
   </script>
   ```
3. Χρησιμοποιώντας εξωτερικό αρχείο .js
   ```
   <script src="app.js"></script>
   ```

JavaScript χρήση (2/3)

Πως το χρησιμοποιώ στην σελίδα/εφαρμογή μου:

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1 user-
scalable=no">
  <script src=“app.js” type=“text/javascript”></script>
  <style rel=“stylesheet” href=“app.css" />
  <title>Page/App title</title>
 </head>
 <body>
  <!-- Το σώμα της εφαρμογής/σελίδας -->

 </body>
</html>
```


JavaScript χρήση (3/3)

Το <script> μπορεί να τοποθετηθεί και στο τέλος του <body>:

- Βελτιώνει το χρόνο εμφάνισης της σελίδας
- …αλλά μπορεί να αλλάξει η εμφάνισή της, αφού φορτωθεί

Η χρήση εξωτερικού αρχείου

- Διαχωρίζει τον κώδικα από την HTML
- Διευκολύνει την συντήρηση του κώδικα
- Πραγματοποιείται Caching των αρχείων -> γρηγορότερο φόρτωμα σελίδας

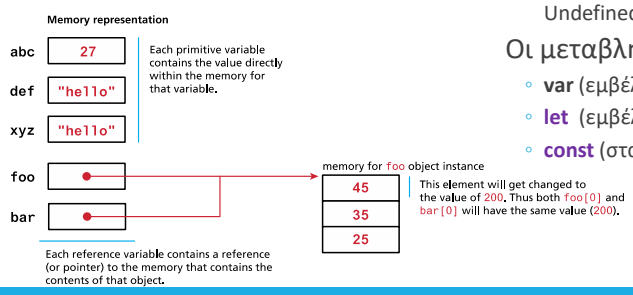
Μεταβλητές και Τύποι Δεδομένων

```
var abc = 27;
var def = "hello";
```
variables with primitive types

```
var foo = [45, 35, 25];
```
variable with reference type (i.e., array object)

```
var xyz = def;
var bar = foo;
```
these new variables differ in important ways (see below)

```
bar[0] = 200;
```
changes value of the first element of array



Δύο βασικοί τύποι:

- Τύποι αναφοράς (*type Object*)
  - Περιλαμβάνει τα arrays
- Πρωτογενείς τύποι (primitive)
  - Boolean, Number, String, Null, Undefined

Οι μεταβλητές δηλώνονται με:

- **var** (εμβέλεια συνάρτησης)
- **let** (εμβέλεια βρόχου)
- **const** (σταθερά με εμβέλεια βρόχου)

Ένα πρώτο script

first-js.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>First JS Example</title>
    <script src="script.js"></script>
  </head>
  <body>
  </body>
</html>
```

script.js

