

Нормален поток vs. Flexbox модел vs. Grid модел. Responsive Layout

ЕКІП 11



ЕКИПЪТ

Димитър Кърцъков
Лидия Шкортова
Оля Атанасова
Росица Тодорова
Стоян Тинчев



Съдържание

01

Нормален поток,
какво е Grid и
FlexBox

Роси

02

Flexbox model в
детайли

Лидка

03

Grid model в
детайли

Митко

04

Responsive layout
и Media Queries

Стоян

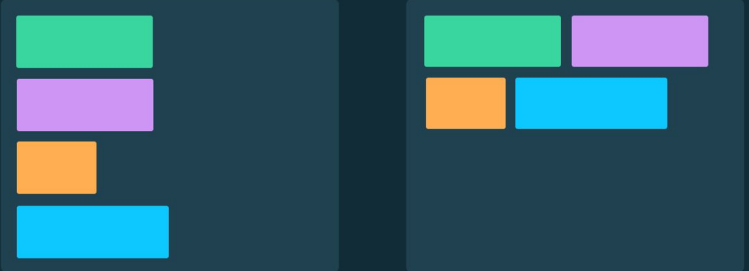
05

Сравнение и
обобщение

Оля

Какво представляват блоковите и вградените елементи

BLOCK VS INLINE



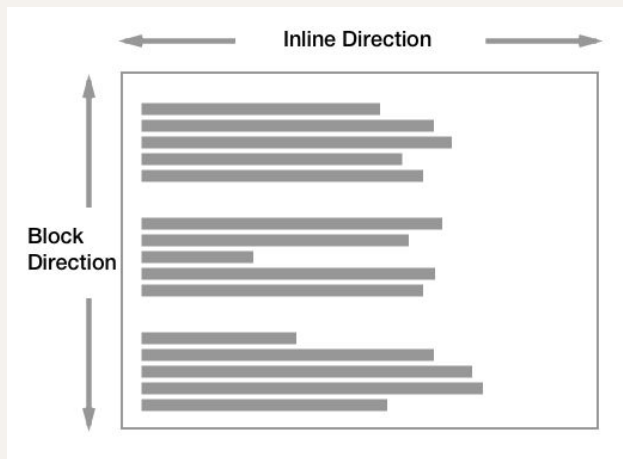
display: block;
Block elements stack, regardless of their width.

display: inline;
Inline elements flow from one line to the next.

Block Elements	Inline Elements
<ul style="list-style-type: none">• <p>• <h1>• • <hr>	<ul style="list-style-type: none">• <a>• • <i>•

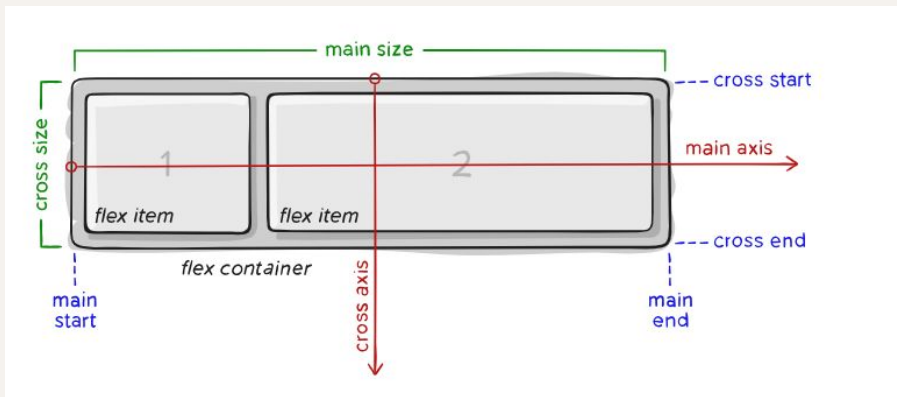
DIFFERENCE Between

Какво означава нормален поток ?



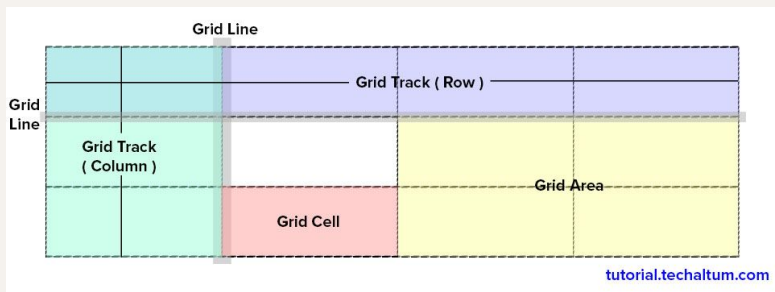
Какво е Flexbox model?

