



# CSS Cascading Style Sheets

**Αθανάσιος Ανδρούτσος**



# Περιεχόμενο HTML σελίδων

Προγραμματισμός στο Web

- Αυτό που έχουμε δει μέχρι στιγμής είναι τι είναι η HTML και πως χρησιμοποιείται για να εισάγουμε περιεχόμενο σε HTML documents
- Είδαμε το πώς μπορούμε με τη χρήση HTML στοιχείων (HTML Elements) να εισάγουμε **πολυμεσικό περιεχόμενο**, όπως κείμενο, **επικεφαλίδες** `<h1> ... <h6>` και **παραγράφους** `<p>`, **εικόνες** `<img>`, **λίστες** `<ul>` `<ol>` `<dl>`, **πίνακες** `<table>`, **υπερσυνδέσμους** `<a>`, και **φόρμες** `<form>`
- Μπορούμε να εισάγουμε και video και audio (podcast)



# Attributes - Ιδιότητες

- Είδαμε επίσης ότι μπορούμε να εισάγουμε και ιδιότητες στα opening tags που εξειδικεύουν περαιτέρω την βασική λειτουργία των HTML elements
- Για παράδειγμα έχουμε δει: **href**, **target**, **src**, **alt**, **width**, **height**, **style** και υπάρχουν και άλλες ιδιότητες

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, ini
7   <title>Document</title>
8 </head>
9 <body>
10  <p title="I am mouse over Lorem">Lorem ipsum dolor sit
11    Perferendis aspernatur repellat quod reiciendis ne
12    similique quasi est optio. Dolorum voluptatem repe
13    ut soluta dolor ad eum, sed explicabo beatae?
14  </p>
15 </body>
```

Για παράδειγμα αν  
θέλουμε να  
εμφανίσουμε mouse-  
over text μπορούμε να  
το κάνουμε με την  
ιδιότητα **title**



# Video (1)

## Προγραμματισμός στο Web

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10  <video width="320" height="240" controls>
11    <source src="E:\CodingFactory\videos\java\Java14322.mp4" type="video/mp4">
12    Your browser does not support the video tag.
13  </video>
14
15  <br>
16
17  <iframe width="560" height="315" src="https://www.youtube.com/embed/u4ZoJKF_VuA"
18    title="YouTube video player" frameborder="0" allow="accelerometer; autoplay;
19    clipboard-write; encrypted-media; gyroscope; picture-in-picture"
20    allowfullscreen>
21  </iframe>
22 </body>
23 </html>
```

- Συνήθως δίνουμε και διαστάσεις width και height
- Αν δεν βρεθεί το embedded video, μπορούμε να δώσουμε και μήνυμα

**<video>** tag για embedded videos και **<iframe>** tag από YouTube (copy/paste από το Share) και γενικά από Streaming services



# Video (2)

## Προγραμματισμός στο Web

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10   <video width="320" height="240" controls>
11     <source src="E:\CodingFactory\videos\java\Java14322.mp4" type="video/mp4">
12     Your browser does not support the video tag.
13   </video>
14
15   <br>
16
17   <iframe width="560" height="315" src="https://www.youtube.com/embed/u4ZoJKF_VuA"
18     title="YouTube video player" frameborder="0" allow="accelerometer; autoplay;
19     clipboard-write; encrypted-media; gyroscope; picture-in-picture"
20     allowfullscreen>
21   </iframe>
22 </body>
23 </html>
```

Το `<iframe>` έχει κάποια attributes όπως `frameborder`, που είναι ένα περίγραμμα γύρο από το video, με δύο τιμές 0 (`no-border`) και 1 (`border-on`). Η ιδιότητα αυτή είναι depreciated στην HTML5, χάριν του CSS Styling

Το `allow` περιλαμβάνει `accelerometer` (δουλεύει με sensor επιτάχυνσης), `gyroscope` (προσανατολισμός συσκευής, `portrait`, `landscape` σε κινητά), `picture-in-picture` για small video όταν βγαίνουμε από το YouTube



# Video (3)

```
<body>
```

```
<iframe width="560" height="315" src="https://www.youtube.com/embed/S0-zTc5Jwxw?mute=1&autoplay=1" title="You  
frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope;  
picture-in-picture" allowfullscreen></iframe>
```

```
</body>
```

- Το `autoplay = "1"` για να παίζει το βίντεο μόλις φορτώνει η σελίδα για λόγους βελτίωσης της εμπειρίας του χρήστη, γίνεται συνήθως `ignore` από όλους τους σύγχρονους browsers (και σίγουρα από τον Chrome)
- Μόνο αν έχουμε κάνει ταυτόχρονα `mute="1"` στην γραμμή URL, ξεκινάει χωρίς φωνή



# Video

- Όταν χρησιμοποιούμε βίντεο, θα θέλαμε ιδανικά στο source να έχουμε το URL κάποιου streaming server και όχι ένα απλό αρχείο βίντεο
- Τα απλά αρχεία βίντεο πρέπει να κατέβουν (download) στον υπολογιστή μας και να εκτελεστούν τοπικά, κάτι το οποίο είναι χρονοβόρο, μειώνει την εμπειρία του χρήστη και δεν μπορεί να χρησιμοποιηθεί σε live streaming
- Οι streaming servers (YouTube, Vimeo, κλπ.) κατεβάζουν σταδιακά το βίντεο (δεν αποθηκεύεται τοπικά), το οποίο εκτελείται ταυτόχρονα, αφού γίνεται και λίγα δευτερόλεπτα buffering ώστε να εξισορροπηθούν τυχόν καθυστερήσεις του δικτύου, χωρίς να γίνει degrade η ποιότητα υπηρεσίας (quality of service) του βίντεο.
- Η καθυστέρηση (latency) και το bandwidth (χωρητικότητα γραμμής) είναι οι δύο βασικές παράμετροι που επηρεάζουν την ποιότητα υπηρεσίας σε διαδικτυακές υπηρεσίες



# Audio

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
10     <audio controls>
11         <source src="E:\CodingFactory\audio\lidakis.mp3" type="audio/mpeg">
12         Your browser does not support the audio element.
13     </audio>
14 </body>
15 </html>
```

- Με το `<audio>` tag εισάγουμε ήχο, που μπορούμε να κατεβάσουμε από διάφορα sites





# Streaming vs live streaming

Προγραμματισμός στο Web

- Το streaming διαφόρων γεγονότων του πραγματικού κόσμου σε πραγματικό χρόνο (π.χ. ένα συνέδριο ή ένα αθλητικό γεγονός) ονομάζεται live streaming
- Η τεχνολογία είναι η ίδια (streaming) απλά θα πρέπει το μηχάνημα (π.χ. κάμερα υπολογιστή) που λαμβάνει την εικόνα να συνδέεται με τον streaming server και στη συνέχεια ο streaming server να μεταδίδει 'live' το βίντεο
- Open source λογισμικό που μπορεί να κάνει κάτι τέτοιο είναι το OBS (Open Broadcaster Software, <https://obsproject.com/>) με τις κατάλληλες YouTube ρυθμίσεις, αφού πρώτα φτιάξουμε κανάλι στο YouTube (μας δίνεται ένα URL και ένας κωδικός, που εισάγουμε στο OBS)

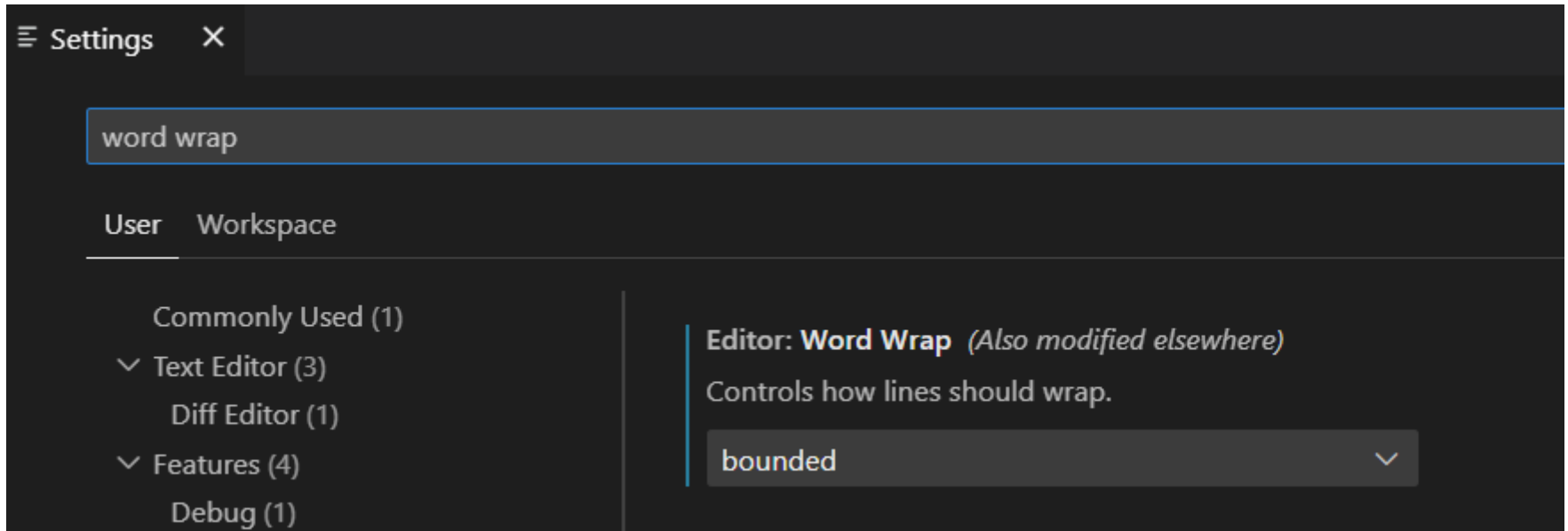


# Browser default styling (1)

- Τα HTML documents είναι αναγνώσιμα από ένα Web Browser.
- Τα Headings παρουσιάζονται μεγαλύτερα από το κανονικό κείμενο, οι παράγραφοι αφήνουν μία κενή γραμμή πριν και μετά, οι σύνδεσμοι έχουν χρώμα και υπογράμμιση ώστε να ξεχωρίζουν από το απλό κείμενο
- Η μορφή του κειμένου που βλέπουμε είναι τα default settings του Browser. Δηλαδή, βασική μορφοποίηση που ο browser εφαρμόζει στο HTML ώστε να μπορεί να είναι αναγνώσιμο ακόμα και αν δεν έχει εφαρμοστεί ειδικότερο styling από τον δημιουργό του εγγράφου



