Front-End Web Development

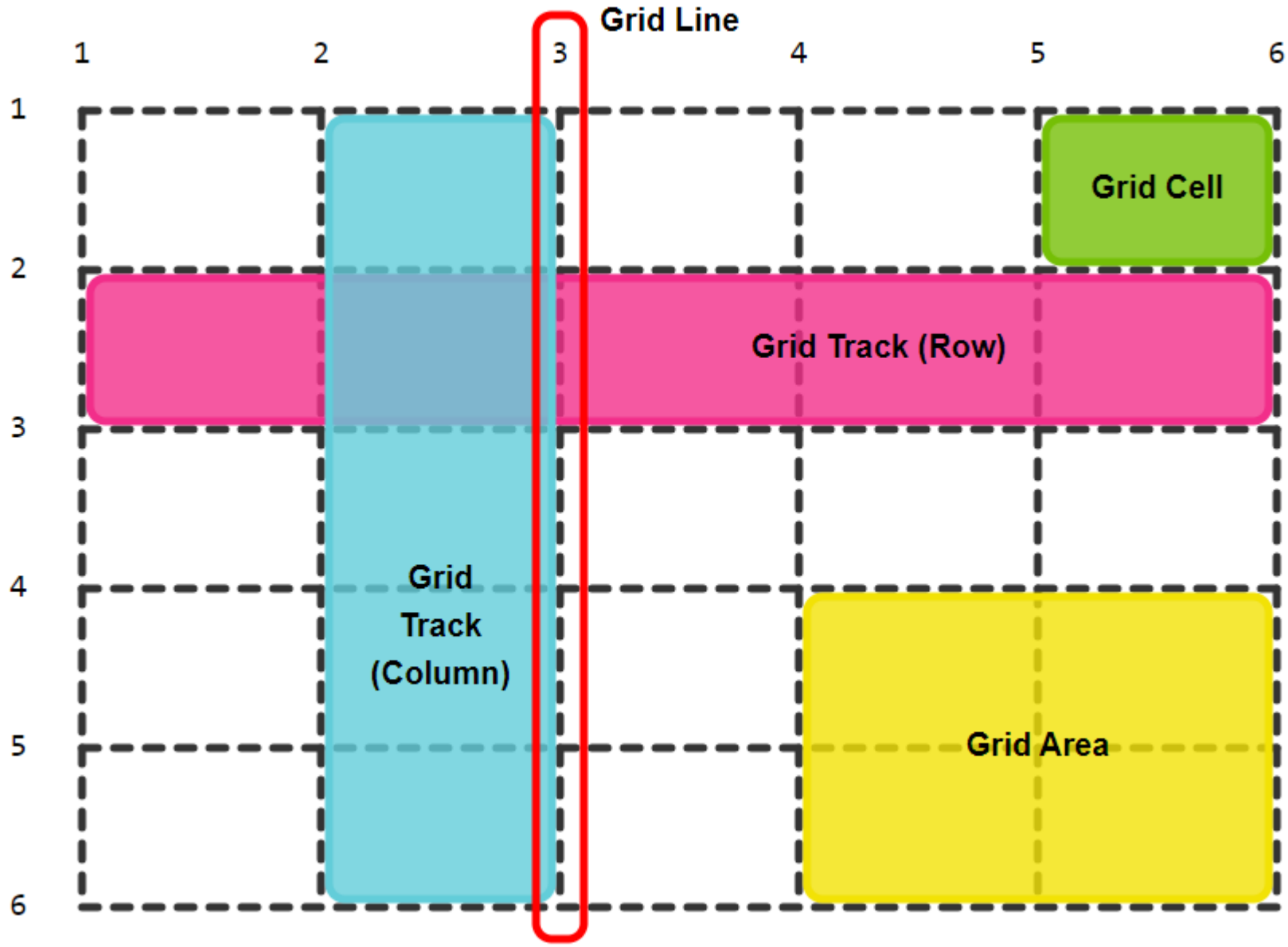
CSS Grid

CSS Grid Layout

- Модулът **CSS Grid Layout** въвежда система за оформление, базирана на мрежова структура, която не е ограничена от конкретната структура на съдържанието. Подходът е „първо оформлението“ - дефинира оформление и след това поставя съдържание вътре в него. Това улеснява проектирането на сложни респонсив уеб страници и адаптирането им към различните устройства и размери на екраните.
- CSS Grid е **двуизмерен** и работи по хоризонталната ос и по вертикалната ос т.е. с редовете и колоните едновременно. Позиционира елементите в отделните клетки на мрежата. Има мощни възможности за контрол на оразмеряването, позиционирането, подредбата и подравняването на елементи в двете измерения, предоставяйки гъвкаво пренареждане на оформленията, което е полезно при изграждане на респонсив дизайн на сайта.
- Елементите на мрежата могат да се припокриват, когато са позиционирани в пресичащи се зони на мрежата или дори когато са позиционирани в непресичащи се зони - чрез отрицателни полета или позициониране.

- Редовете и колоните на мрежата се декларират и оразмеряват или явно чрез експлицитните свойства на мрежата, или се създават имплицитно (генерират се автоматично), когато елементите са поставени извън експлицитната мрежа или има повече елементи, отколкото клетки. Мощната функция за автоматично поставяне позволява лесно да се запълва наличното пространство. Така Grid предлага гъвкаво оформление без ограничение на броя на колоните.
- Grid контейнерите могат да бъдат вложени (subgrids) или смесени с гъвкави flex контейнери, ако е необходимо, за създаване на по-сложни интерактивни оформления.