- Въведение
- Елементи в Flexbox

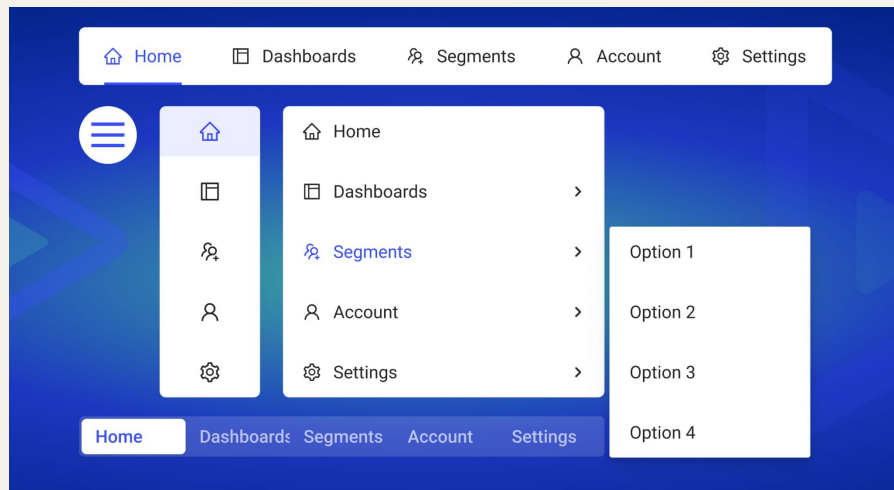
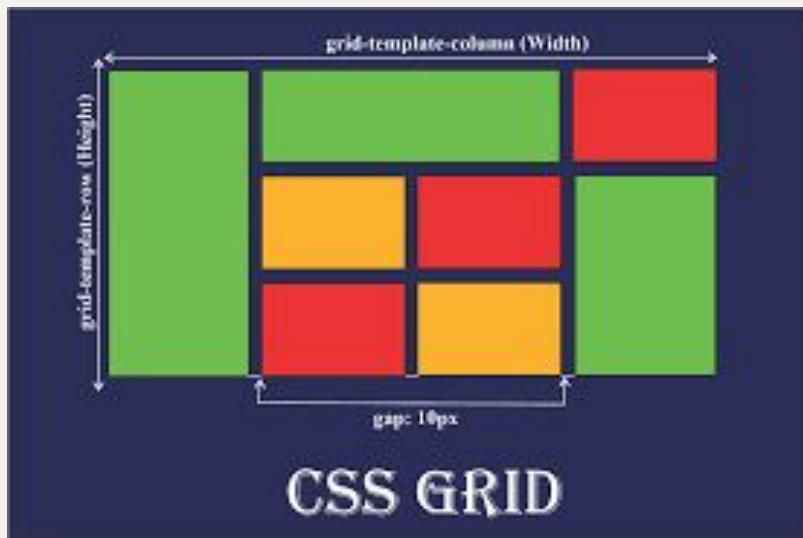


Какво е Grid Model?

- Въведение
- Основни елементи в Grid



Къде има Grid и Flexbox?



Flexbox оформление

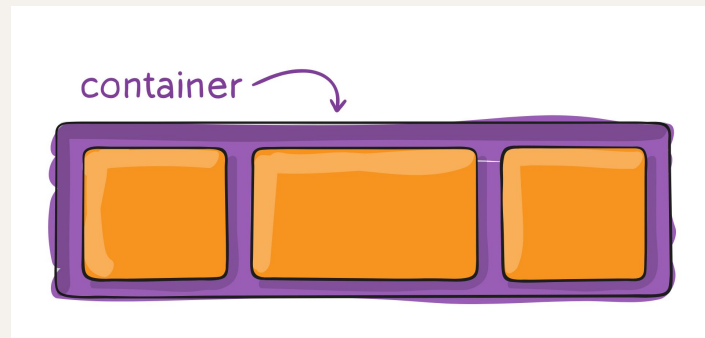


Свойства на flex контейнера

1) display

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

```
.container{  
  display:inline-flex;  
}
```



