

# Προγραμματισμός Διαδικτύου



*JavaScript*



Χ. Γεωργιάδης

geor@uom.edu.gr



Θ.Χ. Κασκάλης

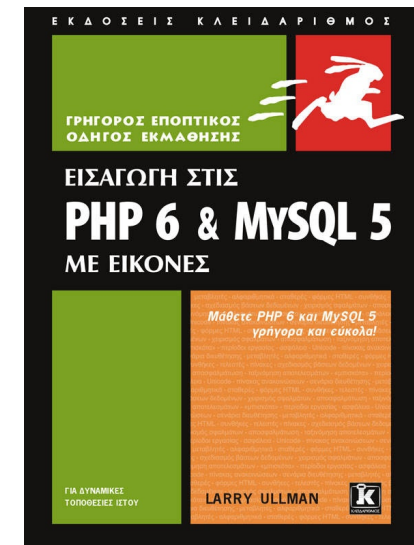
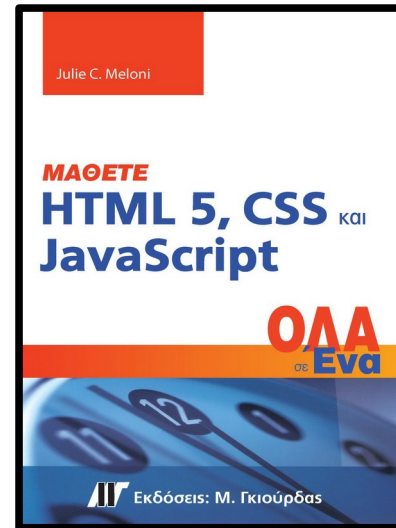
kaskalis@uom.edu.gr

kaskalis@gmail.com



# Εγχειρίδια μαθήματος

- Μάθετε HTML 5 , CSS και JavaScript. Όλα σε ένα
- Εισαγωγή στις PHP 6 και MySQL 5 με Εικόνες





# Τρόπος βαθμολόγησης

- Τελική εργαστηριακή εξέταση



# Περιεχόμενο πρώτων διαλέξεων

- Θα ασχοληθούμε με τις τεχνολογίες του “πελάτη” στον ιστό (client-side technologies):
  - HTML
  - CSS
  - JavaScript
- Άρα... πρέπει να καταλάβουμε την ορολογία!



# Προγράμματα πλοήγησης (browsers)

- Προγράμματα “πελάτες” που απεικονίζουν ιστοσελίδες και “τρέχουν” web εφαρμογές



Mozilla  
Firefox



Google  
Chrome



Microsoft  
Edge



Apple  
Safari

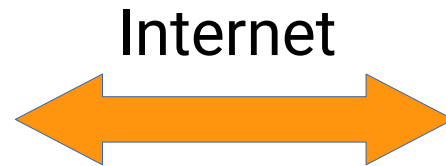


# Τρόπος επικοινωνίας

- Servers (διακομιστές):
  - προγράμματα που «ακούνε» για μηνύματα και εκτελούν διαδικασίες
- Clients (πελάτες):
  - προγράμματα που «συνδέονται» στους servers



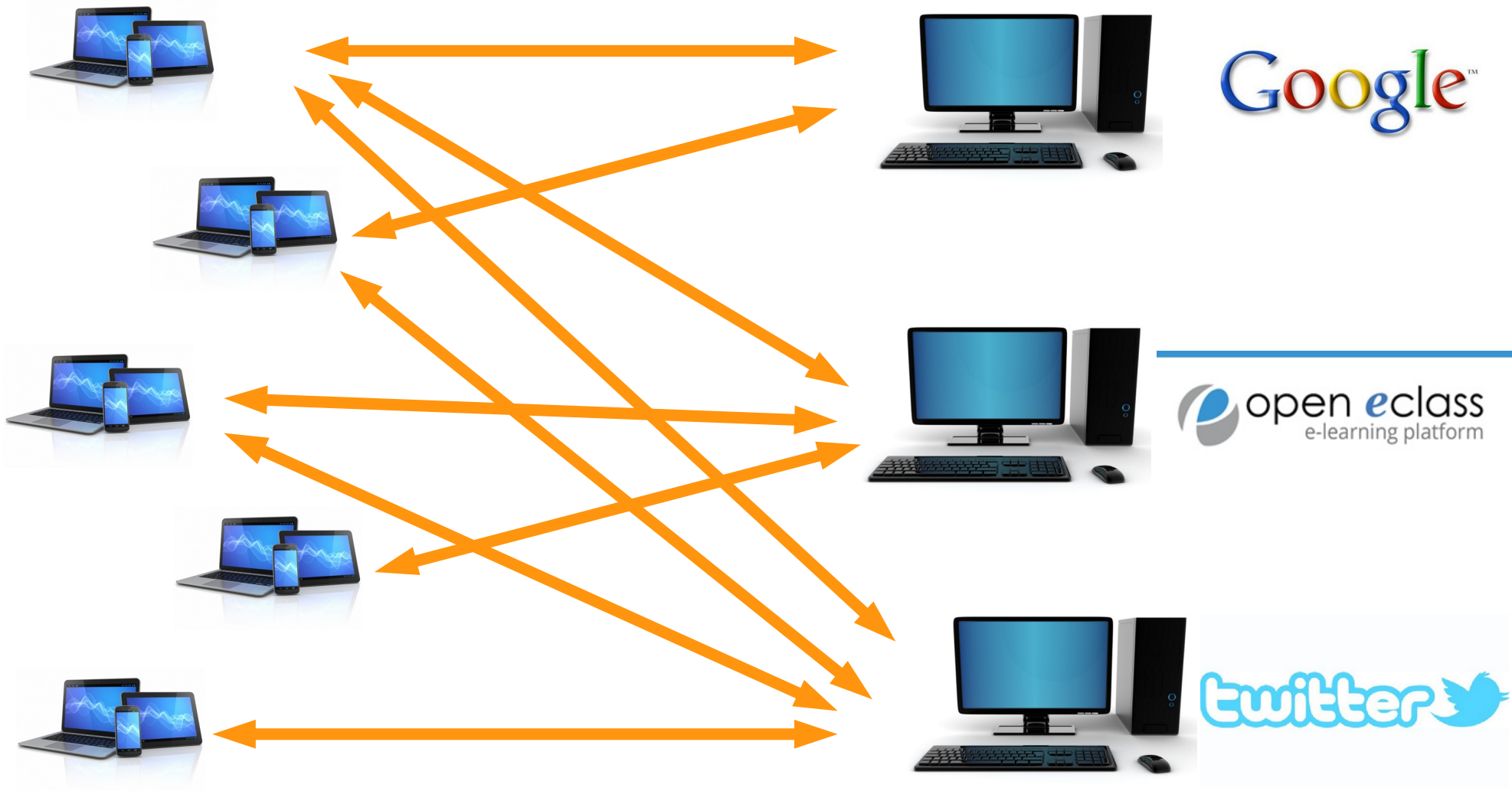
Client



Server

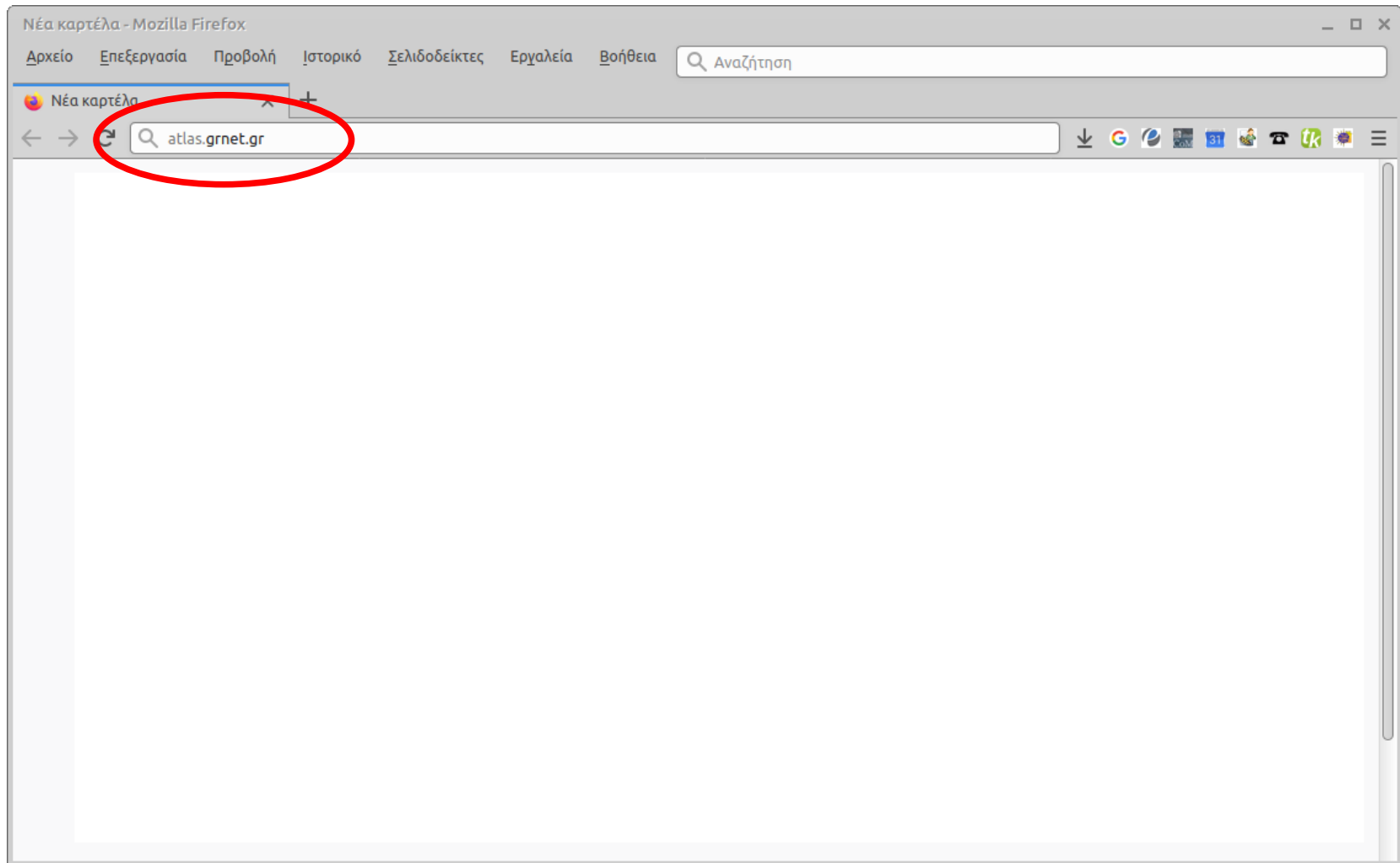


# Πολλοί servers – Πολλοί clients





# Διαδικασία σύνδεσης







# Διαδικασία σύνδεσης

- Ο client ζητά τη μετατροπή της διεύθυνσης atlas.grnet.gr σε διεύθυνση IP και αποστέλλει ένα μήνυμα μέσω πρωτοκόλλου HTTP στη “θύρα” 80 του server ζητώντας του “ένα αρχείο”:

`http://83.212.4.114:80`



# Διαδικασία σύνδεσης

- Ο server δέχεται την αίτηση και, μια που ο client δεν του έχει ζητήσει ρητά κάποιο αρχείο, εκείνος είναι έτσι ρυθμισμένος ώστε να αναζητήσει σε έναν προκαθορισμένο φάκελο ένα προκαθορισμένο αρχείο
  - π.χ. `index.html`, `index.php`, `Default.aspx`

# Δομή αρχείων και καταλόγων (server)



- Ένα αρχείο ρυθμίσεων καθορίζει στον server ποιοι φάκελοι μπορούν να προσπελαστούν από clients και ποια είναι τα προεπιλεγμένα αρχεία που θα στείλουν, αν ο client δεν δηλώνει ρητά επιθυμητό αρχείο.



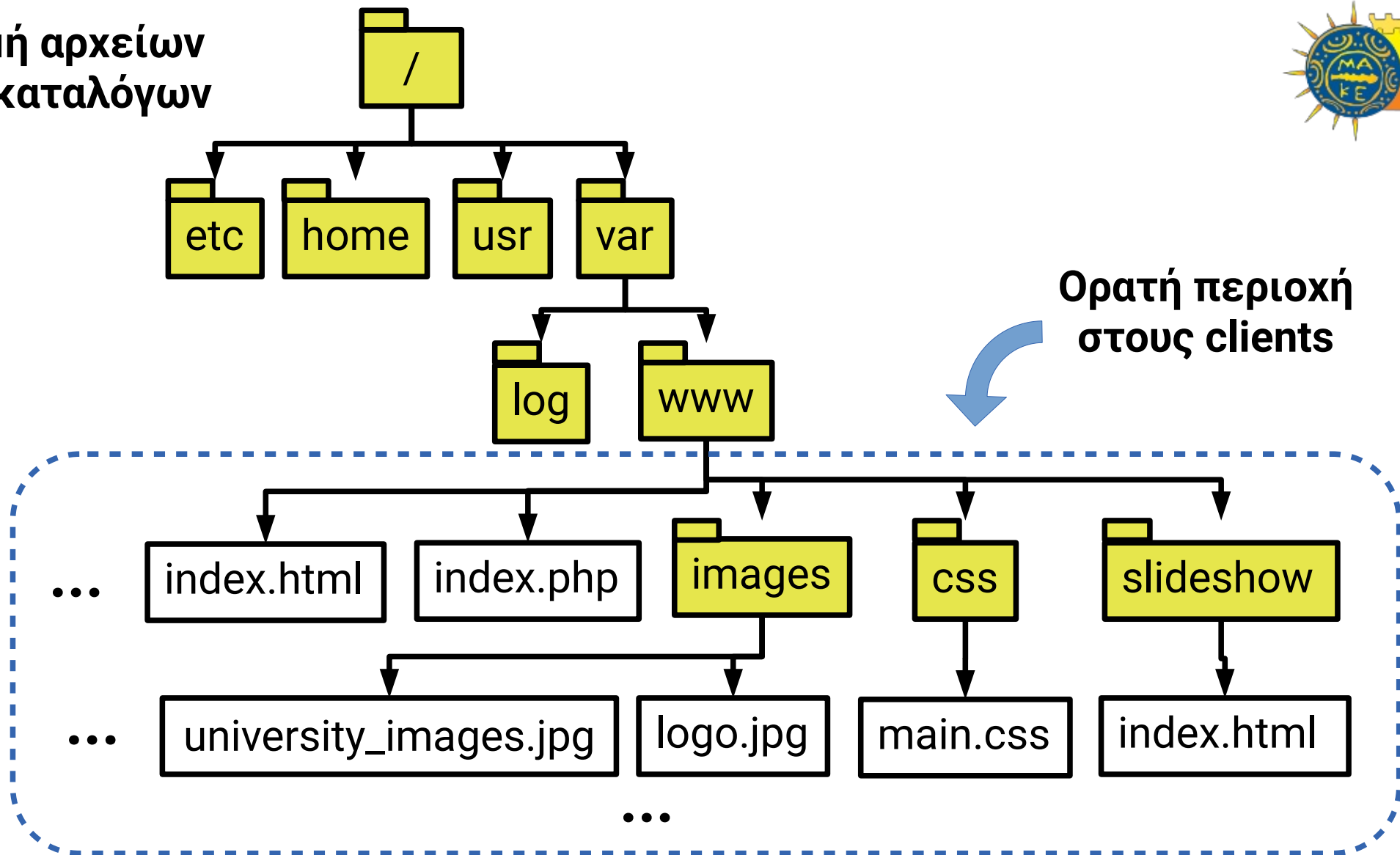
HTTP server

Αρχείο ρυθμίσεων



```
...  
ServerRoot "/var/www"  
...  
DirectoryIndex index.html  
...
```

# Δομή αρχείων και καταλόγων



Π.χ. ζητώντας ο client το αρχείο  
***http://www.κάποιοςserver.com/images/university\_images.jpg***  
στην πραγματικότητα κατεβάζει το αρχείο  
***/var/www/images/university\_images.jpg***  
της πραγματικής δομής αρχείων και καταλόγων



# Έμφαση στις υπηρεσίες πελάτη !

- Άρα οποιοσδήποτε μπορεί να φτιάξει μια δομή φακέλων και αρχείων HTML-CSS-JavaScript, η οποία μπορεί αργότερα να δημοσιευθεί σε έναν ιστότοπο, μέσω ενός server.
- Άρα εμείς μπορούμε να δουλεύουμε τοπικά!



# Δημοσίευση ιστοτόπων ;

- Υπάρχουν δωρεάν αλλά και (βεβαίως) επί πληρωμή λύσεις.
- Λίστα με αξιόλογες δωρεάν λύσεις βρίσκεται στην περιοχή “Σύνδεσμοι” του Open eClass.



# Ιστοσελίδα ή περιεχόμενο ;

- Συχνά χρησιμοποιείται ο όρος “περιεχόμενο” παρά “ιστοσελίδα”.
- Μια ιστοσελίδα (=αρχείο) αναφέρεται σε (=περιέχει) και άλλα αρχεία που πρέπει να μεταφερθούν (π.χ. αρχεία εικόνων, css, javascript, video κ.λπ.)

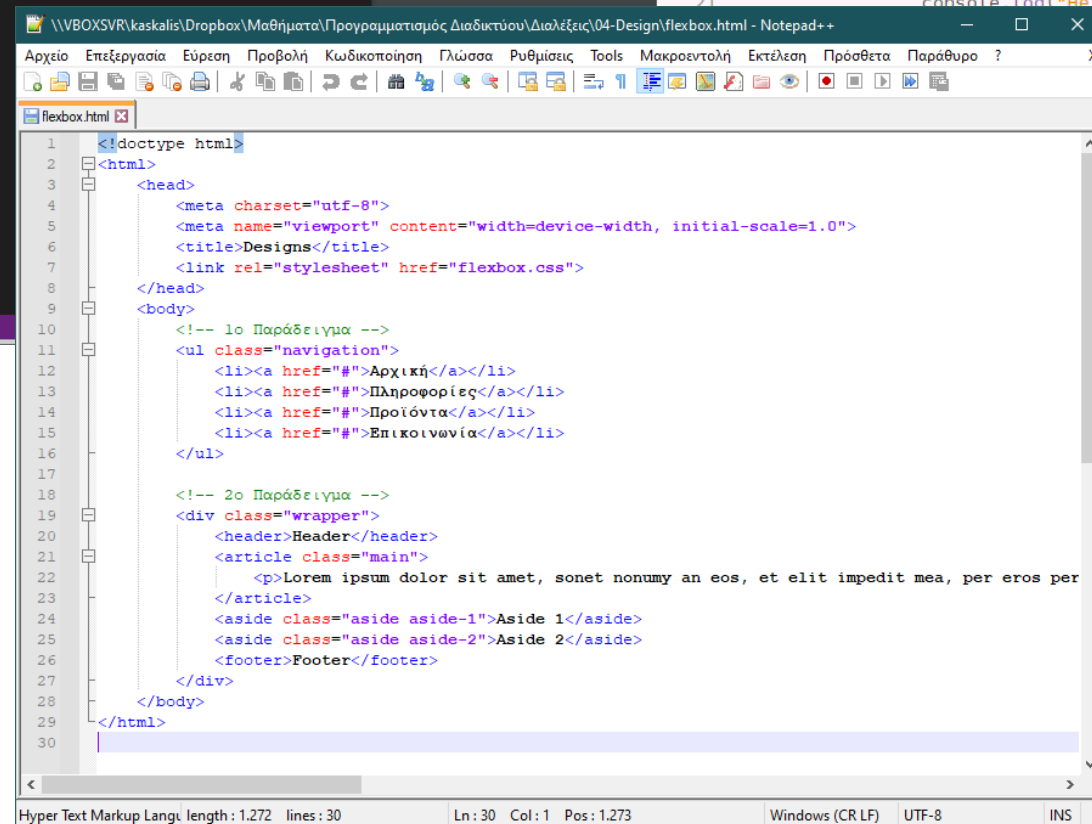
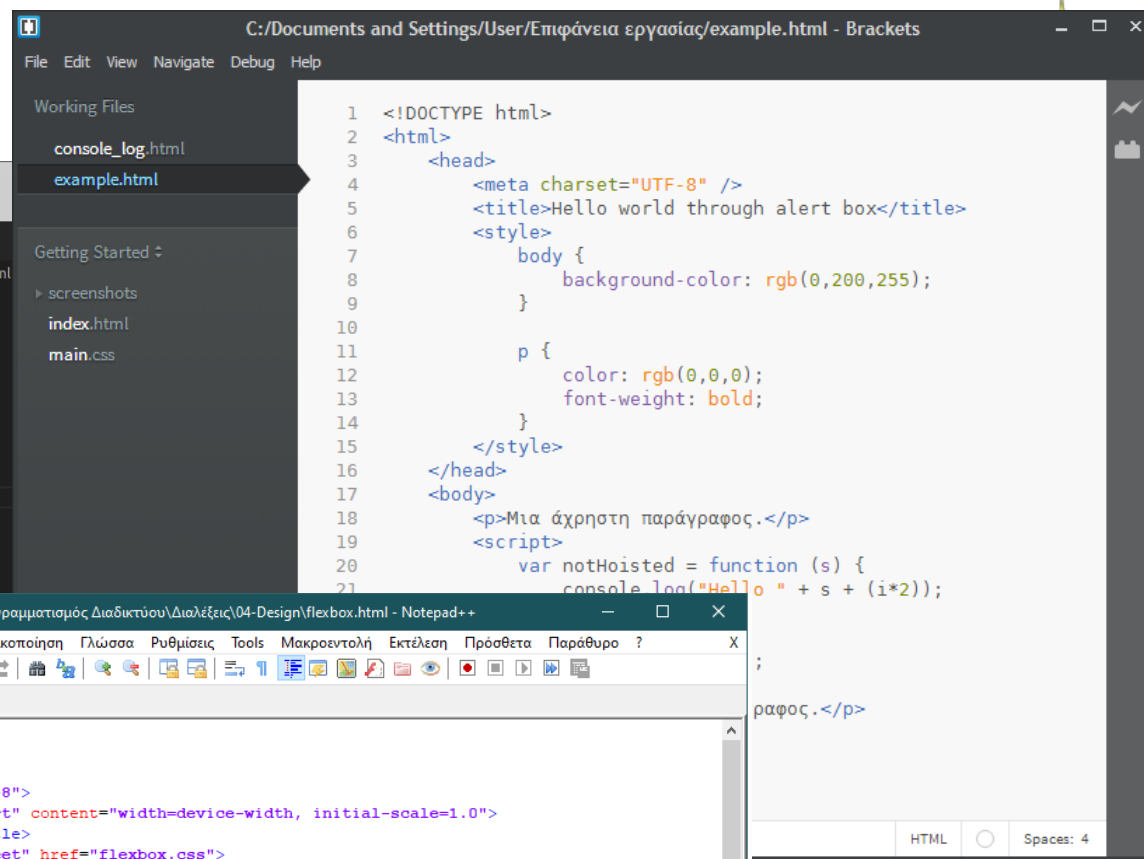
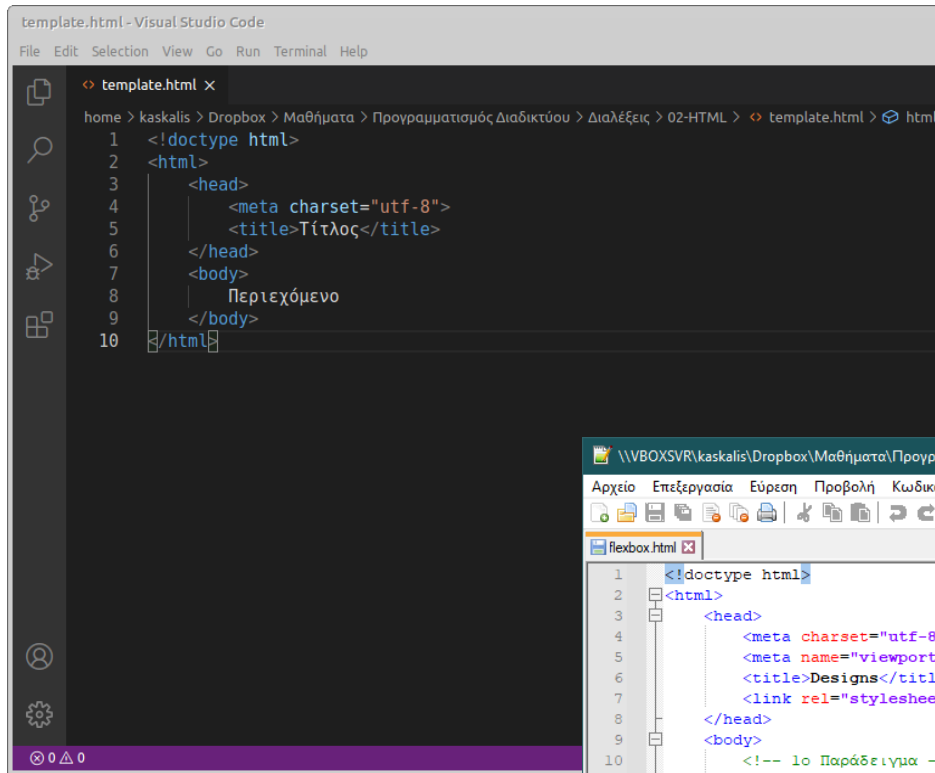