# Word wrap



- Στα Settings αναζητούμε για **Word Wrap** και επιλέγουμε **bounded** ώστε το *Lorem ipsum* και άλλες μεγάλες σε μήκος γραμμές να αλλάζουν αυτόματα γραμμή στους 80 χαρακτήρες



# Browser default styling (2)

```
<> default-styling.html X
chapter2 > <> default-styling.html > html > head > title
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Coding Factory</title>
8  </head>
9  <body>
10
11      <h1>Hello Coding Factory</h1>
12      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Modi molestias ea
13          optio sapiente provident delectus reiciendis blanditiis iure earum culpa
14          ducimus rerum nulla officia nobis magnam recusandae, eaque praesentium velit!
15      </p>
16
17      <a href="https://codingfactory.aueb.gr/">CF -- Links are underlined, unvisited links are blue</a>
18
19      <ul>
20          <li>Java</li>
21          <li>SQL</li>
22          <li>Web</li>
23      </ul>
24
25      <p>Αυτό είναι το default styling του browser</p>
26
27  </body>
28  </html>
```

- Έστω ο παραπάνω HTML κώδικας
- Είναι μόνο περιεχόμενο, δεν έχει κάτι ιδιαίτερο όσο αφορά το styling
- Επομένως, θα εμφανιστεί (θα γίνει render) με το βασικό styling του browser (που μπορεί να αλλάζει λίγο μεταξύ των διαφόρων browsers)



# Browser default styling (3)



- Το default styling όλων των στοιχείων κειμένου `<p>` είναι 16px, black, TimesNewRoman, μία γραμμή κενή από πάνω (16px) και μία γραμμή κενή κάτω



# Browser default styling (4)



- Το default styling των links όταν είναι unvisited είναι υπογραμμισμένα (underlines) και μπλε χρώμα
- Το default symbol στις `<ul>` είναι η τελεία ως bullet. Επίσης κι εδώ το `<u>` αφήνει μία γραμμή πάνω και κάτω κενές

- Το default styling στο `h1` είναι 2em (το em είναι το size του parent container ή το default του browser αν ο parent container είναι ο `<html><body>` - δηλ. 16px αν δεν το έχουμε αλλάξει στον browser)
- `h1` 2em (32px),
- `h2` 1.5em (24px)
- `h3` 1.17em (20.8px)
- `h4` 1em (16px)
- `h5` 0.83em (12.8px)
- `h6` 0.67em (11.3px)



# Styling

- Το Web θα ήταν πολύ βαρετό και συνακόλουθα η εμπειρία των χρηστών πολύ χαμηλή αν η μορφή των σελίδων βασιζόταν μόνο στα default settings των browsers
- Χρειαζόμαστε επομένως κάποιο τρόπο για να διαμορφώνουμε εμείς τον τρόπο παρουσίασης του περιεχομένου των σελίδων μας



# Μορφοποίηση

Προγραμματισμός στο Web

- Όπως ήδη γνωρίζουμε από τη χρήση του MS Word ή αντίστοιχων προγραμμάτων επεξεργασίας κειμένου, εκτός από την εισαγωγή περιεχομένου μπορούμε και να **μορφοποιήσουμε το περιεχόμενο**
- Δηλαδή να επιλέξουμε **γραμματοσειρά**, να αλλάξουμε το **μέγεθος** των γραμμάτων, το **χρώμα** γραμμάτων και υποβάθρου, ή να εισάγουμε **περιγράμματα** σε παραγράφους, καθώς και πιο σύνθετη επεξεργασία, όπως επεξεργασία σε **links**, **λίστες**, **πίνακες**, **εικόνες** ή και multimedia περιεχομένου όπως ήχο και βίντεο





# Περιεχόμενο vs Παρουσίαση

Προγραμματισμός στο Web

- Θα πρέπει επομένως να κάνουμε ένα λογικό διαχωρισμό μεταξύ **περιεχομένου** και **τρόπου παρουσίασης του περιεχομένου (styling)**
- Στο σχεδιασμό μιας web σελίδας μπορεί το ίδιο περιεχόμενο να παρουσιαστεί με διαφορετικούς τρόπους
- Η γλώσσα CSS μας επιτρέπει να ορίζουμε πως παρουσιάζονται τα έγγραφα HTML στους χρήστες, δηλαδή πως τους δίνουμε **styling**



# Παρουσίαση και ευχρηστία

Προγραμματισμός στο Web

- Ένα HTML έγγραφο όπως έχουμε δει είναι ένα έγγραφο κειμένου όπου εισάγουμε περιεχόμενο χρησιμοποιώντας μια mark up language, όπως η HTML
- Το styling ενός εγγράφου και συνακόλουθα ο τρόπος παρουσίασης του στους χρήστες σημαίνει την μετατροπή του εγγράφου σε μορφή εύχρηστη από τους χρήστες



# Browser Engine

- Το Browser engine ή Layout engine ή **rendering** engine είναι το μέρος εκείνο του software των browsers που μετατρέπει τα HTML Documents σε ορατά αντικείμενα στην οθόνη του χρήστη
- Οι browsers όπως ο Chrome (**Blink** Engine), Mozilla (**Gecko** engine), Safari (**WebKit** engine), Edge (**Blink** engine), κλπ. μπορούν και αναπαριστούν HTML αρχεία σε μορφή παρουσίασης στην οθόνη



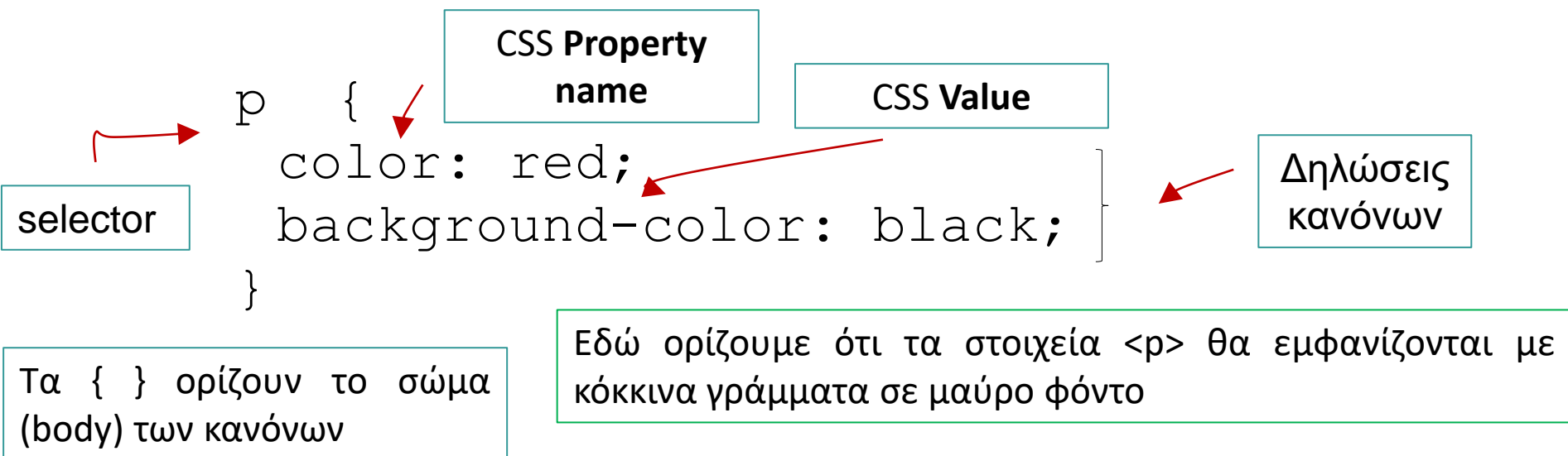
- Συνοψίζοντας, η γλώσσα CSS (Cascading Style Sheet) μας επιτρέπει να εφαρμόζουμε styling στα στοιχεία HTML, να δημιουργούμε Layouts, ακόμα και Animation
- Η CSS είναι μία δηλωτική **γλώσσα ορισμού του τρόπου παρουσίασης των στοιχείων HTML**



# Μορφοποίηση με CSS

Προγραμματισμός στο Web

- Η CSS είναι μία rule-base language, όπου ορίζουμε **κανόνες styling** εφαρμόζονται σε συγκεκριμένα HTML στοιχεία (tags) του εγγράφου μας
- Σύνταξη: **selector { rule1; rule2; ... }**





# CSS Syntax (1)

```
p {  
    color: red;  
    background-color: black;  
}
```

- Οι CSS κανόνες έχουν τη μορφή ζευγών property: value
- Δηλαδή ορίζουμε την **ιδιότητα** (property) του στοιχείου που θέλουμε να μορφοποιήσουμε, μετά ακολουθεί άνω-κάτω τελεία και μετά ορίζουμε την **τιμή** (value) της ιδιότητας