Свойства на flex контейнера

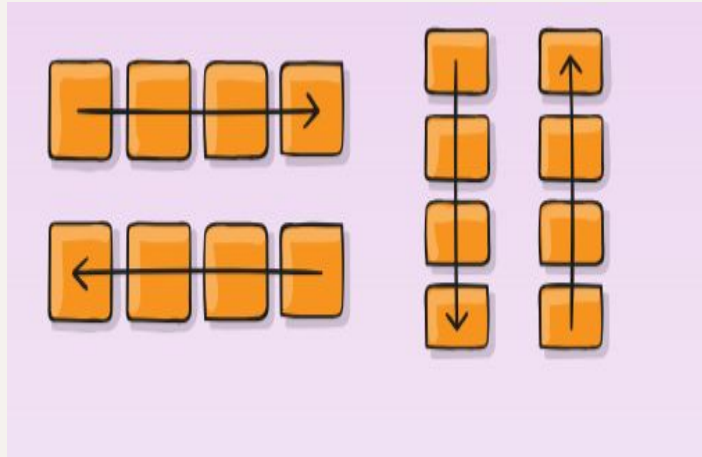
2) flex-direction

row

row-reverse

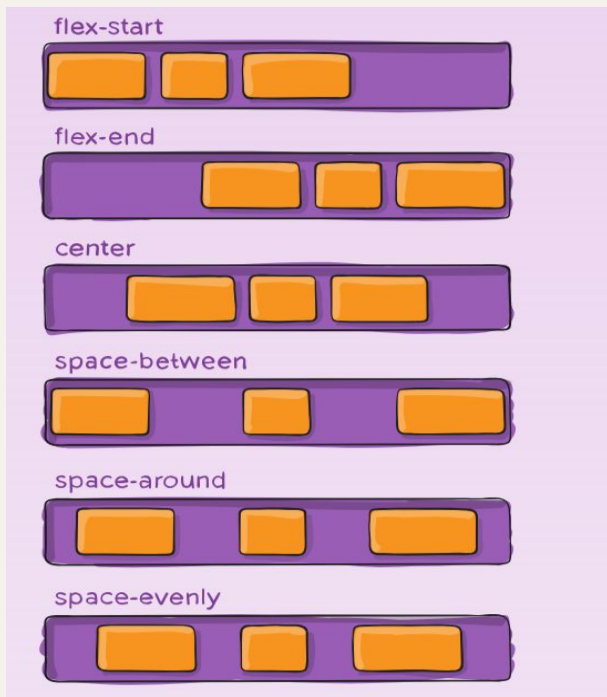
column

column-reverse

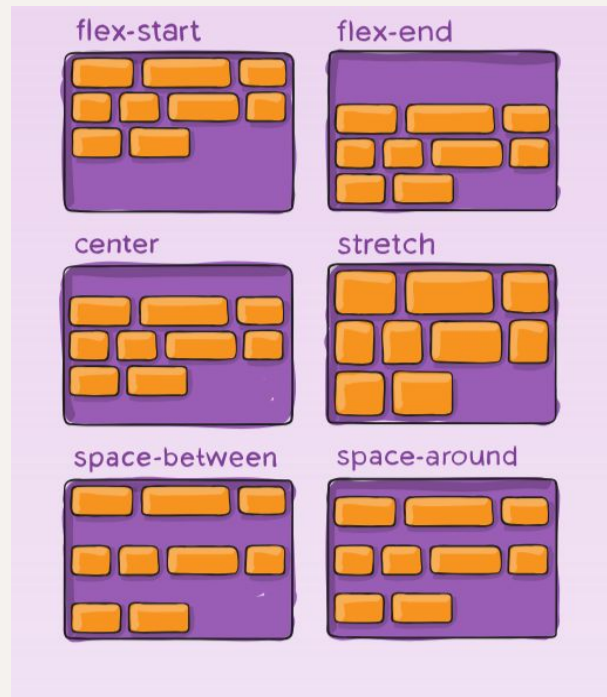


Свойства на flex контейнера

3) justify-content



4) align-items

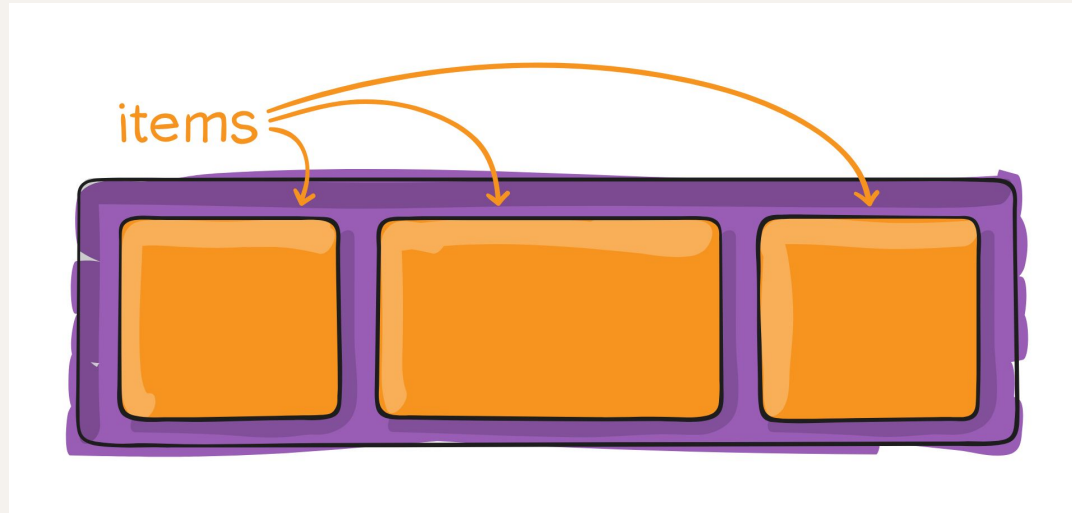


Свойства на flex контейнера

4) gap

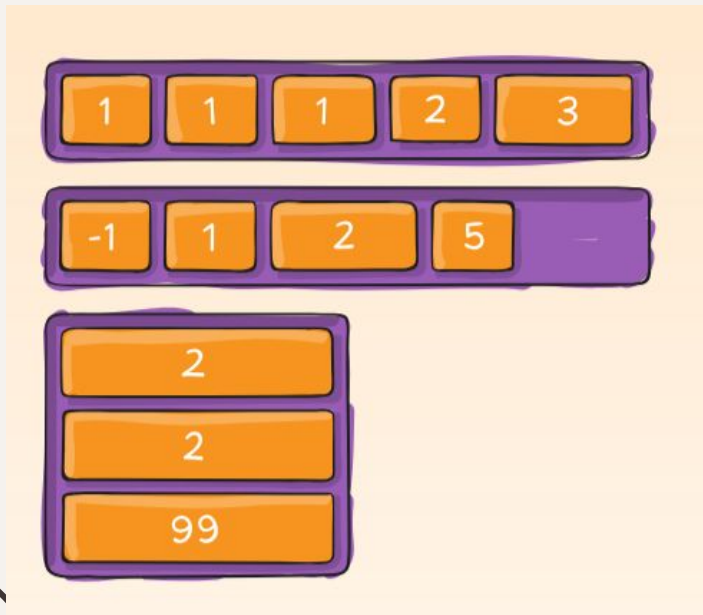


Свойства на flex елементите

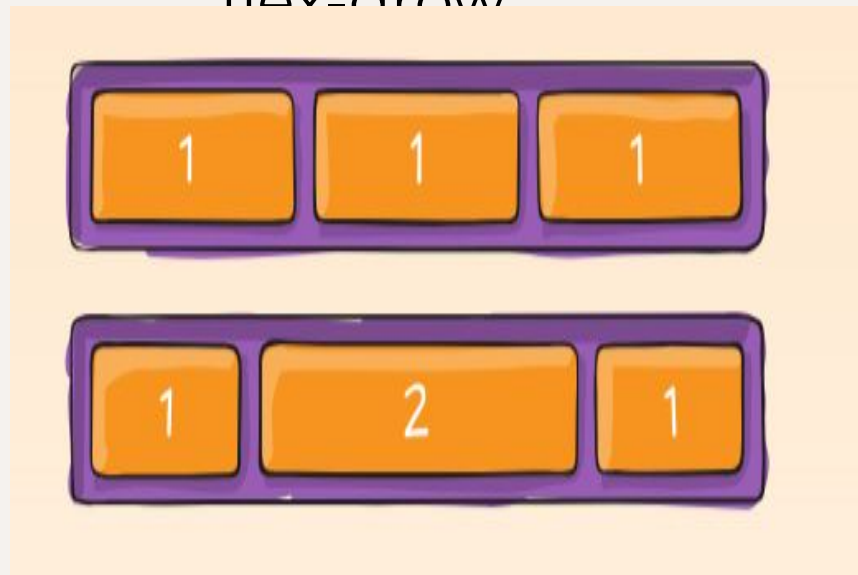


Свойства на flex елементите

1) order



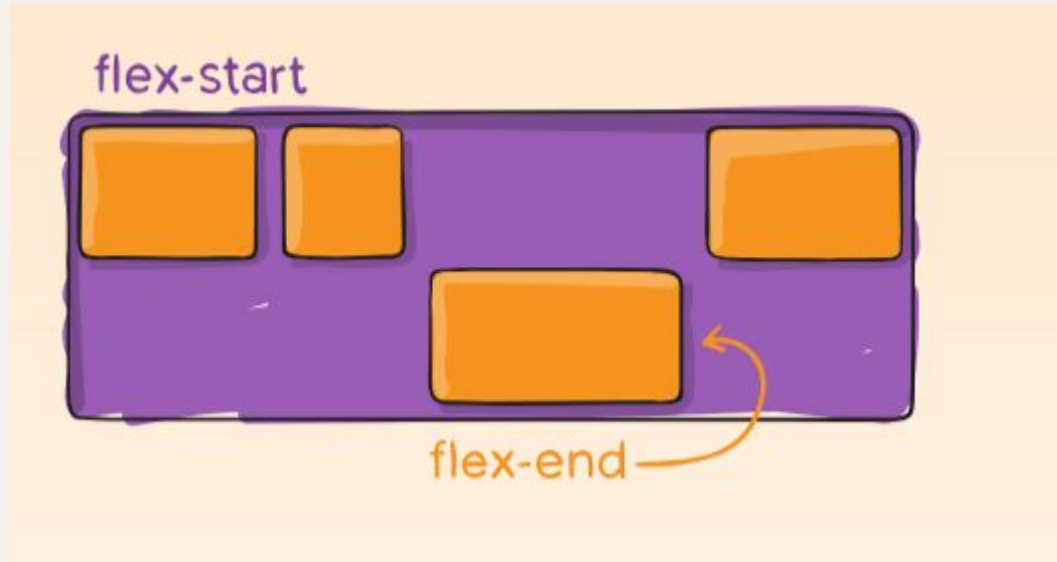
2)
flex-grow