# Εργαλεία γραφής κώδικα...

- Οποιοσδήποτε κειμενογράφος (text editor).
- Προτεινόμενες λύσεις:
  - [Visual Studio Code](https://code.visualstudio.com): <https://code.visualstudio.com>
  - [Atom](https://atom.io): <https://atom.io>
  - [Sublime Text](https://www.sublimetext.com): <https://www.sublimetext.com>
  - [Brackets](http://brackets.io): <http://brackets.io>
  - [Notepad++](https://notepad-plus-plus.org): <https://notepad-plus-plus.org>    :-( windows only )-:
- Θα μάθουμε σύντομα ότι έχουν σημασία:
  - Οι επεκτάσεις αρχείων (.html, .css, .js)
  - Η κωδικοποίηση χαρακτήρων (utf-8)

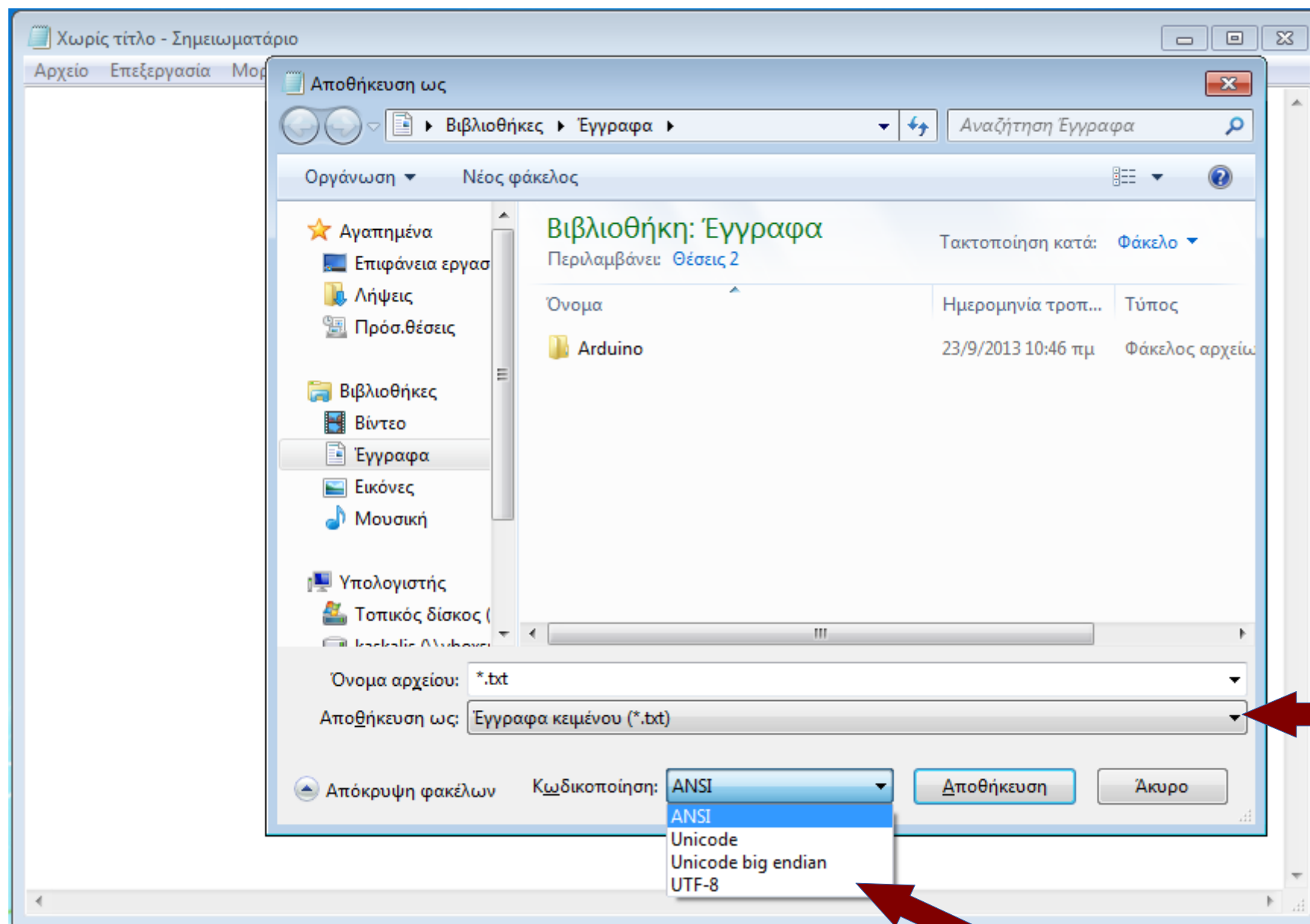


# Visual Studio Code



# Brackets

# Notepad++





# HTML 5 !!

- Θα ασχοληθούμε αποκλειστικά με το πρότυπο:



- Όλα τα προηγούμενα πρότυπα (HTML4, XHTML κ.λπ.) ισχύουν, αλλά ΔΕΝ θα μας απασχολήσουν!



# Βασικές διευθύνσεις

- <https://html.spec.whatwg.org/>
  - HTML 5 Specification
- <https://www.w3.org/Style/CSS/specs.en.html>
  - CSS specifications
- <http://www.ecma-international.org/publications/standards/Ecma-262.htm>
  - ECMAScript (JavaScript) Definition



# Ενδιαφέροντα links

- <http://www.html5rocks.com>
- <http://www.codecademy.com>
- <http://www.lynda.com>
- <http://www.udemy.com>
- <http://htmldog.com>
- <http://html5doctor.com>
- <http://www.w3.org>
- <http://www.w3schools.com>
- <http://help.dottoro.com>
- <http://htmldog.com>
- <http://www.javascriptkit.com>





# Πρότυπο (ελάχιστο) αρχείο html

“template.html”

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Τίτλος</title>
  </head>
  <body>
  </body>
</html>
```

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Τίτλος</title>
  </head>
  <body>
  </body>
</html>
```



# Βασικά δομικά στοιχεία HTML 5

- `<!doctype html>`
  - Απαραίτητη έναρξη αρχείων HTML
- ΠΡΟΣΟΧΗ: Οι browsers είναι πολύ ανεκτικοί σε παραλείψεις. Αυτό δεν σημαίνει ότι εμείς θα τις κάνουμε...



# Βασικά δομικά στοιχεία HTML 5

- Elements 2 ειδών:

1. Με περιεχόμενο (closing elements):

`<element> </element>`

2. Κενές (self-closing elements):

`<element>`

- Προσοχή στην ένθεση ετικετών !





# Element `<html>`

- Περιέχει ολόκληρη την ιστοσελίδα μας
- 

# Element `<head>`

- Περιέχει πληροφορίες και δεδομένα που αφορούν τη σελίδα, αλλά δεν θα αποτελέσουν μέρος της απεικόνισης στο παράθυρο του browser.



# Element <title>

- Βασικότατο!
- Εμφανίζεται:
  - Στη γραμμή τίτλου του browser
  - Στους σελιδοδείκτες και στο ιστορικό του χρήστη
- Έχει μεγάλη σημασία στις μηχανές αναζήτησης.
- Άρα πρέπει να συντάσσεται σωστά!



# Element `<body>`

- Το “σώμα” της ιστοσελίδας, που περιέχει οτιδήποτε πρόκειται να κάνει render (=απεικόνιση) στο παράθυρό του ο browser.
- Η δόμηση του body είναι **λογική**. Δηλαδή περιέχει ετικέτες που περιγράφουν τη δομή της σελίδας.
- Η εμφάνιση είναι θέμα των CSS.



# Κωδικοποιήσεις ιστοσελίδων: ASCII, ISO-8859-7, UTF-8

- Το πρότυπο Unicode (UTF-8) είναι το πλέον καθολικά χρησιμοποιούμενο.
- Αυτή θα είναι η μοναδική μας μέθοδος κωδικοποίησης!



# Element <meta>

- Element μεταδεδομένων (metadata):

`<meta charset="utf-8">`

- Χρησιμοποιείται και για άλλες πληροφορίες (π.χ. λέξεις κλειδιά σελίδας), αλλά προς το παρόν θα μας χρησιμεύσει μόνον για την κωδικοποίηση των αρχείων HTML 5.



# Τα πρώτα μας δομικά elements

- Elements:

`p, h1...h6, ul, ol, li`

- Paragraph, Header, Unordered List, Ordered List, List Item.



<p> ... </p>

- Παράγραφος
- Δομικό element κειμένου
- Το default rendering αφήνει ένα περιθώριο (margin) κάτω από την παράγραφο
- Γενική παρατήρηση: τα διαδοχικά κενά και/ή τα enter αντιμετωπίζονται ως **ένα** απλό κενό.



`<h1> ... </h1> έως <h6> ... </h6>`

- Επικεφαλίδες
- Default rendering:







# <ul> <li>

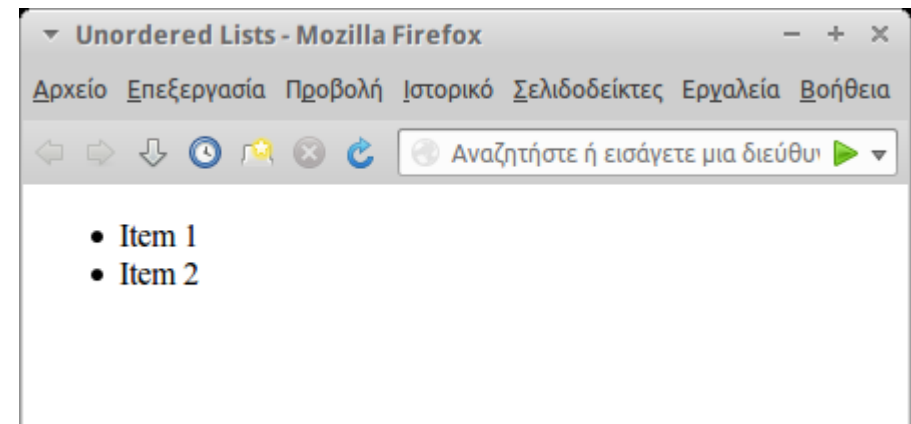
- Μη αριθμημένες λίστες
- Σύνταξη (ένθεση):

<ul>

<li>Item 1</li>

<li>Item 2</li>

</ul>





# <ol> <li>

- Αριθμημένες λίστες
- Σύνταξη (ένθεση):

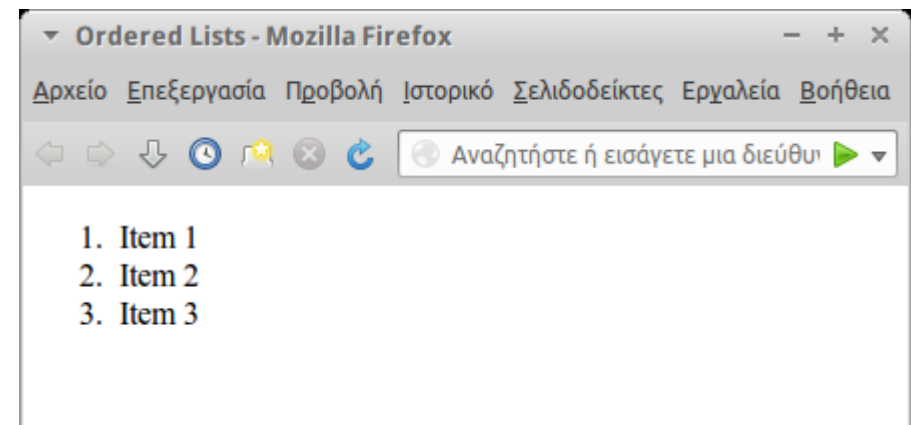
## <ol>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

## </ol>





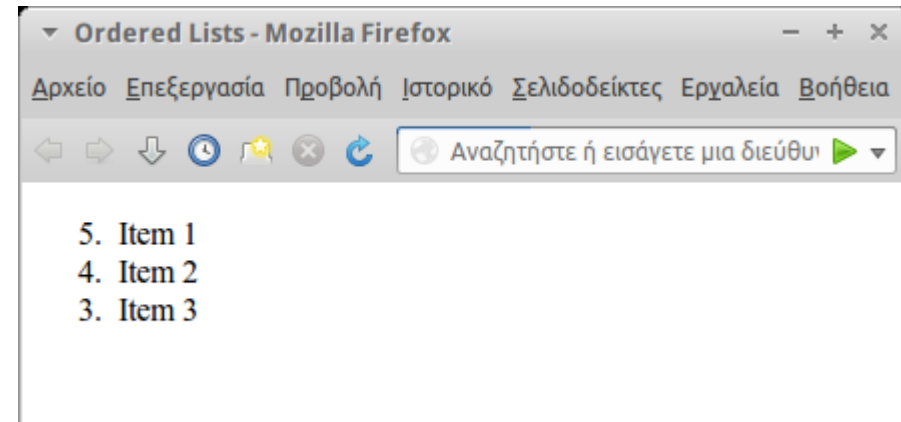
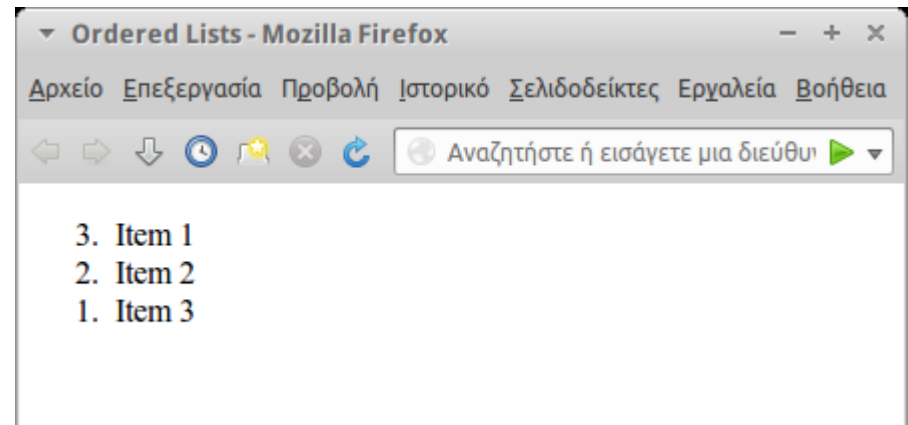
# Χαρακτηριστικά (attributes)

- Τα elements έχουν κατά περίπτωση χαρακτηριστικά (attributes) που δηλώνονται μέσα στην ετικέτα ανοίγματος:

`<ol reversed>`

`<ol start="5">`

`<ol start="5" reversed>`





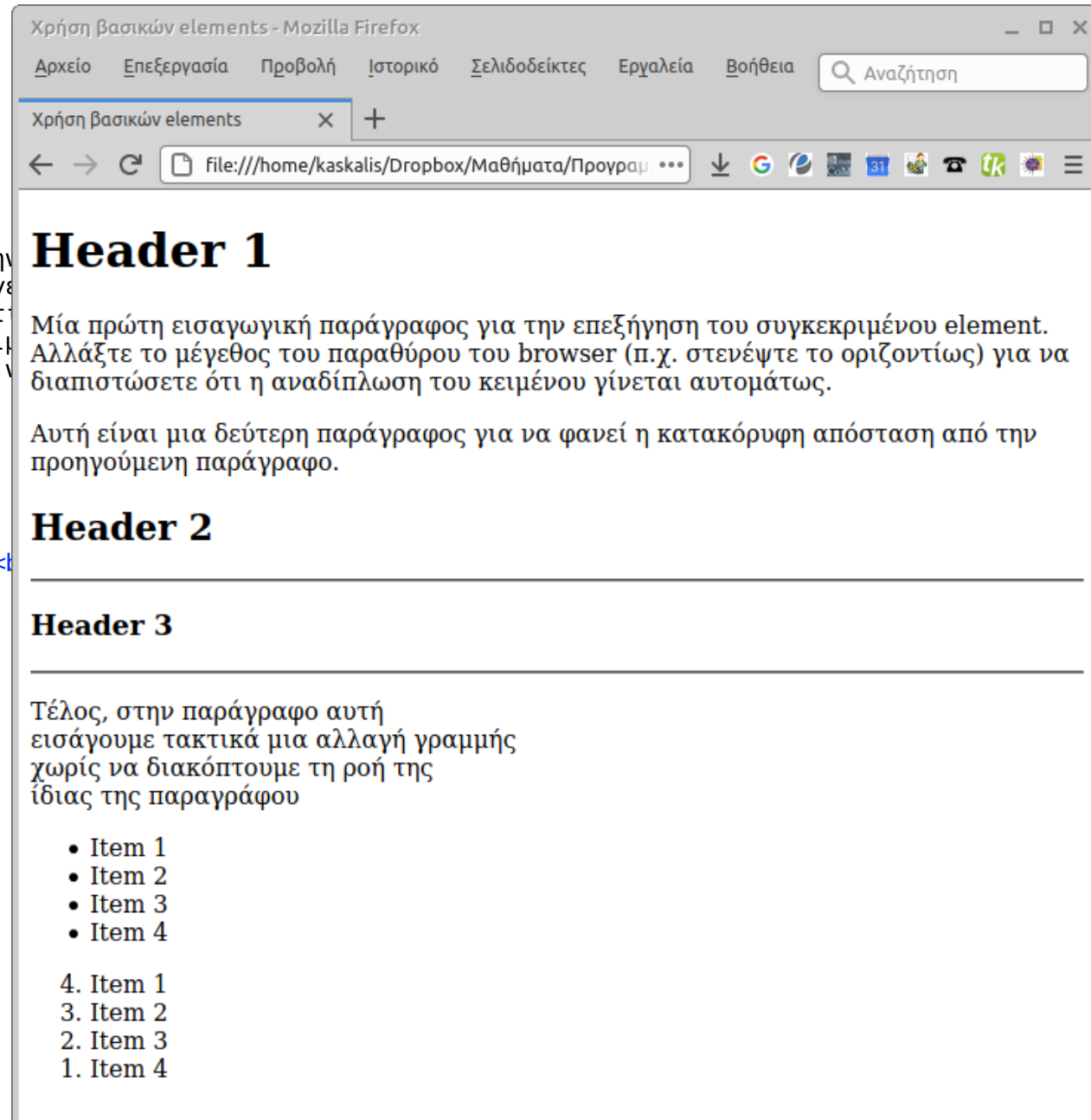
# <hr> και <br>

- Self-closing elements
- hr : Οριζόντια γραμμή (horizontal ruler)
- br : Αλλαγή γραμμής (break)

# Συνολικό παράδειγμα



```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Χρήση βασικών elements</title>
  </head>
  <body>
    <h1>Header 1</h1>
    <p>Μία πρώτη εισαγωγική παράγραφος για την
      συγκεκριμένου element. Αλλάξτε το μέγεθος
      του browser (π.χ. στενέψτε το οριζοντίως) για να
      διαπιστώσετε ότι η αναδίπλωση του κειμένου
      γίνεται αυτομάτως.</p>
    <p>Αυτή είναι μια δεύτερη παράγραφος για να
      φανεί η κατακόρυφη απόσταση από την
      προηγούμενη παράγραφο.</p>
    <h2>Header 2</h2>
    <hr>
    <h3>Header 3</h3>
    <hr>
    <p>Τέλος, στην παράγραφο αυτή<br>
      εισάγουμε τακτικά μια αλλαγή γραμμής<br>
      χωρίς να διακόπτουμε τη ροή της<br>
      ίδιας της παραγράφου.</p>
    <ul>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
      <li>Item 4</li>
    </ul>
    <ol reversed>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
      <li>Item 4</li>
    </ol>
  </body>
</html>
```



# Μελέτη...



<http://html.spec.whatwg.org/multipage/>

[diveintohtml5.info](http://diveintohtml5.info)

[webplatform.github.io/docs/html/tutorials/](http://webplatform.github.io/docs/html/tutorials/)





# HTML ως elements με attributes

- με content:  
`<elem>...</elem>`
- void:  
`<elem>`
- attributes:
  - ως ζεύγος name-value:  
`<elem name="value">...`
  - boolean:  
`<elem name>...`



# void elements

- br, embed, hr, img, input, link, meta, source, track

- Παραδείγματα:

```

```

```
<input type="checkbox" checked>
```



# Element list



([developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5/HTML5\\_element\\_list](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5/HTML5_element_list))

- Root: `<html>`
- Document metadata: `<head>`, `<title>`, `<link>`, `<meta>`, `<style>`
- Scripting: `<script>`
- Sections: `<body>`, `<h1>` – `<h6>`, `<section>`, `<nav>`, `<article>`, `<aside>`, `<header>`, `<footer>`, `<address>`, `<main>`
- Grouping: `<div>`, `<p>`, `<hr>`, `<pre>`, [ `<ol>`, `<ul>`, `<li>` ], [ `<dl>`, `<dt>`, `<dd>` ], [ `<figure>`, `<figcaption>` ]

# Element list



([developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5/HTML5\\_element\\_list](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5/HTML5_element_list))

- Text-level semantics: `<a>`, `<br>`, `<em>`, `<strong>`, `<sub>`, `<sup>`, `<span>`, `<code>`
- Embedded content: `<img>`, `<iframe>`, `<embed>`,  
[ `<video>`, `<audio>`, `<source>` ], `<canvas>`, `<svg>`
- Tabular data: `<table>`, [ `<thead>`, `<tbody>`, `<tfoot>` ],  
[ `<colgroup>`, `<col>` ], [ `<tr>`, `<th>`, `<td>` ],  
`<caption>`
- Forms: `<form>`, [ `<fieldset>`, `<legend>` ], `<label>`,  
`<input>`, `<datalist>`, `<button>`, [ `<select>`,  
`<optgroup>`, `<option>` ], `<textarea>`, `<progress>`,  
`<meter>`



# Default rendering

- Για πολλά από τα elements οι browsers χρησιμοποιούν κάποιους default κανόνες rendering, που βεβαίως μπορούν να τροποποιηθούν εκ των υστέρων.
- Π.χ. μεγέθη γραμμάτων, περιθώρια (box model), block / inline (display).