# CSS Syntax (2)

```
p {  
    color: red;  
    background-color: black;  
}
```

- Τα **css properties** όπως τα color και background-color **είναι τα styles** που μπορούμε να εφαρμόσουμε στα elements που ορίζει ο selector
- Είναι case-insensitive, δηλαδή δεν γίνεται διάκριση μεταξύ πεζών (lowercase) και κεφαλαίων (uppercase)
- Κατά σύμβαση όμως, χρησιμοποιούμε πεζά γράμματα και αν ένα css property αποτελείται από δύο λέξεις, διαχωρίζονται με παύλα (kebab-case)



# CSS Syntax (3)

```
p {  
  color: ■ red;  
  background-color: ■ black;  
}
```

- Οι τιμές που μπορεί να λάβει ένα CSS property εξαρτώνται από τον τύπο της ιδιότητας και ορίζονται τυπικά από το World Wide Web Consortium (W3C)





# CSS Κανόνες

- Σε ένα CSS stylesheet μπορούμε να ορίσουμε πολλούς τέτοιους κανόνες στη σειρά τον ένα μετά τον άλλο

```
h1 {  
  font-size: 2em;  
}  
  
p {  
  color: red;  
  background-color: black;  
}
```

Το em είναι μονάδα μέτρησης στην τυπογραφία. 1 em ισούται με το τρέχον text size, το size δηλαδή του parent element, εν αντιθέσει με το rem (root em) που ισούται με το default size του root element δηλαδή του μεγέθους γραμματοσειράς που έχει ορίσει ο κάθε χρήστης στον browser



# CSS Standard

- Το CSS αρχικά εκδόθηκε (έγινε release) το 1996, ενώ σήμερα η standard έκδοση, είναι η έκδοση CSS3
- Αναπτύχθηκε και τη διαχειρίζεται το CSS Working Group του W3C, <https://www.w3.org/Style/CSS/members.en.html>



# Ορισμός rules (1)

- Που ακριβώς όμως μπορούμε να ορίζουμε CSS rules;
  1. **Inline** -- Με την ιδιότητα **style** μέσα στο opening tag κάθε στοιχείου HTML
  2. **Internal** -- Με το στοιχείο **<style></style>** μέσα στο **<head>**
  3. **External** – Σε ένα **αρχείο με κατάληξη .css**, το οποίο συνδέουμε σε ένα ή περισσότερα αρχεία HTML



# Inline Styling

```
<h1 style="color: ■ red; background-color: ■ black;">Web Development Course</h1>
```



- Όπως βλέπουμε παραπάνω, έχουμε εισάγει την ιδιότητα **style μέσα στο <h1>**
- Το **inline styling** δεν χρησιμοποιείται γενικά, γιατί θα θέλαμε Separation Of Concerns (άλλο concern το περιεχόμενο (HTML) και άλλο το styling (CSS))
- Έχει ωστόσο την **υψηλότερη προτεραιότητα** από τους τρεις τρόπους styling



# Internal Styling

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Web Development</title>
  <style>
    h1 {
      color: red;
      background-color: black;
    }
  </style>
</head>
```



- Μέσα στο `<head>` έχουμε ορίσει το στοιχείο `<style>` μέσα στο οποίο έχουμε δηλώσει rules για τα στοιχεία `<h1>`. Αφού ο selector είναι `h1`, οι κανόνες ισχύουν για όλα τα στοιχεία `<h1>`
- Όπως και το `inline styling`, έτσι και το `internal`, δεν προσφέρει Separation of Concerns (έχουμε σε ένα αρχείο με δύο concerns, περιεχόμενο και styling)



# External Styling

```
index.html X
css > index.html > html > head > link
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device
6   <title>Web Development with HTML/CSS/JavaSc
7   <link rel="stylesheet" href="index.css">
8 </head>

# index.css X
css > # index.css > h1
1 h1 {
2   color: red;
3   background-color: black;
4 }
```

- Δημιουργούμε ένα αρχείο ***index.css*** μέσα στο οποίο εισάγουμε τα CSS rules
- Μέσα στο ***index.html*** εισάγουμε μέσα στο `<head>` (βλ. γραμμή 7) ένα link element και με την ιδιότητα href συνδέουμε με το αρχείο `index.css`
- Όταν ο browser κατεβάσει το `index.html`, 'βλέπει' το `<link>` και κατεβάζει σε 2<sup>ο</sup> βήμα το `index.css`



# Προτιμότερος τρόπος

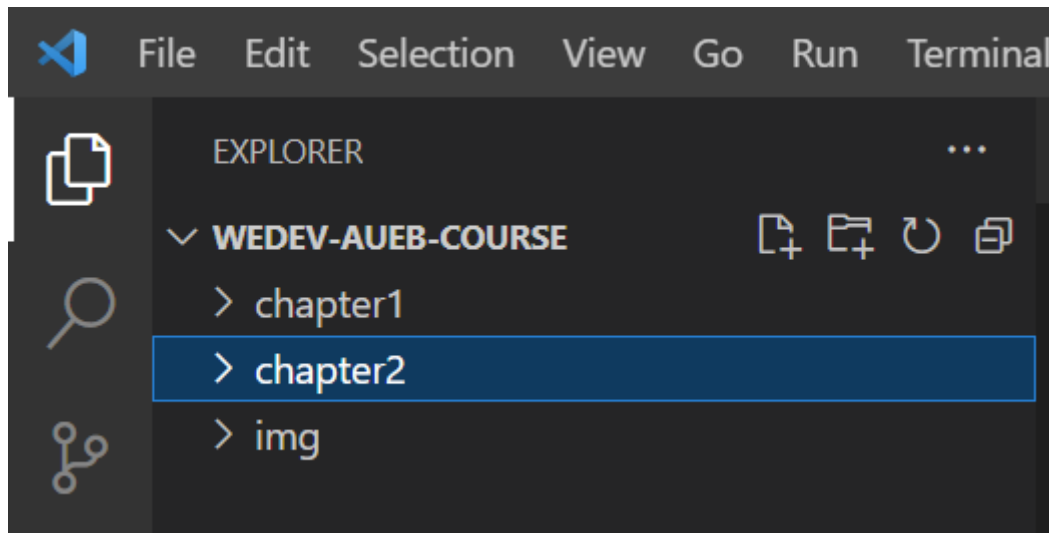
Προγραμματισμός στο Web

- Προτιμότερος τρόπος είναι το **external styling** δηλαδή να ορίζουμε CSS rules σε .css αρχεία ώστε να αποσυνδέουμε και με φυσικό τρόπο την παρουσίαση από το περιεχόμενο
- Έτσι έχουμε **separation of concerns** (δεν μπλέκουμε html και css σε ένα αρχείο)



# Νέος φάκελος chapter2

Προγραμματισμός στο Web




- Δημιουργούμε νέο φάκελο chapter2, ώστε να αναπτύξουμε μικρές εφαρμογές






# Inline Styling

<> hello.html X

chapter2 > <> hello.html >  html

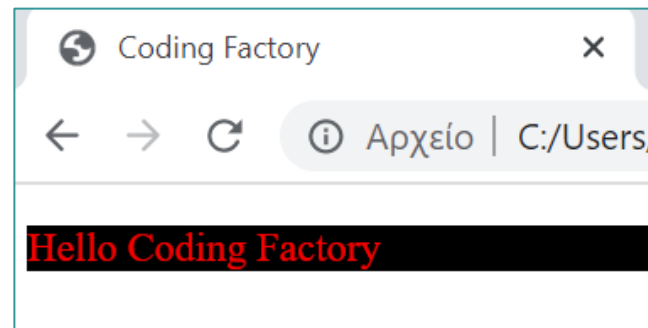
```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Coding Factory</title>
8  </head>
9  <body>
10
11      <p style="color:  red; background-color:  black">Hello Coding Factory</p>
12
13  </body>
14  </html>
```

- Styling με την ιδιότητα style μέσα στο UI Element



# Internal Styling

```
<> hello.html X
chapter2 > <> hello.html > html > head > style > p
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, in
7      <title>Coding Factory</title>
8      <style>
9          p {
10             color: red;
11             background-color: black;
12          }
13      </style>
14  </head>
15  <body>
16
17      <p>Hello Coding Factory</p>
18
19  </body>
20  </html>
```



- Styling μέσα στα `<style></style>` tags, μέσα στο `<head>`



# External Styling (1)

```
chapter2 > css > # hello.css > p
1  p {
2      color: red;
3      background-color: black;
4  }
```

- Στο CSS έχουμε δηλώσει δύο κανόνες που αφορούν τα `<p>`
  1. Το χρώμα των γραμμάτων να είναι κόκκινο
  2. Το χρώμα του παρασκηνίου να είναι μαύρο



# External Styling (2)

The screenshot displays the Visual Studio Code interface with three panels. The Explorer on the left shows a project structure with 'chapter2' containing 'css' and 'hello.html'. The main editor shows 'hello.html' with the following code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>Coding Factory</title>
8   <link rel="stylesheet" href="./css/hello.css">
9 </head>
10 <body>
11
12   <p>Hello Coding Factory</p>
13
14 </body>
15 </html>
```