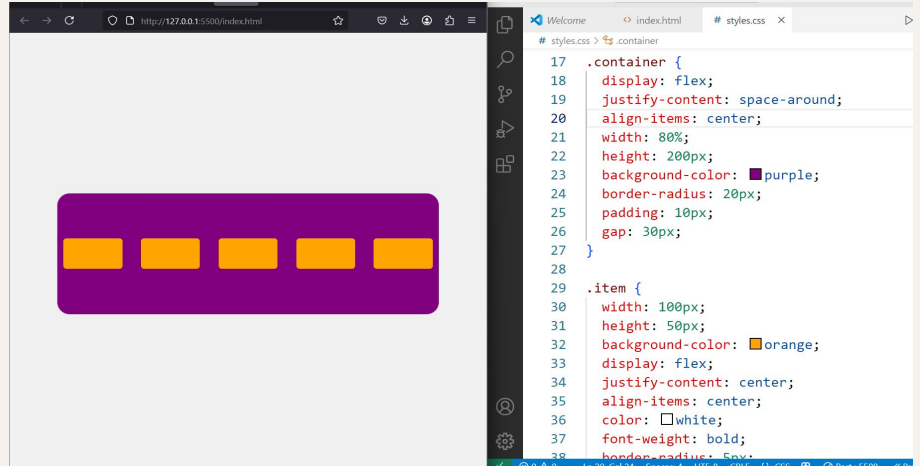
Свойства на flex елементите

3) align-self

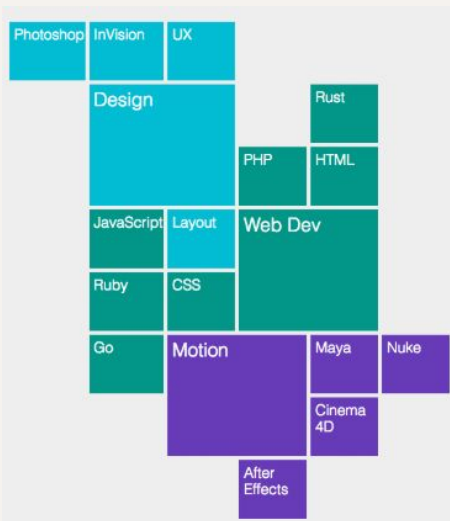
auto
flex-start
flex-end
center



Flexbox froggy & демо



Grid Layout



Димитър Кърцъков

Grid-template-rows/columns

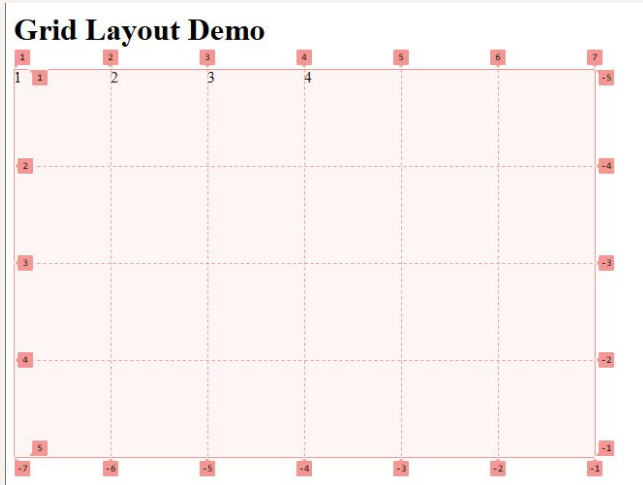
HTML

```
10 <body>|
11   <h1>Grid Layout Demo</h1>
12   <div class="grid-container">
13     <div class="grid-item item1">1</div>
14     <div class="grid-item item2">2</div>
15     <div class="grid-item item3">3</div>
16     <div class="grid-item item4">4</div>
17 </body>
```

CSS

