Σχεδιάζοντας ιστοσελίδες με πλέγμα (Grid)

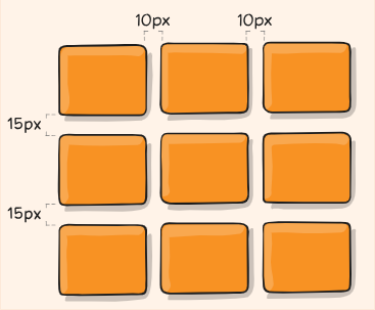
Το **μοντέλο πλέγματος** (**CSS Grid Layout**), στο σχεδιασμό της διάταξης ιστοσελίδων, είναι ένα δισδιάστατο σύστημα που βασίζεται στη λογική του πλέγματος. Η χρησιμοποίηση ενός μοντέλου διάταξης με πλέγμα (**grid**), καθιστά ευκολότερο το σχεδιασμό ιστοσελίδων αποφεύγοντας τη χρήση πλεύσης (float) και τοποθέτησης (positioning) των στοιχείων.

Ως πρώτο βήμα, δημιουργούμε ένα στοιχείο **περιέκτη** (**container**), με ένα στοιχείο block όπως π.χ. το **div** και εσωκλείουμε στις ετικέτες του ένα ή περισσότερα στοιχεία (παιδιά). Το στοιχείο αυτό γίνεται **περιέκτης πλέγματος** (**grid container**) όταν δίνουμε στην **ιδιότητα display** την **τιμή grid** ή **inline-grid**. Αυτόματα όλα τα παιδιά του γίνονται **στοιχεία πλέγματος** (**grid items**).

Όταν φτιάχνουμε ένα πλέγμα τα στοιχεία πλέγματος στοιχίζονται σε κατακόρυφες γραμμές που καλούνται **στήλες** (**columns**) και οριζόντιες γραμμές που καλούνται **σειρές** (**rows**). Καθορίζουμε το πλήθος των στηλών του πλέγματος, δίνοντας στην ιδιότητα **grid-template-columns** τόσες τιμές, που καθορίζουν το εύρος της κάθε στήλης, όσες οι στήλες που επιθυμούμε να έχει το πλέγμα (για παράδειγμα, αν θέλουμε το πλέγμα να έχει 3 στήλες, θα δώσουμε 3 τιμές μέτρησης μεγέθους (px, em, %) π.χ. grid-template-columns: 10% 20px 2em;, καθορίζοντας έτσι και το πλάτος κάθε στήλης. Αν βάλουμε grid-template-columns: auto auto auto auto; θα καθορίσουμε 4 στήλες με το ίδιο πλάτος). Παρομοίως, καθορίζουμε το πλήθος των σειρών του πλέγματος, δίνοντας στην ιδιότητα **grid-template-rows** τόσες τιμές, που καθορίζουν το ύψος της κάθε σειράς, όσες οι σειρές που επιθυμούμε (για παράδειγμα, αν θέλουμε το πλέγμα να έχει 2 σειρές θα δώσουμε 2 τιμές π.χ. grid-template-rows: 40px 3em;, καθορίζοντας έτσι και το ύψος κάθε σειράς. Αν βάλουμε grid-template-rows: auto auto; θα καθορίσουμε 2 σειρές με το ίδιο ύψος).

Μεταξύ των στηλών και μεταξύ των γραμμών δημιουργούνται κάποια κενά διαστήματα (**gap**, **row-gap**, **column-gap**). Δημιουργούνται με αυτό τον τρόπο κελιά που περιέχουν διαδοχικά τα στοιχεία πλέγματος (grid items). Μπορούμε να καθορίσουμε τις διαστάσεις των ενδιάμεσων κενών δίνοντας στις αντίστοιχες ιδιότητες τις σχετικές τιμές (π.χ. **gap**: 10px; **gap**: 10px 15px; **row-gap**: 20px; **column-gap**: 10px;). Η ιδιότητα **gap** είναι σύντμηση των ιδιοτήτων **row-gap** και **column-gap**.

```
.container {
  grid-template-columns: 100px 50px 100px;
  grid-template-rows: 80px auto 80px;
  column-gap: 10px;
  row-gap: 15px;
}
```



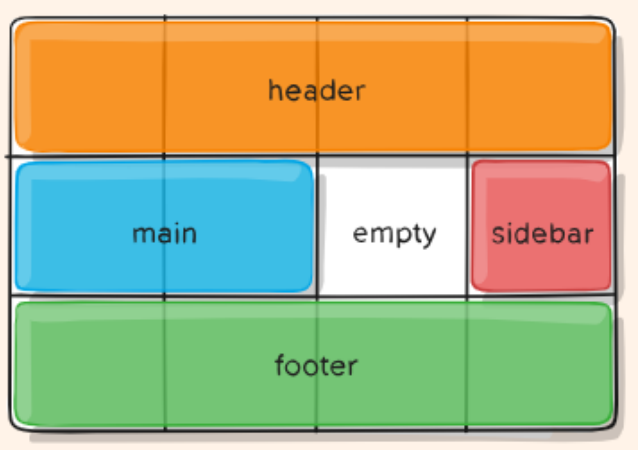
Ένας περιέκτης πλέγματος έχει προκαθορισμένα ένα στοιχείο πλέγματος σε κάθε κελί του. Η θέση στο πλέγμα που θα τοποθετηθεί ένα στοιχείο πλέγματος, σε ότι αφορά στη στήλη, καθορίζεται από τις τιμές που θα δώσουμε στις ιδιότητες **grid-column-start** και **grid-column-end** (οι οποίες δηλώνουν την αριστερή και δεξιά γραμμή που περιβάλλουν το κελί του) ή στη σύντμηση τους **grid-column**. Παρομοίως, η θέση στο πλέγμα που θα τοποθετηθεί ένα στοιχείο πλέγματος, σε ότι αφορά στη σειρά, καθορίζεται από τις

