

1 CSS

Μετά την HTML παρουσιάζεται η CSS, η δεύτερη σημαντική τεχνολογία προγραμματισμού διαδικτυακών εφαρμογών. Το κεφάλαιο ξεκινά με σύντομη ιστορική αναδρομή και παρουσίαση της γλώσσας και των προτύπων της. Στη συνέχεια παρουσιάζονται οι πρώτες βασικές έννοιες των επιλογέων και των οδηγιών CSS καθώς και η έννοια της ιεραρχίας επάλληλων φύλλων στυλ. Το κεφάλαιο συνεχίζει με την εισαγωγή σε βασικές οδηγίες της CSS και με τη συζήτηση για τους μηχανισμούς που διαθέτει η γλώσσα σχετικά με μονάδες τα μεγέθη. Έπειτα παρουσιάζεται διεξοδικά το μοντέλο box της CSS και συζητούνται έννοιες και οδηγίες που αφορούν το υπόβαθρο, το πλαίσιο, την υπερχειλίση, τις εικόνες, τις λίστες κ. ά. Το κεφάλαιο κεφάλαιο κλείνει με την παρουσίαση του μηχανισμού των μεταβάσεων (transitions).

1.1 Η γλώσσα CSS

Η CSS είναι γλώσσα με την οποία περιγράφεται η *παρουσίαση* ενός κειμένου και εφαρμόζεται σε κείμενα που είναι γραμμένα σε κάποια γλώσσα όπως HTML, XML, SVG κ.ά. Με τη χρήση της CSS πραγματοποιούμε το διαχωρισμό του περιεχομένου της ιστοσελίδας από τη παρουσίασή του. Παλιότερα, οι οδηγίες περιγραφής της παρουσίασης περιλαμβάνονταν μέσα στην ίδια την HTML, όπως για παράδειγμα οι ετικέτες **** και **** (bold) που στους φυλλομετρητές εμφανίζονται με έντονα γράμματα. Η χρήση όμως των ετικετών αυτών για να παρουσιάσουμε κείμενο με έντονα γράμματα αποθαρρύνεται και αυτός ο ρόλος ανατίθεται στη γλώσσα CSS, π.χ. με την ιδιότητα font-size.

Η πρώτη έκδοση της CSS, η CSS1, δημοσιεύτηκε το 1996 ως [W3C CSS Recommendation \(CSS1\)](#) και ήταν αποτέλεσμα της συνεργασίας που είχε ξεκινήσει δύο χρόνια νωρίτερα μεταξύ του Νορβηγού Håkon Wium Lie και του Ολλανδού Bert Bos. Η CSS1, όπως και η επόμενη έκδοσή, η CSS Επίπεδο 2, περιγράφονται σε ένα ενιαίο κείμενο χωρισμένο σε κεφάλαια. Η τρέχουσα πρόταση του W3C είναι η CSS2.1 (Level 2 Revision 1). Η CSS Επίπεδο 3 αποτελείται από διαφορετικές, ανεξάρτητες μεταξύ τους, ενότητες. Κάθε ενότητα της CSS3 επεκτείνει τις αντίστοιχες δυνατότητες της CSS2. Έτσι κάθε ενότητα

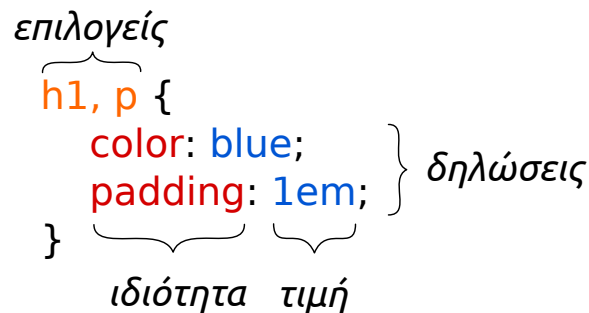
1 CSS

της CSS3 βρίσκεται σε διαφορετικό στάδιο ωρίμανσης, που μπορεί να χαρακτηρίζεται ως Πρόχειρη (Working Draft), Υποψήφια Πρόταση (Candidate Recommendation) και Πρόταση (Recommendation) καθώς και σε διαφορετικό επίπεδο. Για παράδειγμα, η ενότητα CSS Fonts βρίσκεται στο επίπεδο 3 (Lilley, Maxfield, και Daggett 2018) ενώ η ενότητα CSS Flexible Box Module είναι στο επίπεδο 1 (Etemad κ.ά. 2018). Οι ενότητες CSS3 που ολοκληρώνονται ενσωματώνονται σταδιακά στην τρέχουσα CSS2.1 (Etemad, Jr., και Rivoal 2021).

1.2 Κανόνες CSS

Οι οδηγίες της CSS γράφονται σε αρχεία κειμένου που ονομάζονται CSS Stylesheets, φύλλα CSS, και αποτελούνται από έναν ή περισσότερους **κανόνες** (CSS rules). Κάθε τέτοιος κανόνας δίνει οδηγίες στον φυλλομετρητή για το πώς να εμφανίσει συγκεκριμένα στοιχεία της σελίδας. Ένας τέτοιος κανόνας αποτελείται από τα εξής στοιχεία (Σχήμα 1.1):

- **Επιλογέας** (selector). Καθορίζει το στοιχείο ή τα στοιχεία στα οποία έχει ισχύ ο κανόνας.
- **Δήλωση** (declaration). Αποτελείται από μια **ιδιότητα** (property) και την **τιμή** της (value).



```
h1, p {  
  color: blue;  
  padding: 1em;  
}
```

Σχήμα 1.1: Τα συστατικά ενός κανόνα CSS.

Για παράδειγμα, ο παρακάτω κανόνας καθορίζει πως όλες οι παράγραφοι (τα στοιχεία **<p>**) θα έχουν κείμενο με σκούρο γκρι χρώμα:

```
p {  
  color: darkgray;  
}
```

1.3 Άλλες μορφές δηλώσεων CSS

Οι κανόνες CSS είναι ο πιο συνηθής τρόπος που έχουμε για να δώσουμε στον φυλλομετρητή οδηγίες για την εμφάνιση της σελίδας μας. Υπάρχουν ωστόσο και άλλες μορφές κανόνων στη γλώσσα CSS.

1.3.1 at-rules

Οι κανόνες αυτοί αρχίζουν με το σύμβολο @ (at) και δηλώνουν μεταδεδομένα ή άλλες πληροφορίες. Συνήθεις at-rules είναι:

- **@import.** Φορτώνει κανόνες που βρίσκονται σε εξωτερικό αρχείο, π.χ.

```
@import url('page-url.css');
```

- **@media.** Εφαρμόζει το στυλ ανάλογα με το μέσο (media) προβολής, π.χ.

```
@media screen and (min-width: 900px) {  
    ...  
}
```

- **@supports.** Εφαρμόζει το στυλ ανάλογα με το επίπεδο CSS που υποστηρίζει ο φυλλομετρητής, π.χ.

```
@supports not (display: grid) {  
    ...  
}
```

- **@font-face.** Καθορίζει μια γραμματοσειρά.

1.4 Επιλογείς CSS

Οι επιλογείς ενός κανόνα CSS καθορίζουν σε ποια στοιχεία θα εφαρμοστούν οι δηλώσεις. Οι βασικοί επιλογείς είναι ο **καθολικός επιλογέας**, ο **επιλογέας τύπου**, ο **επιλογέας κλάσης**, ο **επιλογέας id** και ο **επιλογέας ιδιότητας**. Απλοί επιλογείς μπορούν να συνδυαστούν σε πιο πολύπλοκους, όπως θα δούμε στη συνέχεια. Η τρέχουσα πρόταση της W3C για τους επιλογείς είναι αυτή του επιπέδου 3 ([Selectors Level 3](#)).

Στη συνέχεια αυτής της ενότητας θα παρουσιαστούν σύντομα οι διαθέσιμοι επιλογείς καθώς και το πώς μπορούν να συνδυαστούν. Η παράθεση των επιλογέων έχει έναν κάπως εγκυκλοπαιδικό χαρακτήρα, καθώς οι επιλογείς είναι πολλοί και παρουσιάζονται

σε συντομία, συνδεδεμένοι από σύντομα παραδείγματα. Συνίσταται να αποκτηθεί επίγνωση όλων των επιλογών, ακόμη και όσον φαίνονται εξωτικοί με την πρώτη ματιά. Σε κάθε περίπτωση ο αναγνώστης μπορεί να ανατρέξει σε αυτή την ενότητα για να διελευκάνει τον τρόπο λειτουργίας των πιο ιδιαίτερων επιλογών και αργότερα.

1.4.1 Καθολικός επιλογέας (Universal selector)

Ο καθολικός επιλογέας, που συμβολίζεται με *, επιλέγει όλα τα στοιχεία, οποιουδήποτε τύπου. Επιλέγονται όλα τα στοιχεία της σελίδας, π.χ.

```
* {
    color: black; /* όλα τα στοιχεία της σελίδας θα έχουν μαύρα γράμματα */
}
```

1.4.2 Επιλογέας τύπου (Type selector)

Ο επιλογέας τύπου επιλέγει όλα τα στοιχεία του συγκεκριμένου τύπου, π.χ.

```
p, h1, ul {
    color: green; /* όλα τα στοιχεία p, h1 και ul θα έχουν πράσινα γράμματα */
}
```

Συμπληρωματικά με τον επιλογέα τύπου, έχουμε και τον επιλογέα ψευδο-στοιχείου (pseudo-element), με τον οποίο μπορούμε να επιλέξουμε ένα *τμήμα* του στοιχείου. Για παράδειγμα, με το ψευδο-στοιχείο **::first-line** μπορούμε να επιλέξουμε την πρώτη γραμμή μιας παραγράφου. Αυτή η γραμμή μπορεί στην οθόνη του χρήστη να περιλαμβάνει διαφορετικό πλήθος λέξεων, που αλλάζει ανάλογα με το διαθέσιμο πλάτος στον εκάστοτε φυλλομετρητή. Άλλα ενδιαφέροντα ψευδο-στοιχεία είναι τα **::before** και **::after**, που εισάγουν περιεχόμενο στην αρχή ή στο τέλος του επιλεγμένου στοιχείου:

```
/* Εισάγει τη λέξη "Προσοχή: " στην αρχή κάθε στοιχείου p με κλάση .warning */
p.warning::before {
    content: "Προσοχή: "
}
```

1.4.3 Επιλογέας κλάσης (Class selector)

Επιλέγονται όλα τα στοιχεία που στην ιδιότητα class τους αναγράφεται ο επιλογέας. Ο επιλογέας κλάσης αρχίζει με μια τελεία . ακολουθούμενη από το όνομα μιας κλάσης π.χ.

1 CSS

```
.first {  
    font-weight: bold; /* τα στοιχεία με κλάση first θα έχουν έντονα γράμματα */  
}  
  

```

1 CSS

```
<!-- κάθε id πρέπει να είναι μοναδικό στη σελίδα -->  
<span id="friendly">Καλωσήλθατε</span>
```

Σημειώστε πως ένα στοιχείο μπορεί να έχει μόνο μια τιμή στην ιδιότητα `id`, δηλαδή δεν επιτρέπεται να έχουμε τιμή στο `id` που να περιέχει το κενό διάστημα. Επίσης, σύμφωνα με το πρότυπο, η τιμή του `id` **πρέπει να είναι μοναδική** στη σελίδα.

Αν χρησιμοποιήσουμε το ίδιο `id` σε περισσότερα στοιχεία στη σελίδα μας, ο φυλλομετρητής δε θα τα αγνοήσει αλλά θα εφαρμόσει τον κανόνα CSS σε όλα τα στοιχεία με το ίδιο `id`. Ωστόσο, ενθαρρύνεται ο κώδικας να είναι σύμφωνος με τα πρότυπα.

1.4.5 Επιλογέας ιδιότητας (Attribute selector)

Ο επιλογέας ιδιότητας επιτρέπει την επιλογή με βάση την ύπαρξη και τις τιμές που μπορεί να έχει κάποια ιδιότητα των στοιχείων. Ο επιλογέας αυτός είναι πιο σύνθετος καθώς υπάρχει σε τέσσερις μορφές, ανάλογα με την τιμή που μπορεί να έχει η ιδιότητα:

- **[ιδιότητα]**

Στην πιο απλή μορφή του, μπορεί να χρησιμοποιηθεί για να επιλέξει τα στοιχεία που έχουν κάποια ιδιότητα, ανεξάρτητα από την τιμή της, π.χ.

```
/* τα στοιχεία που έχουν την ιδιότητα disabled θα εμφανίζονται με μια οριζόντια  
↪ διαγράμμιση */  
[disabled] {  
    text-decoration: line-through;  
}
```

- **[ιδιότητα=τιμή]**

Τα στοιχεία τα οποία στην ιδιότητα έχουν ακριβώς την καθορισμένη τιμή.

```
/* τα στοιχεία με την ιδιότητα type='button' θα εμφανίζονται με μαύρο περίγραμμα */  
[type=button] {  
    border-color: black;  
}
```

- **[ιδιότητα~τιμή]**

Σε μερικές ιδιότητες μπορούν να καταχωρηθεί μια λίστα με τιμές χωρισμένες με κενό διάστημα. Ο επιλογέας `~` επιλέγει τα στοιχεία που στη λίστα τιμών τους περιέχεται η τιμή.