- **Grid Container** - Grid контейнерът е родителски елемент, който съдържа елементите на мрежата.
- **Grid Items** - Мрежовите елементи са директните деца на grid контейнера.
- **Grid Lines** - Линиите на мрежата се отнасят до вертикалните и хоризонталните линии, които разделят мрежата. Хоризонталните линии разделят редовете, а вертикалните отделят колоните. Линиите на мрежата съществуват от двете страни на колона или ред. Те могат да бъдат посочени с цифров индекс или с име. Елемент от мрежата препраща към мрежовите линии, за да определи позицията си в мрежата като използва свойствата за разположение.
- **Grid Tracks** – Трасето / тракът на мрежата, общ термин за колона или ред, се отнася до пространството между две линии на мрежата. Това може да бъде вертикално или хоризонтално пространство.
- **Grid Cells** - Мрежовите клетки са отделните единици, образувани от пресичането на редовите линии и колонните линии.
- **Grid Areas** - Зоната на мрежата се отнася до логическото пространство, заобиколено от четири линии на мрежата. Мрежовата област може да обхваща произволен брой клетки от мрежата.
- **Grid Gaps** - Пространството между редовете и колоните в мрежата.


Grid Line
1
2
3
4
5
6
Grid Cell
Grid Track (Row)
Grid
Track
(Column)
Grid Area

Свойства на CSS Grid Container

- **Настройване на мрежата**

Първа стъпка при grid базирано оформление е да се дефинира grid контейнер.

display: grid | inline-grid; - Превръща HTML елемент в мрежов контейнер.

grid – генерира мрежа на ниво block

inline-grid – генерира мрежа на ниво inline

```
.container {
    display: grid;
    }
```

- **Специфициране на редовете и колоните на мрежата**

На втората стъпка се определят редове и колони в grid контейнер.

```
. container {
    display: grid;
    grid-template-columns: 100px 200px 100px;
    grid-template-rows: 300px 180px;
    grid-gap: 20px;
    }
```

- **grid-template-columns**: column1-size column2-size column3-size ...

Определя броя на колоните и размера по ширината (px, em,%, auto или комбинация от мерни единици 100px 25% 3em;).

grid-template-columns: none|auto|max-content|min-content|length|initial|inherit;

grid-template-columns: 100px 200px 100px;

- **grid-template-rows**: row1-size row2-size row3-size ...

Определя размера (височината) и броя на редовете.

grid-template-rows: none|auto|max-content|min-content|length|initial|inherit;

grid-template-rows: 100px 60px 80px;

Стойност min-content: минималният размер на съдържанието -

ред с текст „E pluribus unum“, минималното съдържание е ширината на думата „pluribus“.

Стойност max-content: максималният размер на съдържанието -

ред с текст „E pluribus unum“, максималното съдържание е дължината на целия текст.

- **CSS дробна единица (Fr)** - разделя наличното пространство в grid контейнера на фракции. Разпределя наличното пространство между колони или редове по гъвкав и динамичен начин.

  grid-template-rows: 1fr 1fr 2fr;

  grid-template-columns: 1fr 2fr 1fr; - разделя наличното хоризонтално пространство на четири равни части: първата колона заема една част, втората колона заема две части и третата колона - една част.

  grid-template-columns: 100px 100px 1fr;

  grid-template-columns: 1.5fr 3fr 4.5fr;

- **Функцията minmax()** дефинира мин. и макс. размер на ред или колона.

Синтаксисът е: minmax(минимален размер, максимален размер)

Функцията minmax() приема 2 аргумента:

grid-template-columns: minmax(100px, 300px) 200px; - първата колона може динамично да варира по ширина минимум 100px до максимум 300px и втора колона с фиксирана ширина от 200px.

Освен стойностите (px, em, % и т.н.), функцията minmax() приема и auto стойност - разтегля въз основа на размера на съдържанието.

grid-template-rows: minmax(auto, 80%) 1fr 3em; - редът автоматично коригира ширината си, за да отговори на съдържанието си, но няма да надвишава 80%.

- **Функцията repeat()** дефинира повтарящите се решетки (grid tracks – колони / редове). Това е полезно за мрежи с елементи с еднакви размери. Осигурява възможност да се посочат колко колони да са в мрежата и след това се оставя браузърите да се справят с отзивчивостта на тези колони.

Синтаксисът е: repeat(брой, размер);

grid-template-columns: repeat(12, 1fr)

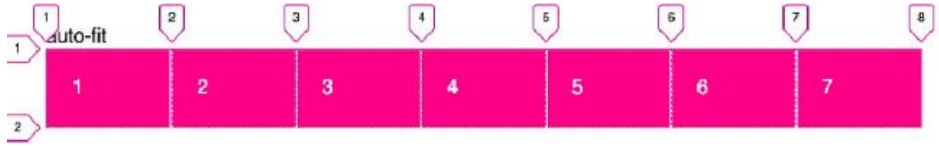
- Ако колоните са много, може да се ‘смачка’ съдържанието – за това се ползва функцията minmax() и ключовите думи за автоматично побиране **auto-fit** или автоматично попълване auto-fill. Така се показват по-малко колони на по-малки размери на прозорците или повече колони, ако екранът позволява повече, без да е необходимо да се пише медийна заявка, която да диктува това реагиращо поведение.

grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));

- При автоматичното побиране auto-fit наличните колони ще се разтегнат, за да запълнят цялата ширина на реда, като преди това са добавени всички колони и после празните са свити.
- С автоматичното попълване auto-fill, браузърът ще позволи на празни колони да заемат мястото в реда с размер, указан от втория параметър.