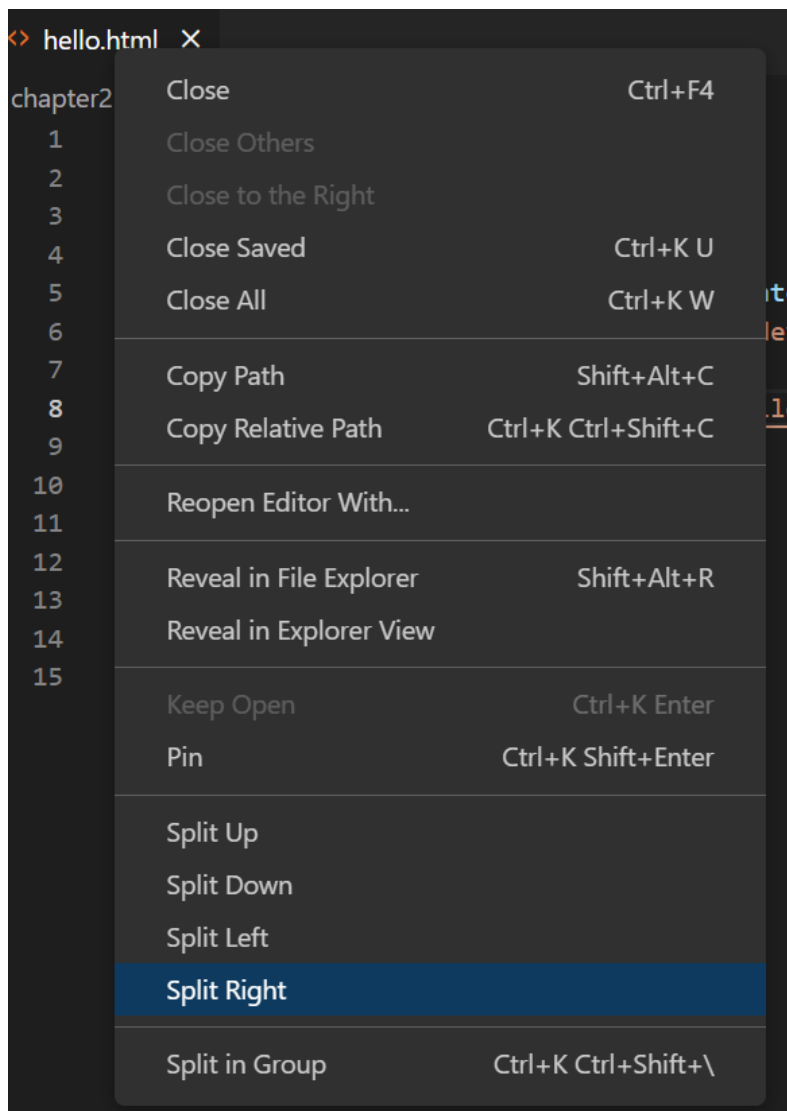
The right panel shows 'hello.css' with the following code:

```
1 p {
2   color: red;
3   background-color: black;
4 }
5
```

- External Styling με `<link>` tag στο `<head>`
- Το αρχείο `hello.css` βρίσκεται στον φάκελο `css`, που είναι υποφάκελος του `chapter2`. Επομένως, μέσα στο `<link>` αναφερόμαστε στο `hello.css`, ως `./css/hello.css` όπου η τελεία συμβολίζει τον τρέχον φάκελο (`chapter2`)



# Split



- Με Alt και click ή καλύτερα alt + double click (για να σταθεροποιηθεί το tab ου ανοίγει) ανοίγουμε ένα file σε ένα νέο tab (split)
- Αν έχουμε δύο tabs μπορούμε να κάνουμε split την οθόνη και duplicate το tab με δεξί κλικ στο tab και split right



# Multiple Selectors

```
<> multiple-selectors.html X
chapter2 > <> multiple-selectors.html > html > body > ul > li
6      <meta name="viewport" content="width=device-width, initial-scal
7      <title>Coding Factory</title>
8  </head>
9  <body>
10
11      <h2>Hello Coding Factory</h2>
12
13      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
14          Quibusdam facilis laborum vero iure et itaque
15          facere quaerat impedit dolorum quos porro asperiores odit
16          eligendi inventore, quis dolores debitis, nostrum amet?
17      </p>
18
19      <a href="https://codingfactory.aueb.gr/">Visit us</a>
20
21      <h2>Course</h2>
22
23      <ul>
24          <li>Programming Languages</li>
25          <li>SQL, Data-Driven</li>
26          <li>Web Development</li>
27      </ul>
28
29      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
30          Quibusdam facilis laborum vero iure et itaque
31          facere quaerat impedit dolorum quos porro asperiores odit
32          eligendi inventore, quis dolores debitis, nostrum amet?
33      </p>
34  </body>
35  </html>
```



- Έστω το HTML αρχείο αριστερά



# Multiple Selectors (1)

Προγραμματισμός στο Web



```
1  body {
2      background-color: gray;
3  }
4
5  p, li {
6      color: white;
7      background-color: pink;
8  }
9
```

- Το body αφορά όλη τη σελίδα
- Μπορούμε να ορίσουμε πολλαπλούς selectors διαχωρίζοντάς τους με κόμμα
- Το list-style-type είναι ένα CSS property που δίνει styling στα bullets των <li>



# Multiple Selectors (2)

Προγραμματισμός στο Web

# multiple-selectors.css X

chapter2 > css > # multiple-selectors.css > ...

```
1  body {
2      background-color: gray;
3  }
4
5  p, li {
6      color: white;
7      background-color: pink;
8  }
9
10 li {
11     list-style-type: square;
12 }
13
```

## Course

- Programming Languages
- SQL, Data-Driven
- Web Development

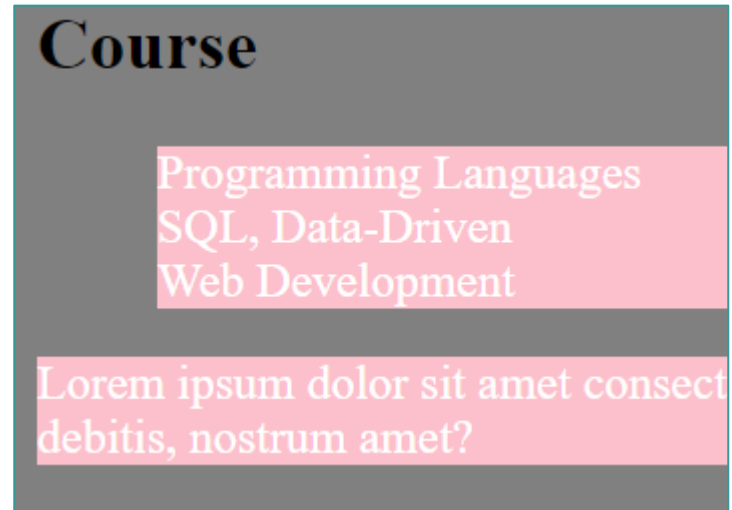
- Μπορούμε να αλλάξουμε το default styling των browsers
- Για παράδειγμα στις unordered λίστες μπορούμε να αφαιρέσουμε την κουκίδα ή να ορίσουμε κάποιο άλλο σύμβολο, π.χ. τετράγωνο (square)





# Αλλαγή default styling

```
# multiple-selectors.css X
chapter2 > css > # multiple-selectors.css > li
1  body {
2      background-color: gray;
3  }
4
5  p, li {
6      color: white;
7      background-color: pink;
8  }
9
10 li {
11     list-style-type: none;
12 }
```



- Όπως αναφέραμε μπορούμε να αλλάξουμε το default styling των browsers
- Για παράδειγμα στις unordered λίστες μπορούμε να αφαιρέσουμε την κουκίδα



# List style

```
li {  
  list-style: circle outside none;  
}
```

- Το list-style είναι σύντμηση του list-style-type, list-style-position, list-style-image

```
li::before {  
  content: '';  
  display: inline-block;  
  height: 10px;  
  width: 10px;  
  margin-right: 20px;  
  background-image: url('./aueblogo.jpg');  
}
```

- Το ::before είναι ψευδο-element για να εισάγουμε content πριν από ένα στοιχείο (στοιχείο που μπορεί να έχει content)
- Εδώ το content είναι κενό αλλά εισάγουμε εικόνα ως bullet (με μικρές διαστάσεις)

- Τα παραπάνω θα τα ξαναδούμε αναλυτικά



# id και class

- Χρησιμοποιώντας ως CSS Selectors, στοιχεία HTML, όπως h1, p, κλπ. επιλέγουμε **όλα** τα αντίστοιχα HTML στοιχεία
- Πως μπορούμε όμως να επιλέγουμε συγκεκριμένα στοιχεία HTML;
- Για να ξεχωρίζουμε τα στοιχεία html εισάγουμε την ιδιότητα **class** καθώς ή/και την ιδιότητα **id**



# id – CSS Property

- Η ιδιότητα **id** **μοναδικοποιεί** (κατά σύμβαση) ένα στοιχείο html
- Αυτό σημαίνει πως κάθε στοιχείο HTML μπορεί να έχει ένα μοναδικό id. Αυτό είναι σύμβαση (convention) ώστε να μπορούμε να επιλέγουμε με μοναδικό τρόπο μέσω του id, HTML στοιχεία.
- Τεχνικά μπορούμε να έχουμε περισσότερα από ένα, ίδια id, αλλά κάτι τέτοιο δεν συνίσταται γιατί τα semantics του id είναι να είναι μοναδικό
- Ο βασικός πρακτικός λόγος χρήσης των id είναι, όπως θα δούμε, μέσα από την JavaScript για να μπορούμε να εντοπίσουμε με μοναδικό τρόπο ένα HTML στοιχείο



# Class – CSS Property

Προγραμματισμός στο Web

- Η ιδιότητα **class** ταυτοποιεί ένα ή περισσότερα στοιχεία html
- Είναι η βασική και περισσότερο χρησιμοποιημένη ιδιότητα στο CSS Styling
- Μέσω της ιδιότητας **class**, δίνουμε ένα όνομα (αναγνωριστικό) στα HTML στοιχεία, που λειτουργεί ως **selector** ώστε να μπορούμε να **επιλέγουμε όχι μόνο με το tag name αλλά και με το class name**