# Block level elements (flow content)



- Εμφανίζονται με αλλαγή γραμμής πριν και μετά, δημιουργώντας ένα αυτόνομο τμήμα περιεχομένου.
- Ουσιαστικά καταλαμβάνουν το πλήρες πλάτος των υποδοχέων τους (containers).

# Inline level elements (phrasing content)



- Ακολουθούν τη ροή του περιεχομένου χωρίς αλλαγή γραμμών.
- Έχουν τόσο πλάτος όσο είναι τα περιεχόμενά τους.

# Εμφώλευση



- Ένα block-level element μπορεί να περιέχει άλλα block-level elements ή inline elements ή data.
- Ένα inline element μπορεί να περιέχει μόνον άλλα inline elements ή data.



- Inline elements: `<a>`, `<audio>`, `<br>`, `<canvas>`, `<code>`, `<em>`, `<strong>`, `<iframe>`, `<img>`, `<script>`, `<span>`, `<sub>`, `<sup>`, `<svg>`, `<video>`, `<button>`, `<input>`, `<label>`, `<meter>`, `<progress>`, `<select>`, `<datalist>`, `<textarea>`



# Ειδικοί χαρακτήρες

Χαρακτ.	Όνομα	Περιγραφή
<	&lt;	less-than
>	&gt;	greater-than
"	&quot;	quotation mark
'	&apos;	apostrophe
&	&amp;	ampersand
©	&copy;	copyright
®	&reg;	registered trademark
™	&trade;	trademark

Χαρακτ.	Όνομα	Περιγραφή
×	&times;	multiplication
÷	&divide;	division
‰	&permil;	per mille
€	&euro;	Euro
§	&sect;	section
°	&deg;	degree
±	&plusmn;	plus-or-minus
	&nbsp;	non-breaking space



# Global attributes

([www.w3.org/TR/html/dom.html#global-attributes](http://www.w3.org/TR/html/dom.html#global-attributes))



- `id`
- `class`
- `style`
- `lang`
- `title`
- ...



# Global attributes

- Event-handler attributes:
  - `onchange`, `onclick`, `onloadstart`,  
`onmousedown`, `onmousemove`, `onmouseout`,  
`onmouseover`, `onmouseup` κ.λπ.
- Δέχονται εντολές JavaScript που τις εκτελούν ως `function` (θα το αποφύγουμε)

# Άλλα attributes

([developer.mozilla.org/en-US/docs/Web/HTML/Attributes](https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes))



- Εξαρτώνται από το εκάστοτε element.

W3C Working Group Note

## HTML: The Markup Language (an HTML language reference)

« elements abbr »

**a** – hyperlink **CHANGED**

The **a** element represents a hyperlink.

### Permitted contents

[Transparent](#) (either phrasing content or flow content ).

### Permitted attributes #

[global attributes](#) & [href](#) & [target](#) & [rel](#) & [hreflang](#) & [media](#) & [type](#)

---

- global attributes**  
Any attributes permitted globally.
- href** = [URL potentially surrounded by spaces](#)  
A URL that provides the destination of the hyperlink. If the [href](#) attribute is not specified, the element represents a **placeholder hyperlink**.
- target** = [browsing-context name or keyword](#) **CHANGED**  
A name or keyword giving a [browsing context](#) for UAs to use when following the

jump

# Περιπτώσιολογία: `<input type="..."`

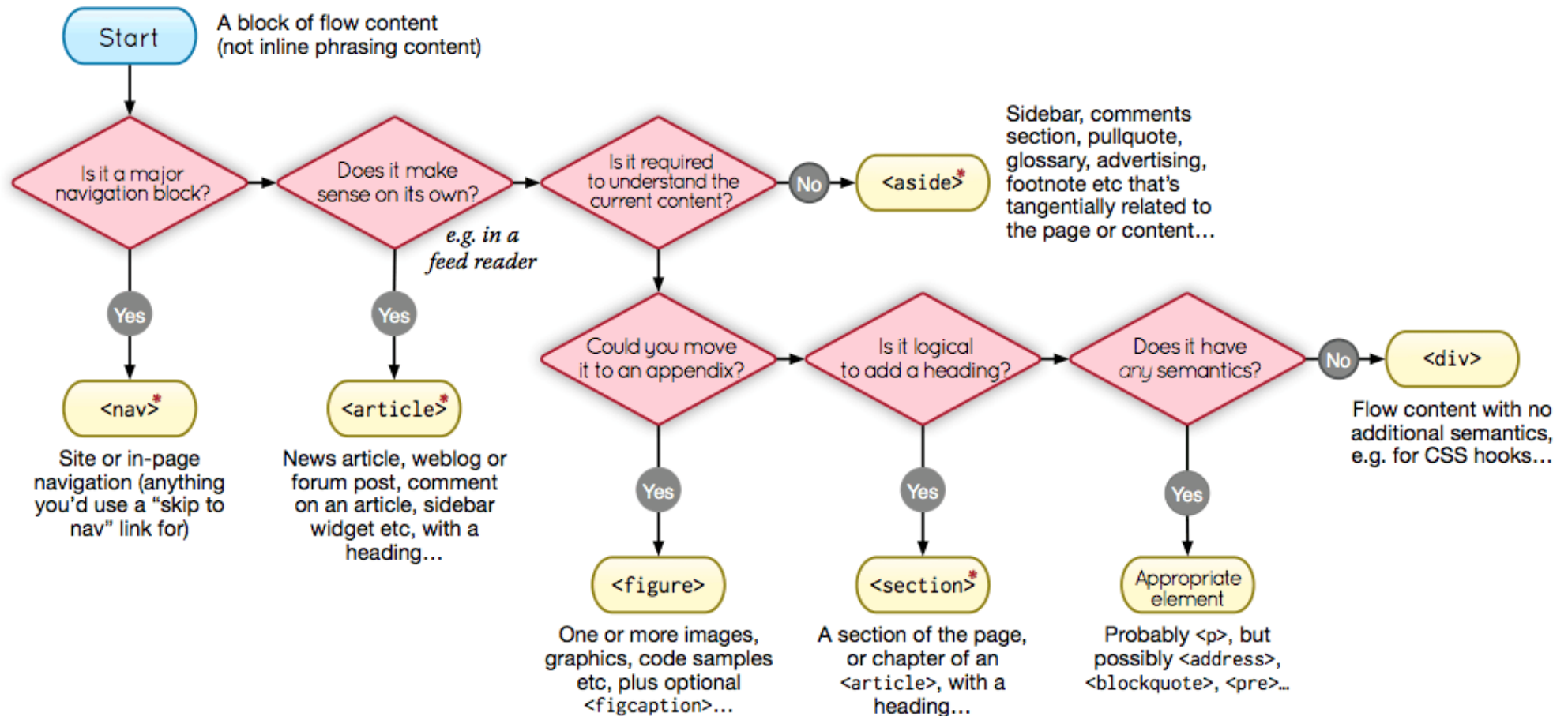


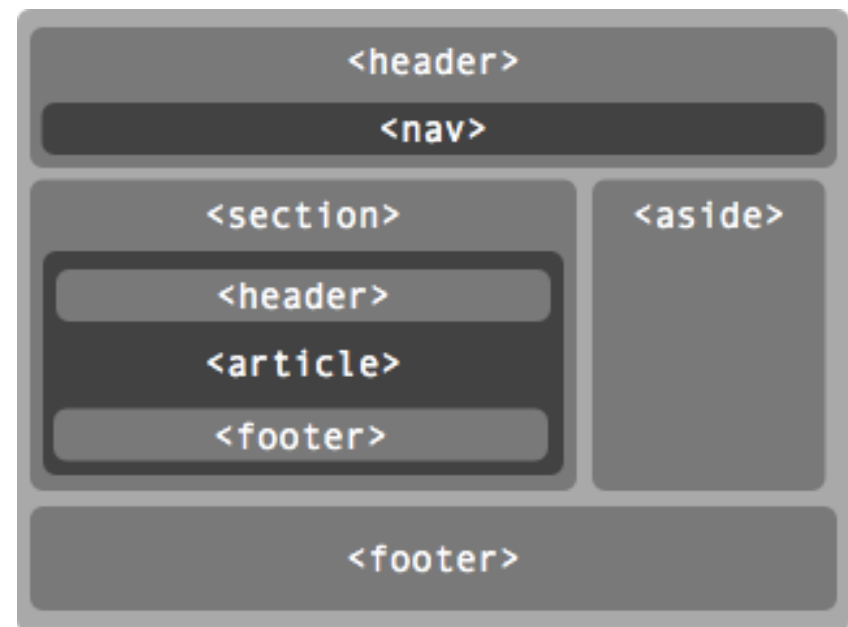
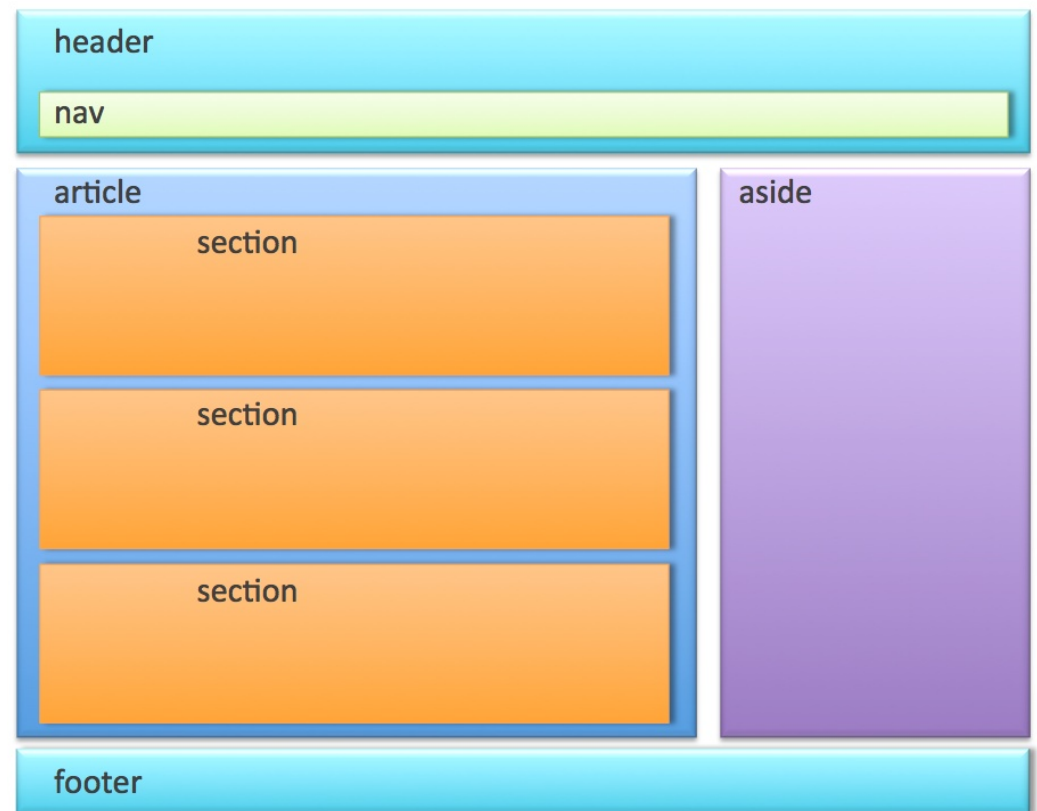
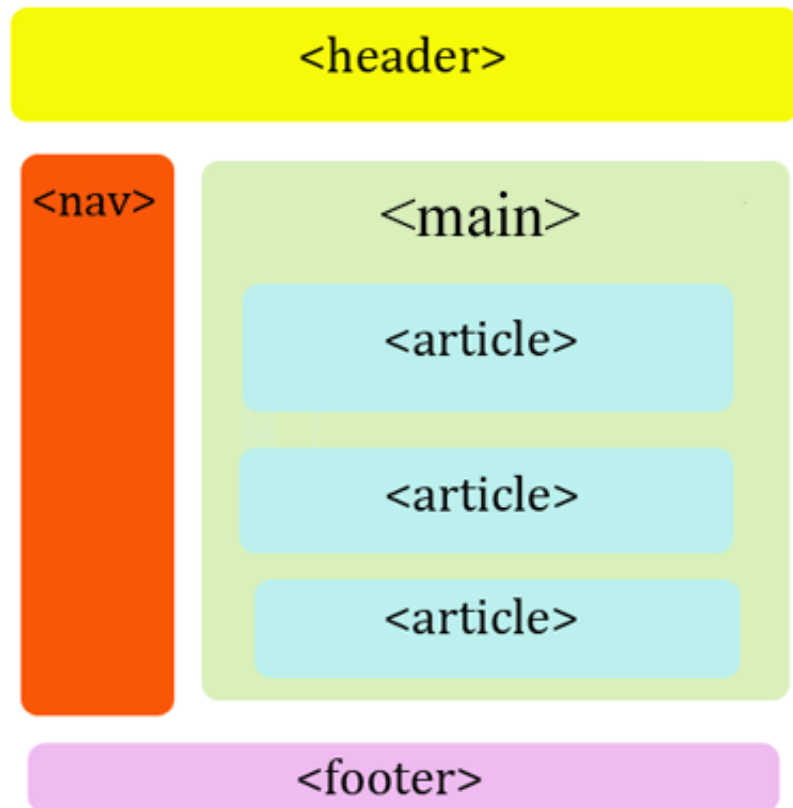
- text
- password
- checkbox
- radio
- button
- submit
- reset
- file
- hidden
- image
- number
- range
- email
- url
- search
- tel
- color
- date
- month
- week
- time
- datetime
- datetime-local



# HTML sectioning elements

- **<body>, <main>**
  - Το κύριο τμήμα του εγγράφου ή της εφαρμογής (μόνον ένα)
- **<article>**
  - Τμήμα του εγγράφου ή της εφαρμογής που θα μπορούσε να αποσπαστεί από το περιβάλλον του
- **<section>**
  - Ενότητα περιεχομένου
- **<h1> - <h6>**
  - Επικεφαλίδες που είναι καλό να εκκινούν κάθε section
- **<aside>**
  - Υλικό περιφερειακά σχετιζόμενο με το περιβάλλον sectioning element
- **<nav>**
  - Σύνολο από links σε άλλες σελίδες ή άλλα sections
- **<header>, <footer>**
  - Υλικό επικεφαλίδας / υποσέλιδου έως και ανά section, article, aside
- **<address>**
  - Πληροφορίες επικοινωνίας







# enter Mobile

```
<meta name="viewport" content="...">
```

- Content (comma separated):
  - **width=**
    - ακέραιος αριθμός (π.χ. 320) ή **device-width**
  - **initial-scale=**
    - αριθμός (π.χ. 1.2)
  - **minimum-scale=**
    - αριθμός (αν είναι 1.0 αποτρέπεται το zoom-out)
  - **maximum-scale=**
    - αριθμός (αν είναι 1.0 αποτρέπεται το zoom-in)
  - **user-scalable=**
    - **no**





# Αντιμετώπιση σφαλμάτων

- Τα σφάλματα στην HTML (και στα CSS) είναι κάτι κοινό και αναμενόμενο.
- Συνηθισμένα σφάλματα:
  - Ορθογραφικά λάθη σε εντολές, ids και attributes
  - Παράλειψη κλεισίματος tags (ή αγκίστρων)
  - Λανθασμένη εμφώλευση
  - Προβλήματα στη χρήση εισαγωγικών
  - Λανθασμένη αναφορά σε εξωτερικά αρχεία



Υπάρχουν πολλά επίπεδα εντοπισμού αυτών των σφαλμάτων:

1. Ορθογραφικά λάθη σε εντολές, attributes, Παράλειψη κλεισίματος tags ή αγκίστρων, Λανθασμένη εμφώλευση, Προβλήματα στη χρήση εισαγωγικών:
  - Χρωματισμός στον editor που χρησιμοποιείτε
  - Χρωματισμός στην εμφάνιση του κώδικα από τον browser (Control-U)
  - Αποτυχία HTML5 validation
  - Προειδοποιητικά μηνύματα στο Web Console



## 2. Λανθασμένη αναφορά σε εξωτερικά αρχεία:

- Προειδοποιητικά μηνύματα στο Web Console

## 3. Ορθογραφικά λάθη σε ids

- Σφάλμα δύσκολο να εντοπιστεί
- Θυμίζει τη δήλωση μεταβλητών σε γλώσσες προγραμματισμού
- Κάποιοι editors χρωματίζουν όλες τις εμφανίσεις μιας λέξης εντός ενός αρχείου (μερικώς βοηθητικό)





# Προβολή κώδικα (Ctrl+U)

```
Αρχείο  Επεξεργασία  Προβολή  Βοήθεια

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8" />
5      <title>Σφάλματα !</title>
6      <link rel="stylesheet" href="my.css" />
7    </head>
8    <body>
9      <p stle="font-weight: normal;" id="firt">Παραδειγματική πρώτη
    παράγραφος.
10
11    <h1>Παραδειγματική επικεφαλίδα</h1>
12
13    <p id="second">Παραδειγματική δεύτερη παράγραφος...</p>
14  </body>
15 </html>
```



# http://validator.w3.org

## Validation Output: 4 Errors

✖ Line 9, Column 43: **Attribute `style` not allowed on element `p` at this point.**

```
<p style="font-weight: normal;" id="first">Παραδειγματική πρώτη παράγραφος...
```

Attributes for element `p`:  
[Global attributes](#)

✖ Line 11, Column 4: **Bad character `η` after `<`. Probable cause: Unescaped `<`. Try escaping it as `&lt;`.**

```
<η1>Παραδειγματική επικεφαλίδα</η1>
```

✖ Line 11, Column 35: **Garbage after `</`.**

```
<η1>Παραδειγματική επικεφαλίδα</η1>
```

✖ Line 15, Column 7: **End of file reached when inside an attribute value. Ignoring tag.**

```
</html>
```



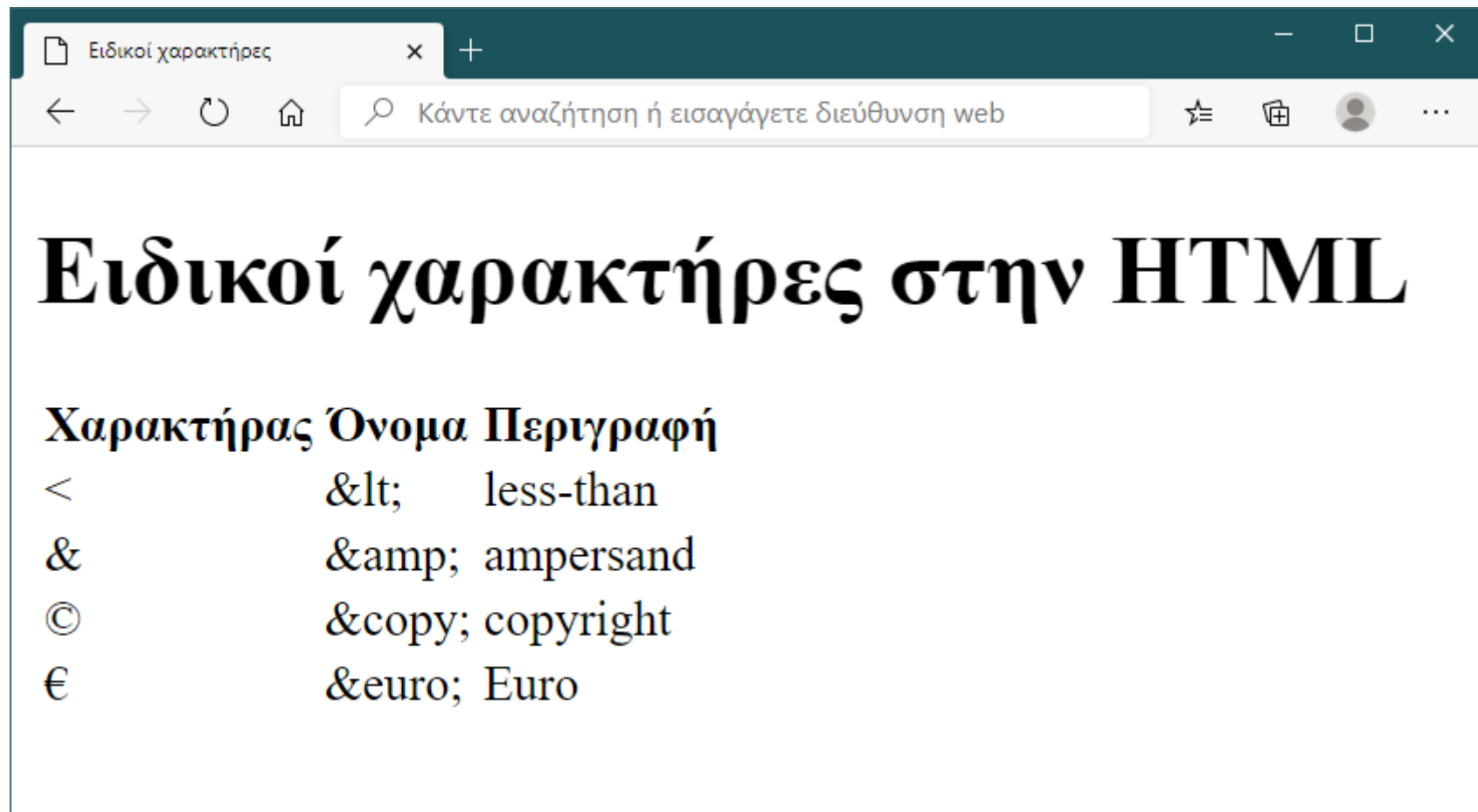
# Εργαλεία Web Developer

- Οι σύγχρονοι browsers ενσωματώνουν εργαλεία web development (Ctrl+Shift+I)
- Μέσω αυτών των εργαλείων μπορούμε να αλληλεπιδρούμε με τη σελίδα και τις ιδιότητές της και να διαπιστώνουμε τυχόν σφάλματα (π.χ. λανθασμένη αναφορά σε αρχείο).



# Άσκηση 1

- Κατασκευάστε τον ακόλουθο πίνακα:



The screenshot shows a web browser window with a single tab titled 'Ειδικοί χαρακτήρες'. The address bar contains the text 'Κάντε αναζήτηση ή εισαγάγετε διεύθυνση web'. The main content area displays a table with the title 'Ειδικοί χαρακτήρες στην HTML'.