```
console.log('Hello, world!');
```

Εκτέλεση JavaScript

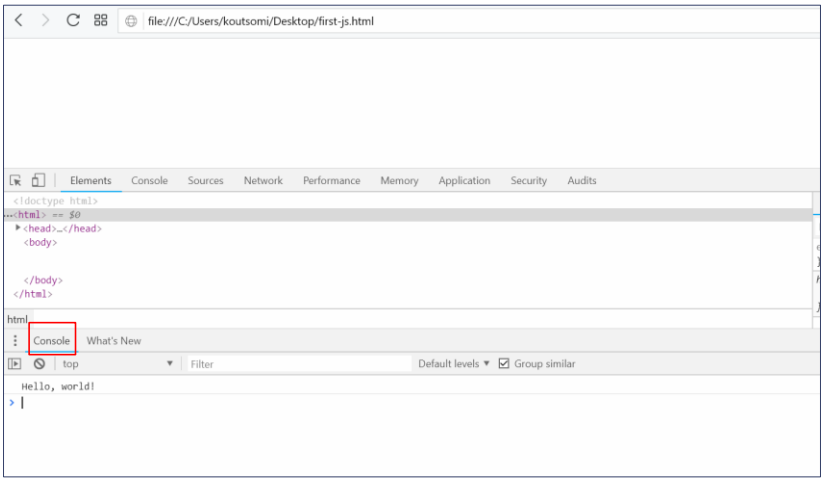
Δεν υπάρχει **μέθοδος main**

- Το script εκτελείται από πάνω προς τα κάτω

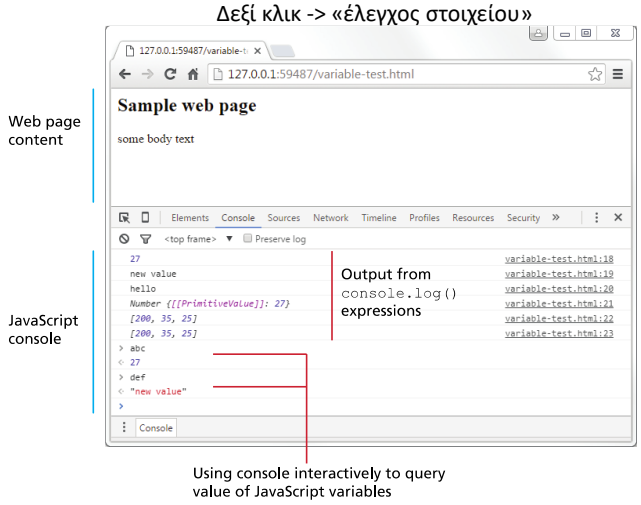
Δεν γίνεται κάτι compile από τον προγραμματιστή

- Η JavaScript μεταγλωττίζεται και εκτελείται δυναμικά από τον browser (Just In Time (JIT) Compilation)

Έξοδος στην κονσόλα



Κονσόλα JavaScript



Έξοδος

alert()
- εμφανίζει ένα pop-up μήνυμα

console.log()
- Εμφανίζει περιεχόμενο στην κονσόλα του browser

document.write()
- Εξάγει περιεχόμενο απευθείας μέσα στο HTML έγγραφο

Προγραμματιστικές δομές

for-loops:

```
for (let i = 0; i < 5; i++) { … }
```

while-loops:

```
while (notFinished) { … }
```

comments:

```
// comment or /* comment */
```

conditionals (if statements):

```
if (...) {
  ...
} else {
  ...
}
```

Σύνταξη παρόμοια με γλώσσες όπως Java/C++/C…

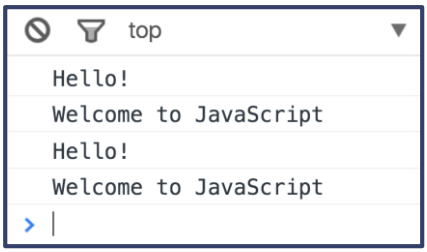
Συναρτήσεις

Δήλωση συνάρτησης (function declaration)

script.js

```
function hello() {
  console.log('Hello!');
  console.log('Welcome to JavaScript');
}

hello();
hello();
```



Console output

Hoisting

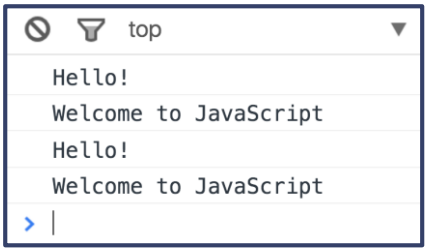
script.js

```
hello();
hello();

function hello() {
  console.log('Hello!');
  console.log('Welcome to JavaScript');
}
```

Λειτουργεί το παραπάνω;
Ναι, λόγω ***ανέλκυσης (hoisting)***
(Σαν να) μετακινείται ο ορισμός της συνάρτησης στην αρχή της εμβέλειας στην οποία δηλώνεται

- Μπορεί να αποφευχθεί
- Δεν είναι καλή πρακτική



Console output

Εμβέλεια μεταβλητών

Εμβέλεια συνάρτησης με το var

```
var x = 10;
if (x > 0) {
  var y = 10;
}
console.log('Value of y is ' + y);
```

```
Value of y is 10
>
```

- Μεταβλητές που έχουν δηλωθεί με "var" έχουν function-level scope και δεν βγαίνουν εκτός εμβέλειας μετά το τέλος του βρόχου. Μόνο στο τέλος της συνάρτησης
- Επομένως μπορούμε να αναφερθούμε στην ίδια μεταβλητή μετά το τέλος ενός βρόχου

Εμβέλεια συνάρτησης με το var

```
function meaningless() {
  var x = 10;
  if (x > 0) {
    var y = 10;
  }
  console.log('y is ' + y);
}
meaningless();
console.log('y is ' + y); // error!
```