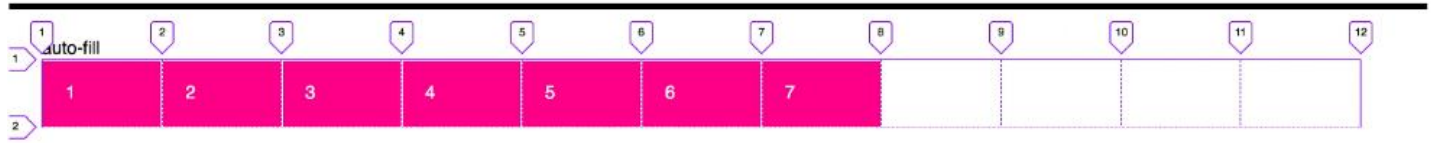
Фракционната единица fr гарантира и в двата варианта, че в случай, че ширината позволява част от колона да се побере, но не и цяла, това пространство да се разпредели върху колоните, които вече се побират, така няма да остане празно място в края на реда.


auto-fill
1
2
3
4
5
6
7


auto-fit
1
2
3
4
5
6
7


auto-fill
1
2
3
4
5
6
7


auto-fit
1
2
3
4
5
6
7


auto-fill
1
2
3
4
5
6
7
8
9
10
11
12


auto-fit
1
2
3
4
5
6
7
13

- Наименоване на линиите на мрежата с имена

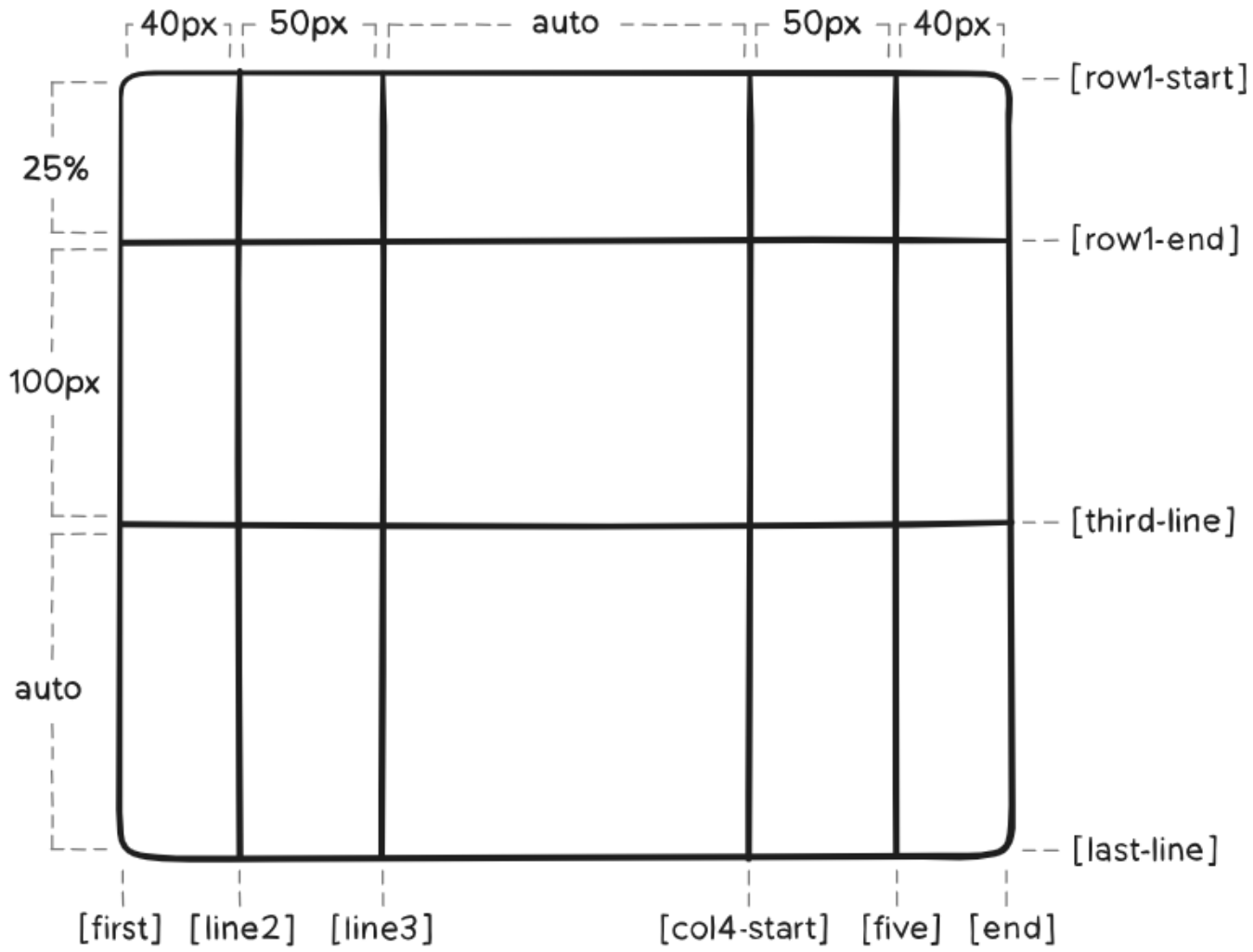
Явното деклариране на имената се прави в grid-template-columns и grid-template-rows

grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];

grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line];

grid-template-rows: [row1-start] 25% [row1-end row2-start] 25% [row2-end];

- Използване - при специфициране на елементите на мрежата.
- Може само ключови линии да се наименоват.