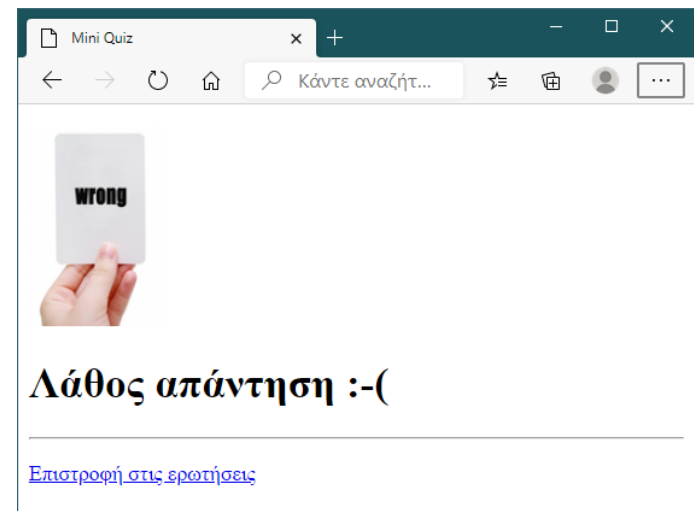
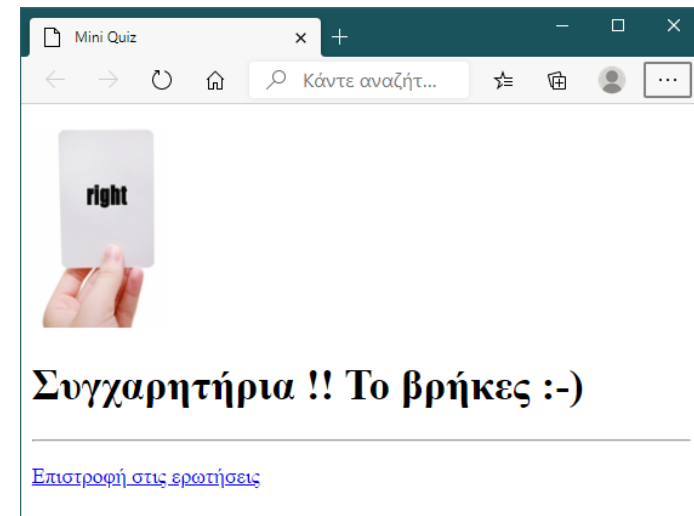
Χαρακτήρας	Όνομα	Περιγραφή
<	&lt;	less-than
&	&amp;	ampersand
©	&copy;	copyright
€	&euro;	Euro



# Άσκηση 2



- Κατασκευάστε ένα mini quiz με τη χρήση 3 αρχείων HTML (αρχεία ex\_02\_q.html, ex\_02\_r.html, ex\_02\_w.html).
- Για εικόνες χρησιμοποιήστε κάποιες που θα κατεβάσετε από το Internet (π.χ. Εικόνες στο Google).



# Cascading StyleSheets (CSS)



*“CSS is a language for styling markup content (such as HTML, SVG or XML)”*

- Επίπεδο παρουσίασης...



# CSS 3

- Σε αντιδιαστολή με τα CSS 2, τα CSS 3 αντί για “μονολιθικά” διαχωρίζονται σε επιμέρους έγγραφα "modules", με διαφορετικό βαθμό ωρίμανσης.
- Περιστασιακά συγκεντρώνονται ως snapshots.

[www.w3.org/Style/CSS/specs](http://www.w3.org/Style/CSS/specs)



# CSS Snapshots

“CSS Snapshot 20XX links to all the specifications that together represent the state of CSS as of 20XX.

With this document, the CSS WG aims to help implementors distinguish between the parts of CSS that are ready for production and the parts that are still experimental.”



# Τρόποι επίκλησης

- 3 τρόποι επίκλησης:
  - inline (style attribute)
  - embedded (style element)
  - external (link element + file) ✓



# style attribute

- Για να αλλάξουμε την εμφάνιση του περιεχομένου ενός HTML tag, χρησιμοποιούμε το attribute *style*:

```
<p style="..."> ... </p>
```

και μέσα στα εισαγωγικά δίνουμε ιδιότητες CSS.



# css\_start.html

- Έχω ένα αρχείο “καθαρής” html και θέλω να κάνω τα γράμματα του **<h1>** πλάγια!

## Μερικές Λίστες

Λίστα με ψώνια:

1. Γάλα
2. Τυρί
3. Αυγά

Λίστα με δουλειές:

- Καθάρισμα σπιτιού
- Βόλτα το σκύλο
- Πλύσιμο πιάτων



# font-style

```
<h1 style="font-style: italic;">
```

Μερικές Λίστες</h1>

## *Μερικές Λίστες*

Λίστα με ψώνια:

1. Γάλα
2. Τυρί
3. Αυγά

Λίστα με δουλειές:

- Καθάρισμα σπιτιού
- Βόλτα το σκύλο
- Πλύσιμο πιάτων





# font-weight

- Ναι... αλλά δεν μου αρέσουν τα έντονα γράμματα...  
(1 element – πολλά properties)

`<h1 style="font-style: italic; font-weight: normal;">`

Μερικές Λίστες</h1>

## *Μερικές Λίστες*

Λίστα με ψώνια:

1. Γάλα
2. Τυρί
3. Αυγά

Λίστα με δουλειές:

- Καθάρισμα σπιτιού
- Βόλτα το σκύλο
- Πλύσιμο πιάτων



# Το κατάλαβα...

- Μάλιστα... Άρα για να κάνω πλάγια όλα τα γράμματα της λίστας με τα ψώνια πρέπει να κάνω κάτι σαν:

<ol>

<li style="font-style: italic;">Γάλα</li>

<li style="font-style: italic;">Τυρί</li>

<li style="font-style: italic;">Αυγά</li>

</ol>

## *Μερικές Λίστες*

Λίστα με ψώνια:

1. Γάλα
2. Τυρί
3. Αυγά

Λίστα με δουλειές:

- Καθάρισμα σπιτιού
- Βόλτα το σκύλο
- Πλύσιμο πιάτων



# Κληρονομικότητα (inheritance)

- Η έννοια της κληρονομικότητας στα CSS:
  - Τα δηλωμένα από το χρήστη properties ενός element ενδέχεται να κληρονομούνται σε όλα τα ένθετα σε αυτό elements

```
<ol style="font-style: italic;">  
  <li>Γάλα</li>  
  <li>Τυρί</li>  
  <li>Αυγά</li>  
</ol>
```



# Εμβόλιμες (inline) εντολές CSS

- Οι εντολές CSS που δηλώνονται με το attribute **style** ονομάζονται εμβόλιμες.
- Κι αν θέλω να κάνω έντονα τα γράμματα **όλων** των παραγράφων;

# Εσωτερικές (internal or embedded) εντολές CSS



- Νέο element: **style**

```
<style>  
    tag {  
        property: value;  
        property: value;  
    }  
</style>
```



# Τροποποίηση <p>

<head>

```
<meta charset="UTF-8">
```

```
<title>Τίτλος</title>
```

```
<link rel="stylesheet" href="myStyles.css">
```

```
<style>
```

```
  p {
```

```
    font-weight: bold;
```

```
  }
```

```
</style>
```

</head>

## *Μερικές Λίστες*

**Λίστα με ψώνια:**

1. Γάλα
2. Τυρί
3. Αυγά

**Λίστα με δουλειές:**

- Καθάρισμα σπιτιού
- Βόλτα το σκύλο
- Πλύσιμο πιάτων



# Ορισμός κλάσεων

- Ναι... αλλά αν θέλω να κάνω έντονα το “γάλα”, τα “αυγά” και το “πλύσιμο πιάτων”;
- Μπορώ να ορίζω κλάσεις εντολών CSS και να χρησιμοποιώ το attribute **class**



# Δημιουργία κλάσης

```
.myBold {  
    font-weight: bold;  
}
```

.....

```
<li class="myBold">Γάλα</li>
```





# Κατοχή ιδιοτήτων πολλών κλάσεων

- Μπορώ να αποδίδω τις ιδιότητες πολλών κλάσεων σε ένα tag:

```
.myBold { font-weight: bold; }
```

```
.myItalic { font-style: italic; }
```

.....

```
<li class="myBold">Γάλα</li>
```

```
<li class="myBold myItalic">Τυρί</li>
```



# Άρα...

- Το style attribute μου χρειάζεται μόνον για αλλαγές σε ένα μόνον element.
- Σωστά... αλλά, αν το element είναι θαμμένο σε ένα τεράστιο html αρχείο, θα είναι δύσκολο να το βρω. Επίσης είναι ωραίο να κρατώ ξεχωριστά τις εντολές CSS από τις HTML.
- Καμιά ιδέα;



# Attribute: **id**

- Μοναδική αναφορά σε ένα element!

....

```
#myID {  
    font-style: italic;  
    font-weight: normal;  
}
```

....

```
<h1 id="myID">Μερικές Λίστες</h1>
```

## *Μερικές Λίστες*

Λίστα με ψώνια:

1. Γάλα
2. Τυρί
3. Αυγά

Λίστα με δουλειές:

- Καθάρισμα σπιτιού
- Βόλτα το σκύλο
- Πλύσιμο πιάτων

- Προσοχή: πάρα πολύ χρήσιμο attribute για τη JavaScript, που θα γνωρίσουμε αργότερα!



# Εξωτερικές (external) εντολές CSS

- Ωραία... έχω μαζέψει όλο το παρουσιαστικό της HTML μέσα στο style element.
- Κι αν θέλω να έχει το ίδιο παρουσιαστικό κάποιο άλλο αρχείο μου HTML; Πρέπει να το αντιγράψω κάθε φορά;
- Νέο tag: **link**



# External CSS

- Δημιουργώ ένα αρχείο με επέκταση `.css` (π.χ. **myStyles.css**), το οποίο περιέχει μόνον κανόνες CSS.
- Συνδέω το αρχείο CSS με το αρχείο HTML ως εξής (περιλαμβάνεται στο **head** element):

```
<link rel="stylesheet" href="myStyles.css">
```



# Style rules

```
selector {  
    property1: value(s);  
    property2: value(s);  
    property3: value(s);  
}
```

- Shorthand properties!



# Selectors

- **X** (element selector)
  - `a { color: red; }`
- **.X** (class selector)
  - `.error { color: red; }`
- **#X** (ID selector)
  - `#example {}`
- Combining selectors
  - `p.warning {}`
  - `a, p.warning {}`



# Selectors

- `X > Y` (child selector)
  - `#container > ul {}`
- `X Y` (descendant selector)
  - `li a {}`
- Attribute selector
  - `a[title], a[href="https://www.uom.gr"] {}`
- Pseudo-classes
  - `a:visited, section:not(#references) {}`
  - `p:lang(el), input[type="checkbox"]:checked {}`
  - [developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes](https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes)





# CSS Specificity

- Inline Styles
- Specificity
  - ID selectors
  - Class selectors, Attributes selectors, Pseudo-classes
  - Elements, Pseudo-elements
- Order of Specification



# Colors, backgrounds, fonts, text

- Ένα σύνολο από properties διαμορφώνουν θέματα όπως:
  - Χρώμα γραμμάτων και φόντου (`color`, `background-color`)
  - Εικόνες στο φόντο των elements (`background-image`, `background-repeat`, `background-attachment`, `background-position`, `background`)
  - Γραμματοσειρές, μεγέθη, στυλ γραμμάτων (`font-family`, `font-style`, `font-variant`, `font-weight`, `font-size`, `font`)
  - Στοιχίσεις, διάστιχο, διαγραμμίσεις (`text-indent`, `text-align`, `text-decoration`, `letter-spacing`, `word-spacing`, `text-transform`, `white-space`, `line-height`, `vertical-align`)
  - Κουκίδες, αριθμήσεις (`list-style-type`, `list-style-image`, `list-style-position`, `list-style`)



# Γραμματοσειρές

- Μεγάλες Κατηγορίες:
  - Μεταβλητού Πλάτους - Σταθερού Πλάτους
  - Serif – Sans Serif (Με ή Χωρίς Απολήξεις)

Times New Roman:	Ίλιον Μάγμα
------------------	----------------

Arial:	Ίλιον Μάγμα
--------	----------------

Courier New:	Ίλιον Μάγμα
--------------	----------------



# Ποια να χρησιμοποιώ;

- Δεν είναι εκ των προτέρων γνωστό ποιες γραμματοσειρές έχει εγκατεστημένες στο σύστημά του ο κάθε χρήστης.
- Προτεινόμενες λύσεις:
  - Χρήση των standards:  
**serif, sans-serif, monospace**
  - Χρήση του κανόνα CSS:  
**@font-face** (θα επανέλθουμε)



# font-family

- Δηλώνουμε όσες γραμματοσειρές θέλουμε, χωρισμένες με κόμμα, και θα αναζητηθούν με αυτή τη σειρά στο σύστημα του χρήστη.

p {

font-family: "Open Sans", Georgia, serif;

}

- Λεπτομέρεια: όταν έχετε πρόβλημα χρήσης διπλών εισαγωγικών, χρησιμοποιήστε μονά (και αντιστρόφως). Π.χ.:

<p style="font-family: 'Open Sans', serif;"> ... </p>



# font-size

- Δηλώνεται με πολλούς τρόπους (συζήτηση παρακάτω):

```
p { font-size: 36px; }
```

```
h1 { font-size: 250%; }
```

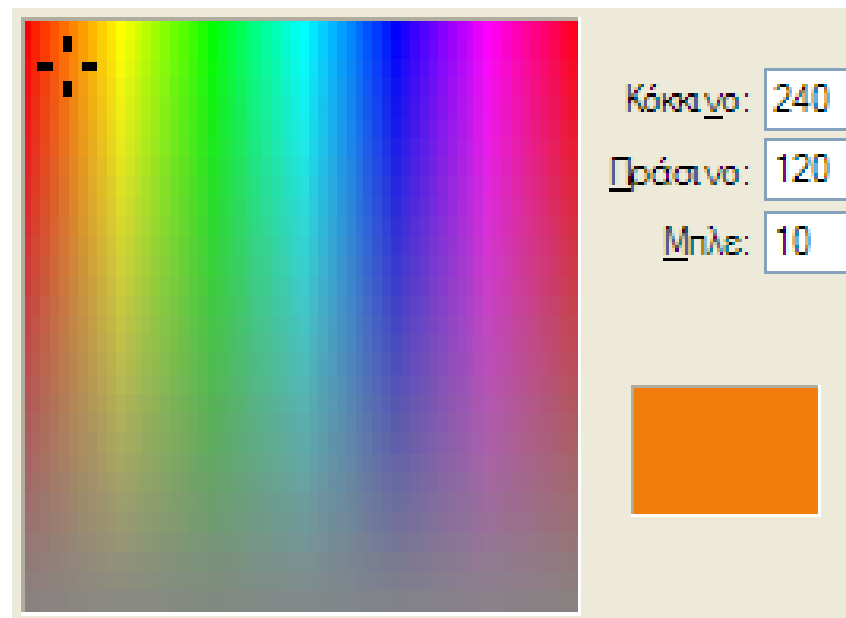


# Χρώματα !

- Τα χρώματα δηλώνονται με πολλούς τρόπους.
- Θα κρατήσουμε τον εξής:

`rgb(0..255, 0..255, 0..255) ;`

↑                      ↑                      ↑  
κόκκινο (r)      πράσινο (g)      μπλε (b)





# Χρώματα !

- Παράδειγμα κόκκινου:

`rgb(255, 0, 0);`

- Σύνολο διαθέσιμων χρωμάτων:

$$256^3 = (2^8)^3 = 2^{24} = 16.777.216$$

[http://www.rapidtables.com/web/color/RGB\\_Color.htm](http://www.rapidtables.com/web/color/RGB_Color.htm)





# color, background-color

- Δήλωση χρώματος για το κείμενο (**color**) και για το φόντο (**background-color**)

```
p {
```

```
    color: rgb(255,255,0);
```

```
    background-color: rgb(0,100,255);
```

```
}
```



# Άλλοι τρόποι δήλωσης χρώματος

- Δεκαεξαδικός κωδικός **#rrggbb**

Π.χ.: **#FF9900**

- Αγγλική λέξη. Π.χ.: **pink**

- Όλα τα ακόλουθα είναι το ίδιο χρώμα:

**#6495ED**      **cornflowerblue**      **rgb(100,149,237)**





# Διαφάνεια

- Μπορούμε να ορίσουμε το επίπεδο διαφάνειας ενός αντικειμένου, επιτρέποντας να φαίνονται τα χρώματα και τα αντικείμενα που βρίσκονται “πίσω” του.
- Η διαφάνεια ορίζεται με την ιδιότητα: **opacity**, η οποία λαμβάνει τιμές από 0 (πλήρως διαφανές) έως 1 (πλήρως αδιαφανές – κανονικό).
- Π.χ.:

**opacity: 0.5;** /\* ημι-διαφανές \*/

**opacity: 0.33;** /\* 1/3 διαφάνειας \*/



# CSS Values and Units

- Keywords:

```
ul {  
    margin: auto;  
    list-style-type: disc;  
    border-left-style: dashed;  
}
```

- Data types:

```
p {  
    margin-left: 10px;  
    font-size: 2em;  
}
```



# CSS data types

- Strings

```
font-family: "Open Sans";
```

- Resource Locators

```
background: url("http://www.xpl.com/pin.gif");
```

- Integer / Real Numbers

```
order: 5; z-index: -10; opacity: 0.33;
```

- Color

```
color: rgb(255, 0, 0);
```

```
rgba(128, 150, 10, 0.5) #f5f5dc #5fc lime
```



# CSS data types

([css-tricks.com/the-lengths-of-css](https://css-tricks.com/the-lengths-of-css))

- Length

	Absolute		Relative	
		to parent (compounding)	to <html>	to viewport
screen	px	em, %	rem	vw, vh, vmin, vmax
print	cm, mm, in, pt	em	rem	



# Fonts

- Εξαιρετικά σημαντικό στοιχείο...
- Generic font families:
  - serif, sans-serif, monospace, cursive, fantasy
- Χρήση ελεύθερων ή δωρεάν fonts
  - **@font - face** rule

[www.w3.org/TR/css3-fonts](http://www.w3.org/TR/css3-fonts)



# Παράδειγμα χρήσης

```
@font-face {  
    font-family: "Clear Sans";  
    src: local("Clear Sans"), url("path/ClearSans-Regular.ttf");  
}
```

```
@font-face {  
    font-family: "Clear Sans";  
    src: local("Clear Sans"), url("path/ClearSans-Bold.ttf");  
    font-weight: bold;  
}
```





# Font links

[www.google.com/fonts](http://www.google.com/fonts)

[edgewebfonts.adobe.com](http://edgewebfonts.adobe.com)

[openfontlibrary.org](http://openfontlibrary.org)

font hosting-  
servicing

[www.fontsquirrel.com](http://www.fontsquirrel.com)

[www.fonts.com](http://www.fonts.com)

[www.fontspring.com](http://www.fontspring.com)

# Άσκηση



- Ξεκινώντας από το αρχείο `ex_01_start.html`, προσθέστε κανόνες και ιδιότητες CSS, προσπαθώντας να φτάσετε στην ακόλουθη αισθητική:

Hints:

Γραμματοσειρές:  
Noto Serif, Open Sans

Στοίχιση  
παραγράφου:  
πλήρης

Αλλαγή φόντου  
όταν το ποντίκι  
αιωρείται στη  
λίστα

## Ανδρική και Γυναικεία Μόδα

ρούχα, παπούτσια, casual, accessories

Το μεγαλύτερο κατάστημα στην Ελλάδα για γυναικεία και ανδρικά ρούχα, παπούτσια και αξεσουάρ σε προσιτές τιμές. Η τελευταία λέξη της μόδας. Νέες αφίξεις καθημερινά. Θα βρείτε ό,τι χρειάζεστε σε είδη ρουχισμού, υποδήματα και αξεσουάρ από επώνυμες μάρκες, για όλα τα απαιτητικά μέλη της οικογένειας, στις καλύτερες τιμές.

Προτιμήστε μας για:

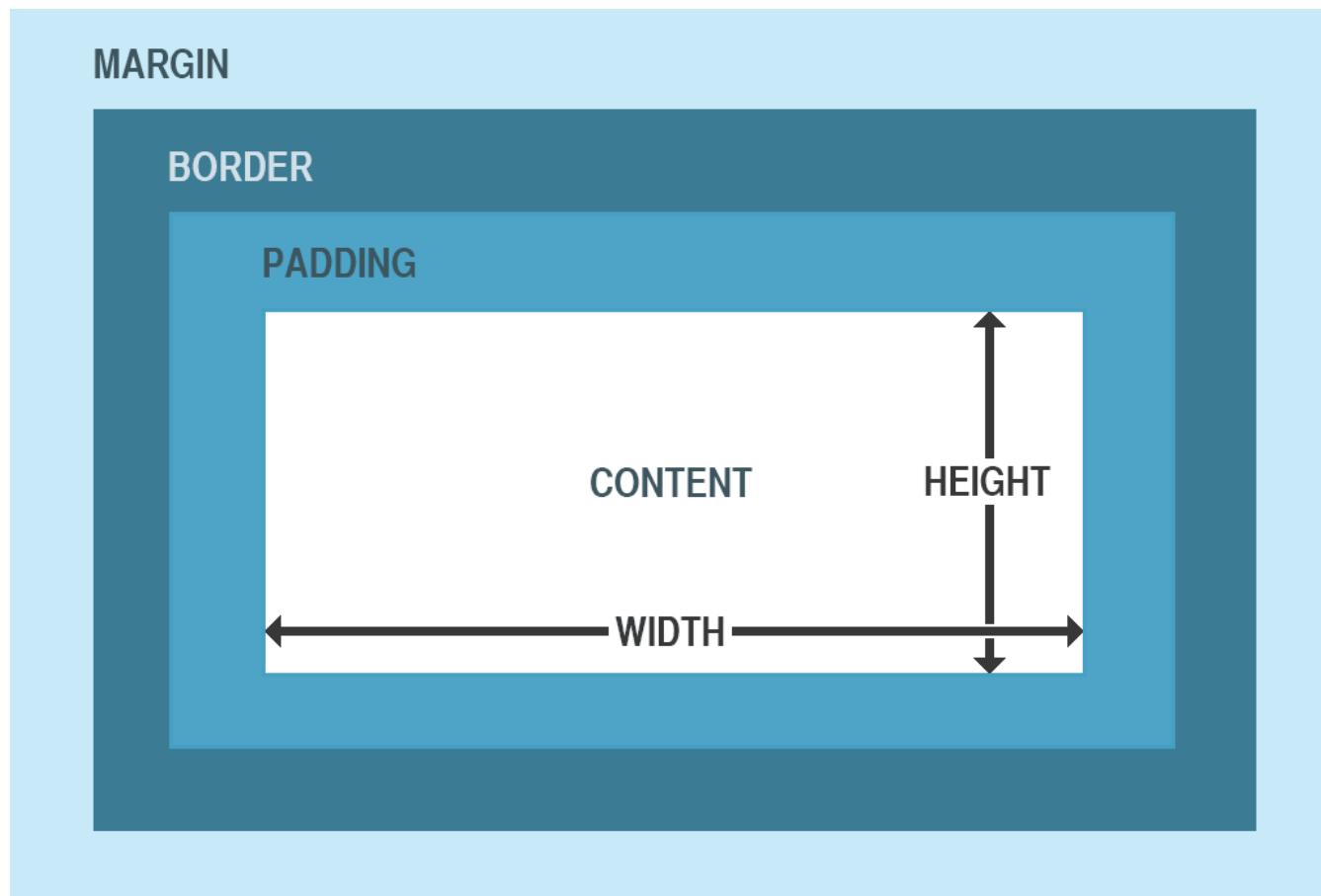
- Ρούχα
- Παπούτσια
- Jeans
- Γυαλιά ηλίου
- Ρολόγια
- Τσάντες
- Μαγιώ

Σας περιμένουμε...