# Id , class naming conventions

- Τα ονόματα (τιμές) που δίνουμε στα id και class είναι case-sensitive, δηλαδή διαφέρουν πεζά και κεφαλαία
- Επίσης, το naming convention για τα **class names** είναι **kebab-case** ενώ για το **id** είναι **camelCase** μιας και τα id όπως αναφέραμε μπορεί να χρησιμοποιούνται στον κώδικα JavaScript



# CSS και class selectors (1)

id-class.html X

chapter2 > id-class.html > html > body > p#auebText

```
6 <meta name="viewport" content="width=device-width, initial-scale=1.
7 <title>AUEB Page</title>
8 <link rel="stylesheet" href="./css/id-class.css">
9 </head>
10 <body>
11
12 
13
14 <p id="auebText">
15     Lorem ipsum dolor sit, amet consectetur adipisicing elit.
16     Mollitia quis odio tempora optio repellat laudantium voluptate,
17     magnam reiciendis voluptates, dignissimos numquam
18     consequatur molestiae libero vitae cumque nihil corporis ad con
19 </p>
20
21 </body>
22 </html>
```

- Στο `<img>` έχουμε δώσει ένα class name “aueb-logo”
- Στο `<p>` έχουμε δώσει ένα id=“auebText”

- Παρατηρούμε τα naming-conventions των aueb-logo και auebText. Τα class names τα δίνουμε με kebab-case. Τα ids καλύτερα να τα δίνουμε με camelCase καθώς όπως αναφέραμε υπάρχουν μέσα στον κώδικα JavaScript όπου εκεί το convention είναι camelCase



# Styling

```
# id-class.css X
chapter2 > css > # id-class.css > #auebText
1  .aueb-logo {
2      width: 800px;
3      height: auto;
4  }
5
6  #auebText {
7      color: white;
8      background-color: grey;
9  }
```

- Στο CSS, ο selector της class **ξεκινά με τελεία** ακολουθούμενη από το class name δηλαδή **.aueb-logo**
- Ο selector του id **ξεκινά με #** και ακολουθείται από το id, δηλαδή **#id**





# id styling

# id-class.css X

chapter2 > css > # id-class.css > #auebText


```
1  .aueb-logo {
2      width: 800px;
3      height: auto;
4  }
5
6  #auebText {
7      color: white;
8      background-color: grey;
9  }
```

- Ωστόσο, ουσιαστικά το id δεν χρησιμοποιείται για styling
- Αναφέρεται μόνο γιατί πιθανά υπάρχουν παραδείγματα σε βιβλία / tutorials (styling με id, όπως αναφέραμε, τεχνικά γίνεται, αλλά σημασιολογικά δεν είναι σωστό!)



# Ο σωστός τρόπος (1)

<> id-class.html X

chapter2 > <> id-class.html >  html

```
6      <meta name="viewport" content="width=device-width, initial-scale=1
7      <title>AUEB Page</title>
8      <link rel="stylesheet" href="._/css/id-class.css">
9  </head>
10 <body>
11
12      
13
14      <p id="auebText" class="aueb-text">
15          Lorem ipsum dolor sit, amet consectetur adipisicing elit.
16          Mollitia quis odio tempora optio repellat laudantium voluptate
17          magnam reiciendis voluptates, dignissimos numquam
18          consequatur molestiae libero vitae cumque nihil corporis ad co
19      </p>
20
21 </body>
22 </html>
```

- Ο σωστός τρόπος είναι το styling να δίνεται μέσω class
- Μπορεί να συνυπάρχει το id με το class μιας και το id εξυπηρετεί άλλο σκοπό που είναι η επιλογή του στοιχείου HTML μέσα από την JavaScript



# Ο σωστός τρόπος (2)

```
chapter2 > css > # id-class.css > body
1  body {
2      background-color: gainsboro;
3  }
4
5  .aueb-logo {
6      width: 800px;
7      height: auto;
8  }
9
10 .aueb-text {
11     color: white;
12     background-color: grey;
13 }
```

- Για επιλογή στοιχείων χρησιμοποιούμε tags (το body είναι tag) ή/και class names (το .aueb-logo είναι class name)
- Υπάρχουν και άλλοι τρόποι επιλογής στοιχείων που θα δούμε στα επόμενα



# Περισσότερα class names

Προγραμματισμός στο Web

<> id-class.html X

chapter2 > <> id-class.html > html > body

```
9    </head>
10   <body>
11
12     <p class="aueb-logo">Lorem ipsum dolor sit amet consectetur adipis
13       ad eaque voluptatem repudiandae. Fugit ducimus corrupti quia c
14     </p>
15     
16
17
18     <h1 class="aueb-text">Coding Factory</h1>
19
20     <p id="auebText" class="aueb-text">
21       Lorem ipsum dolor sit, amet consectetur adipisicing elit.
22       Mollitia quis odio tempora optio repellat laudantium voluptate
23       magnam reiciendis voluptates, dignissimos numquam
24       consequatur molestiae libero vitae cumque nihil corporis ad co
25     </p>
26
27   </body>
28 </html>
```

- Το ίδιο styling μπορούμε να δώσουμε σε περισσότερα από ένα στοιχεία
- Για παράδειγμα το 1ο <p> και το <img> είναι aueb-logo ενώ το <h1> και το 2ο <p> είναι aueb-text



# Αποτέλεσμα

## Προγραμματισμός στο Web



chapter2 > css > # id-class.css > body

```
1 body {
2   background-color: #gainsboro;
3 }
4
5 .aueb-logo {
6   width: 800px;
7   height: auto;
8 }
9
10 .aueb-text {
11   color: white;
12   background-color: #grey;
13 }
```

- Δώσαμε κοινή μορφοποίηση στα `<p>` `<img>` (aueb-logo) και στα `<h1>``<p>` (aueb-text)
- Παρατηρούμε και το `body` που δίνει styling σε όλο το viewport (όλο το ορατό μέρος/σελίδα)



# CSS Selectors tag.class

```
/head>
body>

<p class="aueb-logo">Lorem ipsum dolor sit amet consectetur adipis
  ad eaque voluptatem repudiandae. Fugit ducimus corrupti quia c
</p>



<h1 class="aueb-text">Coding Factory</h1>

<p id="auebText" class="aueb-text">
  Lorem ipsum dolor sit, amet consectetur adipisicing elit.
  Mollitia quis odio tempora optio repellat laudantium voluptate
  magnam reiciendis voluptates, dignissimos numquam
  consequatur molestiae libero vitae cumque nihil corporis ad co
</p>

/body>
/html>
```

```
1  body {
2    background-color: #gainsboro;
3  }
4
5  .aueb-logo {
6    width: 800px;
7  }
8
9  img.aueb-logo {
10   height: auto;
11 }
12
13 .aueb-text {
14   background-color: #blue
15 }
16
17 p.aueb-text {
18   color: #white;
19 }
```

- Μπορούμε να επιλέγουμε με πιο ειδικό τρόπο με συνδυασμό tag.class, π.χ. `img.aueb-logo` ή `p.aueb-text`



# Selectors και κλάσεις

```
1  body {  
2    background-color: gainsboro;  
3  }  
4  
5  .aueb-logo {  
6    width: 800px;  
7  }  
8  
9  img.aueb-logo {  
10   height: auto;  
11 }  
12  
13 .aueb-text {  
14   background-color: blue  
15 }  
16  
17 p.aueb-text {  
18   color: white;  
19 }
```

- Μπορούμε επομένως να περιορίσουμε την εφαρμογή μιας κλάσης σε ένα ή περισσότερα elements
- Το **img.aueb-logo** προστάζει: Επέλεξε κάθε img element που έχει κλάση aueb-logo
- Το **p.aueb-text** προστάζει: Επέλεξε κάθε p element που έχει κλάση aueb-text



# Επιλογή με βάση τη θέση

Προγραμματισμός στο Web

```
chapter2 > <> position-selector.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" co
6      <meta name="viewport" content="width=
7      <title>Programming Languages</title>
8
9  </head>
10 <body>
11
12     <ul>
13         <li><strong>Java</strong></li>
14         <li><em>C#</em></li>
15         <li>Python</li>
16     </ul>
17
18 </body>
19 </html>
```

- Έστω μία λίστα με στοιχεία `<li>` όπου το 1<sup>ο</sup> στοιχείο είναι `<strong>` Γίνεται render σαν bold, αλλά το `<strong>` έχει και semantics και διαβάζεται από τους screen readers
- Το 2<sup>ο</sup> στοιχείο είναι `<em>` (από το emphasis) και εμφανίζεται ως italics αλλά το `<em>` έχει semantics για τους screen readers





# Επιλογή με βάση τη θέση (1)

The screenshot shows a web browser window. On the left, a dark-themed code editor displays a CSS file named 'position-selector.css'. The code is as follows:

```
# position-selector.css
chapter2 > css > # position-selector.css > ul > li
1  ul > li {
2    color: red;
3  }
```

On the right, the browser's content area shows a list of programming languages:

- Java
- C#
- Python

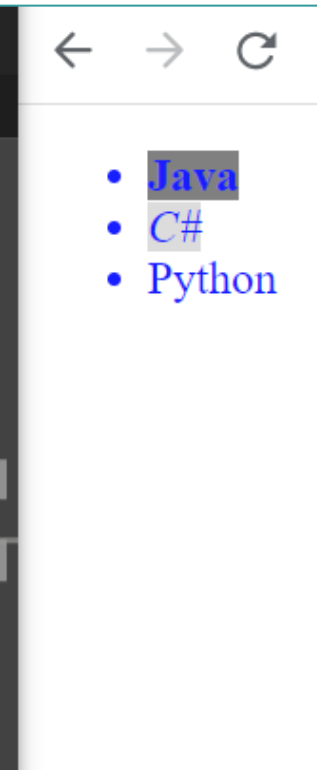
- Μερικές φορές θέλουμε να επιλέγουμε στοιχεία με βάση τη θέση τους στο HTML έγγραφο
- Παραπάνω επιλέγουμε με `>` τους άμεσους απόγονους `<li>` του `<ul>` και δίνουμε styling με κόκκινο χρώμα στο περιεχόμενο



# Επιλογή με βάση τη θέση (2)

Προγραμματισμός στο Web

```
# position-selector.css X
chapter2 > css > # position-selector.css > ul em
1  ul > li {
2      color: blue;
3  }
4
5  ul strong {
6      background-color: gray;
7  }
8
9  ul em {
10     background-color: gainsboro;
11 }
```



- Για να επιλέξουμε απογόνους (όχι μόνο άμεσους) δεν χρησιμοποιούμε `>` αλλά απλά κενό. Παραπάνω επιλέγουμε τα `<strong>` και `<em>` ως απογόνους του `<ul>`



# Επιλογή με βάση τη θέση (2)

Προγραμματισμός στο Web

← → ↻ ⓘ 127.0.0.1:5500/chapter2/position-selector.html

## Coding Factory

Lorem ipsum dolor sit amet consectetur adipisicing elit. Rerum a reprehenderit tempore quae exercitationem? Autem.

- Java
- C#
- Python

```
chapter2 > css > # position-selector.css > h1 + p
1  ul > li {
2      color: blue;
3  }
4
5  ul strong {
6      background-color: gray;
7  }
8
9  ul em {
10     background-color: gainsboro;
11 }
12
13 h1 + p {
14     background-color: aqua;
15 }
```

- Το + σημαίνει: Επέλεξε μία παράγραφο που βρίσκεται αμέσως μετά ένα h1 στο ίδιο επίπεδο ιεραρχίας (**adjacent sibling combinator**)



# Επιλογή με βάση τη θέση (3)

Προγραμματισμός στο Web

- Η επιλογή με βάση τη θέση συνήθως χρησιμοποιείται όταν θέλουμε να επιλέξουμε περισσότερα από ένα στοιχεία, όπως στοιχεία `<li>` μέσα σε ένα `<ul>` ή στοιχεία `<a>` μέσα σε `<li>`
- Πρέπει να χρησιμοποιείται με προσοχή γιατί αν έχουν δοθεί class names στα στοιχεία αυτά και έχει δοθεί styling μέσω του class name, τότε θα δημιουργηθεί conflict και το ποιο styling θα επικρατήσει θα το δούμε στα επόμενα



# Styling Links

- Όταν δίνουμε styling σε links είναι σημαντικό να καταλάβουμε την έννοια των καταστάσεων (states) που βρίσκεται ένα Link καθώς και την έννοια των ψευδο-κλάσεων (pseudo-class) με τις οποίες επιλέγουμε Links που βρίσκονται σε ένα state

State	Ερμηνεία	Pseudo-class
Link	Unvisited link	:link
Visited	Visited link	:visited
Hover	Όταν βάζουμε το ποντίκι πάνω από το link και πριν πατήσουμε κλικ	:hover
Focus	Όταν πάμε με το tab και 'επιλέγουμε' το link () ή προγραμματιστικά με .focus()	:focus
Active	Τη στιγμή που κάνουμε κλικ	:active



# Default Link Styles (1)

- Όλα τα **Links**: είναι underlined
- **Unvisited**: blue (μπλε χρώμα γραμμάτων)
- **Visited**: Purple (Μωβ γράμματα)
- **Hover**: Ο δείκτης του ποντικιού (mouse pointer) γίνεται 'χεράκι' (hand icon)
- **Focus**: Outline (περίγραμμα) γύρω από το link
- **Active**: red (κόκκινα γράμματα στιγμιαία στο κλικ. Κρατήστε το κλικ πατημένο πάνω στο link για να φανεί καλύτερα)



# Default Link Styles (2)

- Μπορούμε με CSS να τροποποιήσουμε τα default styles
- Ωστόσο κάποια default styles είναι χρήσιμο να μην αλλάζουν όπως το focus μιας και το περίγραμμα είναι υποστηρικτικό accessibility tool, εκτός αν χρησιμοποιούμε άλλο τρόπο να καταδείξουμε το focused link



# Styling με βάση το Link state (1)

Προγραμματισμός στο Web

```
<> links.html ×
chapter2 > <> links.html > html > body > a
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Coding Factory AUEB</title>
8  </head>
9  <body>
10
11      <h2>Coding Factory</h2>
12      <a href="https://codingfactory.aueb.gr/" target="_blank">Visit Us</a>
13
14  </body>
15  </html>
```

- Έστω το παραπάνω αρχείο με το `<a>`





# Styling με βάση το Link state (2)

```
chapter2 > css > # links.css > a:focus
1  /* LoVe Fears HAtE
2  * Specific order:
3  * Link, Visited, Focus, Hover, Active
4  */
5
6  a:link {
7      text-decoration: none;
8      color: black;
9  }
10
11 a:visited {
12     color: brown;
13 }
14
15 a:focus {
16     outline: none;
17     background-color: yellow;
18 }
19
20 a:hover {
21     background-color: grey;
22 }
23
24 a:active {
25     background-color: green;
26     color: white;
27 }
28
```

- Δίνουμε styling σε κάθε state
- Έχει σημασία το ordering, μιας και για παράδειγμα το active εμπεριέχει το hover
- Επίσης το hover πρέπει να είναι μετά το visited, διαφορετικά θα εφαρμοζόταν πάντα το visited (εφαρμόζεται ο τελευταίος στη σειρά κανόνας)
- Για να θυμόμαστε μνημονικά τη σειρά εφαρμογής των κανόνων στα Link States: **LoVe Fears HAtE**



# Class reusability (1)

Προγραμματισμός στο Web

- Μερικές φορές μπορούμε να μειώνουμε την πολυπλοκότητα συγγραφής κλάσεων με συνδυασμούς κλάσεων είτε ειδικών κλάσεων που δίνουν styling σε ένα χαρακτηριστικό ή με συνδυασμούς γενικών κλάσεων και ειδικότερων κλάσεων
- Στόχος είναι να μην χρησιμοποιούμε το ίδιο styling rule ξανά και ξανά μέσα σε κλάσεις που μορφοποιούν διαφορετικά στοιχεία αλλά να δημιουργούμε επαναχρησιμοποιήσιμες κλάσεις



# Class reusability (2)

```
<> classes-combination.html X
chapter2 > <> classes-combination.html > html > body > h2.cf-header
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Coding Factory</title>
8      <link rel="stylesheet" href="./css/class-combination2.css">
9  </head>
10 <body>
11
12 <h1 class="aueb-header">Athens University of Economics and Business</h1>
13 <h2 class="cf-header">Coding Factory</h2>
14 <p class="aueb-text">Lorem ipsum dolor sit amet consectetur adipisicing eli
15     Vel quaerat autem libero! Delectus quibusdam eaque deserunt minima temp
16     exercitationem suscipit rem molestiae culpa similique quaerat soluta.
17 </p>
18
19 </body>
20 </html>
```

- Έστω η παραπάνω σελίδα, όπου έχουμε ορίσει class names



# Class reusability (3)

```
chapter2 > css > # class-combination.css > .au
1  .aueb-header {
2      color: ■ white;
3      background-color: ■ brown;
4  }
5
6  .cf-header {
7      color: ■ white;
8      background-color: ■ green;
9  }
10
11 .aueb-text {
12     color: ■ white;
13     background-color: ■ blue;
14 }
```

- Η κλασσική προσέγγιση είναι να ξεκινήσουμε από πάνω προς τα κάτω (top-down) στο HTML και για κάθε στοιχείο να δίνουμε μορφοποίηση
- Κάτι τέτοιο όπως βλέπουμε είναι μεν πιο απλό, αλλά μεγαλώνει ο κώδικας CSS με την έννοια ότι έχουμε λίγες μεγάλες κλάσεις
- Θα μπορούσαμε να εξάγουμε τα κοινά στοιχεία και να έχουμε κάποιες γενικές κλάσεις και κάποιες πιο ειδικές



# Class reusability (4)

```
# class-combination2.css X
chapter2 > css > # class-combination2.css > ...
1  .fg-light {
2    |    color: ■ white;
3  }
4
5  .aueb-header-bg {
6    |    background-color: ■ brown;
7  }
8
9  .cf-header-bg {
10   |    background-color: ■ green;
11 }
12
13 .aueb-text-bg {
14   |    background-color: ■ blue;
15 }
```

- Έχουμε εξάγει τα κοινά στοιχεία (το foreground color που παντού είναι white έχει γίνει μία γενική κλάση) και έχουμε κάποιες πιο ειδικές για το background



# Class reusability (5)

- Αυτή η προσέγγιση μας δίνει περισσότερο reusability, flexibility καθώς και μειώνει την πιθανότητα διαφορετικοί κανόνες να ορίζουν το ίδιο property για το ίδιο HTML element, κάτι το οποίο οδηγεί σε conflicts και μη επιθυμητό αποτέλεσμα στο styling

```
<> classes-combination.html X
chapter2 > <> classes-combination.html > html > body > p.fg-light.aueb-text-bg
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-s
7      <title>Coding Factory</title>
8      <link rel="stylesheet" href="./css/class-combination2.css">
9  </head>
10 <body>
11
12 <h1 class="fg-light aueb-header-bg">Athens University of Econom
13 <h2 class="fg-light cf-header-bg">Coding Factory</h2>
14 <p class="fg-light aueb-text-bg">Lorem ipsum dolor sit amet cons
15     Vel quaerat autem libero! Delectus quibusdam eaque deserunt
16     exercitationem suscipit rem molestiae culpa similique quaera
17 </p>
18
19 </body>
20 </html>
```



# Αποτέλεσμα

Προγραμματισμός στο Web

← → ↻ ⓘ 127.0.0.1:5500/chapter2/classes-combination.html

**Athens University of Economics and Business**

**Coding Factory**

Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga, soluta quis? Vel quaerat autem libero! Dele molestiae culpa similique quaerat soluta.