- **[ιδιότητα|τιμή]**

Ο επιλογέας `|=` επιλέγει όλα τα στοιχεία που η ιδιότητά τους είναι είτε ακριβώς τιμή, είτε αρχίζει με τιμή-. Συχνό παράδειγμα είναι τιμές που αφορούν κωδικό γλώσσας, όπως `en-US`, `en-UK`, `el-GR`, `el-CY` κλπ.

```
/* τα στοιχεία <a> με την ιδιότητα hreflang='el' ή οτιδήποτε αρχίζει με
↪ hreflang='el-.....' εμφανίζονται υπογραμμισμένα */
a[hreflang|="el"] {
    text-decoration: underline;
}
```

Επιπλέον, ο επιλογέας ιδιότητας μπορεί να επιλέξει και με βάση τμήμα της τιμής (substring matching) με τρεις επιπλέον τρόπους:

- **[ιδιότητα^=τιμή]**. Επιλέγει στοιχεία που η τιμή της ιδιότητας **αρχίζει** με τιμή.
- **[ιδιότητα\$=τιμή]**. Επιλέγει στοιχεία που η τιμή της ιδιότητας **τελειώνει** με τιμή.
- **[ιδιότητα*=τιμή]**. Επιλέγει στοιχεία που η τιμή της ιδιότητας **περιέχει** τουλάχιστον μια φορά το αλφαριθμητικό τιμή.

1.4.6 Επιλογέας ψευδοκλάσης (Pseudoclass selector)

Οι επιλογείς ψευδοκλάσης μας επιτρέπουν να επιλέξουμε στοιχεία με άλλον τρόπο, πέρα από την αναζήτηση στο DOM. Οι επιλογείς αυτοί αρχίζουν πάντα με `:` ακολουθούμενο από το όνομα της ψευδοκλάσης. Οι ψευδοκλάσεις μπορούν μεταξύ άλλων να αφορούν τη δυναμική ή στατική κατάσταση ενός στοιχείου, τη θέση του μέσα στο DOM κλπ.

1.4.6.1 Δυναμικές ψευδοκλάσεις

Οι δυναμικές ψευδοκλάσεις είναι οι

- **`:link`**. Επιλέγει τους συνδέσμους που δεν έχει επισκεφθεί ο χρήστης του φυλλομετρητή
- **`:visited`**. Επιλέγει τους συνδέσμους που έχει επισκεφθεί ο χρήστης του φυλλομετρητή,
- **`:hover`**. Επιλέγει τον σύνδεσμο τον οποίο δείχνει ο χρήστης, πάνω από τον οποίο δηλαδή αιωρείται ο δείκτης της συσκευής κατάδειξης (π.χ. του ποντικιού),
- **`:active`**. Επιλέγει τον σύνδεσμο που ενεργοποιείται από τον χρήστη, δηλαδή τον υπερσύνδεσμο στον οποίο έχει κλικάρει ο χρήστης, για όσο διαρκεί το κλικ.
- **`:focus`**. Επιλέγει τον σύνδεσμο που είναι εστιασμένος, που μπορεί να έχει επιλεγεί με το πληκτρολόγιο ή το ποντίκι.

1 CSS

```
a:link { ... }      /* σύνδεσμοι που δεν έχει επισκεφθεί ο χρήστης */
a:visited { ... }   /* σύνδεσμοι που έχει επισκεφθεί ο χρήστης */
a:hover { ... }      /* σύνδεσμοι που από πάνω τους αιωρείται ο δείκτης του ποντικιού */
a:active { ... }     /* ενεργοποιημένος σύνδεσμος */
a:focus:hover { ... } /* σύνδεσμοι που έχουν την εστίαση και που πάνω τους αιωρείται ο
↪ δείκτης του ποντικιού */
```

1.4.6.2 Δομικές ψευδοκλάσεις

Οι δομικές ψευδοκλάσεις επιλέγουν στοιχεία με βάση τη θέση τους στο DOM, με τρόπο που δεν παρέχεται από άλλους επιλογείς.

- **:root**. Επιλέγει το αρχικό στοιχείο, που σε μια σελίδα HTML είναι το στοιχείο `html`.
- **:empty**. Επιλέγει στοιχεία που δεν έχουν παιδιά.

Μια πολύ χρήσιμη κατηγορία ψευδοκλάσεων είναι αυτές που επιλέγουν στοιχεία με βάση τη θέση τους στο DOM, με πιο χαρακτηριστικό τον επιλογέα **:nth-child($An+B$)**. Τα A και B είναι σταθερές που δίνονται από το χρήστη. Επιλέγονται όλα τα στοιχεία που βρίσκονται στη θέση i που υπολογίζεται από την έκφραση $i=An+B$, για μη αρνητικό ακέραιο n . Οι θέσεις των στοιχείων ξεκινούν από το 1. Ας δούμε μερικά παραδείγματα στο παρακάτω απόσπασμα HTML:

```
<body>
  <p>1ο στοιχείο</p>
  <p>2ο στοιχείο</p>
  <p>3ο στοιχείο</p>
  <p>4ο στοιχείο</p>
  <p>5ο στοιχείο</p>
  <p>6ο στοιχείο</p>
  <p>7ο στοιχείο</p>
  <p>8ο στοιχείο</p>
  <p>9ο στοιχείο</p>
</body>
```

Ο επιλογέας **:nth-child($3n$)** θα επιλέξει κάθε 3ο στοιχείο. Ο επιλογέας αυτός ισοδυναμεί με **:nth-child($3n+0$)**. Ξεκινώντας από το $n=0$, η έκφραση $i=3*0+0$ ισούται με 0 και συνεπώς θα επιλεγούν τα στοιχεία για $n=1$, $n=2$ και $n=3$ στις θέσεις 3, 6 και 9 αντίστοιχα. Ο επιλογέας **:nth-child($3n+1$)** θα επιλέξει το 1ο, το 4ο και το 7ο στοιχείο. Αντίστοιχα ο επιλογέας **:nth-child($3n-1$)** θα επιλέξει το 2ο, το 5ο και το 8ο στοιχείο.

Όλα τα στοιχεία που βρίσκονται σε ζυγές ή μονές θέσεις μπορούμε να τα επιλέξουμε και με τις λέξεις-κλειδιά `even` και `odd`, π.χ. **:nth-child(even)**. Μπορούμε να κάνουμε

1 CSS

την επιλογή μετρώντας ανάποδα, από το τέλος προς την αρχή, με τον επιλογέα `:nth-last-child(A n +B)`.

Τέλος, μπορούμε να χρησιμοποιήσουμε και τους επιλογείς `:nth-of-type`, `:nth-last-of-type`, `:first-of-type`, `:last-of-type` που λειτουργούν παρόμοια, με τη διαφορά πως επιλέγονται μόνο στοιχεία ενός συγκεκριμένου τύπου. Για παράδειγμα ο επιλογέας `p:nth-of-type(2n)` επιλέγει μόνο κάθε 2η παράγραφο.

Άσκηση

Ποια στοιχεία θα επιλεγούν με τον επιλογέα `:nth-child(-3 n +1)`;

Απάντηση:

Μόνο το 1ο στοιχείο, δηλ. για $n=0$, που βρίσκεται στη θέση $i=-3*0+1=1$, καθώς για $n \geq 1$ το i θα είναι αρνητικό.

Άσκηση

Γράψτε έναν επιλογέα που να επιλέγει όλα τα στοιχεία σε ζυγές θέσεις και έναν που να επιλέγει αυτά σε μονές θέσεις.

Απάντηση:

Τα στοιχεία σε ζυγές θέσεις μπορούν να επιλεγούν για $A=2$ και $B=0$, δηλαδή `:nth-child(2 n)`. Αντίστοιχα τα στοιχεία σε μονές θέσεις επιλέγονται με τον επιλογέα `:nth-child(2 n +1)`.

1.4.7 Συνδυασμοί επιλογέων

Υπάρχει λοιπόν ένας μεγάλος αριθμός από επιλογείς με τους οποίους μπορούμε να επιλέξουμε στοιχεία και να στοχεύσουμε έτσι το πού θα εφαρμοστούν οι κανόνες μας. Συνδυάζοντάς τους μπορούμε όμως να έχουμε ακόμη πιο σύνθετους επιλογείς.

Στη συνέχεια με A και B σημειώνονται οι επιλογείς που συνδυάζονται.

- **A B.** Είναι ο πιο συνηθής συνδυασμός και επιλέγει τα στοιχεία B που είναι απόγονοι, άμεσοι ή και όχι, των στοιχείων A. Οι δύο επιλογείς A και B χωρίζονται με το κενό διάστημα.
- **A>B.** Σε αυτή την περίπτωση επιλέγονται τα B που είναι **άμεσοι** απόγονοι των A.
- **A~B.** Επιλέγονται τα B που είναι αδέρφια των A, όχι απαραίτητα άμεσα. Αδέρφια είναι τα στοιχεία που έχουν τον ίδιο άμεσο πρόγονο.
- **A+B.** Επιλέγονται τα B που είναι άμεσα επόμενα αδέρφια των A, δηλ. όσα B που ακολουθούν αμέσως μετά από ένα A.

1 CSS

Μπορούμε επίσης, με το , (κόμμα), να κατασκευάσουμε μια λίστα επιλογών, στους οποίους θα εφαρμοστεί ο κανόνας:

- **A,B.** Επιλέγονται τα στοιχεία που επιλέγονται από τον A καθώς και τα στοιχεία που επιλέγονται από τον B. Αν έστω και ένας από τους επιλογείς της λίστας δεν είναι συντακτικά έγκυρος, τότε είναι άκυρη όλη η λίστα. Για παράδειγμα, το

```
h1 {font-color: rebeccapurple}
```

```
h2 {font-color: rebeccapurple}
```

```
h3 {font-color: rebeccapurple}
```

ισοδυναμεί με

```
h1,h2,h3 {font-color: rebeccapurple}
```

1.5 Αλληλουχία και σειρά εφαρμογής των κανόνων

Ένας κανόνας CSS περιέχει ιδιότητες και τις τιμές τους και ο επιλογέας καθορίζει σε ποια στοιχεία θα εφαρμοστούν τα ζεύγη ιδιοτήτων-τιμών. Πολύ συχνά, περισσότεροι από έναν κανόνες (Σχήμα 1.1) μπορεί να περιέχουν δηλώσεις που να αφορούν τις ίδιες ακριβώς ιδιότητες για το ίδιο στοιχείο. Ο μηχανισμός που αποφασίζει ποιες τιμές τελικά θα επικρατήσουν εξετάζει την προέλευση του κανόνα και αν η ιδιότητα έχει σημανθεί ως σημαντική ή όχι, το πόσο στοχευμένος είναι ο επιλογέας καθώς και τη σειρά με την οποία φορτώθηκε. Επίσης, κάποιες από τις ιδιότητες κληρονομούν την τιμή τους από τα στοιχεία που βρίσκονται πιο πριν στο δέντρο, ενώ άλλες ιδιότητες όχι. Αυτό που εκ πρώτης όψεως μπορεί να φαίνεται χαοτικό, μας επιτρέπει ωστόσο να γράφουμε τις ελάχιστες δυνατές δηλώσεις. Ο μηχανισμός εφαρμογής των δηλώσεων είναι τόσο κεντρικός στη CSS που είναι μέρος του ονόματος: ``Cascading" στο Cascading Style Sheets σημαίνει ακριβώς ``φύλλα στυλ" που εφαρμόζονται σε ``αλληλουχία". Η λειτουργία του μηχανισμού αυτού περιγράφεται στην πρόταση Cascading and Inheritance (Etemad και Jr. 2021).

Μπορούμε να συμπεράνουμε πως έχει σημασία η σειρά με την οποία συναντά ο φυλλομετρητής τους κανόνες. Στο τέλος βέβαια, **όλες** οι ιδιότητες **όλων** των στοιχείων θα έχουν τιμές. Αν έχουν οριστεί τιμές σε περισσότερα από ένα σημεία, τότε μόνο μια από αυτές θα επικρατήσει. Ο τρόπος με τον οποίο καθορίζεται η τιμή που θα επικρατήσει εξαρτάται από την εξής αλληλουχία, από το πιο σημαντικό στο λιγότερο σημαντικό:

- την **προέλευση** του κανόνα όπου δηλώνεται η τιμή και αν η τιμή έχει σημανθεί ως σημαντική ή όχι,

- την **ειδικότητα**, δηλ. το πόσο στοχευμένος είναι ο επιλογέας που περιέχει τη δήλωση,
- τη **σειρά εμφάνισης** του κανόνα στον κώδικα,
- το αν η τιμή της ιδιότητας **κληρονομείται** ή όχι.

Στη συνέχεια θα δούμε τα επιμέρους τμήματα της αλληλουχίας με λίγη περισσότερη λεπτομέρεια.