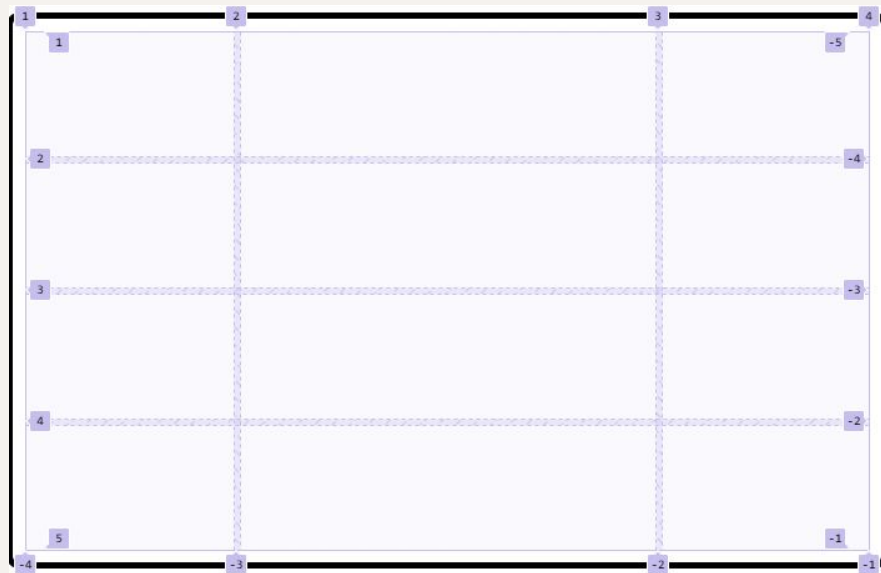
```
# style_demo.css > .grid-container
1 .grid-container{
2   display: grid;
3   grid-template-rows: 100px 100px 100px 100px;
4   grid-template-columns: 100px 100px 100px 100px 100px 100px;
}
```

Demo



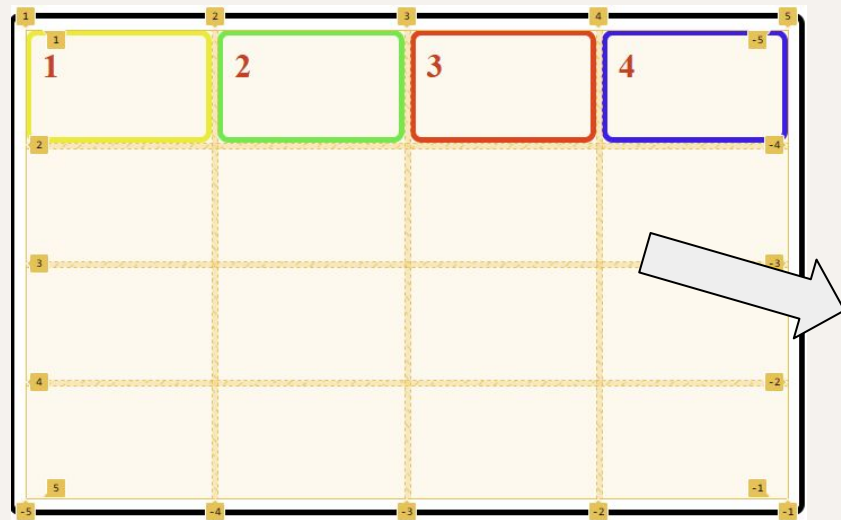
Параметри на grid-template-rows/cols

- пиксели (100px)
- пропорционални части (1fr 2fr)
- Функция repeat()




```
.grid-container{  
  display: grid;  
  grid-template-rows: 100px 100px 100px 100px;  
  grid-template-columns: 1fr 2fr 1fr; /*1fr = една пропорционална част*/  
}
```