40px
50px
auto
50px
40px
[row1-start]
25%
[row1-end]
100px
[third-line]
auto
[last-line]
[first]
[line2]
[line3]
[col4-start]
[five]
[end]

- **grid-template-areas** дефинира области в Grid. Наименоват се елементи с помощта на свойството grid-area и след това се позоваваме на тези имена в grid-template-areas, за да се създаде общото оформление на контейнера.
- Повтарянето на името на мрежова област кара съдържанието да обхваща тези клетки. Точка означава празна клетка. Самият синтаксис осигурява визуализация на структурата на мрежата.

Синтаксисът е: grid-template-areas: none|itemnames;

```
grid-template-areas:
    "име-област1 име-област2 име-област3"
    "име-област4 име-област5 име-област6"
      /* ... допълнителни редове ... */;
```

- Всеки ред в кавичките представлява ред в мрежата, всяко име на област съответства на именувана област.

- grid.html

```
div.container { display: grid;
    grid-template-columns: 1fr 3fr 1fr;          /*  дефиниране на колоните */
    grid-template-rows: 140px 300px 100px;    /* дефиниране на редовете */
    grid-template-areas:
        "header header header"
        "sidebar1 main sidebar2"
        "footer footer footer";          /* дефиниране на областите */
     text-align: center;}
    header {
          grid-area: header;    /* поставяне на header с header grid област */
          background-color: purple; }
    aside.left {  grid-area: sidebar1;  }  /* поставяне на left sidebar с sidebar1 grid област */
    aside.right {  grid-area: sidebar2; }  /* поставяне на right sidebar с sidebar2 grid област */
    main {  grid-area: main;    /* поставяне на main content с main grid област */
          background-color: orange; }
    footer { grid-area: footer; /*поставяне на footer с footer grid област*/
          background-color: skyblue; }
```

друг начин на изграждане – съща визуализация
https://www.w3schools.com/css/tryit.asp?filename=trycss_grid_layout_named

- **grid-template** е съкратено свойство за указване на свойствата в една декларация:

grid-template: grid-template-rows / grid-template-columns | grid-template-areas|none|initial|inherit;

grid-template: 50px 150px 50px / 1fr 1fr 1fr;

- Свойството **grid-column-gap** (column-gap) дефинира разстоянието между колоните в мрежов контейнер.

Синтаксисът е: gird-column-gap: стойност - (px, em, % и т.н.)

grid-column-gap: 20px;

- Свойството **grid-row-gap** (row-gap) дефинира разстоянието между редовете на мрежовия контейнер.

Синтаксисът е: gird-row-gap: стойност - (px, em, % и т.н.)

gird-row-gap: 30px;

Ако не е посочен row-gap, той получава същата стойност като column-gap.

- **Grid-gap** (gap) е съкратено свойство за указване на grid-column-gap и grid-row-gap в една декларация.

grid-gap: 15px 10px; или gap: 15px 10px;

gap: 50px; двете са с тази стойност

- Елементите на мрежата, които не са явно поставени, автоматично се поставят в незаето място в мрежовия контейнер чрез алгоритъма за автоматично поставяне. Ситуации, примерно, когато има повече елементи от мрежата, отколкото клетки в мрежата или когато елемент от мрежата е поставен извън явната мрежа, или колони, които са дефинирани от grid-template-areas, които нямат експлицитен размер, зададен от grid-template-columns.