```
y is 10
▶Uncaught ReferenceError: y is not defined
    at script.js:9
```

Όμως δεν μπορούμε να αναφερθούμε σε μια μεταβλητή εκτός της συνάρτησης στην οποία έχει δηλωθεί

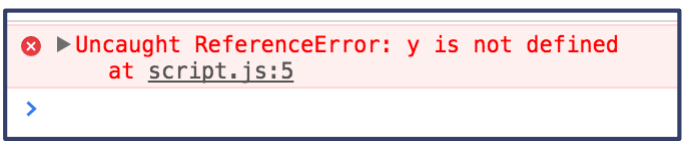
Εμβέλεια `let`

```
function printMessage(message, times) {
  for (let i = 0; i < times; i++) {
    console.log(message);
  }
  console.log('Value of i is ' + i);
}
printMessage('hello', 3);
```

`let` has block-scope so this results in an error

Εμβέλεια `const`

```
let x = 10;
if (x > 0) {
  const y = 10;
}
console.log(y);  // error!
```



Όπως το `let`, το `const` έχει επίσης block-scope, επομένως η πρόσβαση της μεταβλητής εκτός του βρόχου καταλήγει σε *σφάλμα*

Ανάθεση `const`

```
const y = 10;
y = 0;          // error!
y++;            // error!
const list = [1, 2, 3];
list.push(4);   // OK
```

`const` μεταβλητές δεν μπορούν να ανατεθούν εκ νέου (reassigned).

Όμως, το περιεχόμενό τους μπορεί να μεταβληθεί

- (In other words, it behaves like Java's `final` keyword and not C++'s `const` keyword)

Τύποι και συγκρίσεις

Null και Undefined

Ποια η διαφορά τους;

- Το `null` είναι μια τιμή που συμβολίζει την απουσία τιμής («κενή μεταβλητή», όπως στη Java)
- Το `undefined` δίνεται σε μια μεταβλητή που δεν τις έχει ακόμα δοθεί κάποια τιμή.

```
let x = null;
let y;
console.log(x);
console.log(y);
```

```
null
undefined
>
```


Αποτίμηση αληθείας

- Οι μη-boolean τιμές μπορούν να χρησιμοποιηθούν σε δομές ελέγχου και αποτιμώνται λογικά:
  - `null`, `undefined`, `0`, `NaN`, `''`, `""` αποτιμώνται σε `false`
  - Οτιδήποτε άλλο αποτιμάται σε `true`

```
if (username) {
  // username is defined
 }
```

Τελεστές σύγκρισης

JavaScript's == and != δεν λειτουργούν όπως αναμένεται: γίνεται πρώτα αποτίμηση τιμών, πριν τη σύγκριση.

```
'' == '0'  // false
 '' == 0  // true
 0 == '0'  // true
   NaN == NaN  // false
  [''] == ''  // true
false == undefined  // false
  false == null  // false
 null == undefined  // true
```


Τελεστές σύγκρισης

Αντί να διορθωθούν τα == και != , η ECMAScript κράτησε την υπάρχουσα συμπεριφορά και πρόσθεσε τα **===** και **!==**

```
'' === '0'  // false
 '' === 0  // false
 0 === '0'  // false
NaN == NaN  // still weirdly false
  [''] === ''  // false
false === undefined  // false
  false === null  // false
null === undefined  // false
```

Συνίσταται η χρήση των === και !== αντί των == και !=

Arrays και Objects

Arrays

Τα Arrays είναι τύποι Object και χρησιμοποιούνται για να δηλώσουμε λίστες δεδομένων

```
// Creates an empty list
let list = [];
let groceries = ['milk', 'cocoa puffs'];
groceries[1] = 'kix';
```

- Ξεκινούν από το 0 (0-based indexing)
- Mutable
- Μέγεθος μέσω ιδιότητας `length` (όχι συνάρτηση)
- .push()
- .pop()
- concat(), slice(), join(), reverse(), shift(), Και sort()

Επανάληψη σε πίνακα

Μπορεί να χρησιμοποιηθεί το γνωστό for-loop για επανάληψη σε μια λίστα:

```
let groceries = ['milk', 'cocoa puffs', 'tea'];
for (let i = 0; i < groceries.length; i++) {
  console.log(groceries[i]);
}
```

Ή ένα for-each loop μέσω `for...of`:
(intuition: for each item of the groceries list)