τιμές που θα δώσουμε στις ιδιότητες **grid-row-start** και **grid-row-end** (οι οποίες δηλώνουν την πάνω και κάτω γραμμή που περιβάλλουν το κελί του) ή στη σύντμηση τους **grid-row**. Έχουμε επίσης τη δυνατότητα να τοποθετήσουμε ένα στοιχείο πλέγματος σε περισσότερες από μία στήλες ή/και σειρές. Αν θέλουμε ένα στοιχείο πλέγματος (item1) να καταλαμβάνει περισσότερες από μία στήλες, μπορούμε να το πετύχουμε ορίζοντας στις ιδιότητες grid-column-start και grid-column-end ως τιμές την αριστερή και δεξιά γραμμή που θέλουμε να το περιβάλλουν, δηλαδή που ξεκινάει και που τελειώνει αντίστοιχα (π.χ. grid-column-start: 2; grid-column-end: 5; Εναλλακτικά grid-column: 2 / 5; Θα ξεκινάει από την γραμμή στήλης 2 και τελειώνει στην γραμμή στήλης 5). Ομοίως, αν θέλουμε ένα στοιχείο πλέγματος (item2) να καταλαμβάνει περισσότερες από μία σειρές, μπορούμε να το πετύχουμε ορίζοντας στις ιδιότητες grid-row-start και grid-row-end ως τιμές την πάνω και κάτω γραμμή που θέλουμε να το περιβάλλουν τις γραμμές σειρών, δηλαδή που ξεκινάει και που τελειώνει αντίστοιχα (π.χ. grid-row-start: 1; grid- row-end: 6; ; Εναλλακτικά grid- row: 1 / 6; Θα ξεκινάει από την γραμμή 1 και θα τελειώνει στην γραμμή 6). Μπορούμε να ορίσουμε ταυτόχρονα για ένα στοιχείο πλέγματος (item3) μια περιοχή με περισσότερα από ένα συνεχόμενα κελιά και σε στήλες και σε γραμμές, με τη σύντμηση των παραπάνω ιδιοτήτων που είναι η ιδιότητα grid-area (π.χ. grid-area: 2 / 3 / 4 / 6; Αρχίζει από τη γραμμή σειράς 2 και από τη γραμμή στήλης 3 και τελειώνει στη γραμμή σειράς 4 και στη γραμμή στήλης 6).

<table>
<tr><td rowspan="6">item2</td><td colspan="4">item1</td><td></td><td></td></tr>
<tr><td></td><td colspan="3" rowspan="2">item3</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Με την ιδιότητα **grid-area** μπορούμε επίσης να δώσουμε και ένα όνομα σε ένα στοιχείο πλέγματος (π.χ. grid-area: menu;). Το όνομα αυτό το αναφέρουμε επίσης στην ιδιότητα **grid-template-areas** του περιέκτη, η σύνταξη της οποίας παρέχει μια εικόνα του πλέγματος. Η τελεία, ως τιμή δηλώνει ένα άδειο κελί πλέγματος. Η επανάληψη του ονόματος δείχνει ότι επεκτείνεται το στοιχείο σε περισσότερα κελιά, τόσα όσες και οι επαναλήψεις του ονόματος. Συνεχόμενες τελείες χωρίς κενό ανάμεσα τους δηλώνουν συνεχόμενα κενά κελιά. Στο παρακάτω παράδειγμα βλέπουμε πως χρησιμοποιούμε αυτές τις ιδιότητες:

```
.item-1 {grid-area: header;}
.item-2 {grid-area: main;}
.item-3 {grid-area: sidebar;}
.item-4 {grid-area: footer;}
.container {
  display: grid;
  grid-template-columns: 50px 50px 50px 50px;
  grid-template-rows: auto;
  grid-template-areas:
    "header header header header"
    "main main . sidebar"
    "footer footer footer footer";
}
```



Όπως βλέπουμε παραπάνω, έχουμε δώσει όνομα στα **grid items** (.item-1 item-2, item-3, item-4) στην ιδιότητα **grid-area** εκάστου (header, main, sidebar και footer αντίστοιχα) και στη συνέχεια αναφέρουμε αυτά τα ονόματα στην ιδιότητα **grid-template-areas** κάθε γραμμή σε εισαγωγικά. Θα δημιουργηθεί ένα πλέγμα με 3 σειρές και 4 στήλες. Η πρώτη γραμμή θα έχει το στοιχείο item-1 με όνομα header που θα καταλαμβάνει 4 κελιά, γιατί έχουμε γράψει 4 φορές το όνομα header. Η δεύτερη γραμμή θα έχει το στοιχείο item-2 με όνομα main που θα καταλαμβάνει 2 κελιά, γιατί έχουμε γράψει 2 φορές το όνομα main, μετά ένα κενό κελί γιατί έχουμε δώσει τελεία και το στοιχείο item-3 με όνομα sidebar που θα καταλαμβάνει 1 κελί, γιατί έχουμε γράψει 1 φορά το όνομα sidebar. Η τρίτη γραμμή θα έχει το στοιχείο item4 με όνομα footer που θα καταλαμβάνει 4 κελιά, γιατί έχουμε γράψει 4 φορές το όνομα footer.

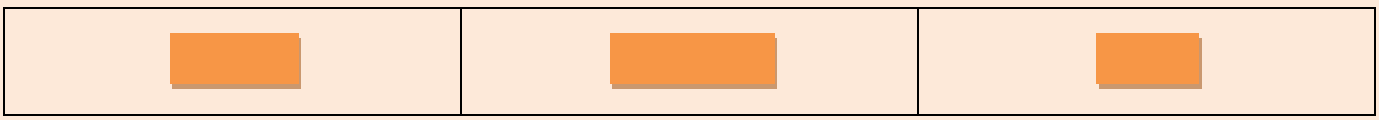
Η οριζόντια στοίχιση των στοιχείων ενός πλέγματος (σε κάθε σειρά) πραγματοποιείται με τη χρήση της ιδιότητας **justify-items**, ενώ η κατακόρυφη (σε κάθε στήλη) με τη χρήση της ιδιότητας **align-items**. Οι δυνατές τιμές που λαμβάνει η ιδιότητα **justify-items** και η περιγραφή τους παρατίθενται στον πίνακα που ακολουθεί:

Τιμή	Περιγραφή
start	Τα στοιχεία στοιχίζονται στην αριστερή (αρχική) πλευρά του κελιού τους.
end	Τα στοιχεία στοιχίζονται στην δεξιά (τελική) πλευρά του κελιού τους.
center	Τα στοιχεία τοποθετούνται οριζόντια στο κέντρο του κελιού τους .
stretch	Τα στοιχεία εκτείνονται και καλύπτουν ολόκληρο το εύρος του κελιού τους . **Προκαθορισμένη τιμή**.