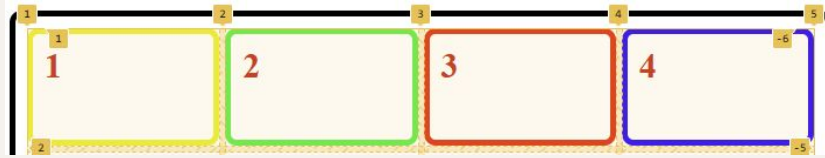
Gap



- задаване на разстояние между отделните елементи на grid-a
- функция repeat(4, 100px)

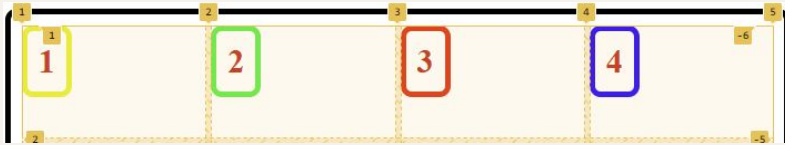
```
# style_demo.css >  .grid-container
1  .grid-container{
2      display: grid;
3      grid-template-rows: repeat(4, 100px);
4      grid-template-columns: repeat(4, 1fr); /* 1fr = 1 пропорционална част*/
5      gap: 5px; /* Разстояние между елементите в контейнера */
```

Подравняване на елементи на grid

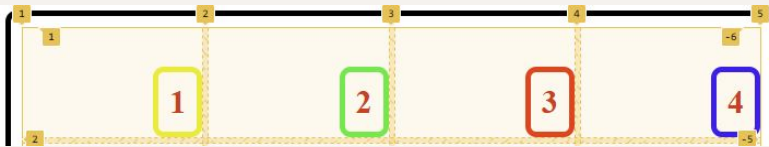


```
align-items: stretch;  
justify-items: stretch;
```

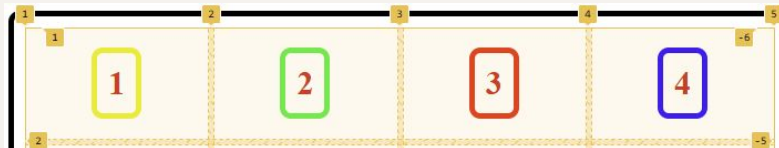
- align-items - по вертикала
- Justify-items - по хоризонтала



```
align-items: start;  
justify-items: start;
```



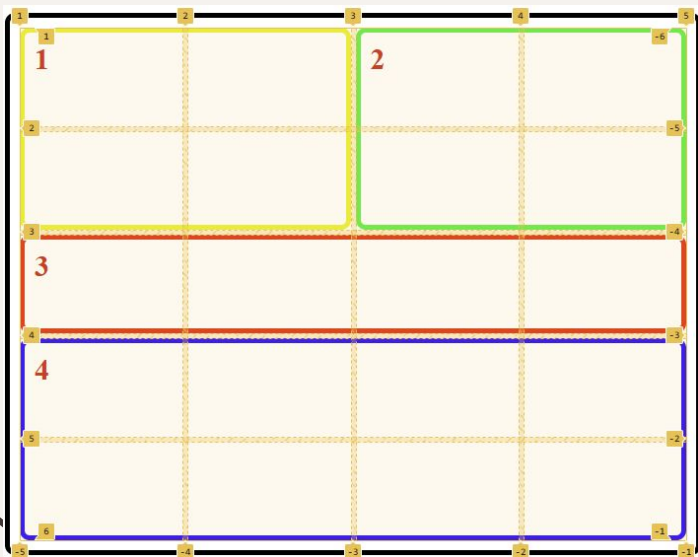
```
align-items: end;  
justify-items: end;
```



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

grid-row, grid-column, grid-area

Няколко различни начина за задаване на големината на отделните елементи. Различен синтаксис, подобна функционалност.



```
.item1{
  grid-row-start: 1;
  grid-row-end: 3;
  grid-column-start: 1;
  grid-column-end: 3;
}
.item2{
  grid-row: 1 / 3; /* rowstart / row-end */
  grid-column: 3 / 5; /* colstart / col-end */
}
.item3{
  grid-area: 3 / 1 / 4 / 5; /* rowstart / colstart / row-end / col-end */
}
.item4{
  grid-area: 4 / 1 / -1 / -1; /* rowstart / colstart / row-end / col-end */
}
```

Online grid-learning game

← → ↺

cssgridgarden.com

☆

📄

☰

D

GRID GARDEN

◀ Level 17 of 28 ▶

How about multiple items? You can overlap them without any trouble. Use `grid-area` to define a second area that covers all of the unwatered carrots.