# Box model

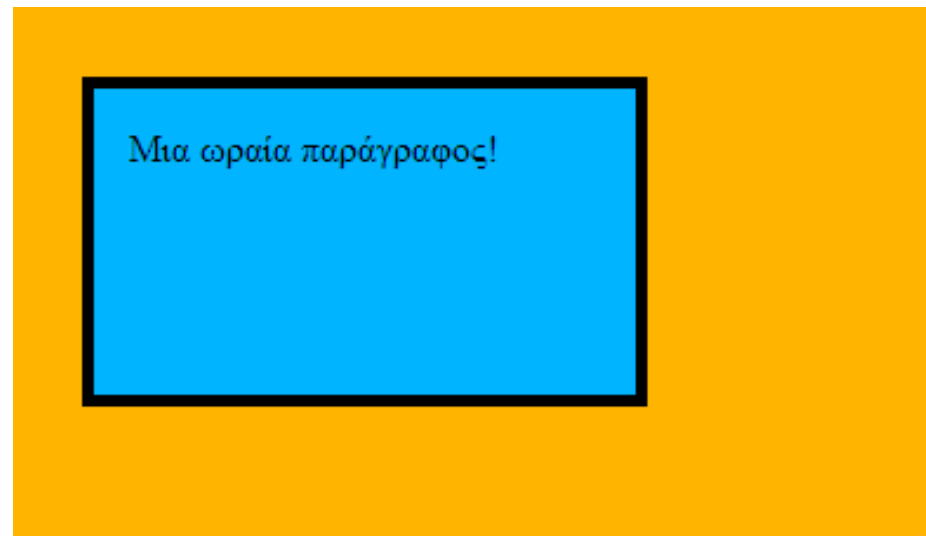
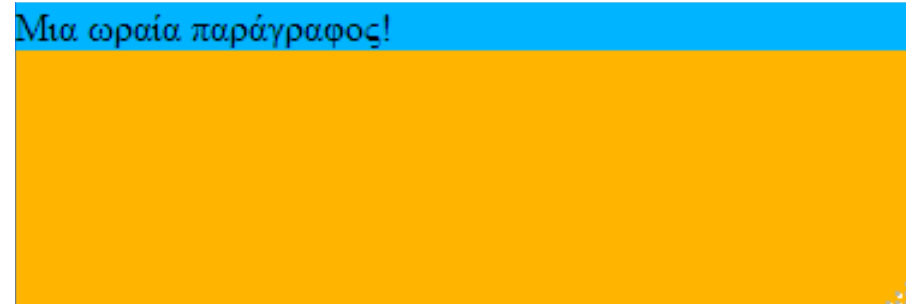
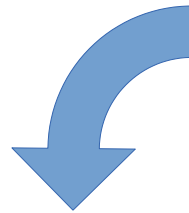
- Κάθε element έχει ένα “μοντέλο πλαισίου” (box model) γύρω του.



# Ορισμός margin, border, padding, width και height στην παράγραφο



```
p {  
  background-color: rgb(0,180,255);  
  margin: 30px;  
  border: 5px solid;  
  padding: 15px;  
  width: 200px;  
  height: 100px;  
}
```





# Ιδιαιτερότητα κελιών πίνακα

- Τα κελιά ενός πίνακα αντιμετωπίζονται ως ξεχωριστά elements (td) και το border τους απέχει.
- Λύση: ιδιότητα CSS του table:

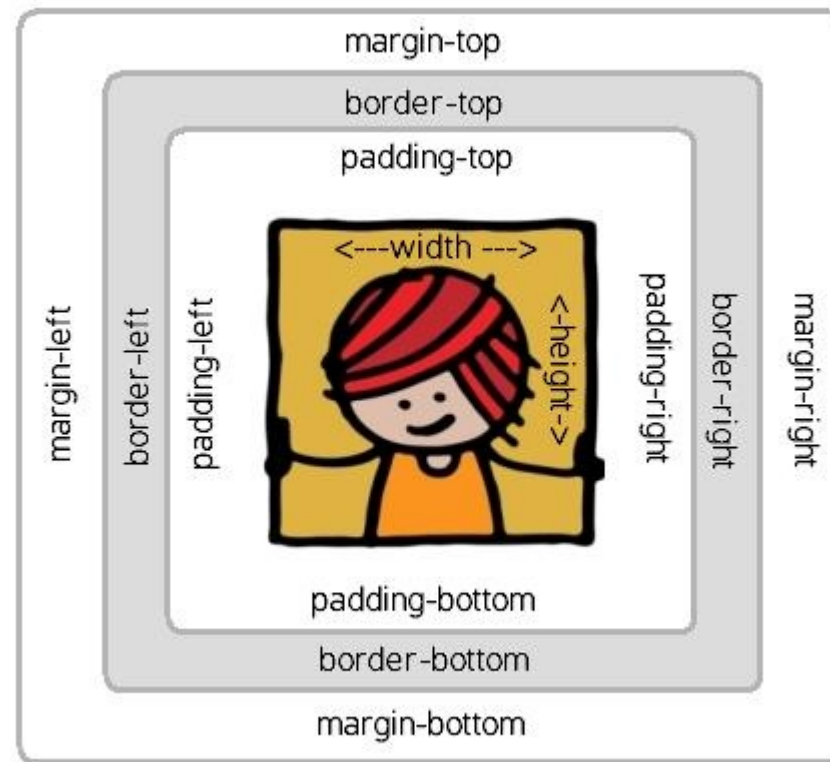
**border-collapse: collapse;**

A/A	Επώνυμο	Όνομα	Βαθμολογία
1	Γεωργίου	Αθανάσιος	9
2	Καλούση	Ελένη	9

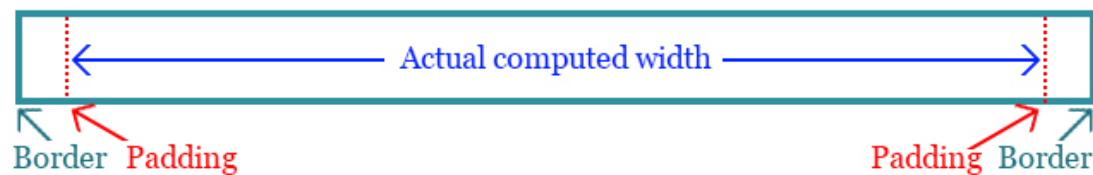
A/A	Επώνυμο	Όνομα	Βαθμολογία
1	Γεωργίου	Αθανάσιος	9
2	Καλούση	Ελένη	9



# Box Model



## WITHOUT BOX-SIZING: BORDER-BOX



## WITH BOX-SIZING: BORDER-BOX





# Διάταξη σελίδας, ροή αντικειμένων

- Η τοποθέτηση των elements μπορεί να γίνει με διάφορους τρόπους
- Τα πιο σημαντικά properties σ' αυτή τη διαδικασία είναι τα:
  - `float`
  - `display`
  - `position`



# Floating elements

- Τα αντικείμενα “βγαίνουν” από την κανονική ροή του κειμένου και “επιπλέουν” αριστερά ή δεξιά εντός του container!
- Κλασική χρήση: ροή κειμένου γύρω από εικόνες
- Πολλαπλά floating elements μπορούν να βρίσκονται το ένα δίπλα στο άλλο, ενόσω υπάρχει χώρος παραθύρου
- Η διακοπή της λειτουργίας γίνεται με το **clear** property
- Απαιτείται προσοχή στο ύψος του container. Το (default) **overflow: visible;** επεκτείνει τα floated αντικείμενα εκτός του container





# display property

- ▷ **none**: Απόκρυψη
- ▷ **inline**: Μέρος μιας συνεχούς γραμμής περιεχομένου, με αναδίπλωση λέξεων (width και height αγνοούνται, top και bottom padding, border και margin απλά επικαλύπτουν τα υπόλοιπα στοιχεία της ροής)
- ▷ **block**: Αλλαγή γραμμής πριν και μετά, πλήρες box model
- ▷ **inline-block**: Συμπεριφέρονται σαν ένα ενιαίο αντικείμενο εντός μιας συνεχούς γραμμής περιεχομένου, πλήρες box model
- ▷ **flex** / **inline-flex**: Συμπεριφορά ως block/inline με βάση το Flexible Box Layout Module
- ▷ **grid** / **inline-grid**: Συμπεριφορά ως block/inline με βάση το Grid Layout Module



# position property

- **static**
  - Ακολουθεί τη ροή της σελίδας (default)
- **relative**
  - Σχετικές τιμές σε σχέση με τη φυσιολογική θέση που θα είχε ως static. Τα relative elements μετακινούνται και επικαλύπτουν/ονται αλλά ο χώρος τους παραμένει στην κανονική ροή.
- **absolute**
  - Σχετικές τιμές σε σχέση με τον πιο κοντινό πρόγονο του element που έχει κάποια τιμή **position** διαφορετική του **static**. Αν δεν υπάρχει τέτοιος πρόγονος, καταλήγουμε στην ίδια τη σελίδα (<html> element, έξω από το <body>!). Τα elements “αποκολλούνται” από την κανονική ροή και το υπόλοιπο κείμενο συνεχίζει σαν να μην υπήρχε absolutely positioned element. Συχνή χρήση: αντικείμενα relative περιέχουν αντικείμενα absolute (π.χ. menus).
- **fixed / sticky**
  - Απόλυτες τιμές σε σχέση με το παράθυρο του browser (fixed), σχετική αγκύρωση (sticky), ακόμη κι αν γίνει scroll η σελίδα



# Ιδιότητες θέσης - Σχόλια

- **top, bottom, left, right**
  - Αποστάσεις της επάνω, κάτω, αριστερής, δεξιάς πλευράς του box του αντικειμένου.
- Επιτρέπονται αρνητικές τιμές και, αν δοθεί ποσοστό (%) αντί π.χ. για `px`, αυτό αναφέρεται στο μέγεθος του στοιχείου που περιέχει το `element` που τοποθετούμε.
- Στις επικαλύψεις η διάταξη ορίζεται μέσω του **z-index** property (initial: 0, επιτρέπονται αρνητικές τιμές)
- Κλασική χρήση του **position**: αναδυόμενα μενού



# At-rules

```
@charset "utf-8";
@import "local.css";
@media (max-width: 600px) {
    aside {
        display: none;
    }
}

@font-face {
    font-family: "Clear Sans";
    src: url("ClearSans-Regular.ttf");
}
```



# Images

	Φωτογραφία φυσικού περιβάλλοντος	Κατασκευασμένη σε Η/Υ
Bitmap	jpg	png
Vector	(αδύνατον)	svg



# Transforms, Transitions, Animation

- Transforms
  - Τροποποιήσεις των elements σε 2D και 3D
- Transitions
  - Ομαλή αλλαγή των τιμών των properties κατά καθορισμένη χρονική διάρκεια
- Animation
  - Δημιουργία animation των αλλαγών των τιμών των properties με χρήση keyframes.



# Transforms

- Τροποποίηση της θέσης και του σχήματος, μέσω αλλαγής του χώρου συντεταγμένων, χωρίς παρέμβαση στην αρχική “ροή” του εγγράφου.

## 2D Transforms

Lorem ipsum dolor sit amet, in eum hinc doming noster. Et postea tacimates mel, ius quodsi aperiri mnesarchum in. Vix purto omittam constituam no, tale habemus civibus ne mei.

Lorem ipsum dolor sit amet, in eum hinc doming noster. Et postea tacimates mel, ius quodsi aperiri mnesarchum in. Vix purto omittam constituam no, tale habemus civibus ne mei.

Lorem ipsum dolor sit amet, in eum hinc doming noster. Et postea tacimates mel, ius quodsi aperiri mnesarchum in. Vix purto omittam constituam no, tale habemus civibus ne mei.

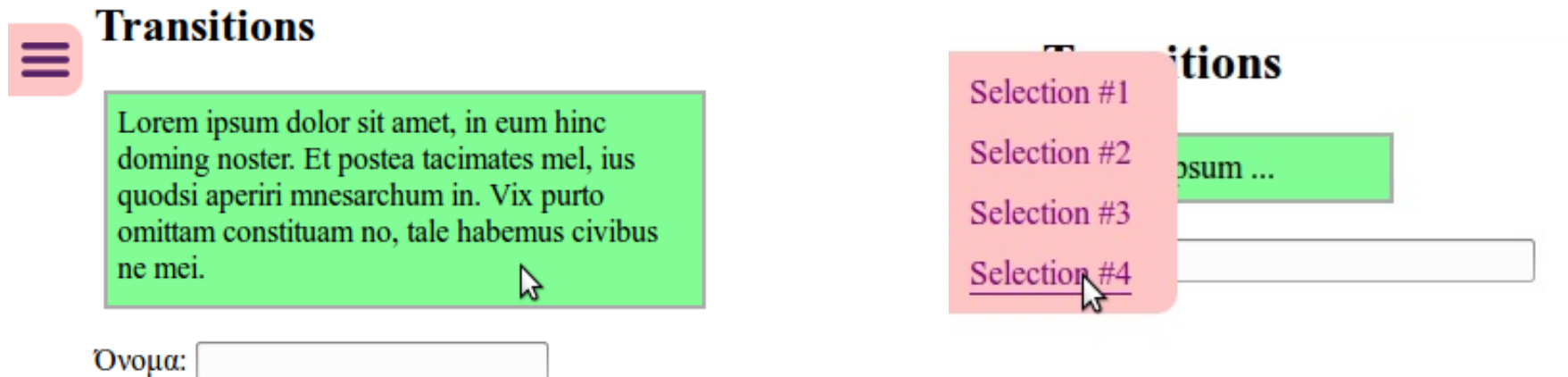
Lorem ipsum dolor sit amet, in eum hinc doming noster. Et postea tacimates mel, ius quodsi aperiri mnesarchum in. Vix purto omittam constituam no, tale habemus civibus ne mei.

Lorem ipsum dolor sit amet, in eum hinc doming noster. Et postea tacimates mel, ius quodsi aperiri mnesarchum in. Vix purto omittam constituam no, tale habemus civibus ne mei.



# Transitions

- Δεν είναι όλα τα properties animatable  
([developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using\\_CSS\\_transitions](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using_CSS_transitions))







# Transitions vs Animation

	Transitions	Animation
Πυροδότηση	Απαραίτητη (π.χ. :hover, JavaScript)	Αυτόματη
Looping	Περίπλοκο (JavaScript)	Απλό
Ενδιάμεσα Σημεία / Keyframes	Αδύνατο	Πλήρως υποστηριζόμενο

JavaScript friendly: Transitions



# Responsive (web) design

- Ευρεία και διαρκώς μεταβαλλόμενη έννοια
- Στόχος:
  - Εμπειρία (ανάγνωση, χρήση, πλοήγηση, ...) ανεξάρτητη του υποκείμενου μέσου προβολής ή εκτέλεσης
  - Media queries, fluid design, proportion-based grids, flexible images, relative units, ...





# Σχετιζόμενα CSS Specifications (μεταξύ άλλων)

- Device Adaptation
- Media Queries
- Flexible Box Layout
- Grid Template Layout
- Multi-column Layout

[www.w3.org/Style/CSS/specs](http://www.w3.org/Style/CSS/specs)



# Device adaptation

- meta viewport tag (το είδαμε)
- CSS Device Adaptation (ωριμάζει...)
  - **@viewport** rule
  - Properties: **min-width**, **max-width**, **width**, **min-height**, **max-height**, **height**, **zoom**, **min-zoom**, **max-zoom**, **orientation**

[www.w3.org/TR/css-device-adapt](http://www.w3.org/TR/css-device-adapt)



# Media queries

[developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media\\_queries](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries)

```
<link rel="stylesheet" media="screen and (device-height: 600px)" href="style.css">
```

```
@media (min-width: 700px) {}
```

```
@media screen and (min-width: 400px) and (max-width: 700px) {}
```



# Layout modes

- Old:
  - block, inline, table, positioned
- New:
  - flexible box, grid (CSS)



# Flexible box (flexbox)

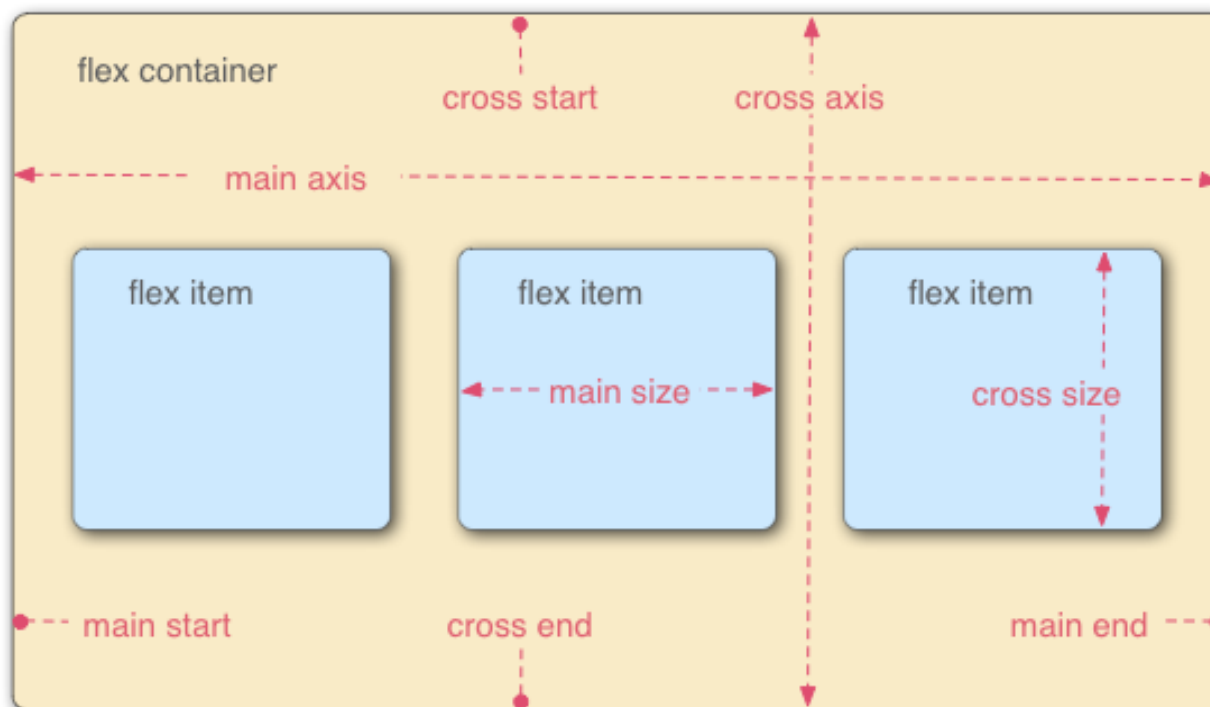
[css-tricks.com/snippets/css/a-guide-to-flexbox/](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)

- Δυνατότητα αλλαγής πλάτους ή/και ύψους των αντικειμένων για τη βέλτιστη κάλυψη του διαθέσιμου χώρου σε κάθε οθόνη
- Ενδείκνυται για τα επιμέρους στοιχεία εφαρμογών και μικρών διατάξεων (layouts)
- Το CSS Grid εξυπηρετεί τις μεγαλύτερες διατάξεις

[blog.stephaniewalter.fr/en/fun-places-learn-css-layout-part-1-flexbox/](https://blog.stephaniewalter.fr/en/fun-places-learn-css-layout-part-1-flexbox/)



- flex container: οποιοδήποτε parent element με display property: flex ή inline-flex
- main axis: ο άξονας ροής των flex items
- πολλαπλές δυνατότητες στοίχισης–εσωτερικών αποστάσεων







# Grid

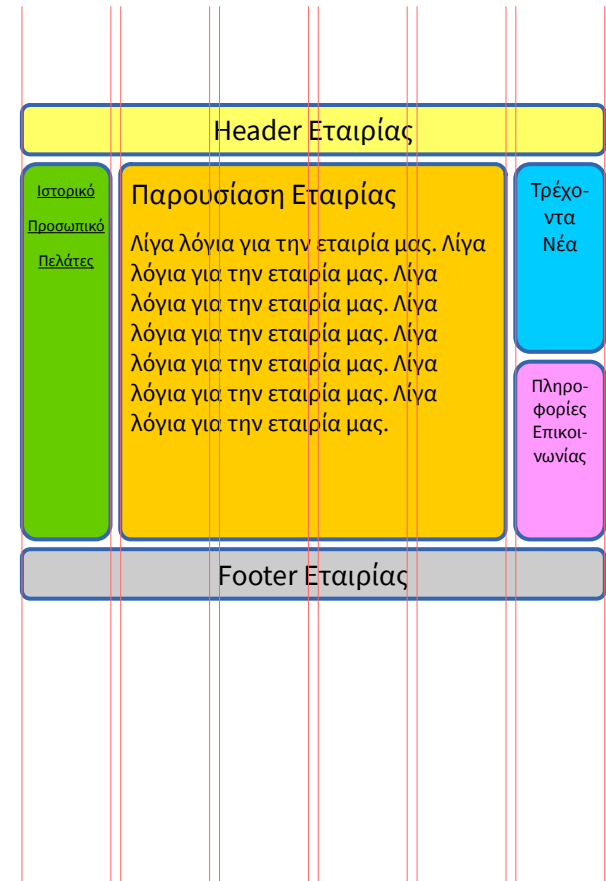
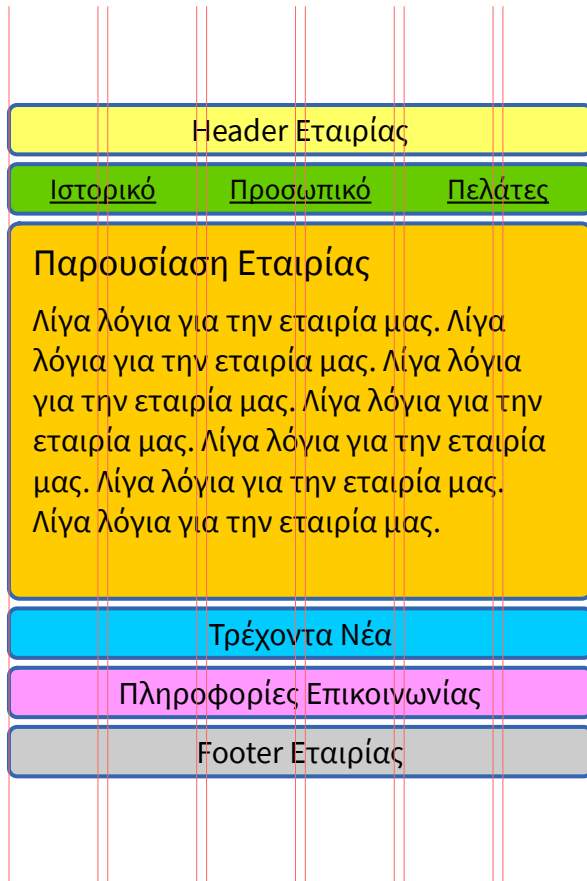
[css-tricks.com/snippets/css/complete-guide-grid/](https://css-tricks.com/snippets/css/complete-guide-grid/)

- Το flexbox προορίζεται (ως επί το πλείστον) για layouts μιας διάστασης
- Σε περίπλοκα δισδιάστατα layouts εξυπηρετεί το grid
- Τα flexbox και grid συνδυάζονται αρμονικά!

[developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Grid\\_Layout#External\\_resources](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout#External_resources)



# Column layout revenge!





# Best practices

- Mobile first, keep it simple
- Fonts
- Images, clipart
- Maximize CSS
- Prototype





# Μελέτη...

[docs.webplatform.org/wiki/css/tutorials](https://docs.webplatform.org/wiki/css/tutorials)

[developer.mozilla.org/en-US/learn/css](https://developer.mozilla.org/en-US/learn/css)

- Meta:

[www.w3.org/Style/CSS/learning](http://www.w3.org/Style/CSS/learning)





# Άσκηση

- Παίξτε τα παιχνίδια:

[flexboxfroggy.com/#el](https://flexboxfroggy.com/#el)

[cssgridgarden.com/#el](https://cssgridgarden.com/#el)



# JavaScript



*“JavaScript is a prototype-based object-oriented, loosely-typed, dynamic, scripting language”*

“Any application that *can* be written in JavaScript,  
*will* eventually be written in JavaScript”

Atwood's Law

- Επίπεδο συμπεριφοράς...



# JavaScript

*“JavaScript is a prototype-based object-oriented, loosely-typed, dynamic, scripting language”*

Run-time code change,  
functional programming,  
closures

Interpreted run-time  
environment

Inheritance through  
object cloning

Run-time type  
change, operations'  
unpredictability



# Προγραμματιστικό στυλ

- Object-oriented
- Imperative
- Functional
  - Closures
  - First class / higher order functions

Functions are  
objects

Functions accept  
functions as  
arguments and  
return functions





# Χρήσεις - ομοιότητες

- It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It is also used in server-side network programming with frameworks such as Node.js, game development and the creation of desktop and mobile applications.

*Wikipedia*

[redmonk.com/sogrady/category/programming-languages](http://redmonk.com/sogrady/category/programming-languages)

- Το συντακτικό της JavaScript προέρχεται από τη C, ενώ ο σχεδιασμός της και η σημασιολογία της έχουν επηρεαστεί από τις Self, Scheme και Lisp.
- Βασίζεται στο πρότυπο [ECMAScript](#)



# Εκτέλεση JavaScript

- Internal και external κώδικας μέσω του **script** element
- Inline (ή embedded) μέσω των element event attributes

```
<body>
  <script>
    var x = 2;
    x = x + 1;
    alert('x is: '+x);
  </script>
  <p>Μια παράγραφος...</p>
  <script src="myFile.js"></script>
  <p onclick="var y = 7; alert('y is: '+y);">Μια άλλη παράγραφος...</p>
</body>
```

- Βοηθητικά, μπορούμε να εκτελέσουμε κώδικα και στη γραμμή εντολών της κονσόλας του browser



# Hello world

- Είναι πολύ κοινό να αρχίζει ο προγραμματισμός με τη δυνατότητα του προγράμματος να μας χαιρετά (να αλληλεπιδρά μαζί μας με έναν βασικό τρόπο)
- Ας κάνουμε τη JavaScript να μας χαιρετήσει με ορισμένους βασικούς τρόπους...
- Θα κάνουμε και ένα μικρό “άλμα στο μέλλον” αρχίζοντας να χρησιμοποιούμε το WEB API object **console** και συγκεκριμένα το method **console.log(...)**



# script element

- Μπορούμε να έχουμε πολλά scripts σε μία σελίδα.
- Μπορεί να τοποθετείται οπουδήποτε εντός του κώδικα HTML και εκτελείται “σειριακά”
- Άρα έχει σημασία η θέση του **script** tag...
- Τα εξωτερικά αρχεία περιέχουν μόνον εντολές JavaScript και συνηθίζεται να χρησιμοποιούμε το επίθεμα **.js**

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

- **Προσοχή:** Μην ξεχνάτε το κλείσιμο του script tag!



# Σφάλματα...

- Όταν κάνουμε συντακτικά σφάλματα, δεν βλέπουμε κάτι στην ιστοσελίδα. Απλά δεν εκτελείται σωστά ο κώδικας JavaScript.
- Στην κονσόλα θα βλέπουμε και τα συντακτικά μας σφάλματα:

```
<script>  
    console.log(Hello world! ');  
</script>
```

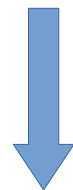


```
07:58:07.948 | ✖ SyntaxError: missing ) after argument list
```



- Ωστόσο, η αποσφαλμάτωση ενδέχεται να έχει ήδη ξεκινήσει από το επίπεδο των χρωματισμών του editor που χρησιμοποιούμε:

```
<script>  
    console.log("Hello world!");  
</script>
```



```
<script>  
    console.log(Hello world!");  
</script>
```



# Ομοιότητες Javascript-C

- Η χρήση του semicolon και των braces
- Η σύνταξη των: αριθμητικών πράξεων, **if**, **switch**, **for**, **while**
- Ίδιες λειτουργίες έχουν οι εντολές **break**, **continue**, **return**, **typeof**



# Παραδείγματα

```
for (i=0; i<20; i++) {  
    b = i + 2;  
}
```

```
while (i<5) {  
    x = a % b;  
    i++;  
}
```

```
if ((x==2) && (y>=7)) {  
    a++;  
    b = a + b;  
} else {  
    a = 2 * b;  
}
```

```
voteable = (age<18) ? 'Too young' : 'Old enough';
```

```
switch (day) {  
    case 6:  
        x = 'Today is Saturday';  
        break;  
    case 0:  
        x = 'Today is Sunday';  
        break;  
    default:  
        x = 'Looking forward to the Weekend';  
}
```





# Διάρθρωση

- Θα κάνουμε μια σύντομη εισαγωγή εισαγωγή στη γλώσσα και στη συνέχεια θα αναλύσουμε περαιτέρω τα χαρακτηριστικά της
- Με **κόκκινο** χρώμα θα επισημαίνουμε τα νέα χαρακτηριστικά της ECMAScript 2015 (**ES6**)

[kangax.github.io/compat-table/es6](https://kangax.github.io/compat-table/es6)



# Statements vs Expressions

- Τα statements “κάνουν πράγματα”
  - Ένα πρόγραμμα είναι μια σειρά από statements
- Τα expressions παράγουν values
- Σχόλια:
  - Οπουδήποτε αναμένεται statement μπορεί να χρησιμοποιηθεί και expression:

```
myFunction(3,4);
```



# Δήλωση μεταβλητών

- “Sloppy” mode (pre ES5):
  - Εμφάνιση νέας μεταβλητής χωρίς κάποια δήλωση: global scope (θα το αποφύγουμε)
- “Strict” mode (ES5):
  - Χρήση του keyword **var** σε συνδυασμό με το statement expression **'use strict'**: function scope με hoisting
- ES6:
  - Χρήση των keywords **const** και **let**: block scope



# Σχόλια

```
var a, b=5, c='test' ;
```

- Μπορούν να αλλάζουν τύπους κατά την εκτέλεση
- Τα ονόματα τους αρχίζουν από γράμμα, underscore ( `_` ) ή dollar sign ( `$` ) και απαγορεύεται να έχουν δεσμευμένα ονόματα:

[www.quackit.com/javascript/javascript\\_reserved\\_words.cfm](http://www.quackit.com/javascript/javascript_reserved_words.cfm)

[developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Reserved\\_Words](http://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Reserved_Words)

[mothereff.in/js-variables](http://mothereff.in/js-variables)



# Σχόλια

- Οι καθολικές μεταβλητές είναι στην πραγματικότητα properties ενός **global object**, που στην περίπτωση των browsers είναι το προκαθορισμένο object **window**  
(test: `var x=5; window.x===x;`)
- Αν χρησιμοποιήσουμε χωρίς να δηλώσουμε μια μεταβλητή μέσω του **var**, αυτή ορίζεται ως global variable (προσοχή!). Για να προστατευτούμε μπορούμε να χρησιμοποιήσουμε τη δήλωση:  
`'use strict';`  
για να έχουμε την εμφάνιση σφάλματος (ES5/strict).  
Το `'use strict';` μπορεί να χρησιμοποιηθεί και εντός της εμβέλειας μιας συνάρτησης.

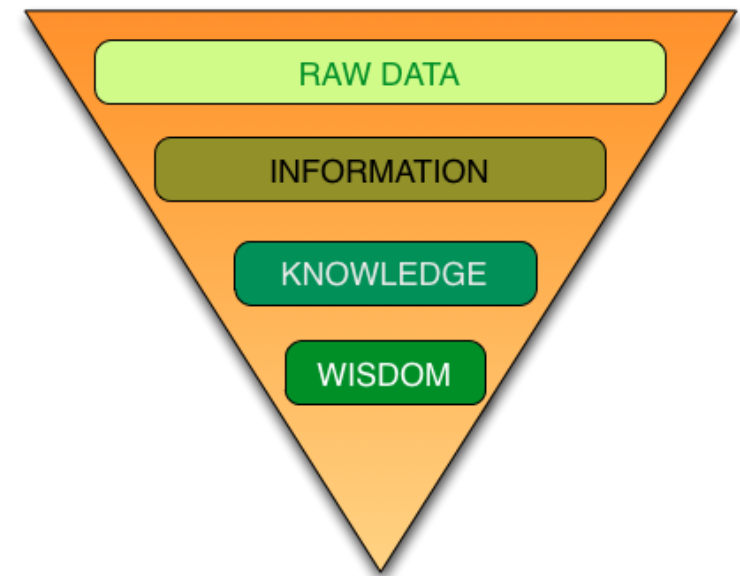
[developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict\\_mode](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode)



# Data types

- Primitive values:
  - number, string, boolean, null και undefined
- Complex values (objects):
  - array, plain object, function και ό,τι άλλο συναντήσουμε...

(ES6) νέο primitive: **Symbol**





# Numbers

- 64-bit real, 54-bit “safe” integers
  - π.χ. **3**, **9 . 81**, **2 . 998e8**
- Special numbers
  - **NaN**, **Infinity** και **-Infinity**
- Προσοχή στους δεκαδικούς! (test: 0.1+0.2)



# Strings

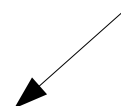
- Αλφαριθμητικά σε μονά ή διπλά εισαγωγικά
- Παρόμοιοι escape characters με τη C (π.χ. `\n`, `\t`, `\\`)
- Όταν σε ένα άθροισμα εμφανιστεί κάποιο string, έχουμε μετατροπή σε string και concatenation

– test: `1+2+3+'4'`

- (ES6) Template strings:

```
var s = 'Hello';  
console.log(`Program: ${s} world`);
```

Grave Accent!







# Boolean

- `true / false`
- “Falsy”: `false, 0, -0, NaN, '', null, undefined`
- “Truthy”: Όλα τα άλλα (συμπεριλαμβανομένων κενών arrays και objects)
- Binary Logical Operators (short-circuiting):
  - `&&` (and): If the first operand is falsy, return it. Otherwise, return the second operand.
  - `||` (or): If the first operand is truthy, return it. Otherwise, return the second operand.
- Έλεγχος ισότητας:
  - `==` : Ισότητα τιμών μετά από ενδεχόμενη μετατροπή (test: `0==false`)
  - `===` : Έλεγχος ταύτισης τιμών και τύπων (test: `0===false`)



# null και undefined

- Η έννοια της έλλειψης περιεχομένου.
- Τυπικά:
  - **undefined** σημαίνει no value ή no existence
  - **null** σημαίνει no object ή emptiness



# Συγκρίσεις και αναφορές

- Τα primitive values συγκρίνονται με βάση την τιμή τους
- Τα complex values (ως objects) είναι αναφορές σε θέσεις μνήμης και συγκρίνονται αυτές οι αναφορές

```
'use strict';  
const a = 7;  
var b = 7;  
console.log( a === b );           // true  
const e = ['uno', 'due'];  
let f = e;  
console.log( e === f );           // true  
console.log( e === ['uno', 'due'] ); // false  
f[0] = 'one';  
f = ['ένα', 'δύο'];  
console.log( e );                 // ['one', 'due']
```



# Arrays (literal notation)

- Μονοδιάστατα:

```
var a = [5, '7', true];  
var b = [];
```

Τι θα γίνει  
αν αλλάξουμε το  
length property;

- Δισδιάστατα:

```
var c = [[1,2],[3,4],[5,6]];  
var d = [[],[ ]];
```

- Δυναμική επαύξηση:

```
var e = ['a', 'b'];
```

```
e[25] = 'z'; // e[2] - e[24] : undefined
```



# Objects

*“An object is a container of properties,  
where a property has a name and a value”*

- Οι ιδιότητες των objects μπορούν να επαυξάνονται (ή να μειώνονται) κατά την εκτέλεση



# Εισαγωγή στα Objects

- Αν είστε ήδη εξοικειωμένοι με την “αντικειμενοστρέφεια” τα επόμενα θα σας φανούν εύκολα :-)
- Οι απλές (βαθμωτές) μεταβλητές των γλωσσών προγραμματισμού αποθηκεύουν στη μνήμη κάτι απλό και μεμονωμένο (π.χ. αριθμητική τιμή ή string).
- Η έννοια των objects αποδεικνύει τη χρησιμότητά της (π.χ.) όταν θέλουμε να αποθηκεύσουμε μια πιο σύνθετη μεταβλητή:
  - Παράδειγμα: Θέλουμε να έχουμε μια μεταβλητή που να αναφέρεται σε φυσικά πρόσωπα και να αποθηκεύει ενιαία στη μνήμη: (α) το μικρό τους όνομα, (β) το επίθετό τους, (γ) το έτος γέννησής τους.



- Αν στο προηγούμενο παράδειγμα σας ήρθε αυθόρμητα η λέξη `object`, είστε ήδη εξοικειωμένοι.
- Αν όχι, τότε πιθανόν να σκεφτήκατε μια μεταβλητή τύπου `Array`, με τρία στοιχεία: όνομα, επίθετο, έτος γέννησης.
- Στις κλασικές γλώσσες προγραμματισμού, όμως, (όχι στην JavaScript...) όλα τα στοιχεία του `Array` πρέπει να είναι του ίδιου τύπου!
- Πώς θα επιτύχουμε να έχουμε 2 strings και 1 number;



# Object properties

- Μια μεταβλητή τύπου object “συσκευάζει” ένα σύνολο άλλων μεταβλητών, οι οποίες ονομάζονται properties (ιδιότητες που αναφέρονται στο object).
- Παράδειγμα δήλωσης:

```
var father = {  
    firstName: 'Γεώργιος',  
    lastName: 'Δημητρίου',  
    yearOfBirth: 1958  
};
```

← κόμμα!





# Προσπέλαση των properties

- Η προσπέλαση των properties ενός object γίνεται με δύο τρόπους:
  - Dot notation:  
**father.lastName**
  - Bracket notation:
    - **father['lastName']**
- Ο δεύτερος τρόπος μας θυμίζει τα arrays σε μια συσχετιστική (associative) μορφή!



# Objects μέσα σε objects

- Αφού τα properties είναι ουσιαστικά μεταβλητές, καταλαβαίνουμε ότι κάποιο property μπορεί να είναι με τη σειρά του object:

```
var father = {  
  firstName: 'Γεώργιος',  
  lastName: 'Δημητρίου',  
  placeOfBirth: {  
    city: 'Θεσσαλονίκη',  
    country: 'Ελλάς'  
  },  
  yearOfBirth: 1958  
};  
console.log(father.placeOfBirth.city);
```



# Επαύξηση των properties

- Τα properties, εκτός από το να τροποποιούνται, μπορούν και να επαυξάνονται:

```
father.occupation = 'Λογιστής';  
console.log(father);
```



# Άρα... (εναλλακτικός τρόπος ορισμού)

κενό object

```
var father = {};
```

```
father.occupation = 'Λογιστής';  
father.firstName = 'Γεώργιος';  
father.lastName = 'Δημητρίου';  
father.yearOfBirth = 1958;
```



# Συναρτήσεις (functions)

*“The best thing about JavaScript is its implementation of functions”*

- Στη βασική τους μορφή είναι σαν τις συναρτήσεις οποιασδήποτε γλώσσας προγραμματισμού, με την επιπρόσθετη δυνατότητα να τις καλούμε με περισσότερα ή και με λιγότερα ορίσματα:

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

```
myVar = square(5);
```



# Πέρασμα ορισμάτων

- Τα ορίσματα στις συναρτήσεις της JavaScript “μεταβιβάζονται κατά τιμή” (pass by value). Ωστόσο, δεν ξεχνάμε ότι η τιμή των objects είναι μια αναφορά (reference) προς αυτά!

```
function testArguments(a, b, c) {  
    a    = 'changed';  
    b.s  = 'changed';  
    c    = { s: 'changed' };  
}
```

```
var a = 'unchanged';  
var b = { s: 'unchanged' };  
var c = { s: 'unchanged' };  
  
testArguments(a, b, c);
```



# function declaration vs expression

- Declaration: “παραδοσιακή” σύνταξη (statement)
- Expression (οπουδήποτε αναμένεται): δήλωση που δημιουργεί value
  - Anonymous function

```
function first() { };  
var second = function () { };  
console.log( typeof first === typeof second ); // true  
console.log( typeof first === 'function' );    // true
```



# Σχόλια

- Άρα οι συναρτήσεις μπορούν να χρησιμοποιηθούν όπως κάθε άλλο value:
  - Να αντιστοιχιστούν σε μεταβλητές
  - Να περαστούν ως ορίσματα σε άλλες συναρτήσεις, αλλά και να επιστρέψουν ως values
  - Να αποτελέσουν values σε object properties. Αυτές θα είναι οι μέθοδοι (methods) των objects.





# Παράδειγμα – Σχόλια

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

```
var myMathObject = {  
    pi: 3.14159265359,  
    square: function (x) {  
        return x * x;  
    }  
};
```

```
var myFactorial = function factorial(num) {  
    if (num < 0) { return -1; }  
    else if (num == 0) { return 1; }  
    else { return (num * factorial(num - 1)); }  
};
```

```
a = add(3,4);  
b = myMathObject.square(3);  
c = myFactorial(4);
```

- ▷ Στα expressions μπορούμε να δίνουμε προαιρετικά ονόματα στις συναρτήσεις για 2 πιθανούς λόγους:
  - ~ Η συνάρτηση είναι αναδρομική
  - ~ Να μην εμφανίζεται ως anonymous στον debugger



# ES6 method definitions

- Εκτός της σύνταξης:

```
myObj = {  
  myProp: 2,  
  myFunc: function (a,b) {  
  }  
}
```

- υποστηρίζεται πλέον και η σύνταξη:

```
myObj = {  
  myProp: 2,  
  myFunc(a,b) {  
  }  
}
```



# Variable scope

- Η εμβέλεια των μεταβλητών είναι η συνάρτηση μέσα στην οποία περιέχονται (δήλωση **var**) ή το block που τα περιέχει (δηλώσεις **const** και **let**)
- Η δήλωση των μεταβλητών μέσω **var** και τα function declarations “ανυψώνονται” (hoisted) στην κορυφή του σώματος της συνάρτησης
- Οι μεταβλητές και τα function declarations έξω από κάθε συνάρτηση είναι καθολικές (global) και πρέπει να αποφεύγονται με κάθε τρόπο (namespace pollution)



# ES6 variables και scope

	Hoisting	Scope
<code>var</code>	Declaration	Function
<code>let</code>	Temporal dead zone	Block
<code>const</code>	Temporal dead zone	Block
<code>function</code>	Complete	Block
<code>class</code>	No	Block
<code>import</code>	Complete	Module-global

```
let x = 1;
if ( true ) {
  console.log(x); // ReferenceError (TDZ)
  let x = 2;
}
```



# Fundamental objects

- Προϋπάρχοντα, δεδομένα objects που λειτουργούν τόσο ως static property namespaces όσο και ως constructors (εκτός των 2 πρώτων):
  - Math, JSON, Number, String, Boolean, Array, Object, Function, Date, ...
  - Error objects

[developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects)



# Μελέτη...

[eloquentjavascript.net](http://eloquentjavascript.net) (Part 1: Language)

- Εισαγωγή στον προγραμματισμό

[speakingjs.com/es5](http://speakingjs.com/es5)

- ES5 αναλυτικά

[exploringjs.com/es6](http://exploringjs.com/es6)

- ES6

- Meta: [jsbooks.revolunet.com](http://jsbooks.revolunet.com)
- Additional: [javascript.info](http://javascript.info)



# Άσκηση

- Ξεκινήστε από το δοθέν αρχείο **ex.html** και κατασκευάστε το αρχείο **ex.js** που θα κάνει τα ακόλουθα (όλες οι τυπώσεις θα γίνονται στην κονσόλα):
  - Θα τυπώνει τις 10 πρώτες δυνάμεις του 2
  - Θα ορίζει και θα χρησιμοποιεί μια συνάρτηση με 3 αριθμητικά ορίσματα και θα επιστρέφει το μεγαλύτερο όρισμα (δοκιμάστε στη συνέχεια να καλέσετε τη συνάρτηση με 3 strings)
  - Θα ορίζει και θα χρησιμοποιεί μια συνάρτηση με 2 αριθμητικά ορίσματα, τα οποία θα ελέγχει αν είναι ακέραιοι αριθμοί μεγαλύτεροι του μηδενός και θα υψώνει τον πρώτο στη δύναμη του δευτέρου. Σε κάθε άλλη περίπτωση θα επιστρέφει undefined.

ex.html

"1η άσκηση"

"1η δύναμη του 2: 2"

"2η δύναμη του 2: 4"

"3η δύναμη του 2: 8"

"4η δύναμη του 2: 16"

"5η δύναμη του 2: 32"

"6η δύναμη του 2: 64"

"7η δύναμη του 2: 128"

"8η δύναμη του 2: 256"

"9η δύναμη του 2: 512"

"10η δύναμη του 2: 1024"

"2η άσκηση"

"Κλήση συνάρτησης με ορίσματα: 10,14,9"

14

"Κλήση συνάρτησης με ορίσματα: 'aa','ba','ca'"

"ca"

"3η άσκηση"

"Κλήση συνάρτησης με ορίσματα: 3,3"

27

"Κλήση συνάρτησης με ορίσματα: 3.2,3"

undefined

"Κλήση συνάρτησης με ορίσματα: -3,3"

undefined

"Κλήση συνάρτησης με ορίσματα: 'aa','ba'"

undefined



# Document Object Model

*“The Document Object Model (DOM) is a cross-platform and language-independent convention (or API) for representing and interacting with objects in HTML, XML and SVG documents”*

- Η πρόσβαση της JavaScript στο περιεχόμενο, στην παρουσίαση και στη συμπεριφορά του