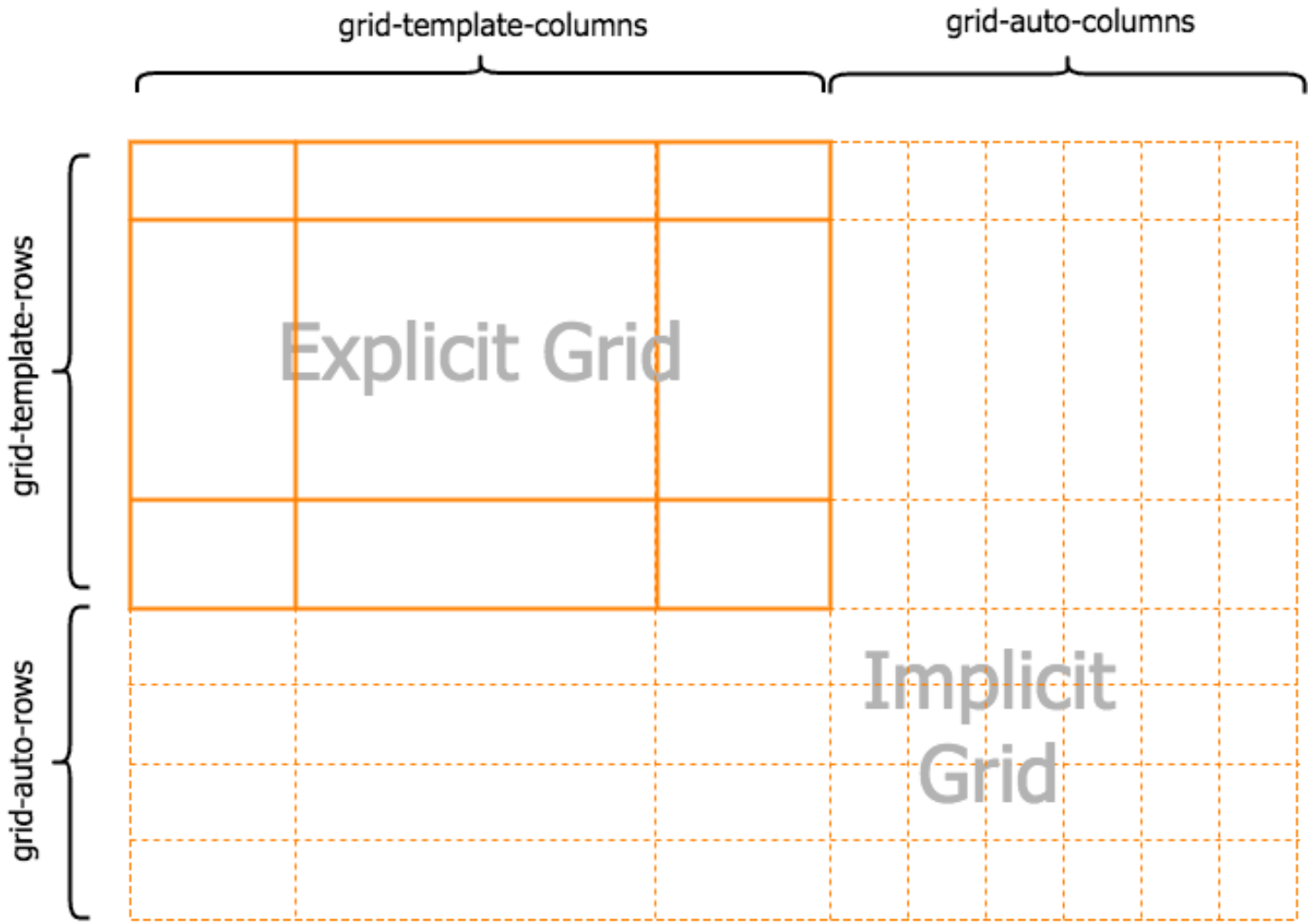
Задаване на автоматични решетки

- grid-auto-columns – Определя размера на автоматично / имплицитно генерираните колони.

  grid-auto-columns: auto|max-content|min-content|length;

- grid-auto-rows - Определя размера на автоматично / имплицитно генерираните редове.

  grid-auto-rows: auto|max-content|min-content|length;


grid-template-columns
grid-auto-columns
grid-template-rows
grid-auto-rows
Explicit Grid
Implicit Grid

- **grid-auto-flow** дефинира разположението на автоматично / имплицитно генерираните елементи от мрежата т.е. как тези елементи се вливат в мрежата.

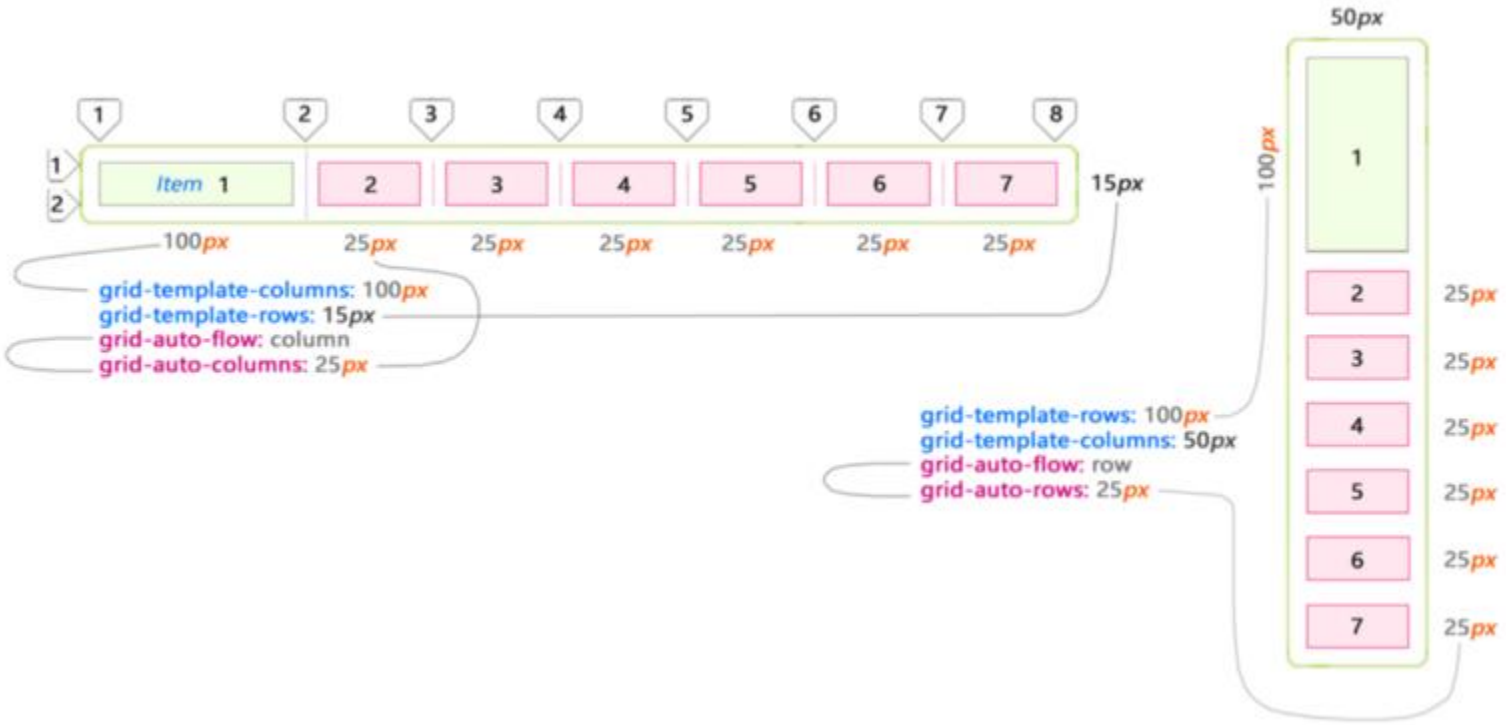
grid-auto-flow: row|column|dense|row dense|column dense;