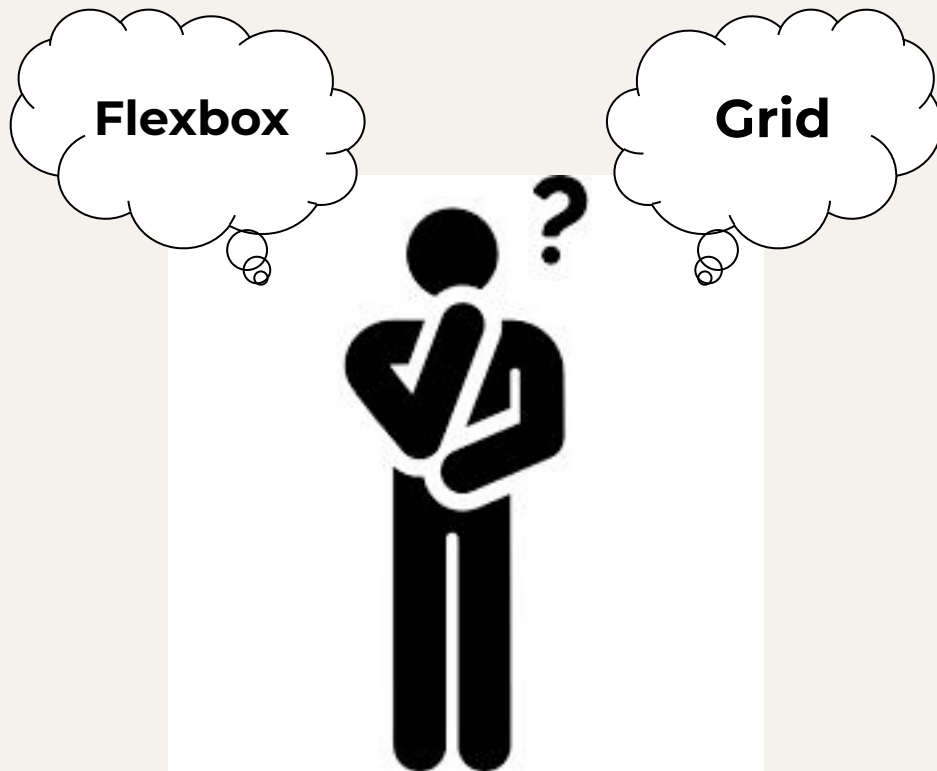
```
1 #garden {
2   display: grid;
3   grid-template-columns: 20% 20% 20% 20% 20%;
4   grid-template-rows: 20% 20% 20% 20% 20%;
5 }
6
7 #water-1 {
8   grid-area: 1 / 4 / 6 / 5;
9 }
10
11 #water-2 {
12   grid-area: 2 / 3 / 5 / 6
13 }
14
```

Next


Grid Garden is created by [Codepip](#) • [YouTube](#) • [Twitter](#) • [GitHub](#) • [English](#)

Want to learn more CSS? Play [Flexbox Froggy](#) or [Anchoreum](#).

Обобщение



Flexbox vs Grid

Характеристика	CSS Grid 	Flexbox 
Подредба	Подрежда елементите в редове и колони	Подрежда елементите в един ред или една колона
Подравняване	Хоризонтално и вертикално (в две посоки)	Само в основната посока (ред/колона)
Размери на елементите	Фиксирани, относителни или автоматични размери за редове и колони	Размерите зависят от съдържанието и свойствата на гъвкавост
Разстояния между елементи	Контролира разстоянието между редове и колони	Контролира разстоянието само между елементите в ред/колона

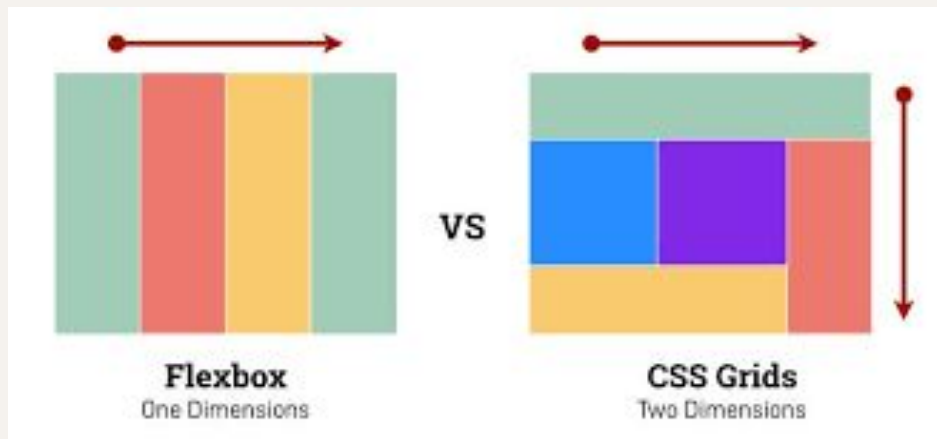
Как да изберем?