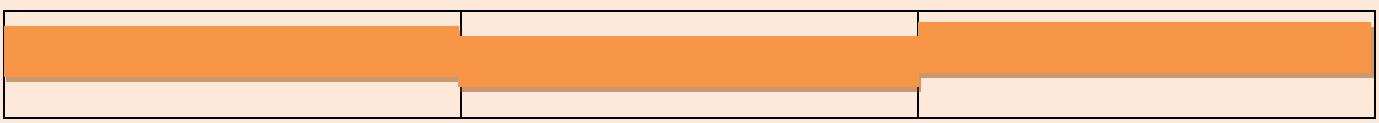
```
.container {justify-items: start;}
```

```
.container {justify-items: end;}
```

```
.container {justify-items: center;}
```



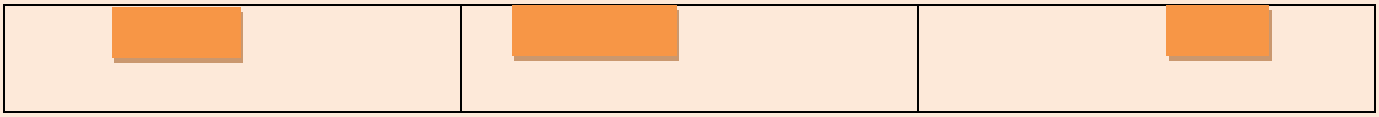
```
.container {justify-items: stretch;}
```



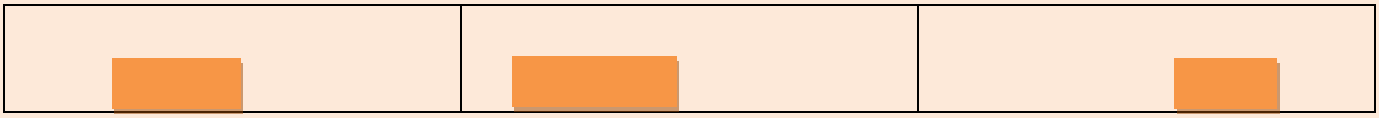
Η κατακόρυφη στοίχιση των στοιχείων ενός πλέγματος (σε κάθε στήλη) πραγματοποιείται με τη χρήση της ιδιότητας **align-items**. Οι δυνατές τιμές που λαμβάνει η ιδιότητα **align-items** και η περιγραφή τους παρατίθενται στον πίνακα που ακολουθεί:

Τιμή	Περιγραφή
start	Τα στοιχεία στοιχίζονται στην πάνω (αρχική) πλευρά του κελιού τους.
end	Τα στοιχεία στοιχίζονται στην κάτω (τελική) πλευρά του κελιού τους.
center	Τα στοιχεία τοποθετούνται κατακόρυφα στο κέντρο του κελιού τους .
stretch	Τα στοιχεία εκτείνονται και καλύπτουν κατακόρυφα ολόκληρο το εύρος του κελιού τους . **Προκαθορισμένη τιμή**.
baseline	Τα στοιχεία στοιχίζονται κατά μήκος της baseline.

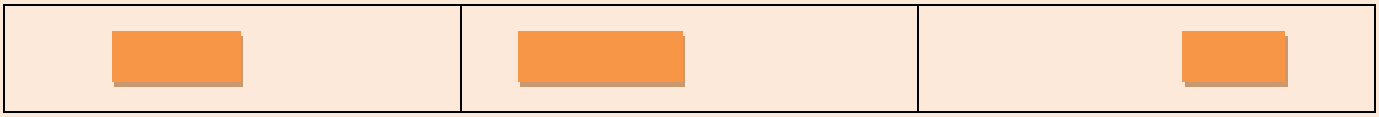
```
.container { align-items: start;}
```



```
.container { align-items: end;}
```

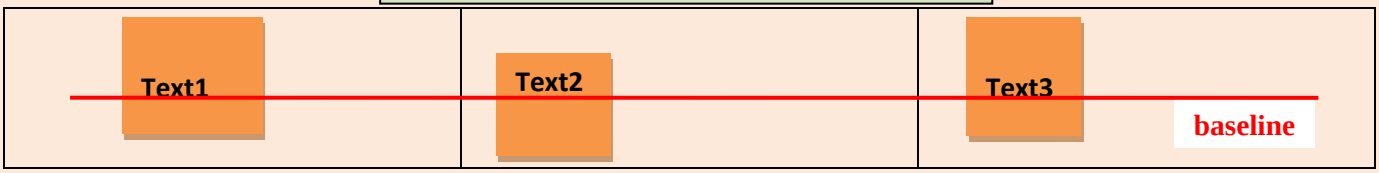


```
.container { align-items: center;}
```



```
.container { align-items: stretch;}
```

```
.container { align-items: baseline;}
```



Η ιδιότητα **place-items** είναι μια σύντμηση των ιδιοτήτων **justify-items** και **align–items**. Η σύνταξη της είναι η ακόλουθη:

place-items: τιμή για **align–items** / τιμή για **justify-items**

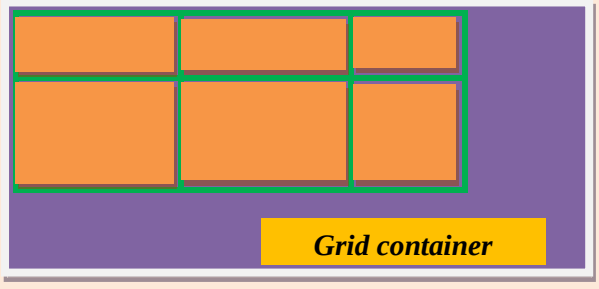
Αν παραληφθεί η δεύτερη τιμή, θεωρείται η ίδια με την πρώτη (π.χ. **place-items: center;** είναι ισοδύναμο με: **align-items: center;** και **justify-items: center;**).