1.5.1 Προέλευση κανόνων

Ένας κανόνας μπορεί να υπάρχει στις εξής τρεις προελεύσεις:

1. Στο λεγόμενο **στυλ φυλλομετρητή** (*user agent style*). Κάθε φυλλομετρητής έχει ενσωματωμένο ένα σύνολο κανόνων CSS, το λεγόμενο και προκαθορισμένο στυλ CSS, το οποίο έχει οριστεί από τον κατασκευαστή του. Το στυλ αυτό καθορίζει πώς θα εμφανιστούν τα στοιχεία, αν δεν υπάρχει άλλη CSS. Για παράδειγμα, το ότι τα γράμματα θα είναι μαύρα σε λευκό φόντο, ή ότι τα **** θα εμφανίζονται με έντονη γραμματοσειρά, ακόμη και αν δεν έχουμε ορίσει δικό μας στυλ, καθορίζεται στο στυλ φυλλομετρητή.
2. Στο **στυλ χρήστη** (*user defined style*) Ο χρήστης μπορεί να ορίσει τα δικά του αρχεία CSS και να ζητήσει από τον φυλλομετρητή να χρησιμοποιεί αυτά αντί για το προκαθορισμένο στυλ φυλλομετρητή. Όπως και το στυλ φυλλομετρητή, το στυλ χρήστη αφορά όλες τις ιστοσελίδες που φορτώνει ο χρήστης στο φυλλομετρητή. Καθώς το στυλ χρήστη είναι ρητή ενέργεια του χρήστη, οι κανόνες που περιέχει έχουν μεγαλύτερη προτεραιότητα από το στυλ φυλλομετρητή.
3. Στα στυλ που παρέχει ο δημιουργός της ιστοσελίδας, με τρεις τρόπους.

1. **Εξωτερικό αρχείο CSS.** Ο δημιουργός της ιστοσελίδας ορίζει ένα ή περισσότερα αρχεία CSS και τοποθετεί το URL τους στο τμήμα <head> της σελίδας, γράφοντας για παράδειγμα

```
<link rel="stylesheet" href="filename-url.css">
```

2. **Εσωτερικό CSS** (ή CSS σελίδας). Ο δημιουργός της σελίδας γράφει τους κανόνες CSS απευθείας στο τμήμα <head> μέσα στην ετικέτα <style>, π.χ.

```
<head>
```

```
...
```

```
<style>
```

```
p {
```

```
    color: darkgray;
```

```
}
```

```
</style>
```

```
..
```

```
</head>
```

Το εσωτερικό CSS αφορά μόνο τη συγκεκριμένη σελίδα, και συνεπώς είναι πιο συγκεκριμένο από το εξωτερικό αρχείο CSS.

3. **Ενσωματωμένο ή inline στυλ.** Δηλώσεις CSS μπορούν να γραφούν και μέσα σε κάθε ετικέτα HTML, στην ιδιότητα `<style>`. Οι δηλώσεις αυτές έχουν εφαρμογή μόνο για το συγκεκριμένο στοιχείο, για παράδειγμα:

```
<p style="color: darkgray"> ... </p>
```

Οι παραπάνω πηγές κανόνων CSS είναι ταξινομημένες από τη λιγότερο στην περισσότερο σημαντική: στυλ φυλλομετρητή < στυλ χρήστη < εξωτερικό στυλ < εσωτερικό στυλ < ενσωματωμένο στυλ. Ο δημιουργός μιας ιστοσελίδας δεν έχει έλεγχο στις δύο πρώτες πηγές κανόνων CSS, το προκαθορισμένο στυλ και το στυλ χρήστη.

Από τις τρεις επιλογές που έχει ο δημιουργός ιστοσελίδων, το εξωτερικό CSS είναι η προτιμώμενη πρακτική. Ο λόγος είναι πως το εξωτερικό CSS μπορεί να αντιστοιχεί σε πολλές σελίδες HTML που συνθέτουν έναν ιστότοπο. Έτσι αν θέλουμε να διορθώσουμε κάποιον κανόνα CSS που να αφορά όλες τις σελίδες του ιστοτόπου, π.χ. στο μενού, αρκεί τότε να τον αλλάξουμε σε ένα μόνο σημείο, στο εξωτερικό CSS. Στο άλλο άκρο, αποθαρρύνεται η χρήση του ενσωματωμένου (inline) στυλ, καθώς οι δηλώσεις αυτές έχουν τη μεγαλύτερη προτεραιότητα. Έτσι αν θέλουμε να διορθώσουμε κάτι, θα πρέπει να το κάνουμε σε όλα τα στοιχεία HTML του ιστοτόπου μας στα οποία έχουμε εφαρμόσει τον ενσωματωμένο κανόνα, ή να χρησιμοποιήσουμε τη λέξη `!important`.

Όλοι οι κανόνες CSS είναι είτε μη σημαντικοί (normal) είτε σημαντικοί (important). Σημαντικοί είναι αυτοί που η τιμή τους τελειώνει με `!important`, για παράδειγμα:

```
span .serious {
  color: red !important;
}
```

Μια σημαντική (`!important`) ιδιότητα έχει αυτομάτως μεγαλύτερο βάρος, δηλ. ανεξάρτητα σε ποια πηγή τη συναντήσαμε, η τιμή της θα επικρατήσει. Ωστόσο τί γίνεται σε περίπτωση που η ίδια ιδιότητα έχει σημανθεί σαν σημαντική σε περισσότερες από μια πηγές; Τότε η προτεραιότητα επικράτησης θα **αντιστραφεί**. Συνολικά λοιπόν έχουμε την σειρά προτεραιότητας μιας δήλωσης, από την περισσότερο προς τη λιγότερο σημαντική, που φαίνεται στον πίνακα 1.1:

1 CSS

Πίνακας 1.1: Προτεραιότητες δηλώσεων ανάλογα με την πηγή στην οποία βρίσκονται. Οι δηλώσεις που έχουν να κάνουν με *ενεργές* μεταβάσεις (transitions) έχουν τη μεγαλύτερη προτεραιότητα (1) ενώ αυτές που αφορούν τα ενεργά animation έχουν μεγαλύτερη προτεραιότητα μόνο από τις κανονικές δηλώσεις.

Σειρά προτεραιότητας	Πηγή κανόνων
1	Δηλώσεις μετάβασης (transition)
2	!important στο στυλ του φυλλομετρητή
3	!important στο στυλ του χρήστη
4	!important στο στυλ του δημιουργού
5	Animation
6	Στο στυλ του δημιουργού
7	Στο στυλ του χρήστη
8	Στο στυλ του φυλλομετρητή

Συνεπώς, όταν ο φυλλομετρητής συναντήσει δύο διαφορετικές τιμές για την ίδια ιδιότητα ενός στοιχείου, θα επιλέξει αυτή με τη μεγαλύτερη προτεραιότητα, που προκύπτει από την πηγή προέλευσης του κανόνα.

Ερώτηση

Γιατί το στυλ χρήστη έχει μικρότερη προτεραιότητα από το στυλ δημιουργού, παρόλο που το πρώτο είναι ρητή ενέργεια του χρήστη;

1.5.2 Ειδικότητα

Αν η προέλευση δυο ή περισσότερων κανόνων είναι η ίδια, τότε θα επικρατήσει αυτός με τον πιο συγκεκριμένο επιλογέα. Το πόσο συγκεκριμένος είναι ο επιλογέας μετράται με την **ειδικότητά** του (specificity). Η ειδικότητα ενός επιλογέα συντίθεται από τρία συστατικά στοιχεία:

- το πλήθος των **επιλογέων id**,

1 CSS

- το πλήθος των **επιλογέων κλάσης** (π.χ. `.first`), των **επιλογέων ιδιότητας** (π.χ. `[type=button]`), και των **επιλογέων ψευδοκλάσης** (π.χ. `:link`),
- το πλήθος των **επιλογέων τύπου** (π.χ. `p`) και των **επιλογέων ψευδοστοιχείων** (όπως το `::first-line`).

Για παράδειγμα ο επιλογέας `#friendly` έχει ειδικότητα (1, 0, 0), καθώς αποτελείται από έναν επιλογέα `id` ενώ ο `#friendly #animal` έχει ειδικότητα (2, 0, 0). Αντίστοιχα ο επιλογέας `p.first` em έχει ειδικότητα (0, 1, 2) καθώς αποτελείται από δύο επιλογείς τύπου (`<p>` και ``) και έναν επιλογέα κλάσης (`.first`). Επικρατεί πάντα ο επιλογέας που έχει τη μεγαλύτερη τιμή από αριστερά προς τα δεξιά, δηλ. η ειδικότητα (1, 0, 0), είναι πιο ισχυρή από την (0, 1, 2). Στην περίπτωση που δύο επιλογείς έχουν την ίδια ειδικότητα θα επικρατήσει αυτός που συναντιέται τελευταίος.

1.5.3 Σειρά εμφάνισης

Αν δεν αρκούν τα δύο προηγούμενα κριτήρια για να αποφασιστεί η τιμή που θα επικρατήσει, δηλ. αν οι κανόνες βρίσκονται στο ίδιο επίπεδο προέλευσης (πίνακας 1.1) και οι επιλογείς έχουν την ίδια ειδικότητα, τότε θα επικρατήσει η πιο πρόσφατη δήλωση, δηλ. αυτή που συνάντησε τελευταία ο φυλλομετρητής όταν φόρτωσε την ιστοσελίδα:

```
/* Τελικά θα έχουμε κίτρινα γράμματα σε μαύρο φόντο */
p {
    color: yellow;
    background-color: navy;
}
p {
    background-color: black;
}
```

1.5.4 Κληρονομικότητα

Όπως αναφέρθηκε, όλες οι ιδιότητες όλων των στοιχείων πρέπει υποχρεωτικά να έχουν μία τιμή. Αν η αλληλουχία δεν αποδώσει τιμή σε μια ιδιότητα, τότε θα χρησιμοποιηθεί η κληρονομικότητα. Όλες οι ιδιότητες έχουν προδιαγραφεί με μια **αρχική τιμή** (initial value). Για παράδειγμα η αρχική τιμή της ιδιότητας `font-size` είναι `medium` (Lilley, Maxfield, και Daggett 2018). Αν λοιπόν δεν προκύψει τιμή από την αλληλουχία, τότε, ανάλογα την ιδιότητα, η τιμή αυτή θα είναι είτε η αρχική τιμή της, είτε θα κληρονομηθεί η τιμή (inherited value) από τον άμεσο πρόγονο στο DOM. Το αν θα γίνει αυτό εξαρτάται από το αν η ιδιότητα είναι κληρονομούμενη ή όχι, το οποίο προδιαγράφεται από το

1 CSS

πρότυπο της CSS. Ένας εμπειρικός κανόνας είναι πως γενικά, οι τιμές ιδιοτήτων που σχετίζονται με την εμφάνιση του κειμένου κληρονομούνται. Αντίστοιχα, γενικά δεν κληρονομούνται οι τιμές ιδιοτήτων που αφορούν την εμφάνιση των στοιχείων, π.χ. τα περιθώρια ή το περίγραμμα.

Για παράδειγμα, ας ορίσουμε στη CSS μας τις εξής τιμές για τη γραμματοσειρά και το μέγεθός της:

```
body {  
    font-size: 1.2em;  
    font-family: serif;  
}
```

Οι τιμές αυτές θα εφαρμοστούν σε όλο το κείμενο της σελίδας μας, π.χ. στα στοιχεία `<p>` που πιθανώς περιέχονται στην σελίδα μας,¹ καθώς τόσο το `font-size` όσο και το `font-family` κληρονομούνται, όπως μπορούμε να διαπιστώσουμε στο CSS Fonts Module Level 3 (Lilley, Maxfield, και Daggett 2018), ή στην τεκμηρίωση του MDN (contributors 2021). Ωστόσο αν ορίσουμε επιπλέον τον εξής κανόνα, που αφορά το εξωτερικό περιθώριο του στοιχείου `<body>`

```
body {  
    margin: 1em;  
}
```

τότε αυτό θα εφαρμοστεί μόνο για το στοιχείο `<body>`, καθώς οι τιμές των ιδιοτήτων `padding`, `border`, `margin` δεν κληρονομούνται, όπως μπορούμε να διαπιστώσουμε στο αντίστοιχο πρότυπο [Etemad:20:CBM] ή την τεκμηρίωση του MDN.

Η συμπεριφορά αυτή μπορεί να αλλάξει ρητά. Μπορούμε δηλ. να ζητήσουμε η ιδιότητα να πάρει την αρχική ή την κληρονομημένη τιμή χρησιμοποιώντας τις τιμές `initial` και `inherit`:

```
p {  
    padding: inherit;  
    background-color: initial;  
}
```

1.6 Τιμές και μονάδες απόστασης και χρώματος

Κάθε ιδιότητα έχει έναν *τύπο* αποδεκτών τιμών. Οι τύποι αυτοί καλύπτουν διάφορες κατηγορίες όπως κείμενο, απόσταση, αριθμητική τιμή, γωνία, χρόνο, συχνότητα,

¹Εκτός φυσικά και αν στη CSS μας υπάρχουν πιο στοχευμένες οδηγίες για τα στοιχεία `<p>`.