grid-auto-flow: row; - поставя елементи, като запълва всеки ред, добавяйки нови редове, ако е необходимо.

grid-auto-flow: column; - поставя елементи, като попълва всяка колона, добавяйки нови колони, ако е необходимо.

grid-auto-flow: dense; – казва на алгоритъма за автоматично поставяне да се опита да запълни празнините по-рано в мрежата, ако по-късно се появят по-малки елементи или с непоследователен размер.

Съответно по редове и по колони - row dense|column dense



- Пример - има 7 елемента, но е дефинирана само една колона. Допълнителните 6 елемента от мрежата ще бъдат оразмерени имплицитно. Ако grid-auto-flow: column, ще се завърши с 6 колони. Ако grid-auto-flow: ред, ще се завърши с 6 реда.

- **grid** е съкратено свойство за указване на свойствата в една декларация:

grid: none|grid-template-rows / grid-template-columns|grid-template-areas | grid-template-rows / [grid-auto-flow] grid-auto-columns | [grid-auto-flow] grid-auto-rows / grid-template-columns | initial|inherit;

- grid: none - по подразбиране. Не специфицира размери.
- grid: grid-template-rows / grid-template-columns – задават се размери на редове / колони
- grid: grid-template-areas - задава размера на мрежовите области
- grid-template-rows / grid-auto-columns - указва размера на редовете и автоматичния размер на колоните
- grid-auto-rows / grid-template-columns - указва автоматичния размер и задава на колоните размер
- grid-template-rows / grid-auto-flow grid-auto-columns – указва размера на редовете и начина на поставяне на елементи и автоматичното оразмеряване на колоните.
- grid-auto-flow grid-auto-rows / grid-template-columns - указва начина на автоматичното поставяне на елементи и автоматичното оразмеряване на редовете и размера на колоните на мрежа.

Подравняване на мрежата

- **justify-items** контролира хоризонталното подравняване (ред) на всички елементи на мрежата в рамките на съответните клетки на мрежата. За да има ефект на подравняване, елементите на мрежата се нуждаят от свободно пространство.

Възможните стойности на свойството justify-items са start, end, center, stretch.

```
justify-items: end;
```

- **align-items** - контролира вертикалното подравняване (колона) на елементите на мрежата в рамките на съответните клетки на мрежата. За да има ефект на подравняване, елементите на мрежата се нуждаят от свободно пространство.

Възможните стойности на свойството align-items са start, end, center, stretch, baseline.

```
align-items: center;
```

- **place-items** е съкратено свойство за указване на align-items и justify-items в една декларация.

```
place-items: start center;
place-items: center;
```

и двете свойства са с тази стойност

Подравняване на мрежата

- **justify-content** контролира хоризонталното подравняване на цялото съдържание в grid контейнера.

Възможните стойности на свойството justify-content са start, end, center, stretch, space-around, space-between и space-evenly.

```
justify-content: stretch;
```

- **align-content** контролира вертикалното подравняване на цялото съдържание в grid контейнера.

Възможните стойности на свойството align-content са start, end, center, stretch, space-around, space-between и space-evenly.

```
align-content: start;
```

- **place-content** е съкратено свойство за указване на align-content и justify-content в една декларация.

Синтаксисът е: place-content: align-content justify-content

```
place-items: center;
```

Тук първата стойност задава align-content, а втората стойност задава justify-content. Ако втората стойност е пропусната, първата стойност се присвоява и на двете свойства.

```
place-content: center end;
```

```
place-content: end; и двете са с тази стойност
```

Специфициране на елементите на мрежата

- Трета стъпка е поставяне на елементи в отделните клетки на мрежата. Те могат да бъдат всякакви HTML елементи, които трябва да се позиционират в оформлението на мрежата.
- Определянето на местоположението на елемент се позовава на конкретни линии на мрежата. Така подреждането е независимо от момента на появяване в потока.

- **grid-column-start** дефинира на коя колонна линия ще започне елементът.

grid-column-start: auto|span n|column-line;

auto - стойност по подразбиране. Елементът ще бъде поставен, следвайки потока.

column-line - указва на коя колонна линия да започне – пореден номер или име, което да се отнася до назована колонна линия на мрежата.

span n - указва броя на колоните, които елементът ще обхваща или ако е име – елементът ще се простира, докато достигне колонната линия с предоставеното име.