Σε μερικές περιπτώσεις όπου τα grid items έχουν σταθερά μεγέθη (δηλαδή ορισμένα με ευέλικτες μονάδες μέτρησης όπως τα px), το συνολικό μήκος του πλέγματος μπορεί να είναι μικρότερο του αντίστοιχου μήκους του περιέκτη. Σε αυτές τις περιπτώσεις μπορούμε να προσδιορίσουμε τη στοίχιση του πλέγματος μέσα στον περιέκτη του με τις ιδιότητες **justify-content** (οριζόντια στοίχιση) και **align-content** (κατακόρυφη στοίχιση).

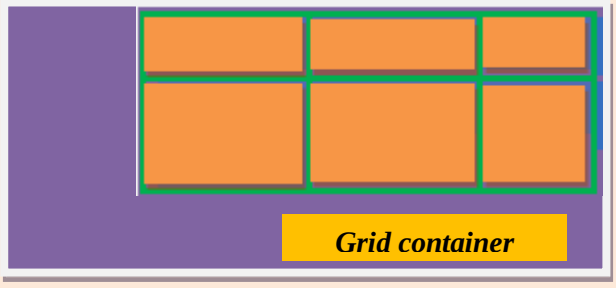
Οι δυνατές τιμές που λαμβάνει η ιδιότητα **justify-content** και η περιγραφή τους παρατίθενται στον πίνακα που ακολουθεί:

Τιμή	Περιγραφή
start	Τα στοιχεία στοιχίζονται στην αριστερή (αρχική) πλευρά του περιέκτη.
end	Τα στοιχεία στοιχίζονται στην δεξιά (τελική) πλευρά του περιέκτη .
center	Τα στοιχεία τοποθετούνται οριζόντια στο κέντρο του περιέκτη.
stretch	Τα στοιχεία εκτείνονται και καλύπτουν ολόκληρο το εύρος του περιέκτη.
space-between	Το διαθέσιμο κενό μοιράζεται εξίσου ανάμεσα στα στοιχεία, χωρίς κενό με τις πλευρές του περιέκτη.
space-around	Τα στοιχεία στοιχίζονται στον περιέκτη με το διαθέσιμο κενό να μοιράζεται εξίσου ανάμεσα στα στοιχεία. Το κενό ανάμεσα σε δύο στοιχεία είναι διπλάσιο από το κενό μεταξύ του πρώτου ή του τελευταίου στοιχείου με τις πλευρές του περιέκτη (πλευρικά κενά), γιατί περιέχει δύο κενά, ένα από το αριστερό και ένα από το δεξιό στοιχείο.
space-evenly	Η διαφορά με την παραπάνω τιμή είναι ότι όλα τα κενά και τα δύο πλευρικά κενά είναι ίσα με τα κενά μεταξύ των στοιχείων.

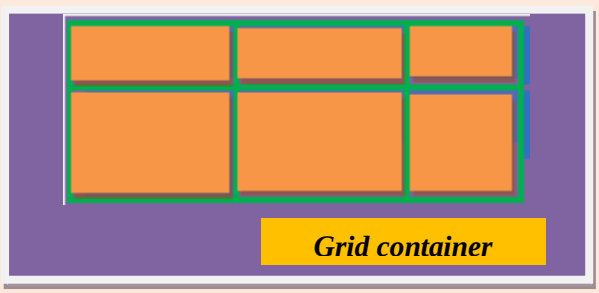
```
.container {justify- content: start;}
```



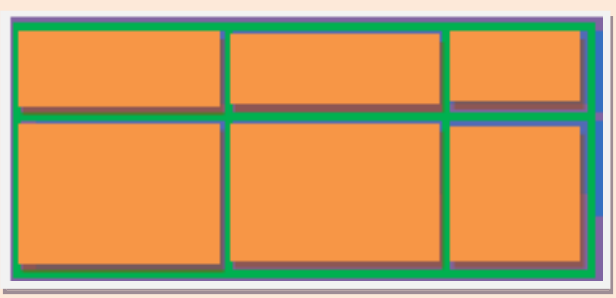

```
.container {justify- content: end;}
```



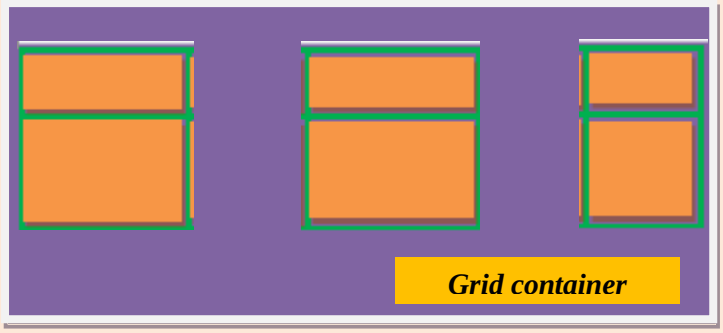

```
.container {justify- content: center;}
```



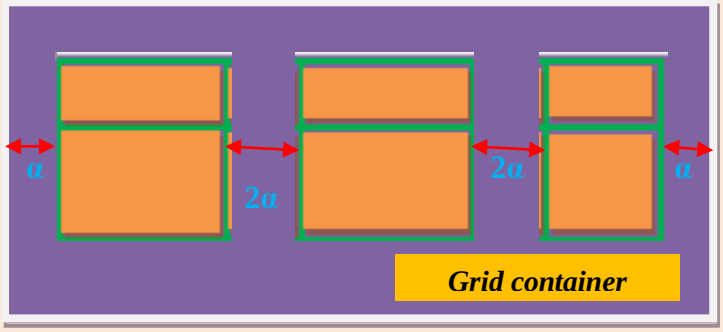

```
.container {justify- content: stretch;}
```



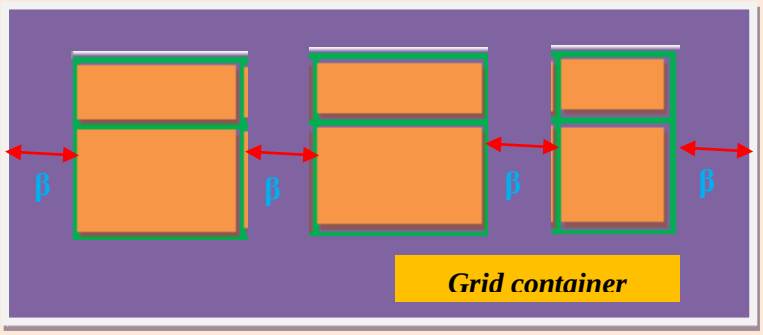
```
.container {justify- content: space-between;}
```




```
.container {justify- content: space-around;}
```




```
.container {justify- content: space-evenly;}
```