1 CSS

ανάλυση κλπ. Τους περισσότερους τύπους θα τους συναντήσουμε σε αυτό και στο επόμενο κεφάλαιο μέσα από τα παραδείγματα και σε αυτή την ενότητα θα δούμε με λεπτομέρεια τις μονάδες που εκφράζουν την απόσταση και το χρώμα.

Φυσικά για να εκφράσουμε κάποιο μέγεθος χρησιμοποιούμε αριθμούς, είτε ακραίους είτε πραγματικούς αριθμούς. Οι αριθμοί αυτοί μπορεί να εμφανίζονται μόνοι τους είτε με κάποια μονάδα μέτρησης, όπως π.χ. κάποια μονάδα που μετράει απόσταση. Αριθμητικοί τύποι μπορούν να εκφραστούν και σαν ποσοστά κάποιας άλλης ποσότητας, π.χ. του πλάτους ενός στοιχείου ή του μεγέθους μιας γραμματοσειράς. Οι μονάδες και τιμές που χρησιμοποιούνται στη CSS περιγράφονται στο στην υποψήφια πρόταση CSS Values and Units (Etemad και Jr. 2019).

1.6.1 Απόσταση

Οι τιμές που εκφράζουν απόσταση μετρούν είτε τη **σχετική** είτε την **απόλυτη** απόσταση. Οι σχετικές αποστάσεις υπολογίζουν την απόσταση με βάση κάποια άλλη απόσταση αναφοράς που μπορεί να είναι είτε σε σχέση με το μέγεθος γραμματοσειράς, είτε σε σχέση με τις διαστάσεις του παραθύρου προβολής (viewport), δηλ. τις διαστάσεις του *κουτιού* μέσα στο οποίο περιέχεται η σελίδα μας. Οι σχετικές αποστάσεις είναι χρήσιμες όταν δεν γνωρίζουμε τις διαστάσεις στις οποίες θα προβληθεί η ιστοσελίδα μας. Διαδεδομένη είναι η χρήση της μονάδας *em*, που είναι ίση με το μέγεθος της γραμματοσειράς του στοιχείου στο οποίο εφαρμόζεται.

Πίνακας 1.2: Σύνοψη των σχετικών μονάδων στο επίπεδο 3 του CSS Values and Units Module (Etemad και Jr. 2019). Στο επίπεδο 4 της ίδιας ενότητας, που είναι όμως ακόμη πρόχειρη (working draft), έχουν εισαχθεί και άλλες μονάδες

Μονάδα	Σχετική με
em	το μέγεθος γραμματοσειράς του στοιχείου
ex	το ύψος του ``x" στη γραμματοσειρά του στοιχείου
ch	το συνολικό πλάτος του ``0" (ZERO, U+0030)
rem	το μέγεθος γραμματοσειράς του ριζικού στοιχείου
vw	1% του πλάτους του παραθύρου προβολής
vh	1% του ύψους του παραθύρου προβολής

1 CSS

Μονάδα Σχετική με

Για παράδειγμα ο κανόνας

```
p {  
  padding: 1em;  
  font-size: 1.2em;  
}
```

ορίζει πως το μέγεθος της γραμματοσειράς του στοιχείου **<p>** θα είναι 120% του κληρονομημένου μεγέθους ενώ το εσωτερικό περιθώριο θα είναι ακριβώς όσο και το μέγεθος της γραμματοσειράς (που είναι πλέον το 120% του κληρονομημένου μεγέθους). Ποια θα είναι όμως τελικά η απόλυτη τιμή αυτού του μεγέθους όταν θα εμφανιστεί στην οθόνη; Αν δεν έχουμε αλλάξει σε κανέναν πρόγονο του **<p>** το μέγεθος της γραμματοσειράς, τότε θα κληρονομηθεί η τιμή που έχει οριστεί για το **<body>**, η οποία θα είναι η τιμή που ορίζει το στυλ φυλλομετρητή. Αυτό συνήθως είναι 16px και συνεπώς τόσο το padding όσο και το font-size θα έχουν τιμή $16 * 1,2 = 19,2\text{px}$.

Εκτός από την em, συνήθης είναι και η χρήση της μονάδας rem. Η διαφορά μεταξύ em και rem είναι πως η δεύτερη αναφέρεται πάντα στο μέγεθος της γραμματοσειράς του **<body>**. Η χρήση της rem είναι ίσως προτιμώτερη, καθώς οι αλλαγές του μεγέθους της γραμματοσειράς με την μονάδα em διαδίδονται αθροιστικά στο DOM:

```
p.warning { font-size: 2em; }
```

```
span.unit { font-size: 2em; }
```

<!-- Η παράγραφος θα έχει μέγεθος γραμματοσειράς 32px ενώ το span 64px -->

```
<p class="warning">Οι αλλαγές του μεγέθους της γραμματοσειράς εμφωλευμένων στοιχείων με  
↪ μονάδες <span class="unit">em</span> δρουν αθροιστικά.</p>
```

Οι αλλαγές του μεγέθους της γραμματοσειράς
εμφωλευμένων στοιχείων με μονάδες **em** δρουν
αθροιστικά.

Επειδή η ιδιότητα font-size είναι κληρονομούμενη (ενότητα 1.5.4), η αλλαγή στο μέγεθος της γραμματοσειράς ενός στοιχείου κληροδοτείται στους απογόνους του και δρα αθροιστικά. Με τη μονάδα rem μπορούμε να το αποφύγουμε αυτό.

1 CSS

Άλλες μονάδες που καθορίζονται από κάποιο μέγεθος γραμματοσειράς είναι οι *ex* και *ch*, η πρώτη σε σχέση με το μέγεθος του γράμματος *x* και η δεύτερη σε σχέση με το μέγεθος του *0*.

Σχετικές αποστάσεις μπορούν να πάρουν τιμές και σε σχέση με τις διαστάσεις του παραθύρου προβολής (*viewport*). Οι μονάδες αυτές είναι οι *vw* και *vh* που είναι ίσες με το 1% της του πλάτους ή του ύψους του *viewport*. Αν μας ενδιαφέρει πάντα η μικρότερη ή η μεγαλύτερη από αυτές τις δύο διαστάσεις, μπορούμε να χρησιμοποιήσουμε τις *vw* και *vmax*.

Πρέπει να σημειωθεί πως στην περίπτωση της κληρονομικότητας, οι σχετικές τιμές δεν κληρονομούνται, αλλά κληρονομείται η τιμή στην οποία τελικά θα καταλήξει ο υπολογισμός της σχετικής απόστασης.

Απόλυτες αποστάσεις μπορούν να οριστούν επίσης με πολλές μονάδες. Οι απόλυτες αποστάσεις είναι χρήσιμες όταν γνωρίζουμε τις διαστάσεις προβολής της ιστοσελίδας. Η πιο συχνά χρησιμοποιούμενη απόλυτη μονάδα είναι τα *πίξελ* (*px*), που αντιστοιχεί σε 1/96 της ίντσας. Όλες οι απόλυτες αποστάσεις φαίνονται στον πίνακα [1.3].

Πίνακας 1.3: Οι απόλυτες μονάδες και η αντιστοιχία τους (Etemad και Jr. 2019).

Μονάδα	Όνομα	Αντιστοιχία
cm	εκατοστά	1cm = 96px/2.54
mm	χιλιοστά	1mm = 1/10 του εκατοστού
Q	τέταρτα του χιλιοστού	1Q = 1/40 του εκατοστού
in	ίντσες	1in = 2.54cm = 96px
pc	πίκα	1pc = 1/6 της ίντσας
pt	σημεία	1pt = 1/72 της ίντσας
px	πίξελ	1px = 1/96 της ίντσας

1.6.2 Χρώμα

Χρωματικές τιμές μπορούν να καθοριστούν είτε με κάποιο από τα διαθέσιμα βασικά ονόματα χρωμάτων (*black*, *silver*, *gray*, *white*, *maroon*, *red*, *purple*, *fuchsia*, *reen*, *lime*,

1 CSS

olive, yellow, navy, blue, teal, aqua), είτε χρησιμοποιώντας κάποιο από τα εκτεταμένα χρώματα. Η πλήρης λίστα των χρωμάτων αυτών μπορεί να βρεθεί μεταξύ άλλων στο πρότυπο CSS Color Module Level 3 (Çelik κ.ά. 2018).

Είναι όμως δυνατό, αντί για τα ονοματισμένα χρώματα, να οριστεί το χρώμα με αριθμητικές τιμές. Με τις τιμές αυτές επιλέγεται ένα χρώμα από τα χρώματα που είναι διαθέσιμα στο χρωματικό χώρο sRGB (sRGB colorspace), που είναι με τη σειρά του ένας από τους πολλούς χρωματικούς χώρους που βασίζονται στο μοντέλο RGB. Αν και το sRGB δεν είναι σε θέση να αποτυπώσει όλα τα δυνατά χρώματα, και ούτε καν όλα τα χρώματα που μπορούν να αποτυπώσουν οι σύγχρονες οθόνες, είναι ωστόσο το πρότυπο που επικρατεί και σήμερα για την κωδικοποίηση χρωμάτων στο διαδίκτυο. Αναμιγνύοντας τα τρία συστατικά χρώματα, κόκκινο, πράσινο και μπλε (red, green, blue) του sRGB μπορούμε να παράξουμε όλα τα διαθέσιμα χρώματα. Στη CSS έχουμε στη διάθεσή μας 1 byte για κάθε ένα από τα τρία χρώματα, και άρα μπορούμε να επιλέξουμε τιμές από 0 ως 255 (ή από 0 ως FF στο δεκαεξαδικό σύστημα), και αυτό μπορούμε να το κάνουμε με διαφορετικούς τρόπους:

```
p { color: #f10; } /* ισοδυναμεί με #ff1100 */
p { color: #ff1100; }
p { color: rgb(255, 17, 0); }
p { color: rgb(100%, 6.7%, 0%); }
```

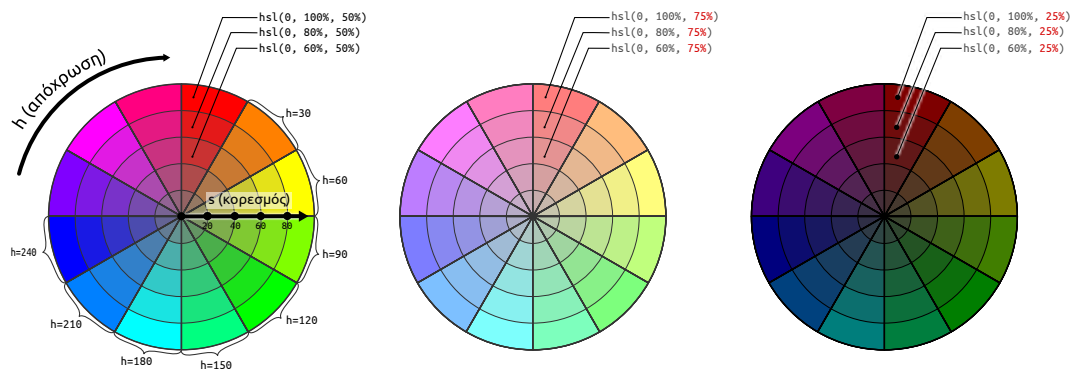
Συμπληρωματικά με την rgb() μπορούμε να χρησιμοποιήσουμε την rgba() για να δηλώσουμε και διαφάνεια με έναν πραγματικό αριθμό από 0, για πλήρη διαφάνεια, μέχρι 1, για πλήρη αδιαφάνεια. Μπορούμε να χρησιμοποιήσουμε και ποσοστό:

```
p { color: rgba(255, 17, 0, 1); } /* τελείως αδιαφανές */
p { color: rgba(100%, 6.7%, 0%, 50%); /* ημιδιαφανές */ }
```

Αντί για τα τρία βασικά χρώματα κόκκινο, πράσινο και μπλε, μπορούμε να ορίσουμε χρώμα με την αναπαράσταση HSV (hue, saturation, luminance - απόχρωση, κορεσμός, φωτεινότητα) [1.2]. Η απόχρωση, το ποιο χρώμα θέλουμε, δηλώνεται με έναν ακέραιο από 0 ως 360. Ο αριθμός αυτός δηλώνει γωνία στο χρωματικό κύκλο HSV, όπου 0=360=κόκκινο, 120=πράσινο και 240=μπλε. Ο κορεσμός, δηλ. το πόσο χρώμα θέλουμε, δηλώνεται με ένα ποσοστό, όπου το 0% είναι το γκρι. Αντίστοιχα και η φωτεινότητα, όπου 0% είναι το μαύρο, 100% το λευκό και 50% η ``κανονική'' φωτεινότητα. Όπως και με την rgb, έχουμε στη διάθεσή μας δύο τρόπους για να ορίσουμε χρώματα hsn, χωρίς ή με διαφάνεια:

```
p { color: hsl(255, 100%, 50%); } /* κόκκινο */
p { color: hsla(255, 100%, 50%, 0.5); /* ημιδιαφανές κόκκινο */ }
```

1 CSS



Σχήμα 1.2: Στην αναπαράσταση HSV η γωνία h επιλέγει την απόχρωση, ξεκινώντας από τις 0° για το κόκκινο. Στο συγκεκριμένο σχήμα επιλέγονται 12 χρώματα, ανά 30° . Όσο ο μειώνεται ο κορεσμός (s) από το 100% προς το 0%, το χρώμα πλησιάζει προς το γκρι, εδώ σε 5 βήματα του 20%. Ο παράγοντας l καθορίζει τη φωτεινότητα του χρώματος. Ο τρόπος αυτός επιλογής χρώματος είναι κάπως πιο προβλέψιμος σε σχέση με την αναπαράσταση RGB.