# Mock-up

Coding Factory

Lorem ipsum dolor sit amet consectetur adipisicing elit. Aliquid nostrum, qui, velit animi, fugit quae nesciunt quaerat veniam harum consectetur optio nam delectus cum libero suscipit ratione voluptatem. Facilis, rem?



- Java
- C#
- Python

Coding Factory

Lorem ipsum dolor sit amet consectetur adipisicing elit. Aliquid nostrum, qui, velit animi, fugit quae nesciunt quaerat veniam harum consectetur optio nam delectus cum libero suscipit ratione voluptatem. Facilis, rem?



- Java
- C#
- Python

- Ας υποθέσουμε ότι έχουμε να αναπτύξουμε το παραπάνω σχέδιο. Παρατηρήστε τα δύο λογικά μέρη, τα οποία είναι λογικοί containers. Οι λογικοί containers απλά ομαδοποιούν τα UI Elements για λόγους κοινής μορφοποίησης





# HTML (1)

## Προγραμματισμός στο Web

```
<> div-hello.html X
chapter2 > <> div-hello.html > html > body > div.container-fluid.bg-primary.
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Coding Factory</title>
8      <link rel="stylesheet" href="./css/div-hello.css">
9  </head>
10 <body>
11
12     <div class="container bg-dark">
13         <h2 class="aueb-text">Coding Factory</h2>
14         <p class="aueb-text bg-primary">Lorem ipsum dolor sit amet consectetur adipisicing elit.
15             Aliquid nostrum, qui, velit animi, fugit quae nesciunt
16             quaerat veniam harum consectetur optio nam delectus cum
17             libero suscipit ratione voluptatem. Facilis, rem?
18         </p>
19
20         
21
22         <ul class="courses-wrapper">
23             <li class="course-text">Java</li>
24             <li class="course-text">C#</li>
25             <li class="course-text">Python</li>
26         </ul>
27     </div>
```

- Παρατηρούμε το `<div>` που είναι ένας container για να ομαδοποιήσουμε λογικά, όπως αναφέραμε, τη δομή της σελίδας
- Το 1<sup>ο</sup> div που φαίνεται αριστερά περιλαμβάνει διάφορα στοιχεία HTML



# HTML (2)

```
31 <div class="container-fluid bg-primary ">
32   <h2 class="aueb-text bg-primary">Coding Factory</h2>
33   <p class="aueb-text bg-dark">Lorem ipsum dolor sit amet consectetur adipisicing elit.
34     Aliquid nostrum, qui, velit animi, fugit quae nesciunt
35     quaerat veniam harum consectetur optio nam delectus cum
36     libero suscipit ratione voluptatem. Facilis, rem?
37   </p>
38
39   
40
41   <ul class="courses-wrapper">
42     <li class="course-text bg-dark">Java</li>
43     <li class="course-text bg-dark">C#</li>
44     <li class="course-text bg-dark">Python</li>
45   </ul>
46
47 </div>
48 </body>
49 </html>
```

- Το ίδιο ισχύει και για το 2<sup>ο</sup> <div>, ομαδοποιεί HTML UI elements. Εδώ το <div> έχει ως class, container-fluid και bg-primary



# CSS (1)

chapter2 > css > # div-hello.css > ...

```
1  body {
2    background-color: gray;
3  }
4
5  .container {
6    width: 80%;
7    border: 1px solid black;
8  }
9
10 .container-fluid {
11   width: 90%;
12   border: 1px solid black;
13 }
14
15 .aueb-text {
16   font-family: 'Times New Roman', Times, serif;
17   font-size: 1.2rem;
18   font-weight: 400;
19   color: antiquewhite;
20 }
21
22 .fg-light {
23   color: white;
24 }
25
26 .bg-dark {
27   background-color: black;
28 }
```

- Το styling αφορά τα χρώματα αλλά και τη δομή, όπως στην κλάση container που είναι ο λογικός container του 1<sup>ου</sup> div, και έχει width και border που θα δούμε πιο αναλυτικά στα επόμενα
- Ο container-fluid με την ίδια λογική δίνει styling στο 2ο div και συνολικά στα στοιχεία του 2<sup>ου</sup> div
- Επίσης, δίνουμε styling και στο text με font-family, font-size, font-weight (και γενικά font-\*)
- Θα δούμε στα επόμενα περισσότερα για typography.



# CSS (2)

```
30 .bg-primary {
31     background-color: blue;
32 }
33
34 .icon-small {
35     width: 150px;
36     height: auto;
37 }
38
39 .courses-wrapper {
40     background-color: aquamarine;
41 }
42
43 .course-text {
44     font-family: Arial, Helvetica, sans-serif;
45     font-size: 1.1 rem;
46     font-weight: bold;
47     color: beige;
48 }
```

- Επίσης τα επιμέρους στοιχεία έχουν styling ανάλογα με το στοιχείο
- Το icon-small είναι εικόνα, το courses-wrapper είναι το <ul> που είναι container για τα <li> και το course-text δίνει styling στο text κάτι που θα δούμε στα επόμενα






# Παράδειγμα (1)

```
7 <title>Coding Factory</title>
8 <link rel="stylesheet" href="./css/selectors.css">
9 </head>
10 <body>
11
12 <div class="container">
13   <h2 class="cf-title">Hello Coding Factory</h2>
14   <p class="cf-text">Lorem ipsum dolor, sit amet consectetur adipisicing
15     Beatae repudiandae voluptates rerum velit libero molestiae dicta
16     maxime incidunt, necessitatibus exercitationem magnam quasi!
17 </p>
18
19 <a href="https://codingfactory.aueb.gr/" target="_blank">Visit Us</a>
20
21 <h2>Courses</h2>
22 <ul>
23   <li>Programming Languages</li>
24   <li>SQL</li>
25   <li>Web Development</li>
26 </ul>
27
28 <p>Lorem ipsum, dolor sit amet consectetur adipisicing elit.
29   Illum expedita quos libero quod accusantium architecto ad ipsum n
30   ipsam iusto dicta quibusdam! Earum quibusdam dolore quas
31   exercitationem.
32 </p>
33 </div>
34
35 </body>
36 </html>
```



# Παράδειγμα (2)

chapter2 > css > # selectors.css >  ul > :nth-child(3)

```
1  * {
2    |   margin: 10px 2px;
3  }
4
5  .container > h2:first-of-type {
6    |   font-size: 24px;
7  }
8
9  .container > p:first-of-type {
10   |   color:  red;
11 }
12
13 ul > :first-child {
14   |   font-weight: bold;
15 }
16
17 ul > :nth-child(3) {
18   |   color:  green;
19 }
```

- Το \* είναι universal selector και επιλέγει όλα τα HTML στοιχεία. Το margin (που θα δούμε πιο αναλυτικά όταν αναφερθούμε στο Box Model) δίνει ένα περιθώριο πάνω-κάτω και αριστερά-δεξιά σε pixels
- Το first-of-type αναφέρεται στο 1ο στοιχείο μίας λίστας στοιχείων HTML, ενώ το first-of-child αναφέρεται στον πρώτο απόγονο ενός parent element όπως εδώ το <ul>
- Επίσης μας παρέχεται η συνάρτηση nth-child() όπου επιλέγουμε τον νιοστό απόγονο ενός parent element



# Selectors Ανασκόπηση

Προγραμματισμός στο Web

Στοιχείο	Επιλογή
h1	Επέλεξε όλα τα στοιχεία h1
a:link	Επέλεξε όλα τα a με state link, δηλαδή normal unvisited link
.my-class	Επέλεξε όλα τα στοιχεία με όνομα κλάση my-class (μπορεί να είναι πολλά στοιχεία)
#myId	Επέλεξε το στοιχείο με όνομα id myId (είναι ένα στοιχείο)
*	Επέλεξε όλα τα στοιχεία (To asterisk character είναι Universal Selector)
.box p	Επέλεξε τα p που είναι απόγονοι των στοιχείων box
.box > p	Επέλεξε τα p που είναι άμεσοι απόγονοι των στοιχείων box
.box li:first-child	Επέλεξε το πρώτο στοιχείο li μεταξύ των στοιχείων li που βρίσκονται μέσα σε στοιχεία με όνομα κλάσης .box
h2: first-of-type	Επέλεξε το 1 <sup>ο</sup> στοιχείο μίας λίστας <h2> στοιχείων
h1, h2, .box	Επέλεξε όλα τα h1 h2 και στοιχεία με όνομα κλάσης box



# Cascade & Specificity

Προγραμματισμός στο Web

- Υπάρχουν περιπτώσεις που το ίδιο στοιχείο HTML το επιλέγουν περισσότεροι από ένας selectors. Το ερώτημα είναι ποιος υπερικχύει;
- Η CSS έχει κανόνες που ορίζουν ποιος selector είναι πιο ισχυρός όταν υπάρχει conflict
- Οι κανόνες αυτοί ονομάζονται **cascade** και **specificity**





# Cascade

```
p {  
  color: red;  
}  
  
p {  
  color: blue;  
}
```

Έστω ότι έχουμε δύο κανόνες για p elements. Τελικά η παράγραφος θα είναι μπλε. Αυτό συμβαίνει επειδή ο ορισμός του CSS κανόνα που κάνει την παράγραφο μπλε εμφανίζεται (μέσα στο stylesheet) **μετά τον κανόνα** που κάνει την παράγραφο κόκκινη

- Ο παραπάνω κανόνας ονομάζεται **cascade rule**. Όταν έχουμε **τον ίδιο selector** περισσότερες από μία φορές, **υπερισχύει ο τελευταίος κανόνας**



# Specificity

```
.special {  
  color: red;  
}  
  
p {  
  color: blue;  
}
```

- Έστω ότι έχουμε δύο κανόνες, ο πρώτος για μία παράγραφο που έχει σημανθεί με όνομα κλάσης `.special` και ένας επόμενος κανόνας που αφορά γενικά όλες τις παραγράφους, άρα και την παράγραφο `special`
- Τελικά η παράγραφος θα είναι κόκκινη. Αυτό συμβαίνει επειδή **μία κλάση θεωρείται πιο specific σε σχέση με το tag name**, δηλαδή πιο ειδική στον ορισμό ενός CSS στοιχείου. Επομένως ακυρώνει τον άλλο πιο γενικό ορισμό που αφορά το HTML στοιχείο.