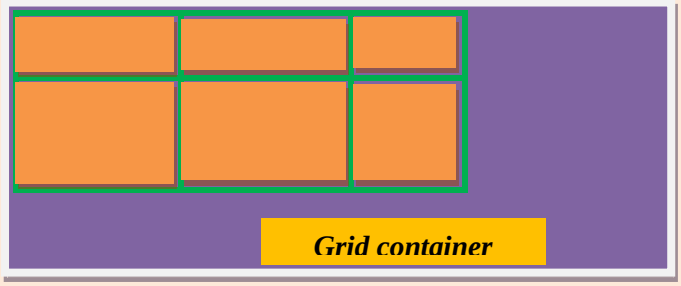


Οι δυνατές τιμές που λαμβάνει η ιδιότητα **align-content** και η περιγραφή τους παρατίθενται στον πίνακα που ακολουθεί:

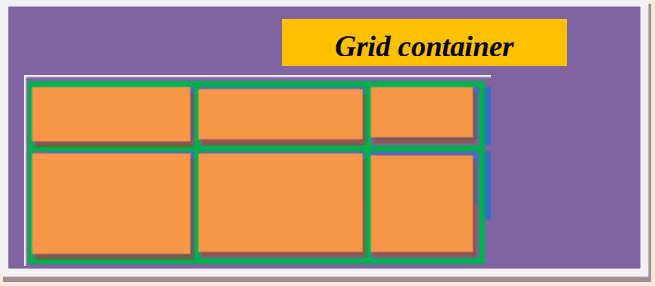
Τιμή	Περιγραφή
start	Τα στοιχεία στοιχίζονται στην πάνω (αρχική) πλευρά του περιέκτη.
end	Τα στοιχεία στοιχίζονται στην κάτω (τελική) πλευρά του περιέκτη .
center	Τα στοιχεία τοποθετούνται κατακόρυφα στο κέντρο του περιέκτη.
stretch	Τα στοιχεία εκτείνονται και καλύπτουν ολόκληρο το εύρος του περιέκτη.
space-between	Το διαθέσιμο κενό μοιράζεται εξίσου ανάμεσα στα στοιχεία, χωρίς κενό με τις πλευρές του

	περιέκτη.
space-around	Τα στοιχεία στοιχίζονται στον περιέκτη με το διαθέσιμο κενό να μοιράζεται εξίσου ανάμεσα στα στοιχεία. Το κενό ανάμεσα σε δύο στοιχεία είναι διπλάσιο από το κενό μεταξύ του πρώτου ή του τελευταίου στοιχείου με τις πλευρές του περιέκτη (πλευρικά κενά), γιατί περιέχει δύο κενά, ένα από το αριστερό και ένα από το δεξιό στοιχείο.
space-evenly	Η διαφορά με την παραπάνω τιμή είναι ότι όλα τα κενά και τα δύο πλευρικά κενά είναι ίσα με τα κενά μεταξύ των στοιχείων.

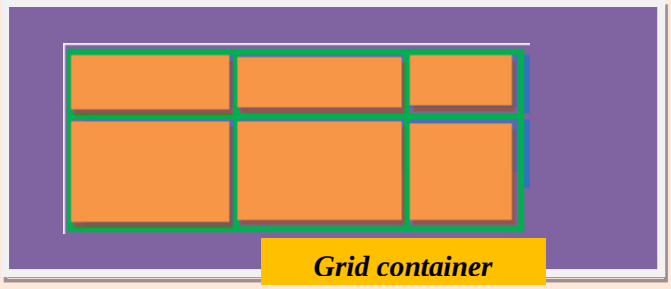
```
.container { align -content: start;}
```



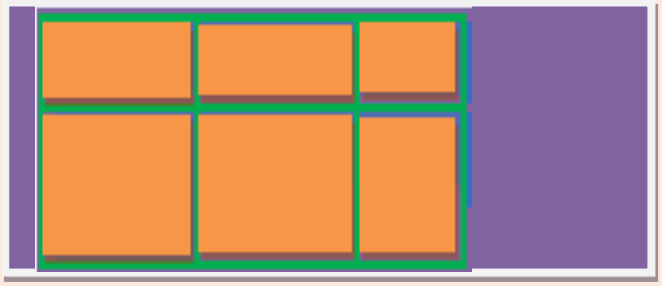

```
.container { align-content: end;}
```



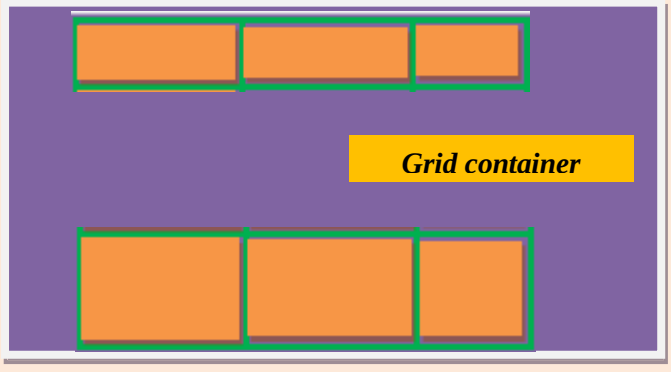

```
.container { align -content: center;}
```



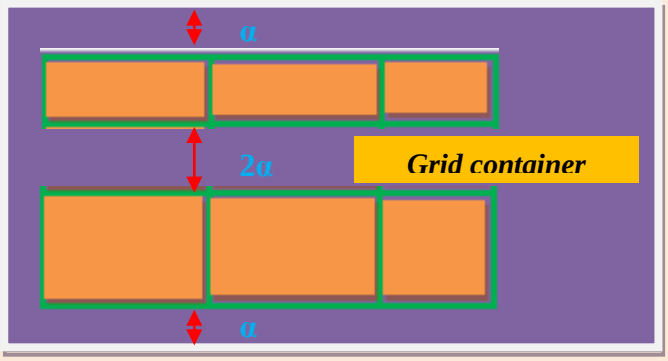

```
.container { align -content: stretch;}
```