- grid-column-end дефинира на коя колонна линия ще завърши елементът.

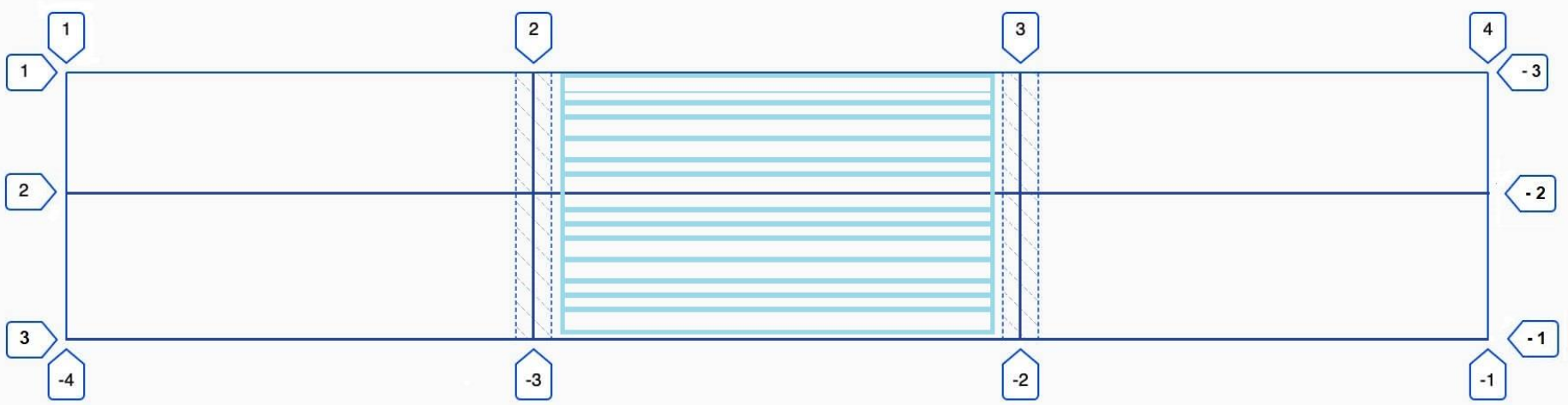
  grid-column-end: auto|span n|column-line;
- grid-row-start дефинира на коя хоризонтална линия ще започне елементът.

  grid-row-start: auto|span n|row-line;
- grid-row-end дефинира на коя хоризонтална линия ще завърши елементът.

  grid-row-end: auto|span n|row-line;

Ако не са декларирани grid-column-end и grid-row-end, елементът ще обхваща 1 трак по подразбиране.

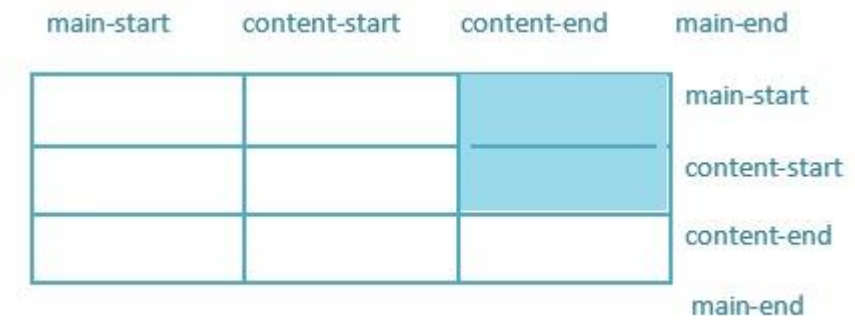
```
.item1 {
  grid-row-start: 1;
  grid-row-end: 3;
  grid-column-start: 2;
  grid-column-end: 3;
    }
```



- https://www.w3schools.com/cssref/tryit.php?filename=trycss_grid-column-start

Пример за специфициране на елемент с използване на имена за указване на мрежови линии

```
.grid-container {
        display: grid;
    grid-template-columns: [main-start] 1fr [content-start] 1fr [content-end] 1fr [main-end];
    grid-template-rows: [main-start] 100px [content-start] 100px [content-end] 100px [main-end];
            }
      .item1 {
       grid-column-start: content-end;
       grid-column-end: main-end;
       grid-row-start: main-start;
       grid-row-end: content-end;
            }
```



- **grid-row** е съкратено свойство за указване на свойствата в една декларация:

  grid-row: grid-row-start / grid-row-end;

  grid-row: 1 / 3;

  grid-row: main-start / content-end;

- **grid-column** е съкратено свойство за указване на свойствата в една декларация:

  grid-column: grid-column-start / grid-column-end;

  grid-column: 2 / 3;

https://www.w3schools.com/css/tryit.asp?filename=trycss_grid_item

- **grid-area** - указва размера и местоположението на мрежовия елемент, използва се и за присвояване на име на елемента. Наименуваните елементи могат да бъдат препратени към свойството grid-template-areas на мрежовия контейнер. Това е и съкратено свойство за указване на свойствата в една декларация:

grid-area: itemname | grid-row-start / grid-column-start / grid-row-end / grid-column-end;

grid-area: 1 / 2 / 5 / 6;

1 - grid-row-start, 2 - grid-column-start, 5 - grid-row-end, 6 - grid-column-end

grid-area: 1 / 2 / span 4 / span 4;

https://www.w3schools.com/css/tryit.asp?filename=trycss_grid_flexible_order

Подравняване на елементите на мрежата

- **justify-self** Подравнява елемент от мрежата вътре в една клетка по хоризонталната ос (ред).