- Ο παραπάνω κανόνας ονομάζεται **Specificity rule**. Οι CSS κανόνες των κλάσεων υπερισχύουν των κανόνων που αφορούν τα tag names των ίδιων στοιχείων



# Specificity

- Επίσης ένας id selector είναι πιο specific από ένα class selector, ο οποίος όπως είδαμε είναι πιο specific από ένα tag selector
- Ο universal selector είναι γενικός κανόνας και έχει specificity 0



# Υπολογισμός specificity (1)

Προγραμματισμός στο Web

- Το Specificity είναι τυπικά ένας τριψήφιος αριθμός (000 – 111) που υπολογίζεται από τρία πεδία: **ID, CLASS, TYPE**
- Για παράδειγμα ο selector `.box` έχει specificity weight: 010 (γιατί είναι κλάση), ενώ ο `#myId` έχει specificity 100 (γιατί είναι id), επομένως ( $100 > 010$ ) το id υπερισχύει
- Το inline styling έχει υψηλότερο specificity από όλα τα παραπάνω



# Υπολογισμός specificity (2)

```
/* specificity: 0101 */  
#outer a {  
    background-color: red;  
}  
  
/* specificity: 0201 */  
#outer #inner a {  
    background-color: blue;  
}  
  
/* specificity: 0104 */  
#outer div ul li a {  
    color: yellow;  
}  
  
/* specificity: 0113 */  
#outer div ul .nav a {  
    color: white;  
}
```

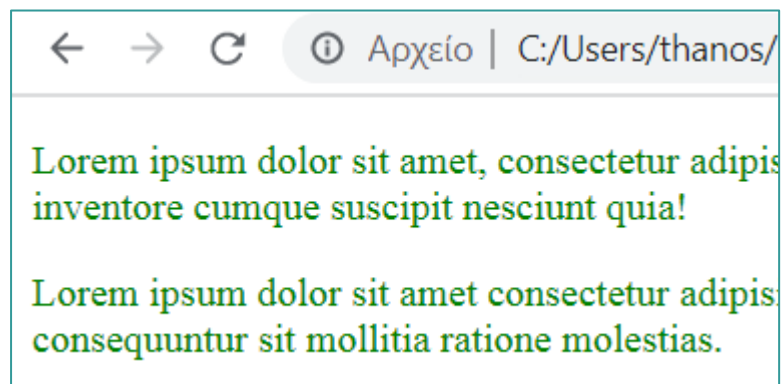
- Αν συμπεριλάβουμε και το inline, το specificity μπορεί να μετρηθεί και με 4 αριθμούς: **inline, id, class, tag**, δηλαδή από το πιο specific προς το πιο γενικό μιας και ξέρουμε ότι πιο specific είναι οι inline ορισμοί, μετά τα id, μετά τα class, και μετά τα tags
- Για παράδειγμα στον 1<sup>ο</sup> ορισμό έχουμε 0 inline, 1 στοιχείο id (το #outer), 0 class, και ένα tag (το a), άρα το specificity είναι 0101
- Στην 2<sup>η</sup> περίπτωση έχουμε 0 inline, 2 id, 0 class και 1 tag, άρα 0201
- Άρα το background-color του a θα γίνει blue γιατί έχει μεγαλύτερο specificity



# !important

```
8      <style>
9          .red-text {
10             color: red;
11         }
12
13         p {
14             color: green!important;
15         }
16
17     </style>
18 </head>
19 <body>
20
21     <p>Lorem ipsum dolor sit amet, conse
22     <p class="red-text">Lorem ipsum dolo
23
24 </body>
25 </html>
```

- Κάποιες φορές που δεν είμαστε βέβαιοι ποιος κανόνας υπερσχύει μπορούμε να εισάγουμε το **!important** μετά τον κανόνα και να του προσδώσουμε το υψηλότερο specificity (υψηλότερο και από το inline)





# !important

- Το !important έχει το υψηλότερο specificity, αλλά αυτό δεν σημαίνει ότι θα πρέπει να χρησιμοποιείται ευρέως. Αντίθετα, ιδανικά δεν θα πρέπει να χρησιμοποιείται ποτέ
- Θα πρέπει να στοχεύουμε στην οργανωμένη χρήση των CSS κανόνων χωρίς να χρειάζεται το !important
- Ιδανικά, μόνο όταν χρησιμοποιούμε frameworks (π.χ. WordPress, Drupal, κλπ.) με έτοιμα CSS, όπου δεν έχουμε πρόσβαση στα αρχεία CSS (Core CSS Modules), μπορούμε να ορίζουμε δικά μας custom CSS με !important



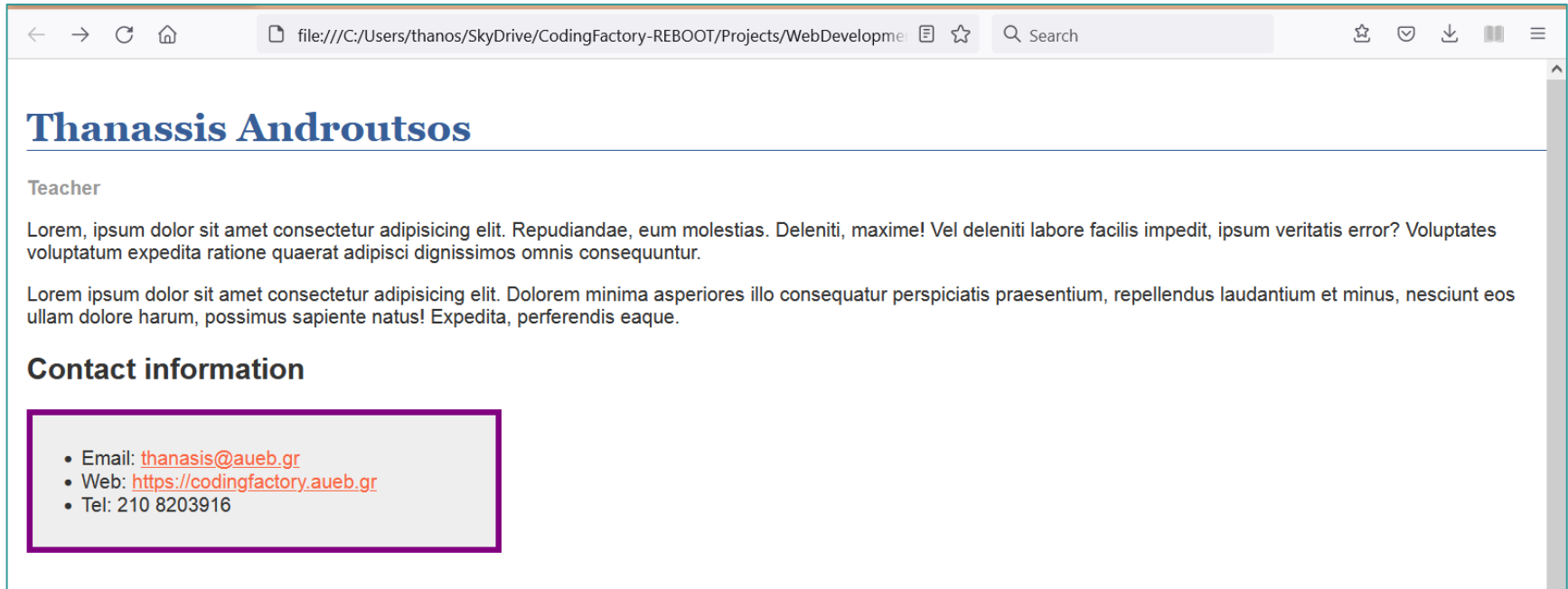
# Invalid properties

- Αν μία ιδιότητα σε ένα CSS κανόνα είναι μη έγκυρη -για παράδειγμα αν δώσουμε με λάθος σύνταξη ή αν δώσουμε μία ιδιότητα που δεν υποστηρίζεται από τον browser- θα αγνοηθεί από το CSS engine
- Δεν θα δώσει κάποιο λάθος





# Άσκηση 1 - Bio



- Δημιουργήστε το δικό σας Βιογραφικό στο Web. Κάντε μία έρευνα σε άλλες σελίδες online βιογραφικών και κάντε μία αποδελτίωση της πληροφορίας και στη συνέχεια συνθέστε τη δική σας εκδοχή σελίδας βιογραφικού