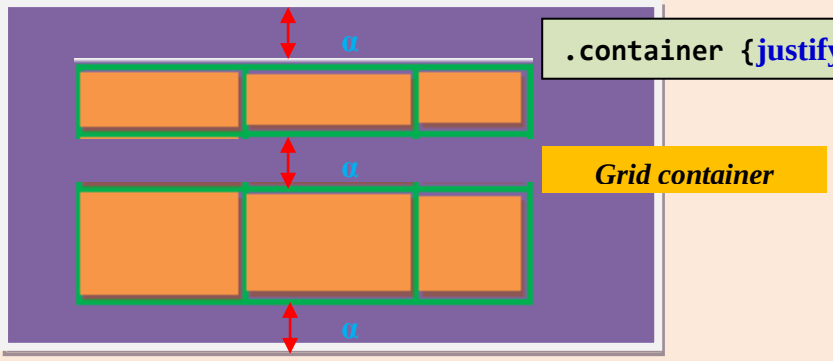
```
.container {justify-content: space-between;}
```




```
.container {justify-content: space-around;}
```




```
.container {justify-content: space-evenly;}
```



Η ιδιότητα **place-content** είναι μια σύντμηση των ιδιοτήτων **justify-content** και **align–content**. Η σύνταξη της είναι η ακόλουθη:

place-content: τιμή για **align–content** / τιμή για **justify-content**

Αν παραληφθεί η δεύτερη τιμή, θεωρείται η ίδια με την πρώτη (π.χ. **place-content: center;** είναι ισοδύναμο με: **align-content: center;** και **justify-content: center;**).

Αν θέλουμε να στοιχίσουμε ένα grid item, ξεχωριστά από τα υπόλοιπα στοιχεία του πλέγματος, χρησιμοποιούμε την ιδιότητα **justify-self** για την οριζόντια στοίχιση τους και την ιδιότητα **align-self** για την κατακόρυφη στοίχιση τους. Οι δυνατές τιμές που λαμβάνουν είναι οι γνωστές: start, end, center και stretch με την περιγραφή που έχουν και παραπάνω.

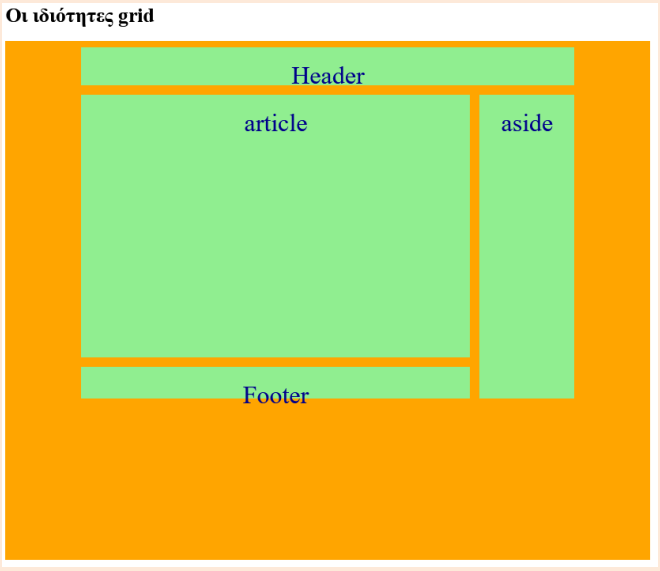
```
<!DOCTYPE html>
<html>
<head>
<style>
  .item1 { grid-area: header; grid-column: 1 / 4;}
  .item2 { grid-area: article; grid-row: 2 / 4;}
  .item3 { grid-area: aside; grid-row: 2 / 5; }
  .item4 { grid-area: footer; grid-row: 4 / 5;}
.grid-container {
  display: grid;
  width: 1000px;
  height: 800px;
  grid-template-columns: 300px 300px 150px;
  grid-template-rows: 60px 200px 200px 50px;
  grid-template-areas:
    'header header header  '
    'article article aside'
    'footer footer aside';
  gap: 15px;
  justify-content: center;
  align-items: normal;
  background-color: orange;
  padding: 10px;
}
.grid-container > div {
  background-color: lightgreen;
  text-align: center;
  padding: 20px;
  font-size: 40px;
  color: darkblue;
}
</style>
</head>
<body>
<h1>Οι ιδιότητες grid</h1>
<div class="grid-container">
  <div class="item1">Header</div>
  <div class="item2">article</div>
  <div class="item3">aside</div>
  <div class="item4">Footer</div>
</div>
</body>
</html>
```

Η ιδιότητα **place-self** είναι μια σύντμηση των ιδιοτήτων **justify-self** και **align–self**. Η σύνταξη της είναι η ακόλουθη:

place-self: τιμή για **align–self** / τιμή για **justify-self**

Αν παραληφθεί η δεύτερη τιμή, θεωρείται η ίδια με την πρώτη (π.χ. **place-self: center;** είναι ισοδύναμο με: **align-self: center;** και **justify-self: center;**).

Παραδείγματα πλέγματος αναφέρονται στη συνέχεια.



Ειδικές μονάδες μέτρησης μεγεθών

Σε ένα πλέγμα μπορούμε να χρησιμοποιήσουμε και ορισμένες ειδικές μονάδες μέτρησης μεγέθους των στηλών και των γραμμών του, με λέξεις κλειδιά, οι οποίες είναι οι ακόλουθες:

- **min-content**, καθορίζεται ως η διάσταση του μικρότερου δυνατού μεγέθους για να χωρέσει το κείμενο χωρίς να κόβονται οι λέξεις. Όταν το κείμενο έχει πολλές λέξεις είναι το μέγεθος της μεγαλύτερης λέξης.
- **max-content**, καθορίζεται ως η μικρότερη διάσταση μεγέθους για να χωρέσει ολόκληρο το κείμενο σε μια σειρά. Αν το κείμενο ξεπερνάει τη διάσταση της γραμμής έχουμε υπερχείλιση και δεν φαίνεται το τμήμα που περισσεύει.
- **fit-content**, καθορίζεται το εύρος αυτόματα ανάμεσα σε min-content και max-content. Το container {width: fit-content;}είναι σύντμηση του: container {width: auto; min-width: min-content; max-width: max-content;}
- **fractional units** (**fr**), μοιράζουν αναλογικά τον διαθέσιμο χώρο.
- **auto**, λειτουργεί όπως το **fr** μειονεκτώντας όπου υπάρχει σύγκρουση.