```
for (let item of groceries) {
      console.log(item);
}
```


Object Type

Το object είναι μια συλλογή από ζευγάρια key-value στοιχείων που ονομάζονται ιδιότητες (properties)

Τα objects σε αντίθεση με τις μεταβλητές μπορούν να διατηρούν περισσότερες από μια τιμές και μεθόδους

```
var obj = {};
obj = {name:”John”, lastname:”Doe”,
age:24}
var obj = {};
obj.name = ”John”;
obj.lastname = ”Doe”;
obj.age = 24;
obj[‘age’] =24;
```

Γνωστό και ως:
Java**S**cript **O**bject **N**otation (JSON)

Global object σε όλους τους browsers είναι το **window**.

Επανάληψη σε ένα object

Επανάληψη σε ένα object με χρήση for...in loop:
(for each key in the object)

```
for (key in object) {
      // … do something with object[key]
}
```

```
for (let name in scores) {
   console.log(name + ' got ' + scores[name]);
}
```

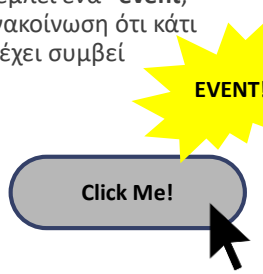
- Δεν μπορεί να χρησιμοποιηθεί for...in σε λίστες. Μόνο σε objects.
- Δεν μπορεί να χρησιμοποιηθεί for...of σε objects. Μόνο σε λίστες.

Events

Γεγονοστραφής προγραμματισμός

Η JavaScript στον φυλλομετρητή είναι κυρίως **event-driven**:
Ο κώδικας δεν τρέχει απευθείας, αλλά εκτελείται όταν συμβαίνει κάποιο γεγονός

Το κουμπί εκπέμπει ένα "**event**," δηλαδή μια ανακοίνωση ότι κάτι συγκεκριμένο έχει συμβεί (πατήθηκε).




```
function onClick() {
  ...
}
```

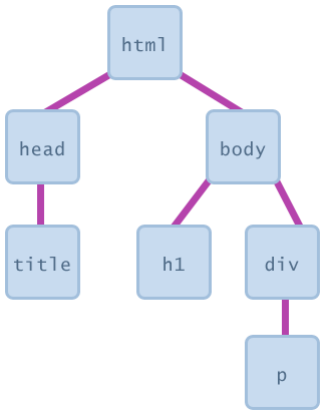
...Μια συνάρτηση που 'ακούει' για αυτό το event, εκτελείται. Η συνάρτηση αυτή ονομάζεται "**event handler**."

Πώς λοιπόν επικοινωνεί η JavaScript με τα στοιχεία της HTML;

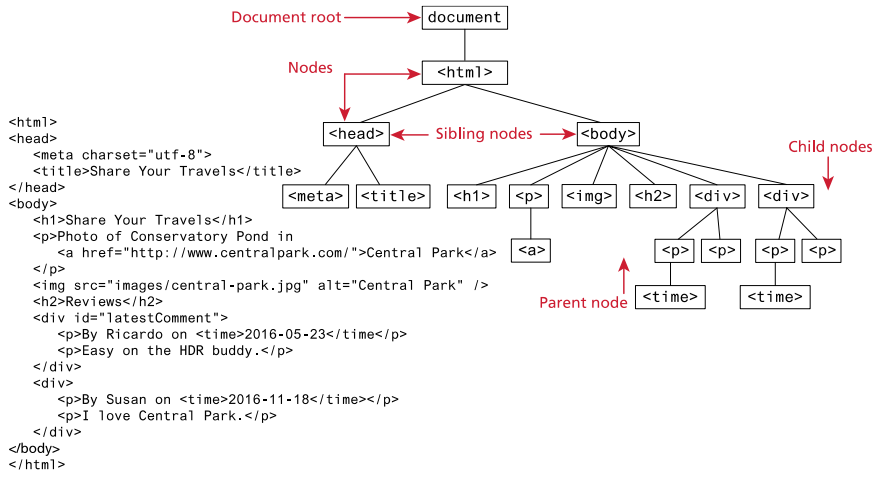
Το DOM

Κάθε στοιχείο της σελίδας είναι προσβάσιμο στην JavaScript μέσω του **DOM**: **Document Object Model**

- Το DOM είναι ένα δένδρο *κόμβων* που αντιστοιχούν στα HTML στοιχεία της σελίδας
- Μπορούμε να τροποποιήσουμε, προσθέσουμε και να αφαιρέσουμε κόμβους του DOM, κάτι που θα επηρεάσει αντιστοίχως τη σελίδα.

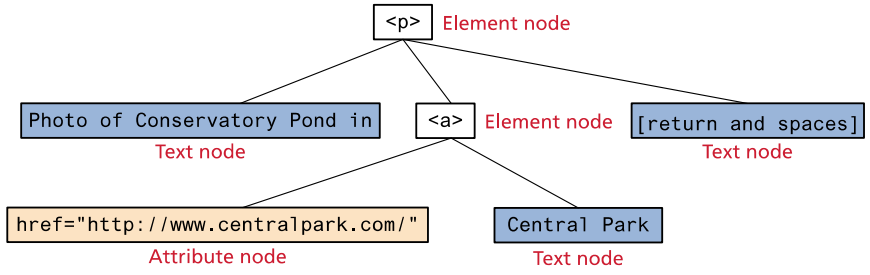


DOM Παράδειγμα



DOM Παράδειγμα