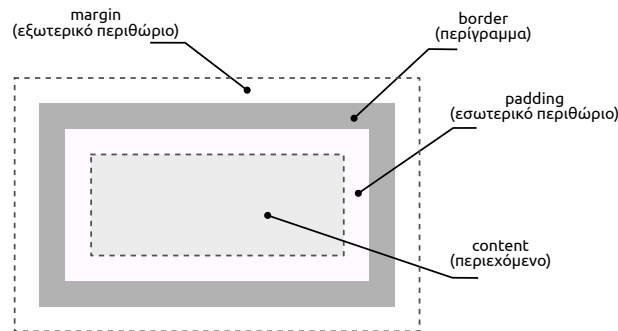
1.7 Το μοντέλο Box

Το μοντέλο box (Etemad 2020) είναι το θεμέλιο της διαδικασίας με την οποία ο φυλλομετρητής διατάσσει τα στοιχεία μιας ιστοσελίδας στην οθόνη. Στο μοντέλο αυτό, τα στοιχεία HTML που συνθέτουν την ιστοσελίδα περιγράφονται σαν ορθογώνια κουτιά, τα οποία καταλαμβάνουν συγκεκριμένο χώρο στην οθόνη και τα οποία μπορούμε να χειριστούμε με τη γλώσσα CSS.

Κάθε τέτοιο κουτί αποτελείται, από μέσα προς τα έξω, από το **περιεχόμενο** (content), το **εσωτερικό περιθώριο** (padding), το **περίγραμμα** (border) και το **εξωτερικό περιθώριο** (margin) (Σχήμα 1.3).

Όπως είδαμε, αν δεν εφαρμόσουμε το δικό μας στυλ CSS και αφήσουμε το φυλλομετρητή να διακοσμίσει τη σελίδα, τότε θα χρησιμοποιηθούν οι τιμές που ορίζονται στο προκαθορισμένο στυλ φυλλομετρητή (default stylesheet), το οποίο είναι ενσωματωμένο σε κάθε φυλλομετρητή.

1 CSS



Σχήμα 1.3: Το μοντέλο box.

Για κάθε κουτί μπορούμε να ορίσουμε την εξωτερική και την εσωτερική του συμπεριφορά, χρησιμοποιώντας την ιδιότητα **display**. Η εξωτερική συμπεριφορά περιγράφει το πώς το κουτί θα συμπεριφερθεί σε σχέση με τα γειτονικά του κουτιά. Υπάρχουν δύο τύποι κουτιών όσον αφορά τον εξωτερικό τύπο: τα κουτιά **inline** και τα κουτιά **block**.

Ένα στοιχείο τύπου **block** καταλαμβάνει από μόνο του όλο τον διαθέσιμο οριζόντιο χώρο, όλη δηλαδή τη γραμμή στην οποία είναι τοποθετημένο. Τέτοια στοιχεία είναι τα `<p>`, `<div>`, `<h1>` κλπ. Αντίθετα, ένα στοιχείο τύπου **inline** καταλαμβάνει οριζόντια μόνο όσο χώρο χρειάζεται για να προβληθεί. Τέτοια στοιχεία είναι τα ``, `` κλπ (Σχήμα 1.4).

Οι δύο αυτοί τύποι κουτιών διαφέρουν και σε άλλα σημεία. Ένα στοιχείο **block** θα είναι μόνο του στην οριζόντια γραμμή όπου βρίσκεται και θα καταλαμβάνει όλο το οριζόντιο εύρος αυτής της γραμμής, χωρίς άλλα στοιχεία αριστερά ή δεξιά του. Τα εσωτερικά και εξωτερικά περιθώρια και το περίγραμμα (`padding`, `border` και `margin`) του κουτιού τύπου **block** γίνονται σεβαστά και ``σπρώχνουν`` τα γειτονικά του στοιχεία που έπονται έτσι ώστε να χωρέσει το κουτί. Επίσης σεβαστές γίνονται και οι τιμές των ιδιοτήτων `width` και `height`.

Αντίθετα, ένα στοιχείο τύπου **inline** εμφανίζεται δίπλα σε άλλα στοιχεία **inline**, το ένα μετά το άλλο στην οριζόντια γραμμή. Αν δεν χωράει στην οριζόντια γραμμή, το στοιχείο **inline** αναδιπλώνεται σε νέα γραμμή και συνεχίζει από κάτω. Τα εσωτερικά και εξωτερικά περιθώρια και το περίγραμμα (`padding`, `border` και `margin`) του κουτιού τύπου **inline** γίνονται σεβαστά, αλλά προκαλούν τη μετατόπιση των γειτονικών στοιχείων μόνο στον οριζόντιο άξονα. Οι τιμές στις ιδιότητες `width` και `height` δεν έχουν κάποιο αποτέλεσμα.

Το αν ένα στοιχείο είναι **block** ή **inline** ορίζεται από την ιδιότητα `display`. Οι αντίστοιχες δηλώσεις είναι `display: block` και `display: inline`.

Η δυνατότητα να αλλάξουμε το πώς προβάλλεται ένα στοιχείο είναι σημαντική. Μας επιτρέπει να επιλέξουμε το στοιχείο HTML που ταιριάζει σημασιολογικά για το σκοπό

μας και να ορίσουμε τον τρόπο προβολής του μέσω της CSS.

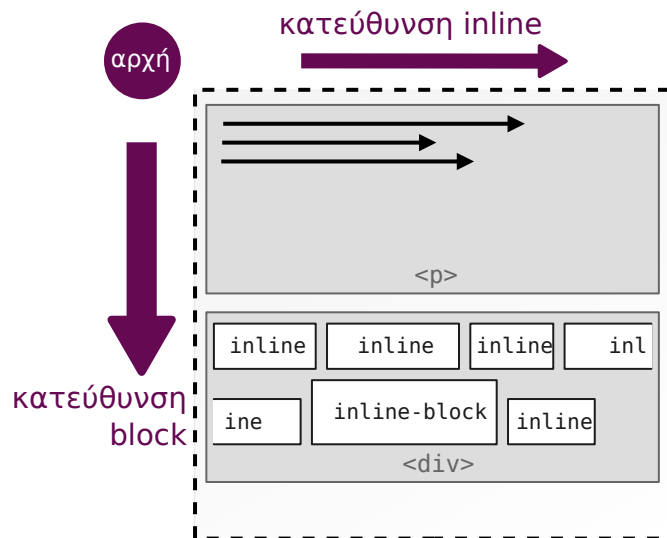
Η τρίτη βασική τιμή της ιδιότητας `display` είναι η `none` που αφαιρεί τελείως το στοιχείο από την προβολή. Το στοιχείο δεν εμφανίζεται στην οθόνη, αλλά ούτε και δεσμεύεται ο χώρος που θα καταλάμβανε.

Όλα τα στοιχεία έχουν μια προκαθορισμένη τιμή για το `display`. Για παράδειγμα το `<p>` ή το `<h1>` είναι στοιχεία `block`. Τα στοιχεία `` ή `` είναι τύπου `inline`.

Μια άλλη τιμή που μπορεί να πάρει η ιδιότητα `display` είναι η `inline-block`. Τα στοιχεία αυτά συμπεριφέρονται σαν `inline`, αλλά οι ιδιότητες `padding` και `margin` έχουν αποτέλεσμα τη μετατόπιση των γειτονικών στοιχείων και στις τέσσερις πλευρές (Σχήμα 1.4).

1.7.1 Κανονική ροή

Η τοποθέτηση στοιχείων **block** από πάνω προς τα κάτω σε ξεχωριστές γραμμές και στοιχείων **inline** από τα δεξιά προς τα αριστερά το ένα δίπλα στο άλλο, ονομάζεται **normal flow**, κανονική ροή.



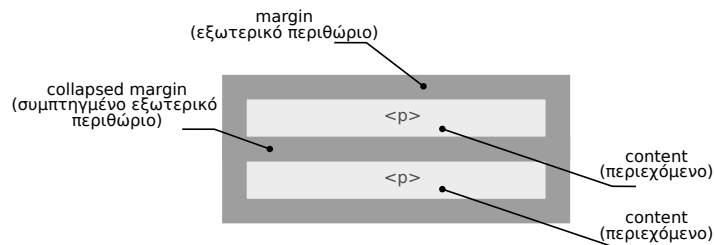
Σχήμα 1.4: Διάταξη των κουτιών `inline` και `block` στην κανονική ροή, για συστήματα γραφής όπως τα δυτικά, όπου γράφουμε από τα αριστερά προς τα δεξιά και από πάνω προς τα κάτω.

Η **κατεύθυνση inline** και **κατεύθυνση block** εξαρτώνται από τη γλώσσα που χρησιμοποιούμε για το περιεχόμενο. Στα Ελληνικά και στις υπόλοιπες δυτικές γλώσσες

1 CSS

γράφουμε από τα αριστερά προς τα δεξιά και από πάνω προς τα κάτω, και αντίστοιχες είναι οι φορές των δύο αυτών κατευθύνσεων. Η κατεύθυνση αυτή αλλάζει με την ιδιότητα `writing-mode`, με την οποία μπορούμε να καθορίσουμε αν το κείμενό μας διατάσσεται οριζόντια, από πάνω προς τα κάτω, ή κάθετα, όπως για παράδειγμα γίνεται σε συστήματα γραφής γλωσσών της Ασίας.

Στην κανονική ροή, γειτονικά εξωτερικά περιθώρια (margin) γειτονικών μπλοκ θα συμπτυχθούν, με κάποιες εξαιρέσεις ([CSS 2.1 Specification](#)). Η συμπεριφορά αυτή είναι γνωστή ως **margin collapse**. Το συμπτυγμένο περιθώριο έχει μέγεθος ίσο με το μεγαλύτερο από τα δύο περιθώρια, ή ίσο με ένα από τα περιθώρια αν τα δύο είναι ίσα. Η σύμπτυξη περιθωρίου συμβαίνει μόνο στην κατεύθυνση block (Σχήμα 1.5). Τα περιθώρια του στοιχείου `:root` δεν συμπτύσσονται.



Σχήμα 1.5: Τα εξωτερικά περιθώρια δύο γειτονικών στοιχείων block συμπτύσσονται.

1.7.2 Διαστάσεις στοιχείων

Για τον προσδιορισμό του μεγέθους που θα έχει ένα στοιχείο έχουν ιδιαίτερη σημασία δύο διαστάσεις, το εξωτερικό του μέγεθος και το εσωτερικό του μέγεθος. Το εξωτερικό μέγεθος ενός στοιχείου καθορίζει το πόσο χώρο χρειάζεται για να προβληθεί το στοιχείο, ενώ το εσωτερικό μέγεθος καθορίζει τις διαστάσεις του περιεχομένου του στοιχείου. Με άλλα λόγια, το εξωτερικό μέγεθος αντιστοιχεί στο εξωτερικό περιθώριο ενώ το εσωτερικό μέγεθος στο περιεχόμενο (Εικόνα 1.3).

Ένα στοιχείο τύπου block παίρνει αυτόματα όλο το πλάτος που του διαθέτει ο άμεσος πρόγονός του, δηλ. το πλάτος του κουτιού που το περιέχει (το κουτί περιεχομένου στην εικόνα 1.3). Αντίθετα, ένα στοιχείο τύπου inline έχει τόσο πλάτος όσο χρειάζεται για να εμφανιστεί το περιεχόμενό του.

Μπορούμε να καθορίσουμε τις εξωτερικές διαστάσεις ενός στοιχείου με τις ιδιότητες `width` και `height`.

1.7.3 Υπερχείλιση

Όταν το περιεχόμενο ενός στοιχείου χρειάζεται περισσότερο χώρο για να προβληθεί από αυτόν που είναι διαθέσιμος, τότε έχουμε υπερχείλιση (overflow). Για τη διαχείριση αυτής της κατάστασης έχουμε διάφορες λύσεις.