# Browser Object Model

- Κάθε καρτέλα του browser είναι ουσιαστικά ένα global object με όνομα **window**, το οποίο περιλαμβάνει ένα τεράστιο σύνολο από properties/methods που σχετίζονται με το ιστορικό, τη διεύθυνση, τα στοιχεία του browser, τις διαστάσεις αναπαράστασης κ.λπ. καθώς και ... το αναπαριστώμενο έγγραφο που δομείται με βάση το *Document Object Model*
- Μικρές εξαιρέσεις στην αντιστοίχιση καρτέλα-**window** αποτελούν:
  - οι ενσωματωμένες ιστοσελίδες (π.χ. μέσω **iframe**) που έχουν το δικό τους **window** (προσπελάζουν το εξωτερικό μέσω του object **window.parent**)
  - ορισμένα properties του **window** που επηρεάζουν ολόκληρο το παράθυρο του browser



# BOM - DOM APIs

- Αμφότερα τα BOM και DOM είναι εντέλει Application Programming Interfaces

[developer.mozilla.org/en-US/docs/Web/API](https://developer.mozilla.org/en-US/docs/Web/API)

- Οι επιμέρους απόγονοι του window (αλλά και το ίδιο) περιλαμβάνουν properties/methods, υλοποιώντας (implement) κάποιο αντίστοιχο interface του API



# Interfaces

- Το **window** object υλοποιεί το **Window** interface, που με τη σειρά του, εκτός των δικών του properties/methods, κληρονομεί ή υλοποιεί properties/methods των **EventTarget**, **WindowTimers**, **WindowBase64** και **WindowEventHandlers** interfaces
- Για παράδειγμα, το **EventTarget** interface υλοποιείται από objects που μπορούν να υποδεχθούν events (π.χ. **window**, **document**, **elements**)
- Ως εκ τούτου, τα properties/methods ενός interface παρουσιάζουν σε γενικότερο επίπεδο τα properties/methods των επιμέρους αντικειμένων (αναμενόμενο)



# Σχόλια

- Επισήμανση: όλα τα properties/methods του **window** είναι global variables, άρα μπορούν να χρησιμοποιηθούν και χωρίς την αναφορά **window.xxx**
- Τα επιμέρους objects (π.χ. **document**, **history**, **location** κ.λπ.) υλοποιούν αντίστοιχα interfaces
- Το DOM ξεκινά από το **document** object

[developer.mozilla.org/en-US/docs/Web/API/Window](https://developer.mozilla.org/en-US/docs/Web/API/Window)



# window object properties

- Όλες οι global variables του χρήστη!
- **document**: Object πρόσβασης στο έγγραφο του παραθύρου (DOM)
- **history**: Object πρόσβασης στο ιστορικό του browser
- **location**: Object χειρισμού της τρέχουσας διεύθυνσης του εγγράφου του browser
- **navigator**: Object πληροφοριών για τον browser
- **console**: Object πρόσβασης στην κονσόλα του browser
- **localStorage, sessionStorage**: Object αποθήκευσης-ανάγνωσης-χειρισμού τοπικών δεδομένων μόνιμα / ενόσω είναι ανοικτό το παράθυρο του browser. Θα τα γνωρίσουμε αναλυτικά αργότερα.
- **screen**: Object πρόσβασης στα στοιχεία της οθόνης όπου εμφανίζεται ο browser
- **innerHeight, innerWidth**: Number ύψους / πλάτους σε pixels του περιεχομένου του παραθύρου του browser, συμπεριλαμβανομένων των scroll-bars
- **pageXOffset, pageYOffset**: Number αριθμού pixels κατά τα οποία έχει γίνει scroll του παραθύρου οριζοντίως / κατακορύφως



# window object methods

- **print()**: Εκτύπωση του τρέχοντος εγγράφου
- **getSelection()**: Object ανάγνωσης του επιλεγμένου από τον χρήστη κειμένου (προσφέρεται και στο **document** object)
- **matchMedia()**: Object ελέγχου ενός media query string. Π.χ.  
`boolean matchMedia(' (min-width: 400px) ').matches`
- **setInterval()**, **clearInterval()**: Ορισμός / εκκαθάριση επαναλαμβανόμενης εκτέλεσης κάποιας συνάρτησης
- **setTimeout()**, **clearTimeout()**: Ορισμός / εκκαθάριση εκτέλεσης κάποιας συνάρτησης μετά από χρονικό διάστημα
- **getComputedStyle()**: Object ανάγνωσης όλων των εφαρμοσμένων CSS styles ενός element (θα επανέλθουμε...)



# location object

- **href**: Επιστρέφει ή θέτει το τρέχον URL (string)
- **hostname, pathname**: Επιστρέφει ή θέτει το τρέχον hostname / pathname (string)
- **protocol**: Επιστρέφει ή θέτει το τρέχον πρωτόκολλο (string)
- **assign(s)**: Φορτώνει ένα καινούριο έγγραφο-ιστοσελίδα με βάση το string **s**
- **replace(s)**: Αντικαθιστά το τρέχον έγγραφο-ιστοσελίδα με το string **s** χωρίς να τροποποιεί το ιστορικό (το πλήκτρο back δεν θα μας οδηγήσει στην αντικατασταθείσα σελίδα)
- **reload(b)**: Ανανέωση του τρέχοντος εγγράφου-ιστοσελίδας. Το προαιρετικό boolean **b**, αν είναι false (default), φορτώνει το έγγραφο από τη μνήμη cache και, αν είναι true, το ξανακατεβάζει από το server



# Παράδειγμα χρήσης `location.replace(s)`

- Αν αλλάξουμε host provider και θέλουμε η παλιά μας ιστοσελίδα να οδηγεί απευθείας στη νέα μας, μπορούμε να αξιοποιήσουμε τη μέθοδο **`location.replace(s)`**

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Redirection</title>
  </head>
  <body>
    <script>
      location.replace('https://www.uom.gr');
    </script>
  </body>
</html>
```





# document object

- Το object εντός του οποίου περιέχεται ολόκληρη η δομή της σελίδας μας
- Ένα έγγραφο HTML, αφού διαβαστεί από τον browser, μετατρέπεται σε ένα δένδρο με κόμβους, το οποίο μπορεί να υποστεί επεξεργασία

[www.w3.org/DOM](http://www.w3.org/DOM)

[developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model](http://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model)



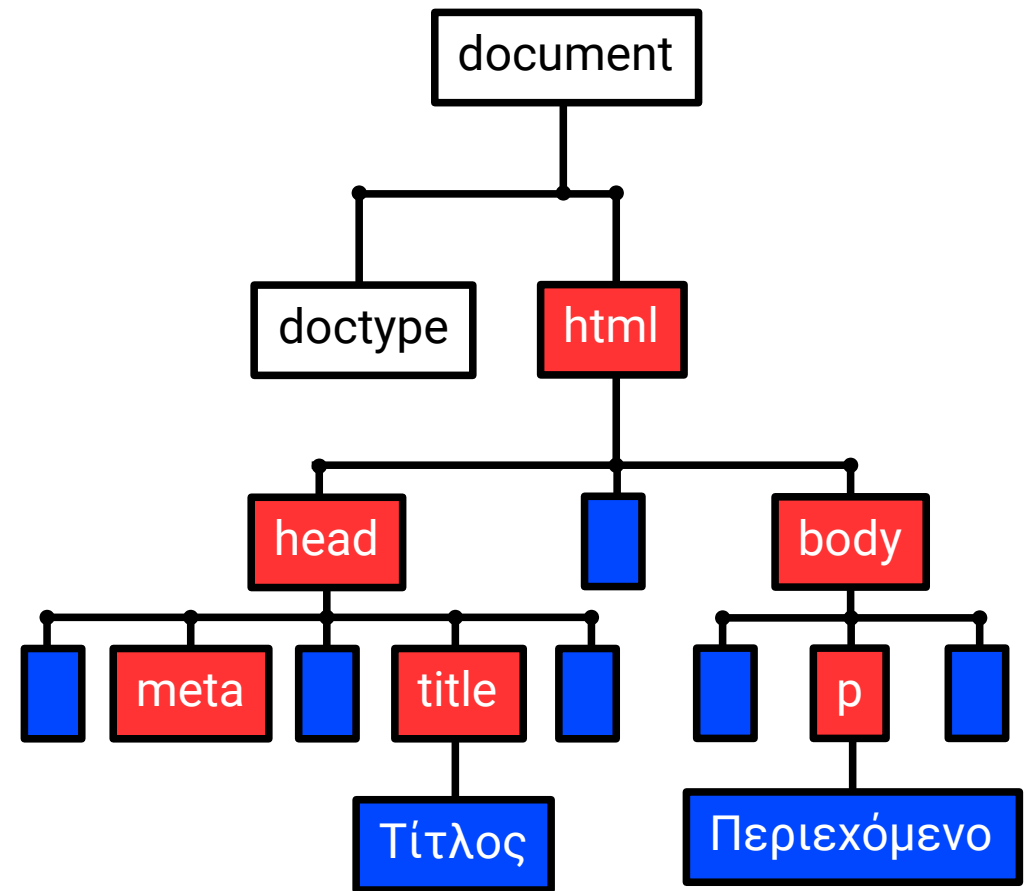
# HTML documents

- Τους κόμβους του δένδρου θα τους διακρίνουμε σε 3 κατηγορίες (αν και υπάρχουν και άλλες λιγότερο χρήσιμες):
  - document node (η ρίζα του δένδρου)
  - element nodes (π.χ.. **<body>**, **<a>**, **<p>**, **<style>**, **<html>** κ.λπ.)
  - text nodes (το απλό κείμενο εντός ενός element node, συμπεριλαμβανομένων κενών και enter)
- Όλοι οι κόμβοι υλοποιούν κάποιο αντίστοιχο interface: **Document**, **Element**, **Text**, αντίστοιχα
- Όλοι οι κόμβοι κληρονομούν από το **Node** interface
- Το document και τα element nodes κληρονομούν και από το **EventTarget** interface



# Δένδρο αντικειμένων

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Τίτλος</title>
  </head>
  <body>
    <p>Περιεχόμενο</p>
  </body>
</html>
```



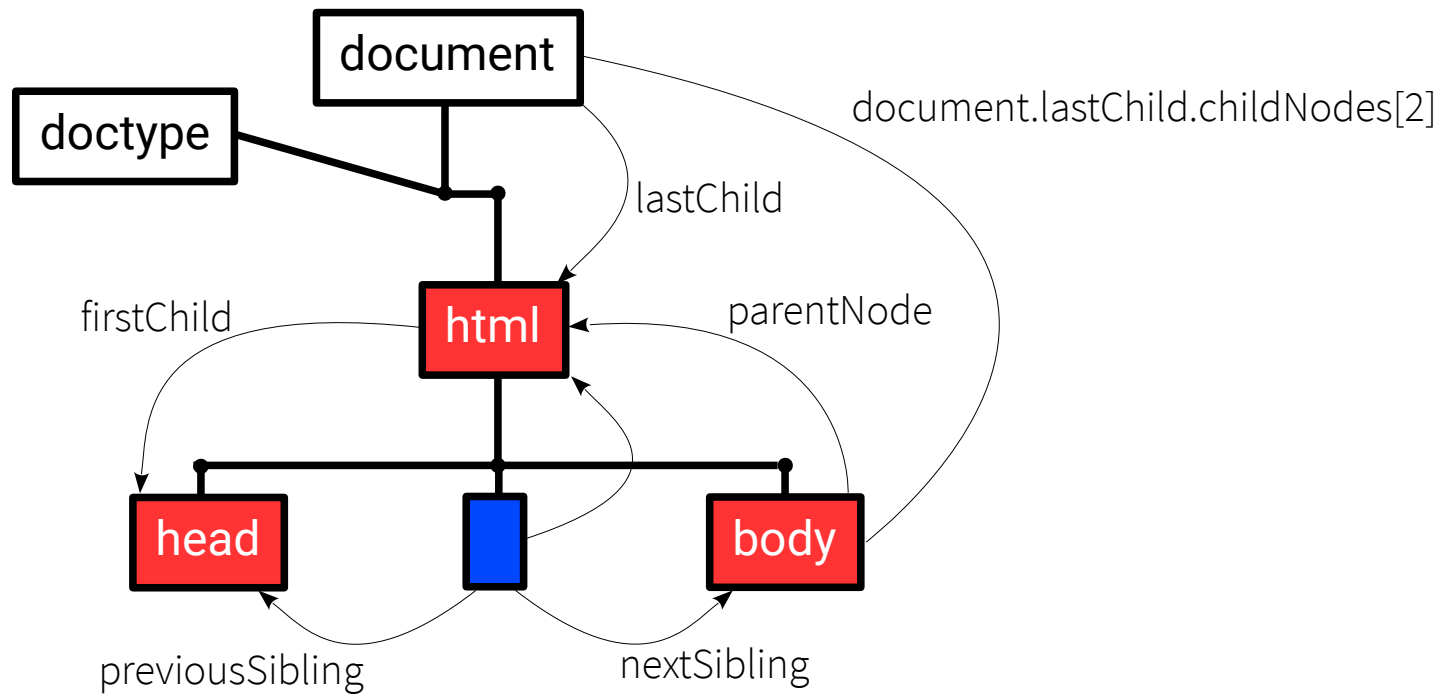
**Element nodes**, **Text nodes**

[software.hixie.ch/utilities/js/live-dom-viewer/](http://software.hixie.ch/utilities/js/live-dom-viewer/)



# Node Interfaces

- Μια που όλα τα στοιχεία του δένδρου κληρονομούν από το Node interface, αποκτά ενδιαφέρον η μελέτη του. Αν κάποιος κόμβος έχει απογόνους (document ή element) ή είναι ο ίδιος απόγονος (element ή text), τότε κληρονομεί και από τα ParentNode και ChildNode interfaces, αντίστοιχα
- Χρήσιμα properties (read only εκτός των 2 τελευταίων):
  - **firstChild, lastChild, previousSibling, nextSibling, parentNode** (parentElement)
  - **childNodes**
    - Array-like (Nodelist) δομή
  - **nodeName, nodeType**
    - #document, #text, P, BODY κ.λπ. / 1, 3, 9
  - ParentNode: **childElementCount, firstElementChild, lastElementChild, children**
    - Πλήθος, πρώτο, τελευταίο, array των περιεχόμενων elements
  - **textContent, nodeValue**
    - Το περιεχόμενο ως text node (nodeValue: #text only)
- Τα χρήσιμα methods αφορούν στην προσθαφαίρεση nodes και θα τα δούμε συνολικά



```
document.lastChild.childNodes[2].nodeName === 'BODY'; // true
```



# document object properties/methods

- **documentElement, head, body, title**
  - Άμεση πρόσβαση στα html, head, body elements και στο κείμενο του title
- **activeElement**
  - Άμεση πρόσβαση στο element που έχει focus
- **forms, images, links, scripts**
  - Λίστες (array-like) των αντίστοιχων στοιχείων του εγγράφου
- **hasFocus ( )**: Boolean για το αν υπάρχει focus στο έγγραφο
- Επιπρόσθετα, ενδιαφέρον έχουν οι ακόλουθες κατηγορίες που θα δούμε στη συνέχεια:
  - Methods πρόσβασης σε elements
  - Methods προσθαφαίρεσης nodes



# Πρόσβαση σε elements

- Η πρόσβαση στα elements μέσω της δενδρικής δομής είναι πολύ περίπλοκη, εξαιρετικά επιρρεπής σε σφάλματα και καθόλου δυναμική
- Μπορούμε να αποκτούμε πρόσβαση στα elements μέσω αμεσότερων μεθόδων





# document object methods πρόσβασης

- Πρόσβαση σε μοναδικό element:
  - `getElementById(str_id)`
  - `querySelector(str_selector)`
- Πρόσβαση σε element list (array-like):
  - `getElementsByClassName(str_className)`
  - `getElementsByTagName(str_tagName)`
  - `getElementsByName(str_name)`
  - `querySelectorAll(str_selector)`





# Σχόλια / παραδείγματα

- Τα methods `getElementsByClassName()`, `getElementsByTagName()`, `querySelector()`, `querySelectorAll()` αποτελούν methods και των elements, επιστρέφοντας descendant elements

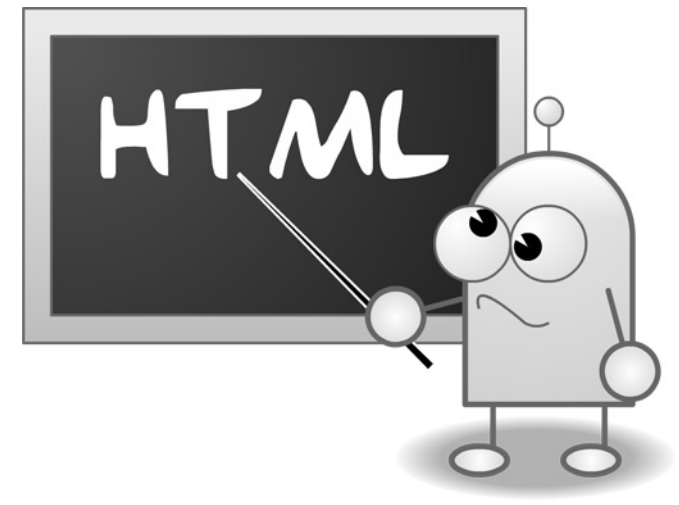
```
const elem = document.getElementById('myNiceID');  
console.log(document.querySelector('div > p').nodeName);  
const third = document.getElementsByClassName('red test')[2];  
const allPars = document.getElementsByTagName('p');  
const myRadioButtons = document.getElementsByName('radioName');  
const matches = document.querySelectorAll('div.note, div.alert');
```

- Προσοχή: Για λόγους παλαιάς συμβατότητας όλα τα id των elements είναι αρχικά global variables (άλλος ένας λόγος προσοχής στο namespace pollution)



# Elements

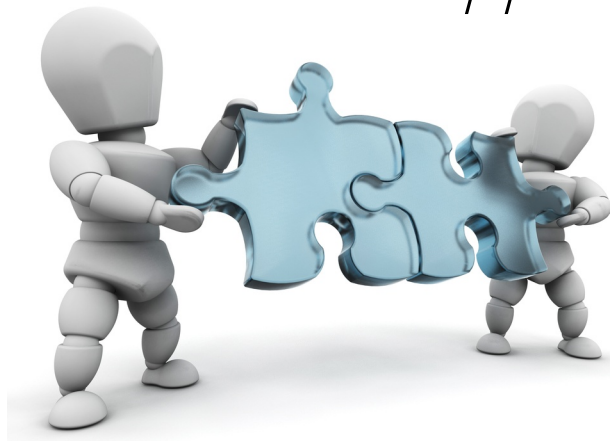
- Αφού αποκτήσουμε πρόσβαση στα elements, το ενδιαφέρον γι' αυτά αναπτύσσεται σε 3 τομείς (εκτός των events):
  - Διαχείριση attributes / properties
  - Διαχείριση CSS styles
  - Προσθαφαίρεση περιεχομένου





# Events

*“Event-driven programming is a programming paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs/threads. Event-driven programming is the dominant paradigm used in graphical user interfaces and other applications.”*



- Τελικά το κύριο μέρος του προγραμματισμού σε JavaScript επί του DOM (ή και εκτός αυτού... Node.js) γίνεται μέσω event listeners



# Σχόλια

- Μπορούμε να επιχειρήσουμε μια κατηγοριοποίηση σε events που προέρχονται από:
  - πράξεις του χρήστη (π.χ. click, keypress, scroll)
  - την ολοκλήρωση μιας διαδικασίας
  - την παρέλευση κάποιου χρονικού διαστήματος
- Η κατηγοριοποίηση δεν σημαίνει ότι κάποιο event μιας κατηγορίας δεν προέρχεται από event μιας άλλης κατηγορίας (π.χ. ένα χρονικό event μπορεί να προκαλέσει scroll της σελίδας και να πυροδοτηθεί ένα αντίστοιχο event)
- Τα events σχετίζονται με κάποιο από τα DOM-BOM objects



# Mouse events σε elements

- **mousedown**
  - Πίεση του αριστερού ή δεξιού πλήκτρου
- **mouseup**
  - Αφέθηκε το αριστερό ή δεξιό πλήκτρο
- **click**
  - Click του αριστερού πλήκτρου από το χρήστη
- **contextmenu**
  - Click του δεξιού πλήκτρου από το χρήστη
- **dblclick**
  - Double-click του χρήστη
- **mousemove**
  - Κίνηση του ποντικιού καθώς αιωρείται πάνω από το στοιχείο
- **mouseover**
  - Το ποντίκι ήρθε για πρώτη φορά επάνω από το στοιχείο
- **mouseout**
  - Το ποντίκι έφυγε έξω από το στοιχείο



# Περαιτέρω mouse events

- Υπάρχουν και άλλα mouse events:
  - drag, dragend, dragenter, dragleave, dragover, dragstart, drop, mousewheel, scroll
- αλλά δεν θα μας απασχολήσουν εδώ!