Възможните стойности на свойството justify-self са start, end, center, stretch.

```
justify-self: stretch;
```

- **align-self** Подравнява елемент от мрежата вътре в една клетка по вертикалната ос (колона).

Възможните стойности на свойството align-self са start, end, center, stretch.

```
align-self: start;
```

- **place-self** е съкратено свойство за указване на align-self и justify-self в една декларация.

Синтаксисът е: place-self: align-self justify-self

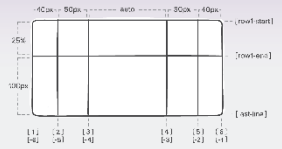
```
place-self: start center;
place-self: center;
```

Ако втората стойност е пропусната, първата стойност се присвоява и на двете свойства.

CSS Grid

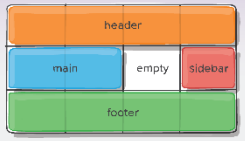
a guide from CSS-TRICKS

grid-template-columns grid-template-rows



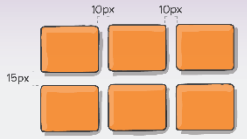
```
.container {
  grid-template-columns: <value> | <name>;
  grid-template-rows: <value> | <name>;
}
```

grid-template-areas



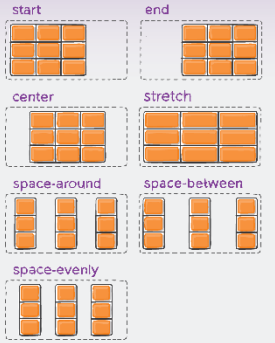

```
.container {
  grid-template-areas: "<name> | . | none";
}
```

column-gap, row-gap, gap




```
.container {
  column-gap: <value>;
  row-gap: <value>;
  gap: <row-gap> <column-gap>;
}
```

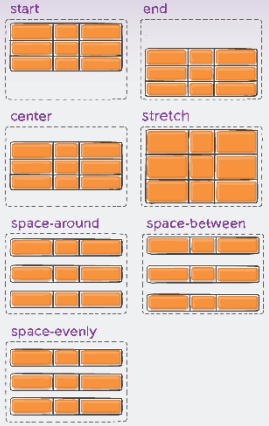
justify-content




```
.container {
  justify-content: start | end | center |
  stretch | space-around | space-between |
  space-evenly;
}
```

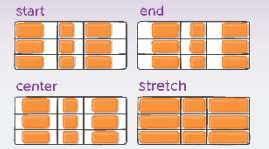
```
.container {
  display: grid; /* or inline-grid */
}
```

align-content



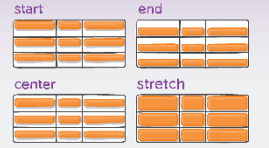

```
.container {
  align-content: start | end | center |
  stretch | space-around | space-between |
  space-evenly;
}
```

justify-items



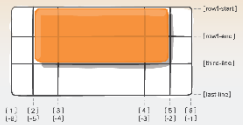

```
.container {
  justify-items: start | end |
  center | stretch;
}
```

align-items



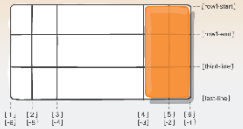

```
.container {
  align-items: start | end |
  center | stretch;
}
```

grid-column, grid-row



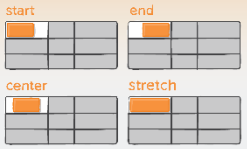
```
.item {
  grid-column: <start-line> / <end-line> |
  <start-line> / span <value>;
  grid-row: <start-line> / <end-line> |
  <start-line> / span <value>;
}
```

grid-area



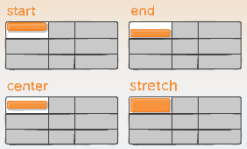
```
.item {
  grid-area: <row-start> / <column-start>
  / <row-end> / <column-end> | <name>;
}
```

justify-self




```
.item {
  justify-self: start | end | center |
  stretch;
}
```

align-self




```
.item {
  align-self: start | end | center |
  stretch;
}
```

Респонсив дизайн с Grid

- Чрез функциите на CSS Grid се осъществява и респонсив дизайн. Използват се и медийни заявки, за да дефинират различни мрежови структури и правила за позициониране въз основа на размера или ориентацията на екрана на устройството.

```
.container { display: grid;
grid-template-columns: 1fr 1fr;
}
@media screen and (min-width: 500px) {
. container {
grid-template-columns: 1fr 1fr 1fr; } }
@media screen and (min-width: 800px) {
.wrapper {
grid-template-columns: 1fr 1fr 1fr 1fr; } }
```

Респонсив дизайн с Grid

- Друг вариант

```
.grid-container { display: grid;
  grid-template-areas:
    'header header header header header header'
    'left middle middle middle middle right'
    'footer footer footer footer footer footer';
}
@media (max-width: 600px) {
        .grid-container  {
    grid-template-areas:
      'header header header header header header'
      'left left left left left left'
      'middle middle middle middle middle middle'
      'right right right right right right'
      'footer footer footer footer footer footer';
  }
}
```

https://labs.jensimmons.com/2019/01-006.html

https://alialaa.github.io/css-grid-cheat-sheet/

https://cssgridgarden.com/