✓ Използвай Flexbox, когато:

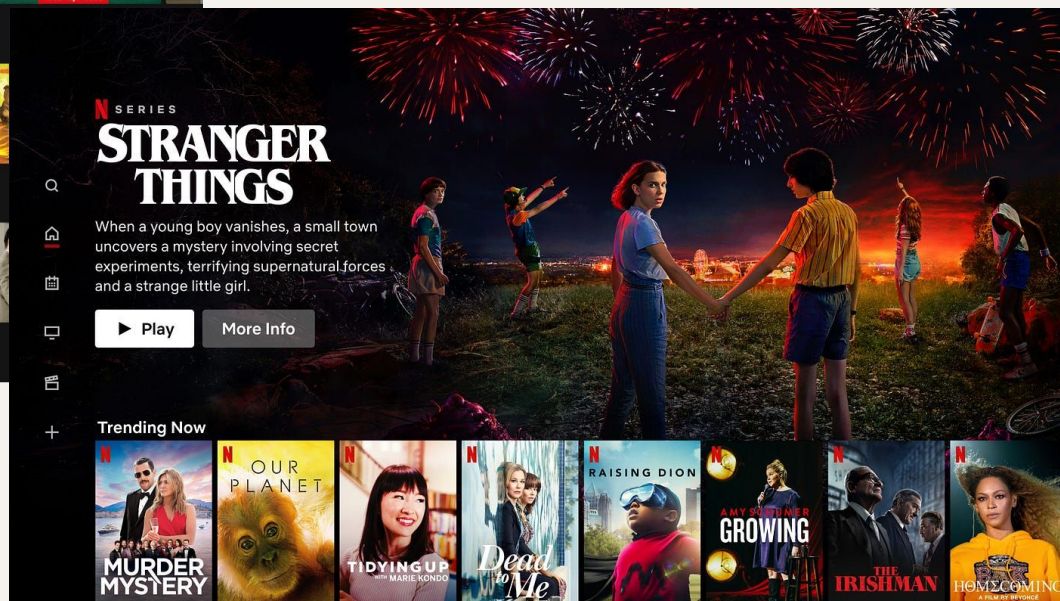
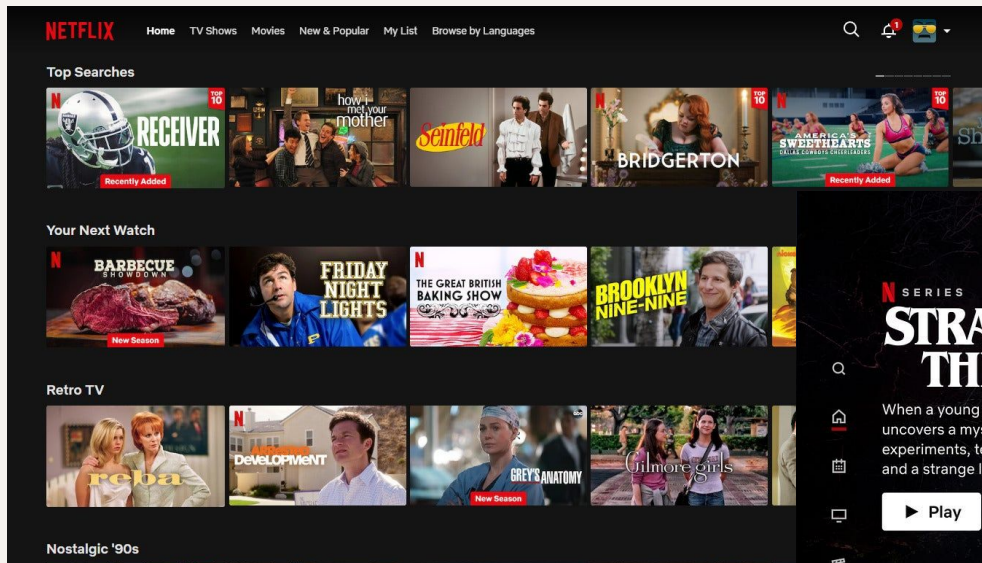
- Подреждаш елементи в един ред или една колона
- Искаш динамично адаптиране на съдържанието
- Трябва бързо и лесно да подравниш елементи

✓ Използвай Grid, когато:

- Създаваш цялостен макет на страницата
- Искаш да разделиш съдържанието в редове и колони
- Имаш нужда от точно позициониране на елементите



Пример



The image features a light gray background with dark gray wavy lines in the corners, creating a decorative border. The main text is centered and reads:

Responsive Layout & Media queries

Media Queries

```
@media [media_type] and (condition) {  
    /* CSS стилове, приложени само когато условието е вярно */  
}
```

Основни компоненти:

1. Media Type

- Определя типа устройство:
 - screen
 - print
 - all

```
@media screen and (max-width: 600px) {  
    body { background: red; }  
}
```

```
@media print {  
    body { background: white; color: black; }  
    .navigation { display: none; }  
}
```

Media Queries

2. Media Features (условия)

- Задават специфични характеристики, например:
 - `max-width`
 - `min-width`
 - `orientation`
 - `resolution`

```
@media (min-width: 768px) and (max-width: 1200px) {  
    .container { width: 750px; }  
}
```

Media Queries

```
@media screen and (min-width: 600px) and (orientation: landscape) {  
    /* стилове */  
}
```

Логически оператори

- **and** – комбинира условия, всички трябва да са изпълнени

```
@media (max-width: 600px), (orientation: portrait) {  
    /* стилове */  
}
```

- **,** (запетая) – действа като логическо OR (или)

- **not** – отрича цялото условие

```
@media not all and (orientation: landscape) {  
    /* стилове */  
}
```

RESPONSIVE WEB DESIGN



DESKTOP

Благодарим Ви за
вниманието! :)