# Σχόλια

- Τα mouse events είναι ο κατ' εξοχήν τρόπος να μελετήσουμε τα χαρακτηριστικά και τον τρόπο λειτουργίας των events.
- Συγκεκριμένα θα δούμε:
  - Πώς ορίζουμε τη συμπεριφορά ενός event
  - Πώς “αναδύονται” (διαδίδονται) τα events
  - Πώς αναλύουμε τις ιδιότητες ενός event



# Event listener

- Προκειμένου να εκτελεστεί κάποιο τμήμα κώδικα όταν πραγματοποιείται κάποιο event, οφείλουμε να δημιουργήσουμε κάποιο αντίστοιχο event listener στο σχετιζόμενο object
- Γενικά, η μηχανή “παρακολουθεί” διαρκώς τη γέννηση events, τα βάζει σε μια σειρά και εκτελεί τα αντίστοιχα event listeners





# Παράδειγμα: click σε button (1/2)

```
<button>Press me...</button>
```

Μη συνιστώμενη μέθοδος: ορισμός των event attributes:

```
<button onclick="this.innerHTML='Thank you!';">Press me...</button>
```

ή

```
var button = document.querySelector('button');
```

```
button.onclick = function (evt) {  
    this.innerHTML = 'Thank you!';  
}
```



# Παράδειγμα: click σε button (2/2)

- Συνιστώμενη μέθοδος, που επιτρέπει την επέκταση των listeners:

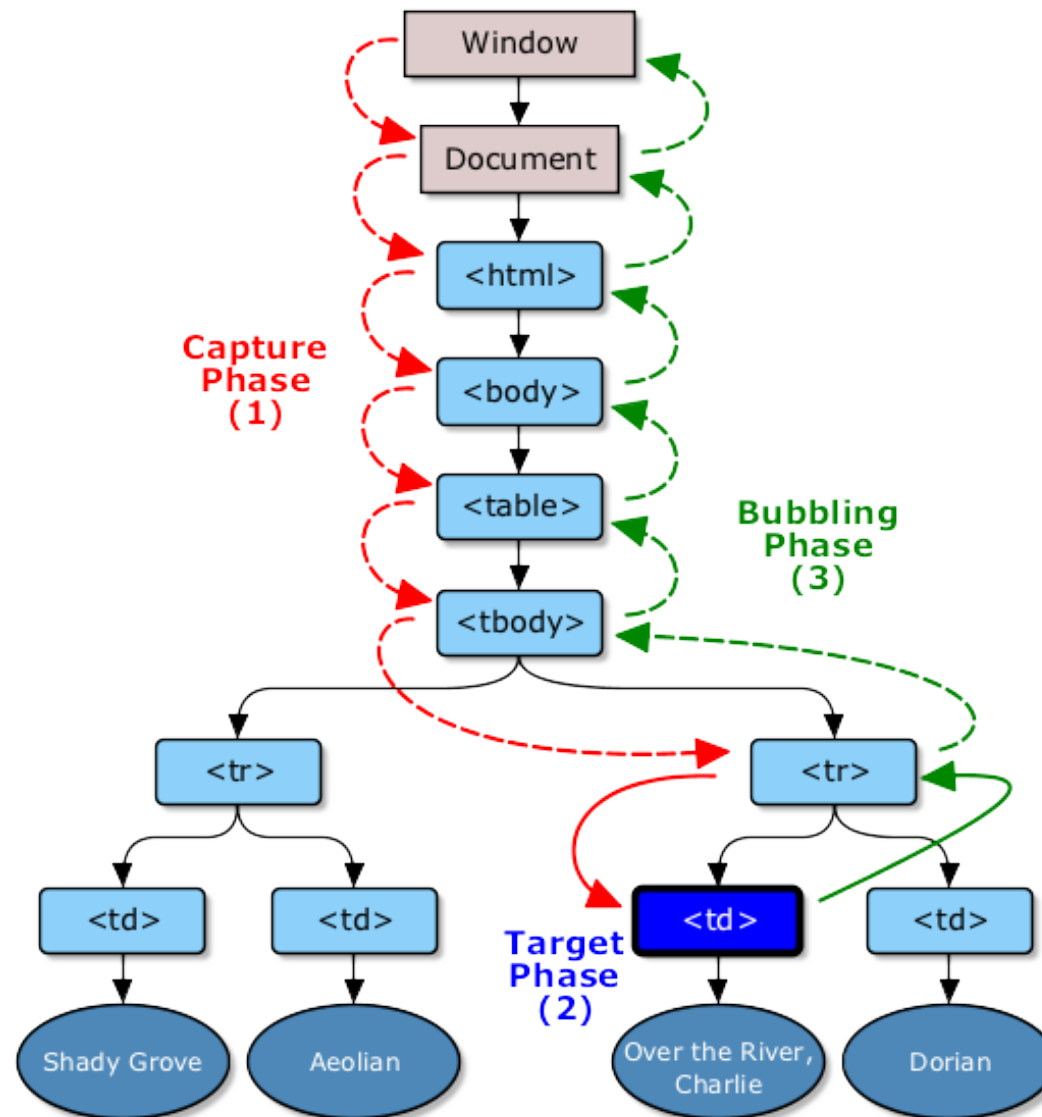
```
const button = document.querySelector('button');
```

```
button.addEventListener('click', function (evt) {  
    this.innerHTML = 'Thank you!';  
}, false);
```

- Η τρίτη (boolean) παράμετρος σχετίζεται με τη φάση πυροδότησης (true/false → capture/bubble)



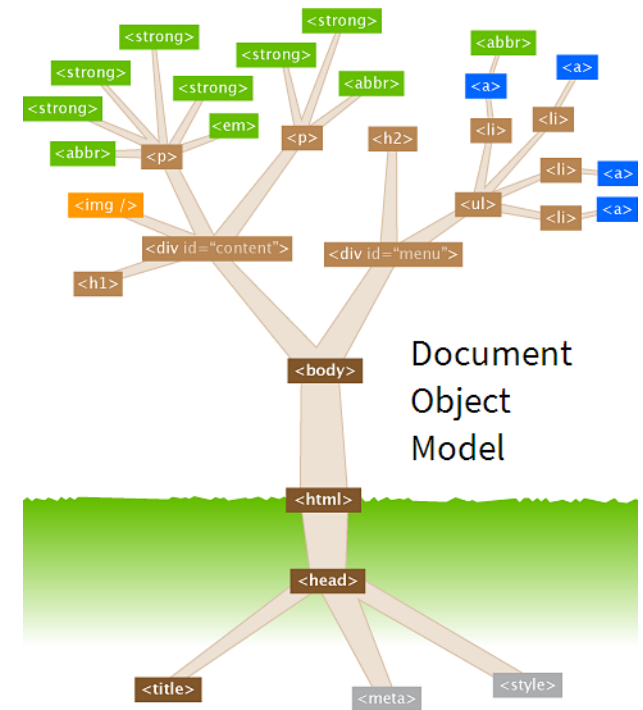
# Φάσεις πυροδότησης





[developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model)

- [www.w3.org/DOM/DOMTR](http://www.w3.org/DOM/DOMTR)



# Άσκηση



Ξεκινήστε από το δοθέν αρχείο **ex.html** και κατασκευάστε το αρχείο **ex.js** που θα υλοποιεί την λειτουργικότητα που φαίνεται δίπλα:

Hint:

Κάθε element object έχει το property **style** που, με τη σειρά του, έχει ως properties όλα τα CSS properties σε μορφή camelCasing. Παράδειγμα:

**document.body.style.fontSize**

Επαυξήστε με περισσότερα πλήκτρα ή elements, αλλάζοντας περαιτέρω την αισθητική και λειτουργικότητα.

Μια ωραία παράγραφος!

Κίτρινο φόντο σελίδας

Λευκό φόντο σελίδας

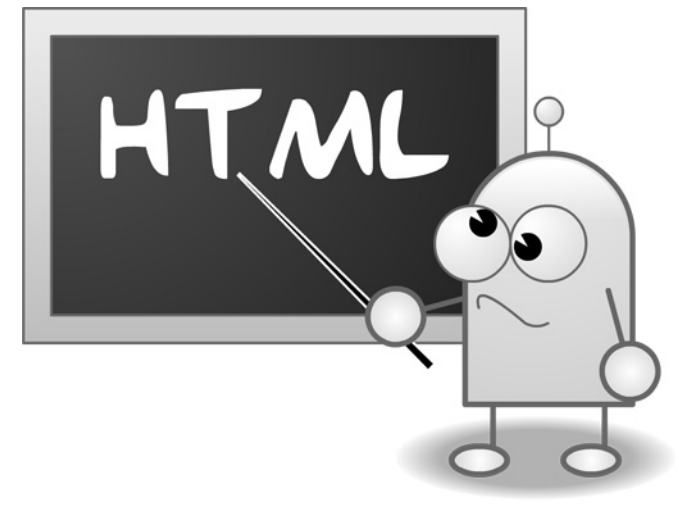
Κίτρινο φόντο παραγράφου

Λευκό φόντο παραγράφου



# Elements

- Αφού αποκτήσουμε πρόσβαση στα elements, το ενδιαφέρον γι' αυτά αναπτύσσεται σε 3 τομείς (εκτός των events):
  - Διαχείριση attributes / properties
  - Διαχείριση CSS styles
  - Προσθαφαίρεση περιεχομένου





# Attributes / Properties (1/2)

- Τα attributes ενός element προέρχονται από ρητές δηλώσεις της HTML
- Ωστόσο, όταν υλοποιηθεί το DOM, τα elements αποτελούν objects, που περιέχουν πολλά properties, ορισμένα από τα οποία αντιστοιχούν σε (lower-case) attributes του element (αρχικοποιούνται από αυτά και/ή έχουν 1-προς-1 αντιστοίχιση μαζί τους)
- Κάθε επιμέρους element κληρονομεί properties από μια σειρά αντίστοιχων interfaces (Element, HTMLElement, HTML**El\_Name**Element)

[developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model#HTML\\_interfaces](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model#HTML_interfaces)

- Π.χ. ένα **input** element κληρονομεί από τα interfaces Element, HTMLElement, HTMLInputElement)
- Στα αντίστοιχα interfaces θα βρούμε και περαιτέρω χρήσιμα properties που σχετίζονται με το συγκεκριμένο element (π.χ. **checked** και **defaultChecked** boolean properties κάποιου checkbox ή radio button)



# Attributes / Properties (2/2)

- Τα attributes είναι διαχειρίσιμα μέσω των ακόλουθων element object properties / methods (**attr**: string):
  - **attributes**: read-only array-like property, κάθε στοιχείο του οποίου είναι ένα object με χρήσιμα properties τα **name** και **value** του αντίστοιχα ορισμένου attribute
  - **hasAttributes()**, **hasAttribute(attr)**: boolean ελέγχου ύπαρξης οποιουδήποτε / κάποιου συγκεκριμένου attribute
  - **getAttribute(attr)**, **setAttribute(attr, value)**, **removeAttribute(attr)**: Ανάγνωση / εγγραφή / διαγραφή attributes (για boolean attributes οποιοδήποτε **value** τα κάνει **true** και γίνονται **false** μέσω του **removeAttribute**)
- Η 1-προς-1 αντιστοίχιση attributes-properties σημαίνει ότι μπορούμε να αλλάζουμε οποιοδήποτε από αυτά, αλλά σε ορισμένες περιπτώσεις (π.χ. στοιχεία φορμών **value** ή **checked**) τα attributes διατηρούν τις αρχικές τιμές της HTML, ενώ τα properties έχουν τις τρέχουσες ορισθείσες τιμές





# Σχόλια / παραδείγματα

- Η διαχείριση των attributes μέσω αντίστοιχων properties είναι απλούστερη και γενικότερη (προτιμήστε την), ωστόσο:
  - είναι αποκλειστικά case-sensitive
  - όταν αναφέρονται σε διεύθυνση (π.χ. **href**, **src**) αναφέρουν το πλήρες URL του αντίστοιχου πόρου

```

```

```
....
```

```
const el = document.querySelector('img');  
console.log(el.alt === el.getAttribute('ALT')); // true  
el.alt = 'Nice picture';  
console.log(el.getAttribute('alt')); // "Nice picture"  
console.log(el.src === el.getAttribute('src')); // false  
el.setAttribute('title', 'Simple Image');  
el.removeAttribute('alt');
```



# Διαχείριση CSS styles (1/2)

- Σε ό,τι αφορά τη διαχείριση των classes του element, χρήσιμα είναι τα παρακάτω element object properties:
  - **className**: CSS class (reserved word attribute) του element (read/write space delimited string)
  - **classList**: Array-like συλλογή των CSS classes του element, την οποία μπορούμε να διαχειριστούμε με τα ακόλουθα δικά της methods, που δέχονται strings ως ορίσματα:
    - **add(s)**: Προσθήκη κλάσης
    - **remove(s)**: Αφαίρεση κλάσης
    - **toggle(s)**: Εναλλαγή κλάσης
    - **contains(s)**: Boolean ελέγχου ύπαρξης κλάσης



# Διαχείριση CSS styles (2/2)

- Τα element objects έχουν το (αναμενόμενο) attribute-property **style**, που όμως ξεπερνά την κλασική έννοια του in-line CSS styling που μάθαμε
- Συγκεκριμένα, πρόκειται για ένα object που έχει ως properties όλα τα γνωστά CSS properties σε γραφή camelCase
- Προσοχή: χρησιμοποιείτε το για θέση και όχι για ανάγνωση ιδιοτήτων CSS, μια που συγκεντρώνει μόνον τα in-line CSS properties
- Για την ασφαλή ανάγνωση των CSS properties ενός element προτιμήστε το window object method **getComputedStyle(element)**, που επιστρέφει object παρόμοιο με το προαναφερθέν **style**



# Παραδείγματα

```
.....  
<style>  
  p { font-size: 26px; }  
</style>  
</head>  
<body>  
  <p id="someID">Κείμενο</p>  
.....
```

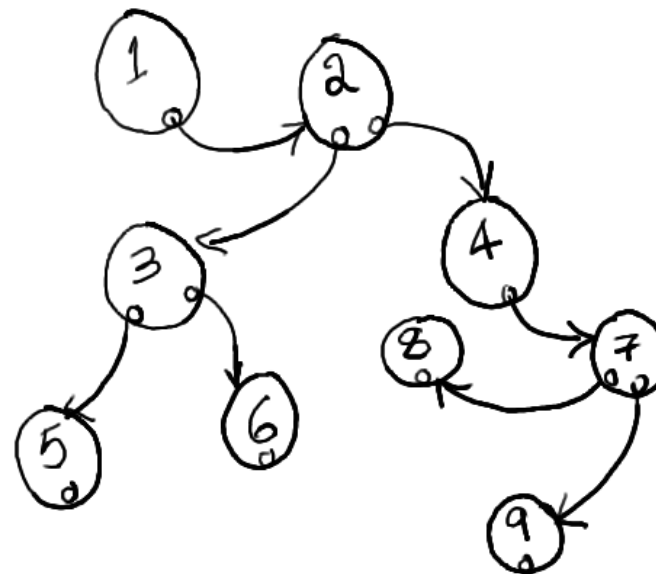
```
const el = document.getElementById('someID');  
const elStyle = getComputedStyle(el);
```

```
el.style.fontWeight = 'bold';  
el.style.backgroundColor = 'rgb(255,0,0)';  
if (el.style.fontSize === '26px') { // NO GOOD !!  
  el.style.fontSize = '46px';  
}  
if (elStyle.fontSize === '26px') {  
  el.style.fontSize = '46px';  
}
```



# Προσθαφαίρεση περιεχομένου

- Οι 3 συνηθέστερες περιπτώσεις είναι:
  - Να αλλάξουμε το περιεχόμενο ενός element
  - Να διαγράψουμε ολοσχερώς ένα element
  - Να δημιουργήσουμε ένα νέο element





# Διαθέσιμα εργαλεία

- `document.createElement(str), document.createTextNode(str)`
  - Επιστρέφουν `element / text nodes` που μπορούν στη συνέχεια να προστεθούν σε κάποιο `element` (χρήση και για `image preloading`)
- Element object property `innerHTML`
  - Ανάγνωση ή εγγραφή ολόκληρου του περιεχομένου του `element` (βολικότατο)
- Element object method `insertAdjacentHTML(position, text)`,
  - Προσθήκη HTML string `text`, ανάλογα με το string `position` (`'beforebegin'`, `'afterbegin'`, `'beforeend'`, `'afterend'`) (επίσης βολικότατο)
- Element object methods `appendChild(child), removeChild(child), replaceChild(newChild, child), insertBefore(child, existingChild)`
  - Προσθήκη κόμβου / αφαίρεση κόμβου / αντικατάσταση κόμβου / προσθήκη πριν από κόμβο (επιστρέφεται το `child`, οι μετακινήσεις υπαρχόντων `nodes` γίνονται αυτόματα)
- Το node object method `cloneNode(deep)`
  - Επιστρέφει αντίγραφο του κόμβου ενδεχομένως περιλαμβάνοντας τα περιεχόμενά του (boolean `deep=true`)



# Σχόλια

- Το **innerHTML** property και το **insertAdjacentHTML** method χρησιμοποιούνται συχνότατα
- Απαιτείται προσοχή σε πιθανά κενά ασφαλείας (cross-site scripting)
- Αν πρόκειται να προσθέσετε με methods ένα νέο περίπλοκο element, πρώτα κατασκευάστε το πλήρως και προσθέστε το ως τελευταία κίνηση
- Η χρήση του **cloneNode()** απαιτεί προσοχή διότι αντιγράφει όλα τα attributes (π.χ. duplicate IDs)



# Παραδείγματα

```
let p = [ document.getElementById('firstID'),  
          document.getElementById('secondID') ];  
  
p[0].innerHTML = 'Goodbye <span>cruel</span> world...';  
  
let t1 = document.createTextNode('Goodbye '),  
    t2 = document.createTextNode('cruel'),  
    t3 = document.createTextNode(' world...'),  
    s   = document.createElement('span');  
  
s.appendChild(t2);  
p[1].appendChild(t1);  
p[1].appendChild(s);  
p[1].appendChild(t3);  
  
p[0].innerHTML === p[1].innerHTML; // true
```





# Επιστροφή στα events

- Η μέθοδος **addEventListener()** ανήκει στο EventTarget interface, το οποίο υλοποιείται από διάφορα objects, συμπεριλαμβανομένων των **window**, **document** και των elements
- Η διαδοχική χρήση της **addEventListener()** για το ίδιο event στο ίδιο object επαυξάνει τις πράξεις που θα εκτελεστούν κατά την πυροδότηση
- Η μέθοδος **removeEventListener()** αναιρεί κάποια αντίστοιχη **addEventListener()** και εξυπηρετείται καλύτερα αν αποφύγουμε τη χρήση anonymous function



# Αλληλεπίδραση με εικόνες

- Καθώς μπορούμε να μεταβάλλουμε τα attributes οποιουδήποτε element:

**setAttribute(s1, s2)**

- προφανώς μπορούμε και να τροποποιούμε τις εικόνες ενός εγγράφου επεμβαίνοντας απλώς στο href attribute.
- Σχόλιο: ας θυμηθούμε και τα προϋπάρχοντα arrays **links** και **images**.



<p>Εικόνες ζώων:

<a id="penguin" href="#">πιγκουίνος</a>,

<a id="cow" href="#">αγελάδα</a>,

<a id="elephant" href="#">ελέφαντας</a>.</p>

<p style="text-align: center;"></p>

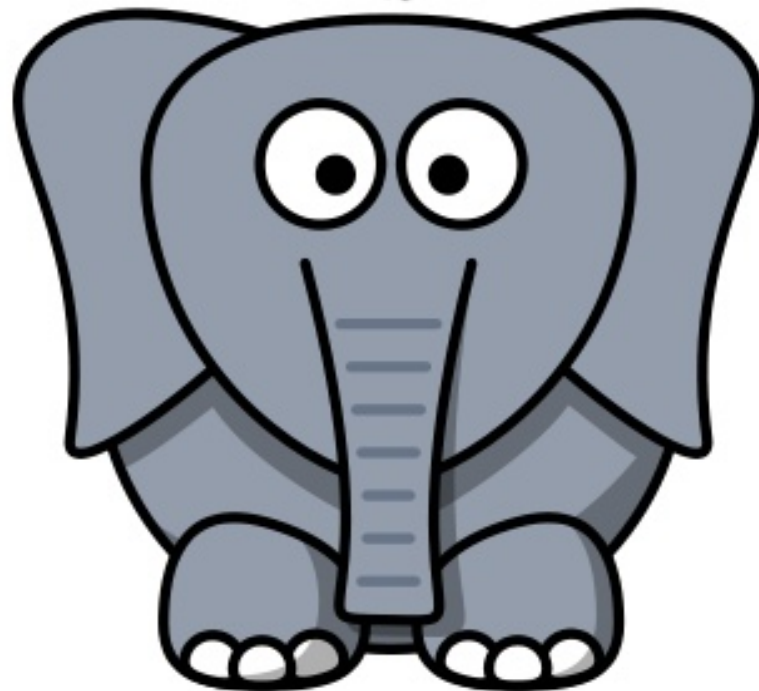
<script>

var i;

```
for (i = 0; i < document.links.length; i++) {  
    document.links[i].addEventListener('click', function (ev) {  
        document.images[0].setAttribute('src', this.id+'.png');  
    });  
}
```

</script>

Εικόνες ζώων: πιγκουίνος, αγελάδα, ελέφαντας.





# Event object

- Σε κάθε event listener function περνά ως όρισμα ένα αντικείμενο που υλοποιεί το Event interface
- Χρήσιμες μέθοδοί του:
  - **preventDefault()**: Αποτρέπει την default συμπεριφορά του browser (π.χ. anchors, checkboxes)
  - **stopPropagation()**: Αποτρέπει την περαιτέρω διάδοση του event στις επακόλουθες φάσεις
  - **stopImmediatePropagation()**: Όπως η προηγούμενη αλλά επιπρόσθετα αποτρέπει και την οποιαδήποτε εκτέλεση άλλων event listeners που έχουν επαυξήσει το συγκεκριμένο event στο συγκεκριμένο object (πιο άμεσο)
- Αν ορίζετε listeners μέσω των on... attributes (DOM0 way), μπορείτε να κάνετε **return false** σε κάποιο event listener function, κάτι που ισοδυναμεί με την κλήση αμφότερων των **preventDefault()** και **stopPropagation()** (αποφύγετέ το)



# Event object properties

- **bubbles, cancelable, defaultPrevented, isTrusted**: boolean που επισημαίνουν αν το event αναδύεται, μπορεί να ακυρωθεί, έχει αποτραπεί η default πράξη, προέρχεται από τον browser και όχι από script
- **eventPhase**: number 4 δυνατών τιμών: Event.NONE = 0, Event.CAPTURING\_PHASE = 1, Event.AT\_TARGET = 2, Event.BUBBLING\_PHASE = 3
- **currentTarget, target**: το object του event listener function / το object του target phase
- **timeStamp**: number που δηλώνει milliseconds (χρήσιμο για αφαιρέσεις)
- **type**: string που δηλώνει το event
- Στα παραπάνω methods και properties προστίθενται και εκείνα που αφορούν το κάθε συγκεκριμένο event (event subclasses)

[developer.mozilla.org/en-US/docs/Web/API/Event](https://developer.mozilla.org/en-US/docs/Web/API/Event)



# Mouse event properties

- **button**
  - Αριθμός: 0: αριστερό πλήκτρο, 1: μεσαίο πλήκτρο, 2: δεξιό πλήκτρο
- **clientX, clientY**
  - Αριθμός σε pixels της απόστασης του ποντικιού οριζοντίως / κατακορύφως από την επάνω αριστερά γωνία του εγγράφου
- **screenX, screenY**
  - Αριθμός σε pixels της απόστασης του ποντικιού οριζοντίως / κατακορύφως από την επάνω αριστερά γωνία της οθόνης



# Events

[developer.mozilla.org/en-US/docs/Web/Events](https://developer.mozilla.org/en-US/docs/Web/Events)

- Μεγάλο πλήθος
- Πολλές κατηγοριοποιήσεις





# Event delegation

- Όταν πρόκειται να αντιστοιχίσουμε παρόμοια event listeners σε πολλαπλά objects (π.χ. κελιά πίνακα) είναι προτιμότερο να αντιστοιχίσουμε ένα event listener σε κάποιο parent object (π.χ. πίνακα) και να χειριζόμαστε το εκάστοτε αντικείμενο μέσω του event object property **target**
- Οφέλη: Λιγότερα event listeners, συνεπής αντίδραση όταν προστίθενται elements δυναμικά
- Αν και συνήθως το **target** αρκεί, σε περίπτωση που πρέπει να αναφερθούμε σε κάποιο parent element, μπορούμε απλά να “ανεβούμε” το DOM tree μέσω κάποιου **while**





# JavaScript animation

- Εκτός από την εναλλαγή CSS ιδιοτήτων που εμπεριέχουν animation, μπορούμε να δημιουργήσουμε κίνηση και μέσω της JavaScript:
  - **setTimeout()**, **setInterval()**
- Προφανώς, η λύση της JavaScript υπερέχει όταν θέλουμε να έχουμε προγραμματιστικό έλεγχο της κίνησης (π.χ. παιχνίδια)
- Αποτελεσματικότερη μέθοδος δημιουργίας JavaScript animation είναι η χρήση του window object method **requestAnimationFrame()**



# Άσκηση 1

- Κατασκευάστε έναν απλό calculator της μορφής:

3 4

Calculate...

Result:

3 4

Calculate...

Result: **12**

3aa 4

Calculate...

3 / 0

Calculate...

Result: **NaN**

Result: **Infinity**



# Άσκηση 2

- Απομονώστε τον κώδικα κίνησης κύκλου από το αρχείο **events\_showcase.html** και προσπαθήστε να τον τροποποιήσετε / εμπλουτίσετε έτσι ώστε να λειτουργεί πιο “ομαλά” και με διαγώνιες κινήσεις.