Η ιδιότητα `overflow` (και οι αντίστοιχες `overflow-` και `overflow-y`) καθορίζει τί θα γίνεται με το περιεχόμενο που ξεπερνά τα όρια του κουτιού του. Οι τιμές που παίρνει είναι

`overflow: visible | hidden | clip | scroll | auto`

- Η προκαθορισμένη τιμή είναι η `visible`, και σημαίνει πως το περιεχόμενο θα υπερχείλσει και θα ζωγραφιστεί έξω από τα όρια του στοιχείου.
- Αν πάρει την τιμή `hidden`, τότε το περιεχόμενο θα εκταθεί μέχρι και τα όρια του εσωτερικού περιθωρίου. Οτιδήποτε ξεπερνά τα όρια αυτά ψαλιδίζεται (clipping). Ωστόσο το περιεχόμενο σε αυτή την περίπτωση είναι διαθέσιμο για να κύλιση (scroll) μέσω κάποιου σκριπτ.
- Αυτό δεν είναι δυνατό με την τιμή `clip`, με την οποία το περιεχόμενο ψαλιδίζεται ώστε να μην ξεπεράσει τα όρια.
- Με την τιμή `scroll`, το περιεχόμενο ψαλιδίζεται (clipped) μεν, αλλά εμφανίζονται μπάρες κύλισης (scrollbars).

1.8 Διακόσμηση στοιχείων

1.8.1 Επιλογή γραμματοσειράς

Ο ορισμός της γραμματοσειράς ενός στοιχείου γίνεται με την ιδιότητα `font-family`, με την οποία ορίζουμε μία ή περισσότερες γραμματοσειρές για το ίδιο στοιχείο. Η ιδιότητα `font-family` είναι κληρονομούμενη (ενότητα 1.5.4), που σημαίνει πως αν οριστεί για παράδειγμα στο στοιχείο `:root` θα ισχύει για όλα τα υπόλοιπα στοιχεία του DOM. Η ιστοσελίδα μας μπορεί να προβληθεί σε υπολογιστή με πολύ διαφορετικά χαρακτηριστικά από αυτά που περιμένουμε και δεν μπορούμε να ξέρουμε ποιες γραμματοσειρές θα είναι διαθέσιμες στους φυλλομετρητές των τελικών χρηστών. Η CSS μας δίνει διάφορους τρόπους για να αντιμετωπίσουμε αυτό το πρόβλημα και να έχουμε και κάποιο έλεγχο στην τελική εμφάνιση του κειμένου στον φυλλομετρητή.

Καταρχήν, μπορούμε να ορίσουμε περισσότερες από μία γραμματοσειρές, και τότε θα εφαρμοστεί η πρώτη από αυτές που είναι διαθέσιμη στον φυλλομετρητή που προβάλλεται η ιστοσελίδα:


```
/* Αν δεν βρεθεί η Helvetica, θα εφαρμοστεί η Verdana, αλλιώς η προκαθορισμένη sans-serif
   ↳ στο φυλλομετρητή */
```

```
body { font-family: Helvetica, Verdana, sans-serif; }
```

Επιτρέπεται όμως να δηλώσουμε και κάποιον από πέντε γενικούς τύπους γραμματοσειρών, τους serif, sans-serif, monospace, cursive, fantasy. Για αυτούς τους γενικούς τύπους, ο εκάστοτε φυλλομετρητής θα χρησιμοποιήσει τις αντίστοιχες προεπιλεγμένες γραμματοσειρές, που μπορεί να διαφέρουν από υπολογιστή σε υπολογιστή. Γιαυτό το λόγο οι γενικοί τύποι γραμματοσειρών συνίσταται να χρησιμοποιούνται σαν εφεδρική λύση, όπως φαίνεται στο παραπάνω παράδειγμα όπου η sans-serif είναι η ύστατη λύση.

Ακόμη πιο χρήσιμη είναι η δυνατότητα να ορίσουμε γραμματοσειρά που βρίσκεται στο διαδίκτυο, αντί για τον υπολογιστή του τελικού χρήστη. Δημοφιλείς βιβλιοθήκες online γραμματοσειρών είναι η [Google Fonts](#), η [Font Library](#), που προσφέρει γραμματοσειρές κυρίως με άδεια [CC BY-SA](#), η [Font Squirrel](#) κ.ά. Οι υπηρεσίες αυτές παράγουν και τον κώδικα που χρειάζεται για να ενσωματωθούν οι γραμματοσειρές στην CSS μας.

Μερικές χρήσιμες ιδιότητες που μπορούν να χρησιμοποιηθούν σε συνάρτηση με τις γραμματοσειρές είναι:

- **font-size.** Καθορίζει το μέγεθος της γραμματοσειράς. Στην ενότητα [1.6.1](#) περιγράφονται οι σχετικές μονάδες.
- **font-style.** Δυνατές τιμές είναι normal, italic, oblique, για κανονική, πλάγια και πλαγιασμένη γραμματοσειρά. Η τελευταία είναι εξομοιωμένη πλάγια που παράγεται δίνοντας κλίση στην κανονική γραμματοσειρά.
- **font-weight.** Το βάρος της γραμματοσειράς μπορεί να καθοριστεί είτε αριθμητικά, από 100, 200, ... ως 900, είτε σε σχέση με το τρέχον βάρος της γραμματοσειράς ως lighter ή bolder, είτε τέλος με τις λέξεις normal και bold που ισοδυναμούν με 400 και 700 αντίστοιχα.
- **text-decoration.** Προσθέτει μια οριζόντια γραμμή στο κείμενο. Δυνατές τιμές είναι οι none, underline (υπογράμμιση), overline (υπεργράμμιση), line-through (διαγράμμιση) και μάλιστα μπορεί να πάρει και δύο τιμές ταυτόχρονα, π.χ. {**text-decoration:** underline overline;}. Μπορεί να αλλάξει και το στυλ της γραμμής με την text-decoration-style καθώς και το χρώμα της με text-decoration-color.

```
p {
  text-decoration-line: underline;
  text-decoration-style: dotted; /* ή double, dashed, wavy, solid */
  text-decoration-color: blue;
```

1 CSS

```
/* ισοδυναμεί με */  
text-decoration: underline dotted blue;  
}
```

- **text-transform.** Μετασχηματίζει το κείμενο σε όλα κεφαλαία: uppercase, όλα μικρά: lowercase, τα πρώτα γράμματα κάθε λέξης κεφαλαία: capitalize, όλα τα γράμματα με το ίδιο πάχος: full-width.

Πέρα από την γραμματοσειρά, μια σειρά από ιδιότητες επηρεάζουν και άλλα χαρακτηριστικά της εμφάνισης του κειμένου και βοηθούν να έχουμε καλύτερο τυπογραφικό έλεγχο. Από τις πιο σημαντικές ιδιότητες, η line-height, ορίζει το ύψος της γραμμής στην οποία περιέχεται το κείμενο. Η line-height δέχεται τιμές σε οποιαδήποτε από τις διαθέσιμες μονάδες μέτρησης (ενότητα 1.6.1), αλλά και τιμές χωρίς μονάδα μέτρησης. Σε αυτή την περίπτωση το line-height θα γίνει <τιμή>*<μέγεθος γραμματοσειράς>. Αυτός είναι και ο προτιμώμενος τρόπος, καθώς το ύψος της γραμμής θα είναι συνεπές, ακόμη και αν αλλάξει το μέγεθος της γραμματοσειράς του στοιχείου λόγω κληρονομικότητας.

Ενδιαφέρον παρουσιάζουν τέλος και οι ιδιότητες text-align και text-indent. Η πρώτη στοιχίζει το κείμενο μέσα στον κοντέινερ και παίρνει τις τιμές left, right, center και justify. Η δεύτερη ιδιότητα δημιουργεί μια εσοχή στην πρώτη γραμμή της παραγράφου, χρησιμοποιώντας κάποια απόλυτη ή σχετική τιμή (ενότητα 1.6.1).

1.8.2 Διακόσμηση λίστας

Όπως είδαμε στην HTML έχουμε δύο είδη λίστας, την μη ταξινομημένη και την ταξινομημένη, και . Για τις μη ταξινομημένες λίστες, η ιδιότητα list-style-type, μπορεί να πάρει τις τιμές disc (αρχική τιμή), circle, square ή none. Μπορούμε να ορίσουμε και δικά μας σύμβολα, χρησιμοποιώντας το at-rule @counter-style.

Για τα στοιχεία των ταξινομημένων λιστών υπάρχει μια σειρά από διαφορετικές απαριθμήσεις που μπορούν να χρησιμοποιηθούν, όπως decimal (η αρχική τιμή), decimal-leading-zero, lower-roman, upper-roman, lower-greek, κ.ά.

1.8.3 Διακόσμηση υποβάθρου

Το υπόβαθρο κάθε στοιχείου της σελίδας μπορεί να μεταβληθεί με διάφορους τρόπους, π.χ. να καθορίσουμε για το υπόβαθρο ένα χρώμα, ή μια διαβάθμιση (gradient) ή μια εικόνα. Ο καθορισμός του χρώματος γίνεται με την ιδιότητα background-color. Κάθε στοιχείο είναι αρχικά διαφανές. Η ιδιότητα background-color δέχεται οποιαδήποτε

έγκυρη τιμή χρώματος (ενότητα [1.6.2]) και χρωματίζει το υπόβαθρο του κουτιού περιεχομένου και του εσωτερικού περιθωρίου (εικόνα 1.3).

Εναλλακτικά, ή και συμπληρωματικά, πάνω από το χρώμα υποβάθρου, μπορούμε να ορίσουμε και μια εικόνα υποβάθρου, με την ιδιότητα `background-image`. Η ιδιότητα αυτή δέχεται μία ή περισσότερες εικόνες, που ζωγραφίζονται η μία κάτω από την άλλη. Για παράδειγμα ο παρακάτω κανόνας, θα δημιουργήσει στο υπόβαθρο μια στοίβα (stacking context, ενότητα 1.12) που περιέχει την εικόνα και από πάνω της τη διαβάθμιση.

```
div {
  background-image:
    radial-gradient(rgba(255, 0, 0, 0.5), rgba(0, 255, 0, 0.5)),
    url("https://openclipart.org/image/400px/335407");
  width: 30vh;
  height: 30vh;
  border: 10px dotted red;
  padding: 10px;
  background-clip: border-box;
  background-origin: content-box;
}
```

Η εικόνα αυτή θα ψαλιδιστεί στο κουτί του πλαισίου (border), που είναι και η προκαθορισμένη τιμή της `background-clip`. Εναλλακτικά μπορεί να ψαλιδιστεί και στο κουτί του περιεχομένου (content) ή το κουτί του εσωτερικού περιθωρίου (padding). Με την ιδιότητα `background-origin`, που παίρνει αντίστοιχες τιμές, ορίζεται ποια θα είναι η αφετηρία της εικόνας, το πλαίσιο, το εσωτερικό περιθώριο ή το κουτί περιεχομένου.

Πολύ χρήσιμες είναι οι τιμές της ιδιότητας `background-repeat`, που καθορίζουν τί θα γίνει αν οι διαστάσεις του κουτιού είναι μεγαλύτερες από την εικόνα. Η εικόνα μπορεί να επαναλαμβάνεται (repeat), που είναι η προκαθορισμένη συμπεριφορά, να επαναλαμβάνεται μόνο κάθετα ή οριζόντια (repeat-y, repeat-x) ή να μην επαναλαμβάνεται (no-repeat). Με τις τιμές round και space έχουμε ακόμη περισσότερο έλεγχο.

1.9 Αντικαθιστώμενα στοιχεία

Οι εικόνες, αλλά και άλλα μέσα που ενσωματώνονται στην ιστοσελίδα, όπως π.χ. ένα βίντεο ή ένα ενσωματωμένο έγγραφο PDF, είναι στοιχεία που το περιεχόμενό τους δεν μπορεί να επηρεαστεί από τη CSS. Μπορούμε να αλλάξουμε βέβαια τη θέση αυτών των στοιχείων, ή την εμφάνιση του κουτιού που τα περιέχει, αλλά όχι το ίδιο το περιεχόμενο. Τα στοιχεία αυτά ονομάζονται *replaced elements*. Συχνά, οι διαστάσεις των στοιχείων

αυτών είναι *φυσικές*, δηλ. προκύπτουν από το ίδιο το στοιχείο, όπως για παράδειγμα οι διαστάσεις μιας εικόνας.

Οι φυσικές διαστάσεις μιας εικόνας σπάνια είναι κατάλληλες για την τελική προβολή. Το σύνηθες αντίθετα, είναι να θέλουμε να προβάλουμε μια εικόνα έτσι ώστε να χωράει σε συγκεκριμένες διαστάσεις. Μπορούμε να ορίσουμε τις διαστάσεις τις εικόνας σε σχέση με τις διαστάσεις

1.10 Μεταβάσεις

Οι μεταβάσεις (transitions) είναι ο πιο απλός τρόπος που έχουμε για κινούμενα εφέ. Αν και είναι σχετικά απλός μηχανισμός, έχει ωστόσο αρκετό βάθος ώστε να καλύψει πολλές ανάγκες.

Όταν αλλάζουμε την τιμή μιας ιδιότητας, π.χ. όταν αλλάζουμε το χρώμα της γραμματοσειράς με τη ψευδοκλάση `:hover` (ενότητα 1.4.6.1), τότε η μετάβαση από την αρχική στην τελική κατάσταση είναι ακαριαία.