```
<p>Photo of Conservatory Pond in
   <a href="http://www.centralpark.com/">Central Park</a>
</p>
```



Προσπέλαση DOM αντικειμένων

Μπορούμε να προσπελάσουμε ένα DOM αντικείμενο που αντιστοιχεί σε ένα στοιχείο HTML μέσω της συνάρτησης querySelector:

```
document.querySelector('css selector');
```

- Επιστρέφει το **πρώτο** στοιχείο που ταιριάζει στον επιλογέα CSS

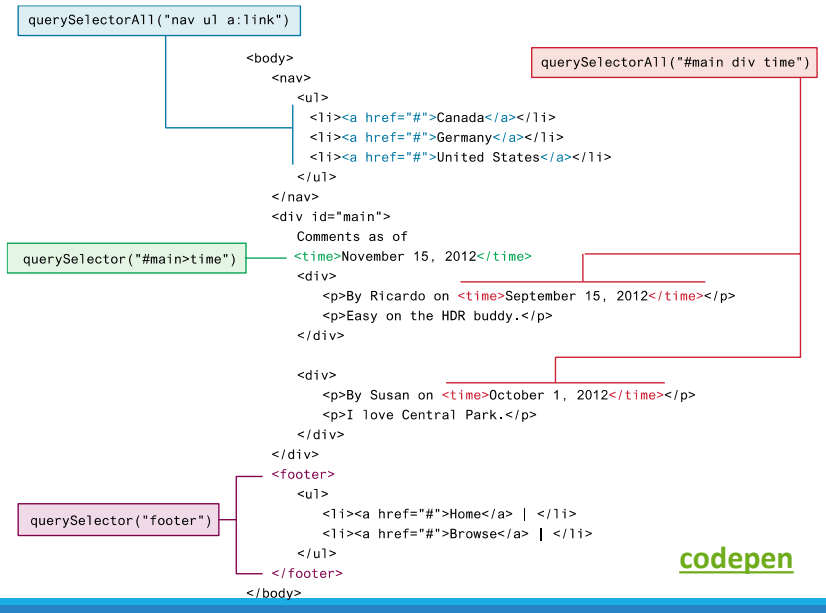
```
// Επιστρέφει το στοιχείο με id="button"
let element = document.querySelector('#button');
```

Άλλες συναρτήσεις επιλογής:

- querySelectorAll()
- getElementById()
- getElementsByTagName()
- getElementsByClassName()

Παλιότεροι τρόποι

QuerySelector



codepen

Προσάρτηση ακροατών (event listeners)

Κάθε DOM αντικείμενο έχει την συνάρτηση:

`addEventListener(`***event name, function name***`);`

- ***event name*** είναι το όνομα (string) του συμβάντος που θέλουμε να ακούσουμε
  - Συνήθως: click, focus, blur, mouseover,…
- ***function name*** είναι το όνομα της συνάρτησης που θέλουμε να εκτελεστεί όταν πυροδοτηθεί το event

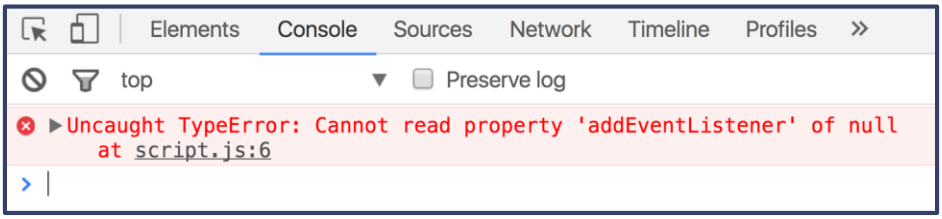