Οι fr units είναι σαν τα ποσοστά. Το άθροισμα τους αντιστοιχεί στο 100% (δηλαδή τον διαθέσιμο χώρο) και κάθε τιμή αντιστοιχεί αναλογικά σε σχέση με τις υπόλοιπες. Π.χ. αν έχουμε:

```
.container {width: 1000px; display: grid; grid-template-columns: 400px 20% 1fr 3fr;}
```

ο περιέκτης θα έχει συνολικό πλάτος 1000px, η πρώτη στήλη θα είναι 400px, η δεύτερη στήλη θα είναι 200px (20%*1000px=200px). Περισσεύουν 1000px-400px-200px=400px για την τρίτη και την τέταρτη στήλη. Για να βρούμε το μέγεθος κάθε μιας, προσθέτουμε τις τιμές τους (1+3=4). Διαιρούμε το εύρος του περιέκτη που περισσεύει, δηλαδή τα 400px με το 4 (400:4=100) και το αποτέλεσμα το πολλαπλασιάζουμε με κάθε μία τιμή. Άρα η τρίτη στήλη θα είναι 1*100px=100px και η τέταρτη στήλη θα είναι 3*100px=300px.

Ειδικές συναρτήσεις

Σε ένα πλέγμα μπορούμε να χρησιμοποιήσουμε και ορισμένες συναρτήσεις για τον καθορισμό του μεγέθους των στηλών και των γραμμών του, οι οποίες είναι οι ακόλουθες:

- **min ( )**, λαμβάνει την μικρότερη τιμή από ένα σύνολο τιμών που χωρίζονται με κόμμα
- **max ( )**, λαμβάνει την μεγαλύτερη τιμή από ένα σύνολο τιμών που χωρίζονται με κόμμα
- **minmax (ελάχιστη τιμή, μέγιστη τιμή)**, καθορίζει ένα μέγεθος ανάμεσα σε ένα ελάχιστο και ένα μέγιστο μέγεθος. Δηλαδή, αν αυξομειώνεται το μέγεθος δεν θα γίνει ποτέ μικρότερο από την ελάχιστη τιμή και δεν θα γίνει μεγαλύτερο από τη μέγιστη τιμή
- **fit-content ( )**, καθορίζει τον διαθέσιμο χώρο ως την μικρότερη τιμή μεταξύ του max-content και της μεγαλύτερης τιμής από το min-content και το όρισμα που βάζουμε στην συνάρτηση. Παράδειγμα: fit-content (100px); Είναι ισοδύναμο με **min (max-content, max (min-content, 100px))**. Το όρισμα δεν μπορεί να είναι fr.
- **repeat (πλήθος, μέγεθος)**, χρησιμοποιείται για να επαναλαμβάνει αριθμό στηλών ή γραμμών, όσες το πλήθος που ορίζουμε στο πρώτο όρισμα και έχουν το μέγεθος που ορίζουμε στο δεύτερο όρισμα

Ορισμένα παραδείγματα:

width: min(50%, 300px);

height: max(100px, 3rem);

.c1 {grid-template-columns: minmax (20px, 20%) 1fr}. Θα δημιουργηθούν δύο στήλες. Η πρώτη στήλη θα έχει εύρος ανάμεσα στην μικρότερη και μεγαλύτερη τιμή από τα 20px και το 20% του περιέκτη.

.c2 {grid-template-columns: repeat (4, 1fr); } Θα δημιουργηθούν 4 στήλες, που θα μοιραστούν τον διαθέσιμο χώρο, γιατί όλες είναι 1fr.

.c3 {grid-template-columns: repeat (2, 50px 1fr); }. Είναι ισοδύναμο με: .c3 {grid-template-columns: 50px 1fr 50px 1fr;}. Θα δημιουργηθούν δηλαδή 4 στήλες. Η πρώτη και η Τρίτη στήλη θα έχουν εύρος 50px, η δεύτερη και τέταρτη θα μοιραστούν τον διαθέσιμο χώρο που περισσεύει, γιατί έχουν 1fr.

Δύο ακόμα λέξεις κλειδιά είναι οι **auto-fill** και **auto-fit**.

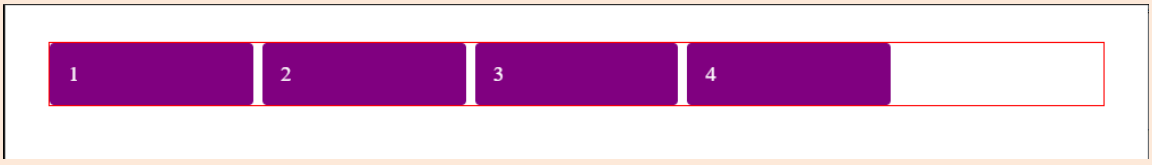
Η **auto-fill** προσαρμόζει σε μια σειρά όσες περισσότερες στήλες είναι εφικτό, ακόμα κι αν είναι άδειες.

Η **auto-fit** προσαρμόζει όλες τις στήλες που υπάρχουν στο χώρο. Εκτείνει τις στήλες για να γεμίσουν τον διαθέσιμο χώρο και να μην μείνει κενό.