Με την ιδιότητα `transition` μπορούμε να περιγράψουμε πιο ομαλές μεταβάσεις, ή, με άλλα λόγια, να αλλάξουμε τις τιμές τους σταδιακά και όχι απότομα. Για παράδειγμα:

```
<a href="">Ένας σύνδεσμος</a>
```

```
a {
  /* color: blue;
  background-color: white; */
  transition-property: background-color color ;
  transition-duration: 500ms;
  transition-timing-function: ease-in-out;
  transition-delay: 0s;
}

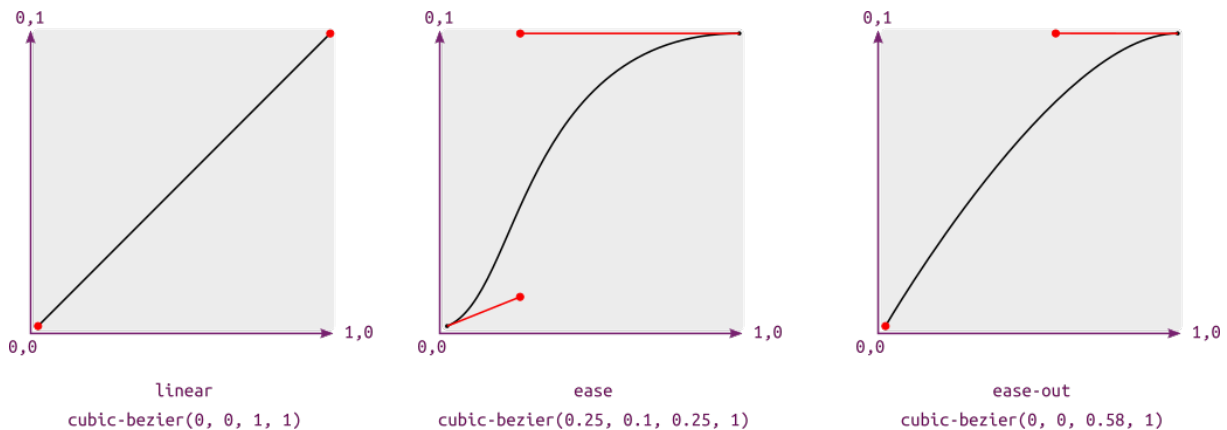
a:hover {
  background-color: navy;
  color: lightyellow;
}
```

Όταν ο δείκτης του ποντικιού αιωρηθεί πάνω από το `<a>` θα σταδιακή μετάβαση από τις αρχικές τιμές των ιδιοτήτων `background-color` και `color` στις τελικές τιμές. Η μετάβαση αυτή θα ξεκινήσει χωρίς καθυστέρηση (`transition-delay`) και θα διαρκέσει 500ms (`transition-duration`). Ο υπολογισμός όλων των ενδιάμεσων καταστάσεων της

1 CSS

μετάβασης θα γίνει από τον φυλλομετρητή χρησιμοποιώντας την συνάρτηση χρονισμού (transition-timing-function).

Η ιδιότητα transition-timing-function μπορεί να πάρει κάποιες έτοιμες συναρτήσεις, όπως ease, ease-in, ease-out, ease-in-out, linear, step-start, step-end. Μπορούμε να καθορίσουμε τη δική μας [συνάρτηση χρονισμού](#) ορίζοντας μια παραμετρική καμπύλη Μπεζιέ με τη συνάρτηση CSS cubic-bezier() ή μια βαθμιδωτή συνάρτηση με την συνάρτηση CSS steps(). Το online εργαλείο [cubic-bezier.com](#) της Λίας Βέρου μας βοηθά αν θέλουμε να κατασκευάσουμε τις δικές μας καμπύλες Μπεζιέ.



Σχήμα 1.6: Παραδείγματα για 3 από τις 5 προκαθορισμένες καμπύλες Μπεζιέ. Οι παράμετροι της συνάρτησης cubic-bezie() είναι οι συντεταγμένες των σημείων ελέγχου.

Αν και δεν μπορούμε να χρησιμοποιήσουμε τη δυνατότητα αυτή σε όλες τις ιδιότητες της CSS, ωστόσο, η [λίστα των διαθέσιμων ιδιοτήτων](#) είναι μεγάλη.

1.11 Η ιδιότητα position

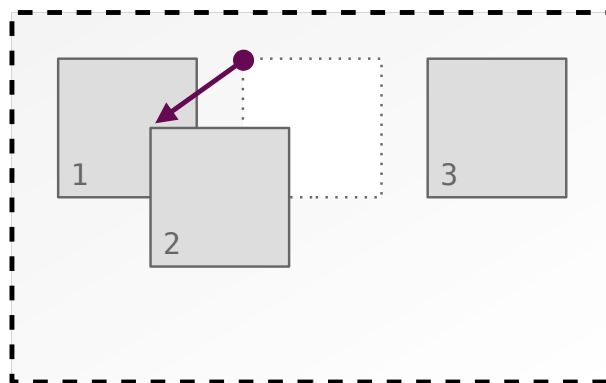
Η ιδιότητα position είναι ένας τρόπος που μας δίνει η CSS από το επίπεδο 2 για να διατάξουμε τα στοιχεία μας έτσι ώστε να τοποθετούνται σε διαφορετική θέση από αυτή στην οποία θα τοποθετούνταν στην κανονική ροή. Με την ιδιότητα position μπορούμε να τοποθετήσουμε στοιχεία σε οποιαδήποτε θέση στη σελίδα. Έχουμε στη διάθεσή μας πέντε τρόπους τοποθέτησης, την στατική, την δυναμική, την απόλυτη, τη σταθερή τοποθέτηση, καθώς και την τοποθέτηση sticky.

1.11.1 Στατική τοποθέτηση (static positioning)

Όπως είδαμε, στην κανονική ροή, η CSS τοποθετεί τα στοιχεία το ένα μετά το άλλο ώστε να μην υπερκαλύπτονται, με αυτό που ονομάζουμε **στατική τοποθέτηση**. Η **στατική τοποθέτηση** δεν είναι τίποτα άλλο παρά η κανονική, συνήθης τοποθέτηση με την οποία τα στοιχεία τοποθετούνται το ένα μετά το άλλο και καταλαμβάνουν το χώρο που χρειάζονται για να προβληθούν, πριν εφαρμόσουμε τις δικές μας οδηγίες CSS. Η **στατική τοποθέτηση** ενεργοποιείται με τη δήλωση `position: static`. Ουσιαστικά σε αυτά τα στοιχεία δεν έχουμε εφαρμόσει κάποια αλλαγή, αφού η `static` είναι η προκαθορισμένη τιμή της ιδιότητας `position`.

1.11.2 Σχετική τοποθέτηση (relative positioning)

Με την ιδιότητα `position: relative` δεσμεύεται ο χώρος που ούτως ή άλλως θα δεσμευόταν για να εμφανιστεί το στοιχείο, αλλά επιπλέον μπορούμε να το μετατοπίσουμε, χωρίς να αλλάξει η θέση των υπόλοιπων στοιχείων. Αυτό είναι σημαντικό: Η θέση των υπόλοιπων στοιχείων δεν αλλάζει και θα μείνουν εκεί που θα ήταν σαν το στοιχείο που μετατοπίζουμε να ήταν `static`.



Σχήμα 1.7: Σχετική τοποθέτηση.

Με τις ιδιότητες `top`, `right`, `bottom`, `left` ορίζουμε μια νέα θέση για το στοιχείο, μετακινώντας το **από την εκάστοτε πλευρά**. Για παράδειγμα η οδηγία

```
#repositioned {
  position: relative;
  bottom: 16px;
  left: 8px;
}
```

1 CSS

θα μετακινήσει το στοιχείο `#repositioned` κατά 16px προς τα πάνω, γιατί θα το σπρώξει από την κάτω πλευρά. Τα επόμενα στοιχεία δεν αλλάζουν θέση και το στοιχείο `#repositioned`, που είναι `relative`, εμφανίζεται πιο κοντά στο χρήστη από ότι το προηγούμενο στοιχείο, δηλ. το στοιχείο `#repositioned` θα υπερακλύπτει το προηγούμενό του στοιχείο.

Αντί για τις ιδιότητες `top`, `right`, `bottom`, `left`, μπορούμε να χρησιμοποιήσουμε και τις `inset-block-start`, `inset-block-end` για μετατόπιση στην κατεύθυνση `block`, ή τις `inset-inline-start`, `inset-inline-end` για μετατόπιση στην κατεύθυνση `inline`. Αυτές οι τέσσερις δηλώσεις αποτελούν και ένα καλό παράδειγμα του τρόπου με τον οποίο εξελίσσεται η CSS. Την ώρα που γράφεται αυτό το κείμενο, οι τέσσερις αυτές ιδιότητες προδιαγράφονται στο CSS Logical Properties and Values Level 1, που είναι όμως σε κατάσταση πρόχειρης έκδοσης (draft). Ωστόσο, η προδιαγραφή έχει ήδη υλοποιηθεί από όλους τους μεγάλους φυλλομετρητές, όπως μπορεί κανείς να δει στο [mdn](#).

Συνοψίζοντας, η ιδιότητα `position` με την τιμή `relative`:

- Μετακινεί το στοιχείο σε σχέση με τη θέση που θα είχε αν δεν επεμβέναμε.
- Δεν επηρεάζεται η θέση κανενός άλλου στοιχείου.
- Η μετακίνηση γίνεται προσδιορίζοντας ποια πλευρά θέλουμε να μετακινήσουμε.
- Ο υπολογισμός γίνεται σε σχέση με την θέση του εξωτερικού περιθωρίου (`margin`).

Αν έχουμε δηλώσει αντιφατικές τιμές, π.χ. `left: 10px; right: 10px`, τότε θα ισχύσει το `left`, γιατί στην κανονική ροή η κατεύθυνση `inline` στο δικό μας σύστημα γραφής είναι από αριστερά προς τα δεξιά. Αντίστοιχα και με τα `top/bottom`, γιατί η κατεύθυνση `block` είναι από πάνω προς τα κάτω.

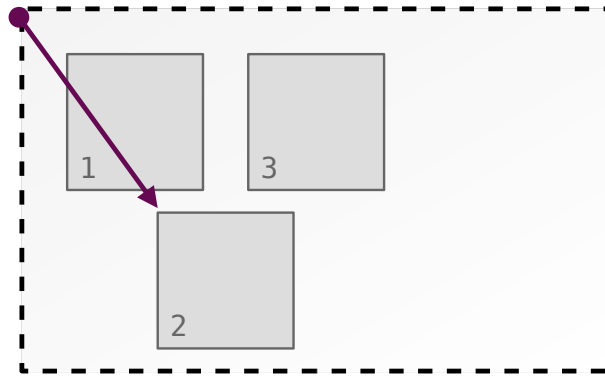
Επιπλέον τα στοιχεία που είναι τοποθετημένα `relative`, δημιουργούν δικό τους `stacking context`, που περιγράφεται στην ενότητα [1.12](#).

1.11.3 Sticky positioning

Το στοιχείο τοποθετείται αρχικά όπως και με την `position: relative`. Η διαφορά είναι πως η θέση προσδιορίζεται με βάση το παράθυρο του `scroll`.

1.11.4 Absolute positioning

Αντίθετα με τη σχετική τοποθέτηση, η δήλωση `position: absolute`, αφαιρεί το στοιχείο από την κανονική ροή τοποθέτησης. Αυτό σημαίνει πως δεν θα δεσμευτεί χώρος για το στοιχείο που τοποθετείται απόλυτα.



Σχήμα 1.8: Απόλυτη τοποθέτηση.

Οι ιδιότητες `top`, `right`, `bottom`, `left` προσδιορίζουν την μετατόπιση της θέσης του στοιχείου σε σχέση με το στοιχείο που το περικλείει, δηλ. σε σχέση με **το πιο κοντινό πρόγονο στοιχείο που δεν έχει `position: static`**. Το στοιχείο τοποθετείται σε σχέση με την εκάστοτε πλευρά του πρόγονου στοιχείου. Για παράδειγμα η οδηγία

```
#absolute-positioned {
  position: absolute;
  bottom: 16px;
}
```

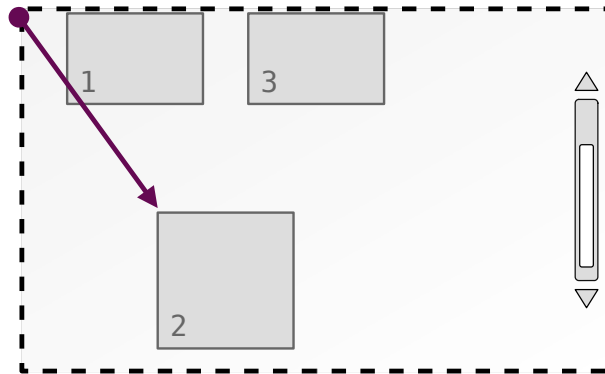
θα μετακινήσει το στοιχείο `#absolute-positioned` κατά 16px προς τα πάνω από την κάτω πλευρά του στοιχείου που το περικλείει.