```
<html>
▼<head>
    <meta charset="utf-8">
    <title>First JS Example</title>
    <script src="script.js"></script>
  </head>
▼<body>
    <button>Click Me!</button>
  </body>
</html>
```

```
function onClick() {
  console.log('clicked');
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```

script.js

```
1  function onClick() {
2    console.log('clicked');
3  }
4
5  const button = document.querySelector('button');
6  button.addEventListener('click', onClick);
7
```



Δημιουργείται σφάλμα γιατί το script φορτώνεται και εκτελείται πριν φορτωθεί η σελίδα

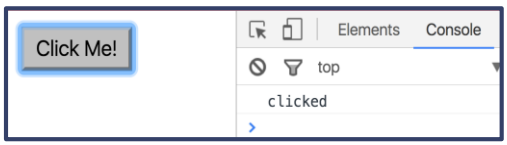
Χρήση defer

Μπορούμε να προσθέσουμε το χαρ/κό `defer` στην ετικέτα script, ώστε η JavaScript να μην εκτελείται μέχρι να φορτωθεί το DOM:

```
<html>
▼<head>
    <meta charset="utf-8">
    <title>First JS Example</title>
    <script src="script.js" defer></script>
  </head>
▼<body>
    <button>Click Me!</button>
  </body>
</html>
```

```
function onClick() {
  console.log('clicked');
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```




```
<script src="script.js" defer></script>
```

Αλλαγή στυλ σε element

```
<style>
    .box {
        margin: 2em; padding: 0;
        border: solid 1pt black;
    }
   .yellowish { background-color: #EFE63F; }
   .hide { display: none; }
</style>
<main>
   <div class="box">
    ...
   </div>
</main>
```

```
var node = document.querySelector("main div");

1 node.className = "yellowish";
    This replaces the existing class specification with this one. Thus the <div> no longer has the box class

2 node.classList.remove("yellowish");
  node.classList.add("box");
    Removes the specified class specification and adds the box class

3 node.classList.add("yellowish");
    Adds a new class to the existing class specification

4 node.classList.toggle("hide");
    If it isn't in the class specification, then add it

5 node.classList.toggle("hide");
    If it is in the class specification, then remove it
```

Equivalent to:

```
1 <div class="yellowish">

2 <div class="">
  <div class="box">

3 <div class="box yellowish">

4 <div class="box yellowish hide">

5 <div class="box yellowish">
```


Αλλαγή του περιεχομένου σε ένα element

```
document.getElementById("here").innerHTML =
"foo<em>bar</em>";
```

Τροποποίηση attributes

- Κάθε DOM object έχει τα attributes του HTML στοιχείου ως properties:

HTML

```
<img src= "cat.png" />
```

JavaScript

```
const element = document.querySelector('img');
element.src = 'tiger.png';
```

Δημιουργία DOM elements

1. Create a new text node

```
var text = document.createTextNode("this is dynamic");
```

"this is dynamic"

2. Create a new empty element

```
var p = document.createElement("p");
```

3. Add the text node to new element

```
p.appendChild(text);
```

 "this is dynamic"

4. Add the element to the

```
var first = document.getElementById("first");
first.appendChild(p);
```


Δημιουργία DOM elements

4. Add the element to the

```
var first = document.getElementById("first");
first.appendChild(p);
```

```
<div id="first">
    <h1>DOM Example</h1>
    <p>Existing element</p>
    <p>this is dynamic</p>
</div>
```

```
<div>
    <h1> "DOM Example" </h1>
    <p> "Existing element" </p>
    <p> "this is dynamic" </p>
</div>
```

codepen