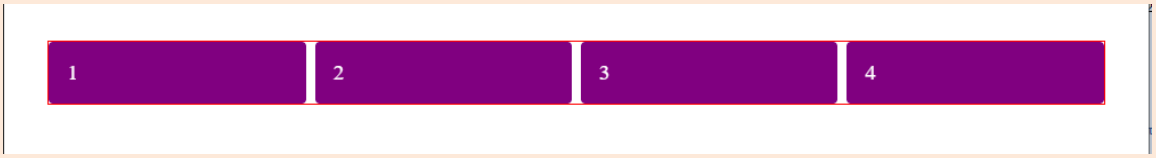
```
<html>
  <head>
    <style>
      .container {
       margin: 40px;
       display: grid;
       border:1px solid red;
       grid-gap: 10px;
       grid-template-columns: repeat(auto-fill, minmax(200px,1fr));
        }
      .item {
       background-color: purple;
       color: #fff;
       border-radius: 5px;
       padding: 20px;
       font-size: 150%;
        }
    </style>
  </head>
  <body>
   <div class="container">
      <div class="item">1</div>
      <div class="item">2</div>
      <div class="item">3</div>
      <div class="item">4</div>
   </div>
  </body>
</html>
```

Στο παράδειγμα μας φτιάχνουμε με το στοιχείο div ένα περιέκτη και δίνουμε το όνομα container στην κλάση του ( ). Το πλαίσιο που τον περιβάλλει θα έχει μια κόκκινη γραμμή πάχους 1px (border:1px solid red;). Βάλαμε και ένα περιθώριο 40px (margin: 40px;) για να έχει έναν αέρα από τις πλευρές του παραθύρου. Με την display: grid; Δημιουργούμε ένα πλέγμα στον περιέκτη, όπου θα τοποθετηθούν τα στοιχεία που περιέχει. Οι στήλες του πλέγματος θα είναι όσες χωρέσουν στο εύρος του παραθύρου, γιατί βάλαμε τη λέξη κλειδί auto-fill. Επειδή έχουμε βάλλει minmax(200px,1fr) στη συνάρτηση repeat, οι στήλες δεν θα είναι μικρότερες από 200px και θα μοιράζονται αναλογικά τον διαθέσιμο χώρο (1fr). Ανάμεσα στα κελιά θα υπάρχει κενό 10px (grid-gap: 10px;).

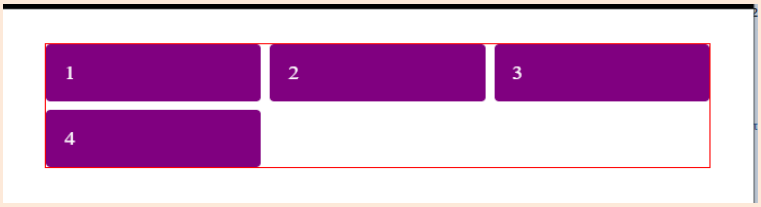
Μέσα στον περιέκτη φτιάχνουμε 4 στοιχεία, πάλι με ένα στοιχείο div και δίνουμε το όνομα item στην κλάση. Στην οθόνη θα δούμε το παρακάτω αποτέλεσμα:

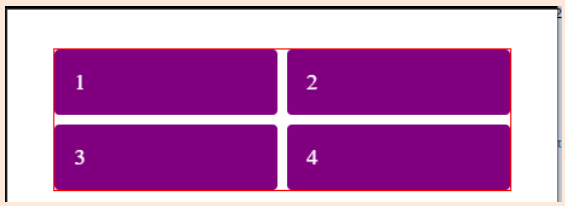


Αν αλλάξουμε το auto-fill και βάλουμε auto-fit, βλέπουμε ότι δεν μένει κενό στο τέλος.

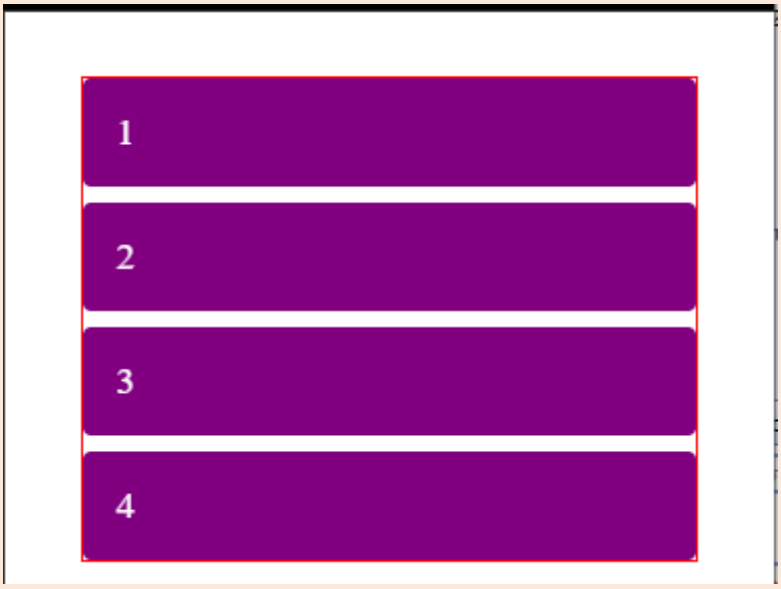


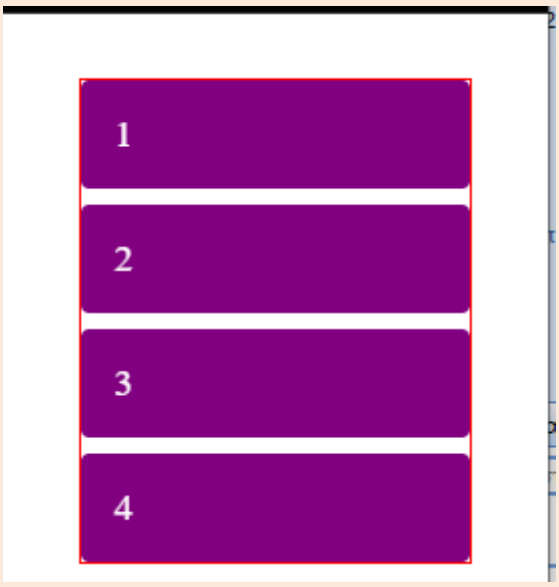
Αν μειώσουμε το εύρος του παραθύρου, θα δούμε και στις δύο περιπτώσεις τα ακόλουθα αποτελέσματα:





Επειδή οι στήλες δεν μπορούν να έχουν εύρος κάτω από 200px, όταν μειώνεται το εύρος του παραθύρου και δεν χωράνε όλα τα στοιχεία, δημιουργούνται 3 στήλες ή 2 στήλες ή μία.





Αν το εύρος του παραθύρου γίνει μικρότερο από 200px, τα στοιχεία δεν χωράνε ούτε σε μία στήλη και έχουμε υπερχείλιση.