Συνοψίζοντας, η τιμή `absolute`:

- Μετακινεί το στοιχείο σε σχέση με το πρόγονο μπλοκ.
- Το στοιχείο δεν συμμετέχει στη κανονική ροή, δε δεσμεύεται χώρος για αυτό.
- Το `margin collapse` δεν ισχύει για στοιχεία με `position: absolute`.

1.11.5 Fixed positioning

Η διαφορά της δήλωσης `position: fixed` από την `absolute`, είναι πως η τοποθέτηση του στοιχείου γίνεται σε σχέση με το παράθυρο προβολής (viewport) του φυλλομετρητή, δηλ. την περιοχή που είναι ορατή μέσα στο παράθυρο του φυλλομετρητή.



Σχήμα 1.9: Fixed positioning.

1.11.6 Τοποθέτηση στοιχείων

Συνοψίζοντας, οι τρόποι τοποθέτησης είναι οι εξής:

- Κανονική ροή (normal flow)
 - position: *static*
 - position: *relative*
 - position: *sticky*
- Absolute positioning
 - position: *absolute*
 - position: *fixed*
- Floats (στην ενότητα 1.13)

1.12 Το stacking context και το z-index

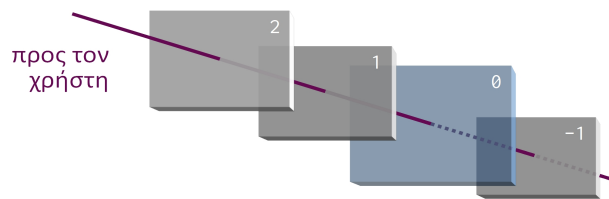
Σύμφωνα με τους κανόνες της κανονικής ροής, ο φυλλομετρητής τοποθετεί τα στοιχεία το ένα μετά το άλλο. Αν όμως κάποια στοιχεία επικαλύπτονται, πρέπει να αποφασιστεί ποιο στοιχείο θα τοποθετηθεί πιο μπροστά δηλ. πιο κοντά στον χρήστη. Επικάλυψη για παράδειγμα μπορεί να συμβεί με την σχετική τοποθέτηση (ενότητα 1.11.2).

Τα στοιχεία που αλληλοεπικαλύπτονται σχηματίζουν μια **στοίβα**, ένα **stack**. Τα στοιχεία αυτής της στοίβας που είναι πιο κοντά μας μπορούν να κρύψουν τα στοιχεία που βρίσκονται πιο πίσω. Το ποιο στοιχείο βρίσκεται πιο μπροστά καθορίζεται από μια σειρά κανόνων, ο πιο βασικός από τους οποίους είναι πως στοιχεία που βρίσκονται πιο βαθιά στο DOM εμφανίζονται πιο μπροστά.

1 CSS

Μια στοίβα στοιχείων, ένα **stacking context**, είναι μια ομάδα στοιχείων που έχουν έναν κοινό πρόγονο. Αυτός ο πρόγονος είναι η ρίζα της στοίβας και τα στοιχεία τοποθετούνται με σειρά μέσα σε αυτή τη στοίβα. Το στοιχείο `<html>` είναι η ρίζα όλων των στοιχείων και συνεπώς σχηματίζει μια στοίβα με τα στοιχεία που περιέχει.

Μπορούμε να χρησιμοποιήσουμε την ιδιότητα `z-index` για να ορίσουμε τη θέση κάποιων στοιχείων στη στοίβα σε σχέση τα υπόλοιπα στοιχεία. Μπορούμε να θεωρήσουμε μια φανταστική γραμμή, έναν νοητό άξονα, τον **άξονα z**, που ξεκινά από τον παρατηρητή μιας σελίδας, κατευθύνεται προς την επιφάνεια του φυλλομετρητή και συνεχίζει προς τα πίσω. Τα στοιχεία που είναι πιο κοντά στον χρήστη εμφανίζονται πιο μπροστά από τα στοιχεία που είναι πιο μακριά από τον χρήστη.



Σχήμα 1.10: Άξονας z.

Ένα στοιχείο που μπορούμε να μετακινήσουμε πάνω στον άξονα z είναι με τη σειρά του ρίζα **για τη δική του στοίβα στοιχείων**. Οπότε, αλλάζοντας την τιμή της `z-index` ενός παιδιού του `<html>`, μετακινούμε προς τα μπροστά ή προς τα πίσω μια ολόκληρη στοίβα στοιχείων, που έχουν τη δική τους διάταξη στο δικό τους άξονα z.

Δεν μπορούμε να αλλάξουμε τη θέση όλων των στοιχείων του DOM με την `z-index`, αλλά μόνο μερικών που **έχουν τοποθετηθεί**. Για ένα στοιχείο λέμε πως **έχει τοποθετηθεί**, όταν έχει αλλάξει η θέση του με την ιδιότητα `position`. Μπορούμε να το δούμε αυτό εξετάζοντας τη σειρά με την οποία ο φυλλομετρητής στοιβάζει τα στοιχεία, από το πιο μπροστά (προς τον χρήστη) στο πιο πίσω (μακριά από τον χρήστη):

- τοποθετημένα στοιχεία για τα οποία έχουμε δηλώσει `z-index >=0`, όπου μεγαλύτερη τιμή σημαίνει πως το στοιχείο βρίσκεται πιο μπροστά,
- στοιχεία `inline` που δεν έχουν τοποθετηθεί,
- στοιχεία `float` που δεν έχουν τοποθετηθεί,
- στοιχεία `block` που δεν έχουν τοποθετηθεί,
- τοποθετημένα στοιχεία με αρνητικό `z-index`,
- το στοιχείο `:root`, που υπάρχει πάντα και περιέχει όλα τα στοιχεία της σελίδας.

Όταν για ένα στοιχείο `<div>` οριστεί μια τιμή για το `z-index`, τότε σχηματίζεται μια **νέα στοίβα στοιχείων** (stacking context) για **τα απόγονα στοιχεία** του `<div>`. Δηλαδή

1 CSS

οι απόγονοι του `<div>` τοποθετούνται σε έναν άξονα z που ανήκει στο `<div>`. Όλη αυτή η στοίβα στοιχείων τοποθετείται με τη σειρά της στον άξονα z της στοίβας στην οποία ανήκει και το `<div>`.

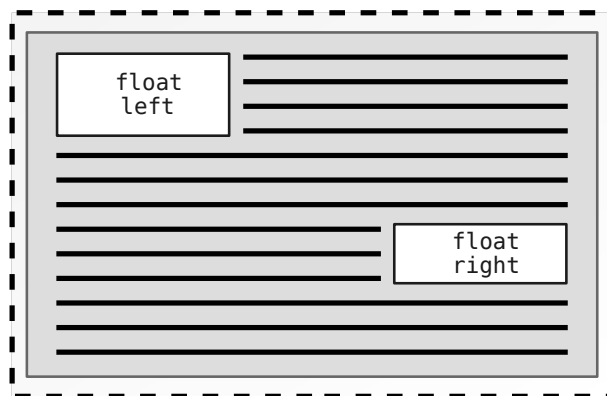
Βέβαια, δεν είναι μόνο το z-index που μπορεί να προκαλέσει το σχηματισμό μιας νέας στοίβας στοιχείων. Οι κανόνες είναι πολλοί, αλλά οι πιο σημαντικοί τρόποι δημιουργίας στοίβας στοιχείων είναι:

- το στοιχείο `:root`
- τοποθετημένο στοιχείο:
 - με absolute ή relative positioning και z-index με αριθμητική τιμή,
 - με fixed ή sticky positioning.

1.13 Float

Με την ιδιότητα float μπορούμε να μετακινήσουμε ένα αντικείμενο ώστε να τοποθετηθεί στην αριστερή ή τη δεξιά της σελίδας, ή καλύτερα, του κοντέινερ που το περιέχει.

Ένα στοιχείο που είναι float θεωρείται πως συμμετέχει στην κανονική ροή, όπως ήταν πριν ενεργοποιηθεί το float. Το περιεχόμενο που ακολουθεί το στοιχείο θα τυλιχτεί γύρω από το στοιχείο αυτό. Αυτό φαίνεται καλύτερα στην εικόνα 1.11.



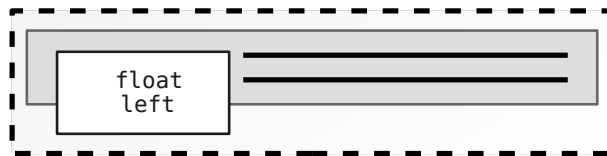
Σχήμα 1.11: Τα δύο λευκά κουτιά έχουν γίνει float.

Η ιδιότητα float μπορεί να έχει τιμή left, right ή none (η προκαθορισμένη τιμή).

Μπορούμε όμως να ζητήσουμε να μην υπάρχει περιεχόμενο στην πλευρά του στοιχείου που είναι float. Δηλ., να μετακινηθεί μεν το float στην άκρη, αλλά το περιεχόμενο που το ακολουθεί να μην ανέβει προς τα πάνω και τυλιχτεί γύρω από το float. Αυτό το

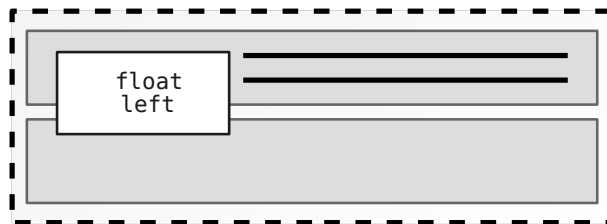
πετυχαίνουμε με την ιδιότητα `clear` η οποία μπορεί να πάρει με τη σειρά της τις τιμές, `left`, `right`, `both` ή **`none`** (η προκαθορισμένη τιμή).

Ενα σημαντικό σημείο είναι πως, ένα στοιχείο που είναι `float`, δεν συνεισφέρει στον υπολογισμό του ύψους του κοντέινερ του. Ο κοντέινερ αυτός παίρνει το ύψος που χρειάζεται για να δείξει το περιεχόμενό του (χωρίς να υπολογίζεται πια το `float`) και το `float` μπορεί εξέχει προς τα κάτω, αν είναι πιο ψηλό από το υπόλοιπο περιεχόμενο (σχήμα ??).



Σχήμα 1.12: Το στοιχείο που είναι `float` (λευκό κουτί), δεν συμμετέχει στον υπολογισμό του ύψους του στοιχείου που το περιέχει.

Μια παρενέργεια είναι το `float` τώρα να ζωγραφίζεται πάνω από το περιεχόμενο που ακολουθεί (σχήμα 1.13).



Σχήμα 1.13: Όταν το ύψος του στοιχείου μέσα στο οποίο περιέχετε το `float` είναι μικρότερο από το ύψος του `float`, το `float` θα επικαλύψει τα επόμενα στοιχεία.

Αυτό το πρόβλημα λύνεται σήμερα εύκολα, κάτι που δεν ίσχυε παλιότερα, με την τιμή `flow-root` στην ιδιότητα `display`. Ορίζοντας λοιπόν `display: flow-root` στον κοντέινερ που περιέχει το `float`, η διάταξη των παιδιών του γίνεται πλέον σε σχέση με αυτόν τον κοντέινερ και συνεπώς ο κοντέινερ επεκτείνεται για να περικλείσει όλα του τα παιδιά, ακόμη και αυτά που είναι `float`. Λέμε λοιπόν πως το στοιχείο με τη δήλωση `display: flow-root` δημιουργεί ένα *block formatting context*, μια περιοχή διάταξης κανονικής ροής. Χωρίς αυτή τη δήλωση, το ρόλο του *block formatting context* τον παίζει το στοιχείο `:root`.

Επίσης νέο *block formatting context* δημιουργούν όσα στοιχεία είναι `float`. Αυτό σημαίνει πως τα παιδιά ενός `float` θα διαταχθούν σε σχέση με το `float`.

Βιβλιογραφία

- contributors, MDN. 2021. 'Font-size - CSS: Cascading style sheets: MDN'. *CSS: Cascading Style Sheets / MDN*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/CSS/font-size#formal_definition.
- Çelik, Tantek, Chris Lilley, L. David Baron, και W3C. 2018. 'CSS Color Module Level 3'. [www.w3.org. https://www.w3.org/TR/css-color-3/#svg-color](https://www.w3.org/TR/css-color-3/#svg-color).
- Etemad, Erika. 2020. 'CSS Box Model Module Level 3'. Candidate Recommendation. W3C.
- Etemad, Erika, David Baron, Rossen Atanassov, και Tab Atkins Jr. 2018. 'CSS Flexible Box Layout Module Level 1'. Candidate Recommendation. W3C.
- Etemad, Erika, και Tab Atkins Jr. 2019. 'CSS Values and Units Module Level 3'. Candidate Recommendation. W3C.
- , 2021. 'CSS Cascading and Inheritance Level 3'. {W3C} Recommendation. W3C.
- Etemad, Erika, Tab Atkins Jr., και Florian Rivoal. 2021. 'CSS Snapshot 2021'. W3C.
- Lilley, Chris, Myles Maxfield, και John Daggett. 2018. 'CSS Fonts Module Level 3 - Basic Font Properties'. {W3C} Recommendation. W3C.